PPO+: EFFECTIVE AND EFFICIENT POLICY OPTIMIZATION WITH SELF-VALUING TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning is central to improving language model reasoning, yet conventional PPO often trails GRPO-style methods and incurs extra cost from a separate critic. At the same time, PPO provides explicit token-level values that can guide learning from sparse rewards, leaving its potential underexplored. We introduce **PPO+**, which improves how value signals guide policy learning, integrates value estimation into the policy itself, and reuses these signals for prompt selection. PPO+ uses both token-level value estimates and within-prompt response comparisons to construct advantages for a single PPO objective. Motivated by our finding that policy representations already contain useful value information, we introduce a *Self-Valuing Transformer (SVT)* that reuses the policy backbone with a small detached value branch. We further introduce *Value-Guided Selection (VGS)*, which combines historical outcomes with early value estimates to prioritize prompts likely to yield mixed groups. Across five mathematical reasoning benchmarks, PPO+ improves over DAPO by 8.09 points on Qwen3-4B while using slightly less overall training time than the GRPO-style baseline. These results suggest that value estimation can become an internal policy capability, enabling stronger and more efficient reinforcement learning.

1 INTRODUCTION

Reinforcement learning with verifiable rewards has become a powerful way to improve mathematical reasoning in language models. Recent methods such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) achieve strong performance by comparing multiple responses to the same prompt, avoiding the need for a learned critic. PPO (Schulman et al., 2017), in contrast, retains an explicit value function and uses it to estimate token-dependent advantages (Schulman et al., 2015). In current reasoning systems, however, conventional PPO is often both less competitive and more expensive.

We argue that this trade-off is not fundamental. A learned value function provides information that group-level outcomes alone cannot: it tracks how expected future return changes along a generated trajectory. The key question is therefore not whether value estimation should be removed, but whether it can be made both more useful for policy learning and less costly to maintain. This motivates our central question: *can we retain explicit value estimation while matching the simplicity and efficiency of critic-free optimization?*

As illustrated in Figure 1, we introduce **PPO+**, which improves both how value signals are used and how they are produced. First, PPO+ combines token-level value advantages with comparisons among responses to the same prompt. These two signals capture different information: values distinguish states within a response, while group comparisons reveal how one response performs relative to alternatives for the same problem. PPO+ brings them together in a single PPO objective by normalizing value-based advantages within each prompt group. This simple change turns a weak PPO baseline into a much stronger policy optimizer in our experiments.

A stronger value-based optimizer still leaves one major cost: the critic itself. We therefore ask *whether value estimation really requires a second full language model*. Our actor–critic analysis shows that actor representations already contain substantial information useful for value prediction, even though later layers still develop role-specific structure. This observation motivates the ***Self-Valuing Transformer (SVT)***. Instead of maintaining a separate critic backbone, SVT reuses the policy representation and adds a small value-specific branch to predict future returns. The value

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.

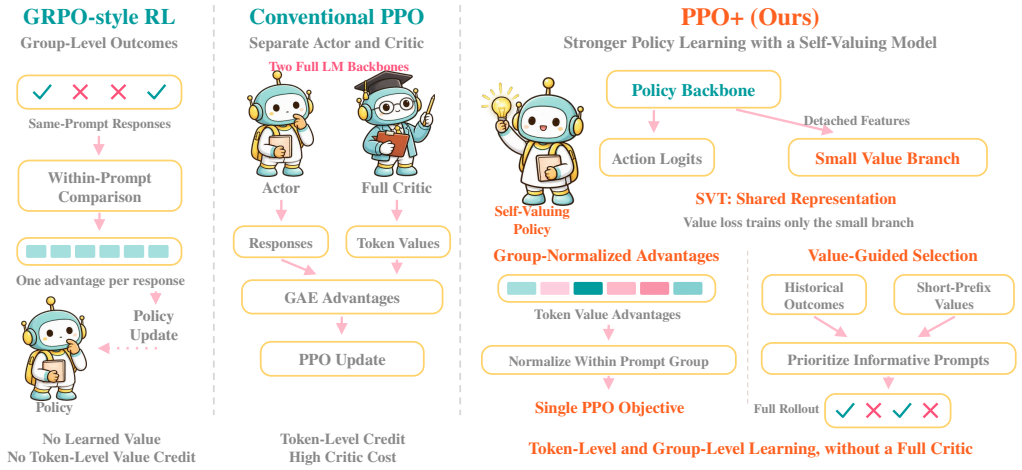

Figure 1: We introduce **PPO+**, which combines group-relative token credit with a self-valuing policy and reuses the same value signal for prompt selection.

branch is trained independently of the policy backbone, allowing the same model to support both acting and valuing without duplicating the full representation pipeline.

Once value becomes an internal capability of the policy, it can also be reused before full-response generation. This matters because dynamically filtered reasoning RL often spends most of its time generating responses that are later discarded when all samples from a prompt are either correct or incorrect. Our measurements show that this sampling cost grows as training progresses. We therefore introduce ***Value-Guided Selection (VGS)***, which combines historical outcomes with value estimates from short prefixes to prioritize prompts that are more likely to produce informative mixed-response groups. The same self-value signal thus serves two purposes: improving policy updates after generation and guiding where generation effort is spent before full rollout.

Across five mathematical reasoning benchmarks, PPO+ consistently outperforms strong group-relative baselines. It improves average accuracy over DAPO by 8.09 points on Qwen3-4B in non-thinking mode, with gains that also transfer to thinking mode and Qwen3-30B-A3B with Routing Replay, while reducing average training time per round.

Our contributions are:

- **Stronger value-based policy optimization.** We combine token-dependent value advantages with within-prompt response comparisons in a single PPO objective, substantially improving PPO for long-context reasoning.
- **Self-valuing policy models.** We show that actor representations contain reusable value information and introduce *SVT*, which replaces a separate full-scale critic with a compact value-specific branch on the policy model.
- **Value-guided data collection.** We introduce *VGS*, which reuses early value estimates together with historical outcomes to prioritize prompts likely to yield informative mixed-response groups before full generation.

2 Pilot Study: Opportunities for Value Reuse

PPO pays for an explicit value function, but that value signal may also create opportunities unavailable to critic-free methods. Before introducing PPO+, we ask two questions: *does value estimation require a separate backbone, and can value help reduce the dominant cost of reasoning RL?*

2.1 Does Value Estimation Need a Separate Backbone?

A separate critic processes the same trajectories as the policy, raising a natural question: does value prediction really require a second representation pipeline? We test this using frozen Qwen3-4B actor and critic models. At several layers, we replace the critic representation with the actor representation from the same layer, either directly or after a learned transformation, and then run the remaining critic layers and value head. Full probe details are provided in Appendix A.1.

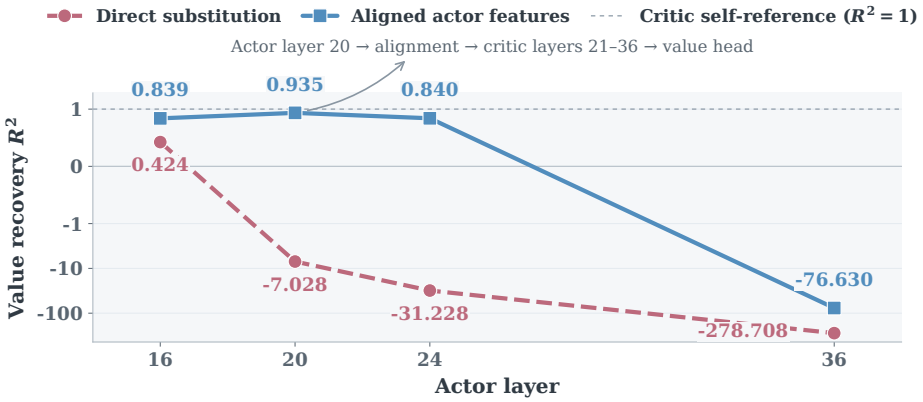

Figure 2: **Recovering critic values from actor representations.** Actor features are substituted directly or aligned before passing through the remaining frozen critic layers and value head.

Figure 2 separates representation similarity from functional reuse. At layer 20, direct substitution fails, while learned alignment recovers critic outputs with $R^2 = 0.935$, despite hidden-state reconstruction of only $R^2 = 0.096$. Recovery varies across layers, and the layer-20 probe still retains 16 critic layers after the attachment point. Thus, actor and critic representations are not directly interchangeable, but the actor already contains information useful for value prediction.

> **Takeaway.** Actor representations contain recoverable value information. This motivates reusing the policy backbone while learning only the value-specific computation on top.

2.2 Why Optimize Sampling?

A lightweight value model addresses only part of PPO's training cost. We therefore examine where the remaining time is spent and how much rollout work is ultimately discarded. We use the same observed training window across methods, with the complete measurement protocol in Appendix A.2.

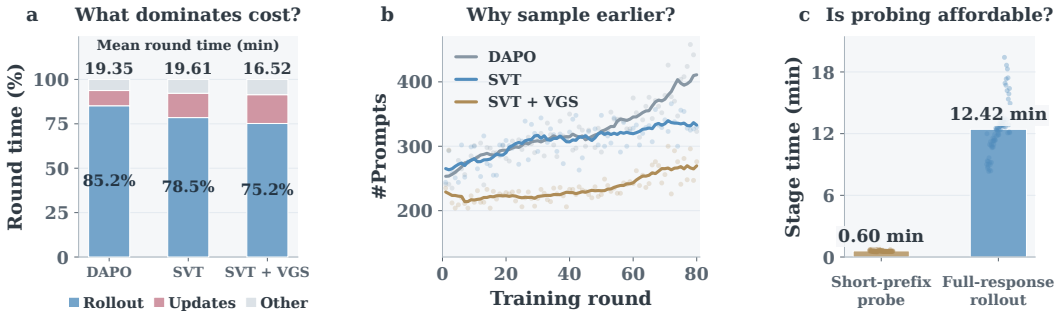

Figure 3: **Rollout dominates training cost.** Qwen3-4B thinking measurements over the first 80 common completed rounds: (a) round-time composition, (b) completed prompt groups required to retain 128 mixed groups, and (c) prefix-probing versus full-rollout time. Appendix A.2 specifies the runs, counting rules, and timing aggregation.

Figure 3a shows that rollout dominates round time, accounting for 85.2% of DAPO, 78.5% of SVT, and 75.2% of *SVT+VGS*. Because dynamic filtering identifies all-correct and all-incorrect groups only after full generation, discarded groups still incur the most expensive part of training.

The same inefficiency appears in sampling demand. Figure 3b shows that *SVT* generates 308.3 prompt groups per round on average to retain 128 mixed groups, compared with 234.9 for *SVT+VGS*. Meanwhile, short-prefix probing takes only 35.8 seconds on average, versus 12.42 minutes for full rollout in the same *VGS* rounds, as shown in Figure 3c.

> **Takeaway.** Full rollout remains the dominant cost, and much of this computation is spent on groups that are later discarded. Because short-prefix probing is far cheaper, value estimates can be used before full generation to focus sampling on more informative prompts.

Together, these studies suggest two forms of value reuse: within the model for value estimation, and before rollout for sampling. These ideas directly motivate *SVT* and *VGS* in PPO+.

3 PPO+

Our pilot studies identify two opportunities: reusing policy representations for value estimation and reducing the cost of full-response generation. PPO+ builds on these insights with three components: group-relative token advantages, a Self-Valuing Transformer, and Value-Guided Selection.

3.1 Group-Relative Advantage Estimation

PPO and GRPO-style methods use complementary signals for credit assignment. PPO uses learned values to distinguish states along a response, while group-relative methods compare responses to the same prompt. PPO+ combines both.

For prompt x_g, we sample K responses $y_{g,i}$ from π_{old} and define $s_{g,i,t} = (x_g, y_{g,i,<t})$. We first compute token-level advantages with GAE (Schulman et al., 2015):

$$A_{g,i,t} = \sum_{l=0}^{L_{g,i}-t} (\gamma\lambda_i)^l \delta_{g,i,t+l}, \qquad \delta_{g,i,t} = r^{\pi}_{g,i,t} + \gamma V(s_{g,i,t+1}) - V(s_{g,i,t}). \tag{1}$$

We then normalize these advantages within each prompt group. To avoid over-weighting long responses, each response contributes equally:

$$\mu_g = \frac{1}{K}\sum_i \frac{1}{L_{g,i}} \sum_t A_{g,i,t}, \quad \sigma_g^2 = \frac{1}{K}\sum_i \frac{1}{L_{g,i}} \sum_t A_{g,i,t}^2 - \mu_g^2, \quad \widehat{A}_{g,i,t} = \frac{A_{g,i,t} - \mu_g}{\sqrt{\max(\sigma_g^2, \epsilon)}}. \tag{2}$$

The resulting advantage remains token-dependent, but is now relative to other responses for the same prompt. We optimize $\widehat{A}_{g,i,t}$ with the standard PPO clipped objective (Schulman et al., 2017), using prompt-balanced aggregation. There is no additional GRPO loss.

3.2 Self-Valuing Transformer

Group-relative token advantages still require a learned value function. Instead of maintaining a separate full critic, SVT reuses representations already computed by the policy.

Let $h_\theta^{(T)}(s)$ denote the policy representation at attachment point T. SVT applies a small value-specific branch:

$$V_\phi(s) = v_\phi\Big(\text{sg}[h_\theta^{(T)}(s)]\Big), \tag{3}$$

where sg stops value gradients from entering the policy backbone.

The policy therefore provides the shared representation, while value supervision updates only the SVT branch. This retains dedicated computation for value prediction without duplicating a full language-model backbone. The branch is trained from terminal verifier outcomes and updated online as the policy changes. Its predictions are used to construct the token advantages above. Architecture and optimization choices are studied in Section 5.

3.3 VALUE-GUIDED SELECTION

SVT reduces the cost of value estimation, but full-response generation remains the dominant training expense. With dynamic filtering, all-correct and all-incorrect groups are discarded only after their responses have been fully generated.

VGS reuses the same value signal before full rollout. For prompt x, let $R_i \in \{0, 1\}$ denote the correctness of its i-th response among K samples. We call the group *mixed* if $0 < \sum_{i=1}^{K} R_i < K$, and train VGS to predict the probability of this event rather than correctness alone.

For each candidate prompt, we generate short prefixes and combine their SVT values with historical outcomes into a standardized feature vector $\widetilde{f}(x)$. A lightweight logistic predictor computes

$$q_\psi(x) = \text{sigmoid}\left(\psi^\top \widetilde{f}(x)\right), \tag{4}$$

where ψ denotes the selector parameters and $q_\psi(x)$ is the predicted mixed-group probability.

Prompts with larger $q_\psi(x)$ are prioritized for full generation, while a small fraction is sampled uniformly for exploration. Selected prefixes are continued rather than regenerated, and sampling proceeds until the target number of mixed groups is collected. The selector is updated only from outcomes observed in previous rounds.

4 EXPERIMENTS

We evaluate whether PPO+ improves both the effectiveness and efficiency of value-based policy optimization. Our experiments address three questions: *(1) Can PPO+ outperform strong group-relative baselines? (2) Where do the gains come from? (3) Do they persist throughout training?*

4.1 EXPERIMENTAL SETUP

Models and data. We evaluate Qwen3-4B in both non-thinking and thinking modes and Qwen3-30B-A3B in non-thinking mode. Training uses DAPO-Math-17k (Yu et al., 2025)[1]. Before training, we sample four responses for each prompt using the corresponding starting model and mode, and use their correctness outcomes to initialize the prompt curriculum.

Prompt curriculum. All methods use the same default prompt curriculum. We focus training on prompts that are partially solved by the current model. Initially, prompts with one to three correct responses out of four are prioritized, while consistently wrong or correct prompts are deprioritized. As training progresses, solved prompts are removed, harder prompts are introduced, and a 5% audit stream periodically revisits previously solved prompts to detect regressions. For each selected prompt, we generate four fresh responses and retain mixed groups until 128 are collected, yielding 512 training responses per round. All completed groups are used to update the curriculum. Appendix B.1 provides the full sampling and update rules.

Baselines and optimization. We compare PPO+ with DAPO (Yu et al., 2025) and conventional PPO (Schulman et al., 2017) with a separate critic. For Qwen3-30B-A3B, DAPO additionally uses Routing Replay (Ma et al., 2025). Our ablations start from PPO and cumulatively add group-relative advantages, replace the separate critic with SVT, and finally add VGS. The actor uses Adam with learning rate 10^{-6}, $(\beta_1, \beta_2) = (0.9, 0.98)$, and weight decay 0.1, with 80 warmup updates followed by a constant learning rate. Each round performs four actor updates on 128 responses, with clip bounds of 0.2 and 0.28 and a KL-loss coefficient of 10^{-3}. The separate critic and SVT each perform eight value updates per round on 64 responses, with learning rates of 5×10^{-6} and 2×10^{-5}, respectively. Before online training, both value models are warmed up for 1,024 updates on 32,768 trajectories from the frozen starting policy, and the lowest-validation-MSE checkpoint initializes training. Non-thinking and thinking runs train for 125 and 100 rounds, respectively, and the best heldout checkpoint is used for final evaluation. Additional settings are in Appendix B.2.

SVT and VGS. SVT uses a detached two-block value branch attached after Transformer layer 36 for Qwen3-4B and layer 48 for Qwen3-30B-A3B. The branch predicts binary verifier correctness,

[1] https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

Table 1: **Reasoning performance and training efficiency.** Scores are avg@k / pass@k (%). Avg. is the unweighted mean across five benchmarks. Rows cumulatively add **Group-Rel. Adv.** (group-relative advantages), **SVT** (Self-Valuing Transformer: a lightweight value branch on the actor), and **VGS** (Value-Guided Selection of training prompts) to PPO.

Method	Performance					Avg.↑	Efficiency	
	AIME 2026 @8	HMMT Nov 2025 @8	HMMT Feb 2026 @8	AIME 2025 @8	Beyond AIME @4		Prompts↓	Time↓ (min)
Qwen3-4B Non-Thinking								
Qwen3-4B	16.3 / 33.3	8.3 / 33.3	17.8 / 27.3	21.7 / 40.0	11.5 / 20.0	15.1 / 30.8	—	—
DAPO	52.5 / 73.3	39.6 / 66.7	33.0 / 42.4	52.5 / 73.3	32.3 / 48.0	42.0 / 60.8	303.4	12.7
PPO	45.8 / 66.7	32.9 / 50.0	27.3 / 42.4	45.8 / 66.7	26.0 / 42.0	35.6 / 53.6	307.4	19.9
+ Group-Rel. Adv.	62.5 / 83.3	42.9 / 66.7	34.9 / 45.5	64.6 / 73.3	**40.5** / 54.0	49.1 / 64.6	311.9	20.5
+ SVT	**64.2** / 80.0	46.3 / **70.0**	39.0 / 48.5	**65.4** / **86.7**	36.0 / **56.0**	**50.2** / **68.2**	307.4	14.3
+ VGS (PPO+, ours)	60.8 / **86.7**	**48.3** / 63.3	**39.8** / **57.6**	62.1 / 76.7	39.3 / 55.0	50.1 / 67.9	**274.3**	**12.2**
Qwen3-4B Thinking								
Qwen3-4B Thinking	63.3 / 76.7	50.8 / 66.7	38.3 / 51.5	63.3 / 83.3	38.5 / 49.0	50.9 / 65.4	—	—
DAPO	68.3 / 83.3	**59.2** / 76.7	39.8 / 54.6	65.0 / 80.0	38.0 / 53.0	54.1 / 69.5	322.0	19.1
PPO	55.8 / 73.3	50.4 / 73.3	36.4 / 54.6	60.4 / 76.7	37.0 / 53.0	48.0 / 66.2	302.9	18.3
+ Group-Rel. Adv.	63.8 / 80.0	58.3 / **80.0**	42.4 / **63.6**	67.5 / 86.7	43.8 / 57.0	55.2 / 73.5	306.7	18.8
+ SVT	**69.2** / **86.7**	57.1 / 76.7	**43.2** / **63.6**	67.1 / **90.0**	44.5 / 56.0	56.2 / **74.6**	311.9	19.7
+ VGS (PPO+, ours)	67.1 / **86.7**	57.5 / **80.0**	42.4 / 60.6	**72.1** / 83.3	**45.5** / **60.0**	**56.9** / 74.1	**234.6**	**17.2**
Qwen3-30B-A3B Non-Thinking								
Qwen3-30B-A3B	19.6 / 43.3	10.0 / 33.3	17.1 / 30.3	17.9 / 46.7	12.8 / 21.0	15.5 / 34.9	—	—
DAPO + RR	65.4 / 83.3	43.8 / 73.3	38.3 / 51.5	62.5 / 83.3	38.3 / 54.0	49.6 / 69.1	302.6	29.4
PPO	52.1 / 70.0	39.6 / 56.7	30.7 / 45.5	50.8 / 66.7	31.8 / 48.0	41.0 / 57.4	300.9	34.9
+ Group-Rel. Adv.	61.7 / 80.0	46.3 / 63.3	34.5 / 48.5	62.5 / 76.7	38.5 / 51.0	48.7 / 63.9	304.3	35.6
+ SVT	**67.5** / **86.7**	52.9 / **76.7**	39.8 / 60.6	**69.2** / **90.0**	**44.0** / **57.0**	**54.7** / **74.2**	308.9	29.9
+ VGS (PPO+, ours)	67.1 / **86.7**	**53.3** / 73.3	**40.2** / **63.6**	68.8 / **90.0**	43.3 / **57.0**	54.5 / 74.1	**276.8**	**27.3**

Reading the table. RR denotes Routing Replay for the MoE baseline. **Prompts**: mean fully generated and scored prompt groups per round (four responses per prompt), targeting 128 retained mixed-outcome groups. Prefix-only and unfinished groups are excluded. **Time**: mean logged time per round in minutes, including recorded evaluation, checkpointing, and waiting overhead.

while the policy reward also includes a soft overlength penalty. VGS scores four prefixes of up to 20 tokens per candidate using their value estimates, prompt-level value, and historical correctness. It reserves 25% of the preferred pool for uniform sampling. Non-thinking runs score 512 candidates and retain 256, while thinking runs score 1,024 and retain 384. Selected prefixes are continued rather than regenerated. Further details are given in Appendices B.3 and B.4.

Generation and benchmarks. Training rollouts use temperature 1.0, top-$p = 1.0$, and no top-k restriction, with a 2,048-token prompt limit and up to 38,912 response tokens within a 40,960-token context. We evaluate on AIME 2026, HMMT November 2025, HMMT February 2026, and AIME 2025 from MathArena (Dekoninck et al., 2026), together with BeyondAIME (ByteDance Seed, 2025). We sample eight responses per problem for the first four benchmarks and four for BeyondAIME. Formal evaluation follows the official recommended settings, using temperature 0.7 and top-$p = 0.8$ for non-thinking mode and temperature 0.6 and top-$p = 0.95$ for thinking mode, with top-$k = 20$ in both. We report avg@k as mean response accuracy and pass@k as the fraction of problems solved at least once. Avg. is the unweighted mean across the five benchmarks. Heldout monitoring is separate from formal evaluation, with remaining details given in Appendix B.5.

Efficiency. We report two measures of training efficiency: the average number of fully generated prompt groups per round and the average logged round time. The former reflects full-response sampling demand, while the latter measures end-to-end training cost. Prompt counts exclude prefix-only candidates and unfinished requests. Round time includes rollout, model updates, and recorded online overhead, but excludes initial value-model warmup. All metrics are averaged over the recorded training rounds. Appendix A.2 provides the detailed timing protocol.

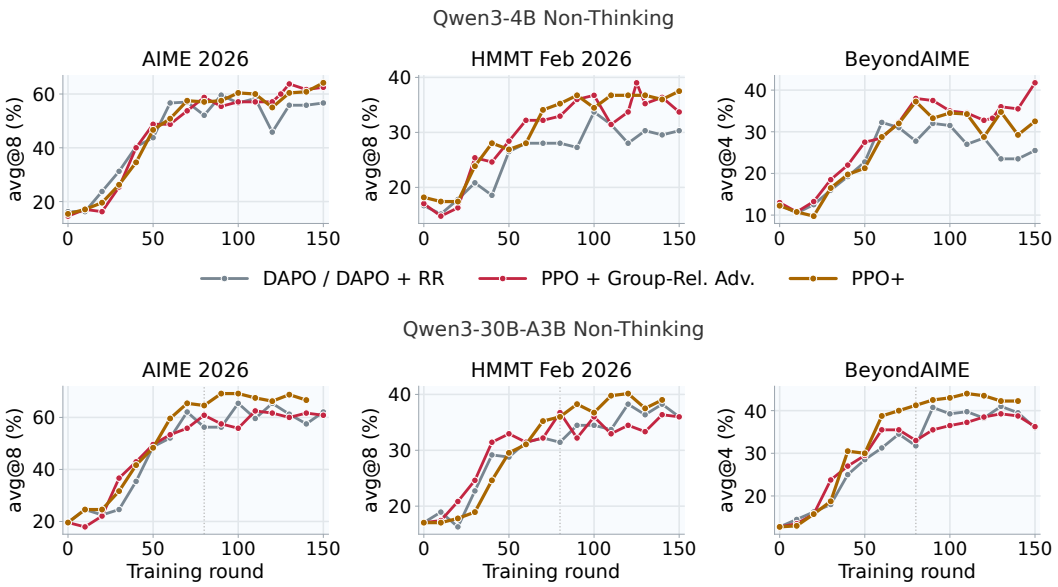

Figure 4: **Training dynamics on non-thinking Qwen3 models.** Comparison of DAPO, PPO with group-relative advantages, and PPO+ with SVT on AIME 2026, HMMT 2026, and BeyondAIME.

4.2 Results and Analysis

PPO+ improves both performance and efficiency. Table 1 shows a consistent pattern across all three model settings. Conventional PPO trails DAPO, whereas PPO+ reverses this gap in both average accuracy and pass@k. The improvement holds across non-thinking and thinking modes and extends to Qwen3-30B-A3B with Routing Replay, indicating that the gain is not tied to a single model scale or reasoning mode. PPO+ also requires fewer fully generated prompt groups and lower average round time across the reported settings. Importantly, the gains appear in both response-level accuracy and problem-level coverage, suggesting that PPO+ improves reasoning quality rather than a single metric. Taken together, these results show that explicit value estimation can remain a useful learning signal without retaining the performance gap and training overhead of conventional PPO.

Each component addresses a different bottleneck. *Group-relative advantages improve how value is used.* They provide the largest initial gain, suggesting that conventional PPO is limited less by the value function itself than by how its signal is converted into policy credit. Token-level values distinguish states along a response, while within-prompt comparisons provide relative information across candidate solutions. Combining the two preserves fine-grained credit while adding the group-relative structure that makes response comparison effective. *SVT makes value estimation more efficient.* Replacing the separate critic with SVT preserves and often improves reasoning performance, supporting the pilot-study finding that useful value information can be recovered from policy representations. This suggests that value prediction benefits from dedicated computation, but does not require a second full backbone. *VGS extends value from optimization to sampling.* It keeps accuracy close to the SVT-only configuration while reducing full-response sampling demand using the same internal value signal before rollout. Together, the three components improve complementary stages of training: credit assignment, value estimation, and rollout allocation.

The improvements persist throughout training. Figure 4 shows that the final gains are not confined to a single selected checkpoint. On Qwen3-4B, group-relative PPO quickly closes the initial gap to the baseline and remains competitive or stronger through later training, while SVT further strengthens performance as training progresses. The same trend is more pronounced on Qwen3-30B-A3B, where PPO+ maintains a clear advantage across much of the later training stage. These curves show that the gains emerge progressively during optimization rather than from an isolated checkpoint. Overall, better use of value signals removes the conventional PPO deficit, while self-valuing preserves and strengthens these gains without a full critic.

Table 2: **SVT design and value-guided selection.** (a) Frozen-actor comparison of value-branch attachment layers. `MSE` measures value error, `AUC` measures correctness discrimination, and `Pair Acc.` measures within-prompt ranking accuracy. (b) Frozen-policy comparison of selection signals on shared candidate responses. `Yield` is the mixed-group rate, `Tokens/mixed` is the generated tokens per retained mixed group, and `AUC` measures mixed-group prediction.

(a) SVT attachment

Layer	MSE↓	AUC↑	Pair Acc.↑
16	0.244	**0.602**	57.6
20	0.245	0.590	57.6
24	0.247	0.574	53.2
32	0.251	0.564	54.4
36	**0.242**	0.577	**67.1**

(b) Prompt selection

Selector	Yield↑	Tokens/mixed↓	AUC↑
Random	14.7	320.9k	0.584
History only	20.1	275.4k	0.661
Prompt value	20.1	282.1k	0.645
Prefix value	28.5	215.3k	0.716
History + prompt value	24.3	243.2k	0.662
History + prefix value	**31.5**	**198.1k**	**0.753**

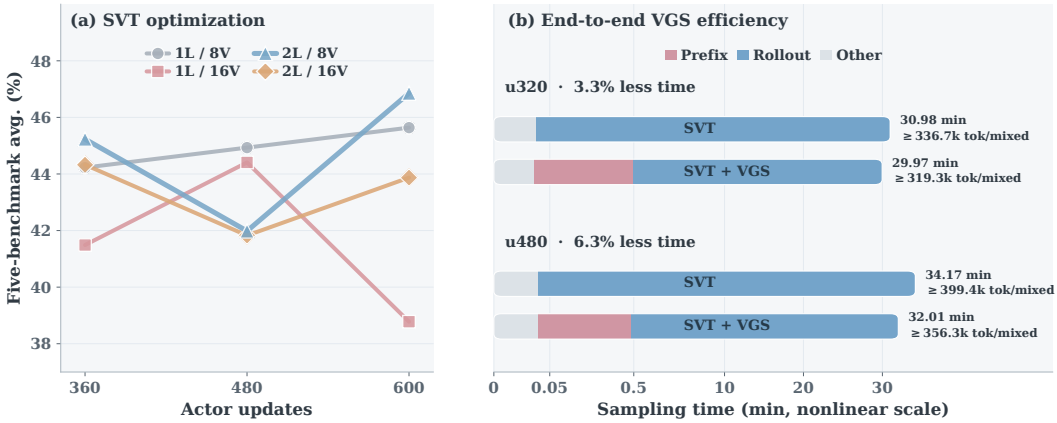

Figure 5: **SVT optimization and end-to-end VGS efficiency.** (a) Five-benchmark averages where available, with heldout calibration otherwise. L denotes branch layers and V updates per round. (b) Mean sampling time over six paired blocks per checkpoint. The nonlinear axis expands short durations. Rollout includes verification and scheduling; token counts exclude cancelled tokens.

5 Analysis

5.1 Self-Valuing Transformer Design

Late features are more suitable for SVT. Table 2(a) compares the same value branch across attachment layers. Earlier layers show slightly stronger global correctness discrimination, while the final layer gives lower value error and much better within-prompt ranking. This suggests that value quality for PPO is not determined by a single global metric. More important is whether the value signal can distinguish responses to the same prompt and remain aligned with the policy features used for generation. Late representations better satisfy this requirement while still supporting compact value-specific adaptation. Together with Section 2, these results support reusing late policy features rather than learning a separate backbone. We therefore attach SVT after layer 36 for Qwen3-4B.

More value optimization is not always better. Figure 5(a) varies branch depth and value-update frequency while keeping actor updates fixed. The plotted scale uses five-benchmark evaluations where available and heldout calibration otherwise, as detailed in Appendix C.2. Additional value updates can help at individual checkpoints, but the gains are not consistent over training, and deeper branches do not uniformly improve performance. Among the tested settings, the two-layer branch with eight updates gives the strongest late result without the extra cost of more frequent value updates. This suggests that SVT needs sufficient capacity to track the evolving policy, rather than aggressive value optimization. We therefore use two value layers and eight updates per round.

5.2 VALUE-GUIDED SELECTION

Value adds information beyond history. Table 2(b) compares selection signals under a frozen policy on shared candidate responses. Historical outcomes outperform random selection, showing that past correctness captures prompt difficulty. Prefix values add current policy information, improving mixed-group yield and reducing generation per retained group. Combining history with prefix value performs best across all metrics and reduces generation by about 28% relative to history alone. Thus, VGS benefits from current value estimates beyond historical difficulty.

Prefixes add useful trajectory information. Prompt-level value is informative, but short-prefix value is substantially more predictive of whether a group will be mixed. This suggests that early trajectory states reveal uncertainty not visible from the prompt alone, motivating prefix probing before expensive full rollouts. History and prefix value are also complementary: history summarizes past outcomes, while prefix value reflects the current policy on new trajectories.

5.3 END-TO-END SAMPLING EFFICIENCY

VGS reduces sampling time. Table 2(b) shows better selection efficiency, but token savings do not necessarily translate into lower wall-clock cost. We therefore compare VGS and uniform selection at two frozen Qwen3-4B non-thinking SVT checkpoints in Figure 5(b). Each checkpoint uses six order-balanced paired blocks with matched model weights, candidate pools, hardware, serving setup, verifier, and a target of 128 mixed groups. VGS reduces mean sampling time by 3.3% after 320 actor updates and 6.3% after 480 updates, averaging 4.9%.

The savings outweigh prefix overhead. Prefix generation and value scoring account for only 1.5% of VGS sampling time, while recorded generation drops from 368.0k to 337.8k tokens per retained mixed group. The timing includes cancellation and waits for serving engines to become idle, with overlapping generation, verification, and refill scheduling reported as one rollout segment. Token counts include completed responses and discarded prefixes but exclude unavailable tokens from unfinished requests, so they are lower bounds. The wall-clock comparison directly measures the sampling benefit of VGS at fixed policies. Appendix C.3 gives the full accounting protocol.

6 RELATED WORK

Policy optimization and value modeling for reasoning. PPO (Schulman et al., 2017) uses value-based advantage estimation (Schulman et al., 2015), while GRPO replaces the critic with within-prompt response comparisons (Shao et al., 2024). DAPO, VAPO, and SAO extend group- and value-based optimization through clipping, dynamic sampling, value pretraining, length-adaptive GAE, and asynchronous training (Yu et al., 2025, Yue et al., 2025, Hou et al., 2026). Building on policy–value sharing in actor–critic methods such as A3C (Mnih et al., 2016), PPO+ combines token-level value credit with within-prompt comparison, while SVT reuses detached policy features with a compact branch trained from terminal instead of step-level labels (Lightman et al., 2023).

Training-data selection and value-guided search. Prioritized experience replay focuses updates on informative transitions (Schaul et al., 2015), while value-guided search uses intermediate values to steer inference. Tree of Thoughts evaluates reasoning paths (Yao et al., 2023), and OVM learns partial-path values for guided decoding (Yu et al., 2023). VGS instead acts before training rollouts, combining history with early values to prioritize prompts likely to yield informative mixed groups. Its goal is higher yield after dynamic filtering, rather than immediate correctness.

7 CONCLUSION

We introduce PPO+, which combines group-relative token advantages, a lightweight self-valuing branch, and value-guided prompt selection. Across three model settings, PPO+ improves over DAPO in mean benchmark accuracy while reducing value-estimation cost. Our analysis shows that policy representations contain useful value information, and that short-prefix values can guide rollouts toward informative mixed groups. Together, these results support self-value as a practical way to improve both credit assignment and reinforcement learning efficiency.

AI USE STATEMENT

Generative AI tools were used to assist with literature discovery, language editing, drafting, plotting scripts, and checks of code, citations, and numerical consistency. All technical content, experimental design, results, and claims were reviewed and verified by the authors.

ETHICS STATEMENT

This work studies reinforcement learning for mathematical reasoning and training efficiency. Results on the evaluated benchmarks should not be interpreted as evidence of reliability in other domains or applications. Estimated quantities, calibrated results, and historical runtime comparisons are distinguished from direct measurements where applicable. The study does not involve human subjects or sensitive personal data.

REPRODUCIBILITY STATEMENT

Section 3 describes the PPO+ method and its three components. Appendix B provides data preparation, optimization settings, SVT architecture and initialization, VGS calibration and scheduling, and evaluation details. Appendices A and C document the pilot studies and supporting ablations. The accompanying configuration manifest records source hashes, dataset metadata, decoding protocols, and selector initialization to support reproduction of the reported experiments.

REFERENCES

ByteDance Seed. Seed1.5-Thinking: Advancing Superb Reasoning Models with Reinforcement Learning. *arXiv preprint arXiv:2504.13914*, 2025. URL https://arxiv.org/abs/2504.13914.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-Rollout Asynchronous Optimization for Agentic Reinforcement Learning. *arXiv preprint arXiv:2607.07508*, 2026. URL https://arxiv.org/abs/2607.07508.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, et al. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Wenhan Ma, Hailin Zhang, Liang Zhao, Yifan Song, Yudong Wang, Zhifang Sui, and Fuli Luo. Stabilizing MoE reinforcement learning by aligning training and inference routers. *arXiv preprint arXiv:2510.11370*, 2025. URL https://arxiv.org/abs/2510.11370.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, et al. Asynchronous Methods for Deep Reinforcement Learning. *arXiv preprint arXiv:1602.01783*, 2016. URL https://arxiv.org/abs/1602.01783.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*, 2015. URL https://arxiv.org/abs/1511.05952.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*, 2015. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, et al. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint arXiv:2305.10601*, 2023. URL https://arxiv.org/abs/2305.10601.

Fei Yu, Anningzhe Gao, and Benyou Wang. OVM, Outcome-supervised Value Models for Planning in Mathematical Reasoning. *arXiv preprint arXiv:2311.09724*, 2023. URL https://arxiv.org/abs/2311.09724.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, et al. VAPO: Efficient and Reliable Reinforcement Learning for Advanced Reasoning Tasks. *arXiv preprint arXiv:2504.05118*, 2025. URL https://arxiv.org/abs/2504.05118.

A Pilot Study Protocols

This appendix details the protocols used in the two pilot studies from Section 2. The first tests whether actor representations can be reused for value prediction. The second analyzes rollout demand and training-time composition. The two studies use different checkpoints and experimental settings and should therefore be interpreted separately.

A.1 Recovering Critic Values from Actor Features

Setup. We use frozen Qwen3-4B PPO actor and critic models from the same training checkpoint. The study contains 202 prompts with four responses per prompt, yielding 808 trajectories. Prompts are split approximately 70/15/15 into 141 training, 30 validation, and 31 test prompts, with all responses from the same prompt kept in the same split. For each response, we collect hidden states at 10%, 30%, 50%, 70%, and 90% of its length, together with the final-token state. This gives $31 \times 4 \times 6 = 744$ test states. Actor and critic parameters remain frozen throughout the study.

Feature substitution and alignment. We probe representations after layers 16, 20, 24, and 36. For direct substitution, the actor representation at a given layer replaces the critic representation at the same depth and is then passed through the remaining critic layers and value head. For learned alignment, the actor representation is first transformed with a rank-256 residual adapter:

$$f(h) = h + \Delta + W_{\text{up}}\,\text{SiLU}\left(W_{\text{down}}\frac{h-\mu}{s}\right), \tag{5}$$

where μ is the training-set mean of the actor features, s is their scalar root-mean-square deviation, and Δ is the mean critic–actor feature difference. The down projection has width 256 and no bias, and the up projection is initialized to zero.

Each layer is fitted independently to minimize hidden-state mean squared error. We use AdamW with learning rate 3×10^{-3}, weight decay 10^{-4}, batch size 256, 600 optimization steps, and gradient-norm clipping at 1.0. Validation MSE is evaluated every 50 steps, and the checkpoint with the lowest validation MSE is used for test evaluation. The learned mapping is applied to the full token sequence before the remaining critic computation, preserving causal attention context.

Metrics and interpretation. For original critic outputs v_n and recovered outputs $\hat{v}_n$, we report

$$R^2 = 1 - \frac{\sum_n(\hat{v}_n - v_n)^2}{\sum_n(v_n - \bar{v})^2}. \tag{6}$$

The target is the original critic output on the same state, not verifier correctness. Hidden-state reconstruction uses the corresponding MSE-based R^2 over flattened hidden features.

At layer 20, learned alignment recovers critic outputs with $R^2 = 0.935$, while hidden-state reconstruction reaches only $R^2 = 0.096$. The probe at this depth still retains 16 critic layers after the attachment point. The study therefore shows that actor features contain functionally useful value information without implying that actor and critic representations are directly interchangeable or that a shallow value branch is sufficient.

A.2 Rollout Demand and Timing

Setup. We compare three Qwen3-4B Thinking runs: DAPO, PPO with a two-block detached SVT value branch attached after layer 36, and the same SVT configuration with VGS. All runs use 32 GPUs. The analysis covers the first 80 completed rounds shared by all three runs.

Each round samples four responses per prompt and retains 128 mixed-outcome groups, yielding 512 training responses. A group is mixed if it contains at least one correct and one incorrect response. For VGS, four prefixes of up to 20 tokens are generated for each candidate prompt, and selected prefixes are continued into full responses. These runs are separate historical training trajectories rather than randomized forks from a common checkpoint, so timing comparisons are descriptive rather than causal.

Round-time decomposition. We decompose each round into rollout, training updates, and other overhead. For stage s, its fraction of total round time is

$$100 \frac{\sum_r t_{r,s}}{\sum_r t_{r,\text{round}}}. \tag{7}$$

Update time includes actor and value updates where applicable. Other time is the residual after subtracting rollout and update time from total round duration. Recorded startup, waiting, and synchronization costs are retained. Initial value-model warmup is excluded from online timing.

Across the common window, rollout accounts for 85.2% of DAPO round time, 78.5% of SVT round time, and 75.2% of SVT+VGS round time.

Completed prompt groups. We count all fully generated and scored prompt groups, including groups later discarded by dynamic filtering. Prefix-only candidates and unfinished requests are excluded, so this metric measures completed prompt groups rather than total generated tokens or total GPU computation.

The mean number of completed prompt groups per round is 321.3 for DAPO, 308.3 for SVT, and 234.9 for SVT+VGS. All three methods target the same 128 retained mixed groups per round. For visualization, individual rounds are shown together with a centered ten-round moving average, using four preceding and five subsequent rounds and truncating the window at the endpoints.

Prefix-probing cost. For SVT+VGS, prefix probing takes 35.8 seconds per round on average, while the logged full-rollout stage takes 745.2 seconds, or 12.42 minutes. The rollout measurement refers to the complete rollout stage of a training round rather than generation of a single response. Prefix-probing time is already included in total round time and is not added again. Its timer does not capture every scoring, calibration, scheduling, or cancellation cost. We therefore use this comparison to characterize the scale of prefix probing, rather than to isolate the net runtime benefit of VGS.

B Training and Evaluation Configuration

This appendix documents the data preparation, online optimization, and runtime settings used in our main experiments. Unless otherwise noted, these settings correspond to the reference PPO+ recipe described in Section 4.1.

B.1 Training Data, Filtering, and Verification

Data preparation. The processed DAPO-Math-17k pool contains 17,182 distinct prompts. For Qwen3-4B, we use 15,464 prompts for training and 1,718 prompt-disjoint prompts for auditing. For Qwen3-30B-A3B, the same 15,464 prompts are used for training, with 859 prompts each for validation and test. Historical correctness labels are generated separately for each model and reasoning mode. These splits are used only for training, curriculum construction, and value diagnostics, and are distinct from the benchmarks used for formal evaluation.

Historical labels and online filtering. Before training, each prompt is sampled four times with the corresponding starting model and mode. For Qwen3-4B non-thinking, generation uses temperature 0.7, top-$p = 0.8$, top-$k = 20$, and a 32,768-token response limit. Thinking-mode labels use temperature 0.6, top-$p = 0.95$, top-$k = 20$, and the same response limit. All responses are regraded with the same verifier used during online training.

The training pool retains prompts from all five correctness buckets, from zero to four correct responses. During online training, correctness is recomputed from four fresh responses sampled at temperature 1.0. Dynamic filtering retains a group only if it contains both correct and incorrect responses. Sampling continues until 128 mixed groups are collected.

The same dynamic curriculum is used by all methods unless stated otherwise. Prompts that are partially solved are prioritized for training, consistently wrong prompts are temporarily deprioritized, and consistently correct prompts are periodically revisited. The hard-prompt cooldown is two rounds, the mastered-prompt audit cooldown is five rounds, and 5% of the sampling queue is reserved for auditing previously solved prompts. All completed groups, including those rejected by

dynamic filtering, update the curriculum state. VGS changes only the ranking within its candidate pool and does not use evaluation-benchmark outcomes.

Verifier and reward. We use a robust final-answer extractor that recognizes boxed answers, explicit English or Chinese final-answer declarations, and standalone mathematical answers on the final line. When multiple explicit answers are present, later declarations take precedence. The extractor does not search for the reference answer. Extracted answers are checked with Math-Verify, and missing or invalid answers receive correctness zero. Verifier failures raise an execution error rather than being silently marked incorrect. Verification uses four workers with a 60-second per-item timeout and a 300-second queue timeout.

The policy reward combines binary correctness with a soft overlength penalty. For the 38,912-token response limit, the penalty is zero through 34,816 tokens and then decreases linearly to -1 over the final 4,096 tokens. Dynamic filtering and SVT targets use unshaped binary correctness rather than the shaped policy reward. Historical runs with different length-penalty settings retain their original configuration when reported.

B.2 Online Optimization and Runtime

Table 3 summarizes the reference online training configuration. Method-specific initialization and deviations from this recipe are described below.

Table 3: **Reference online training configuration.**

Setting	Value
Retained batch	128 prompt groups $\times$ 4 responses
Actor batch / updates per round	128 responses / 4
Value batch / updates per round	64 responses / 8
Actor / SVT learning rate	10^{-6} / 2×10^{-5}
Actor LR schedule	80-update warmup, then constant
Optimizer / weight decay	Adam, $\beta_1 = 0.9$, $\beta_2 = 0.98$ / 0.1
Policy clip, lower / upper	0.20 / 0.28
KL-loss type / coefficient	Low-variance KL / 0.001
Reward KL / entropy coefficient	0 / 0
Discount / adaptive-GAE scale	$\gamma = 1$ / $\alpha = 1.5$
GAE λ	Length-adaptive, clipped to $[0, 1]$
Within-group / between-group weight	1 / 0
Normalization epsilon	10^{-6}
Global advantage whitening	Disabled
Value clip / loss coefficient	0.2 / 1.0
Actor / value gradient-norm limit	1.0 / 5.0
Attention / hidden dropout	0 / 0
Online sampling T, top-p, top-k	1.0, 1.0, unrestricted
Prompt / response / context limit	2,048 / 38,912 / 40,960 tokens
Checkpoint / heldout interval	Every 10 rounds

Policy and value updates. Advantages are computed from the behavior-policy rollout and the corresponding old value predictions, then held fixed within each round. Group-relative PPO uses the normalization and prompt-balanced objective from Section 3.1 and does not add a separate outcome-level policy loss.

The separate PPO critic uses learning rate 5×10^{-6}, batch size 64, and eight value updates per round. Its attention projection parameters are frozen. SVT also performs eight value updates per round with batch size 64, using learning rate 2×10^{-5}. Both value models are warm-started before online training using trajectories from the frozen starting policy. Because the separate critic and SVT differ in architecture and trainable parameters, cumulative ablation rows should be interpreted as complete training recipes rather than single-variable interventions.

DAPO uses group-relative outcome advantages together with dynamic filtering. For Qwen3-30B-A3B, Routing Replay is enabled. Historical thinking-mode continuations retain their recorded dual-clipping and length-penalty settings.

Execution environment. Reference online runs use four nodes with eight GPUs each. Training is implemented with Megatron using full-layer recomputation and FP32 gradient accumulation, while SGLang serves rollout generation. The dense configuration uses context parallelism 2. For Qwen3-30B-A3B, the reference configuration uses tensor parallelism 4, context parallelism 2, expert parallelism 8, and expert tensor parallelism 1.

Actor and value parameters use separate optimizer states. The reference SVT implementation uses FusedAdam without CPU optimizer offload. Serving precision and parallelism differ across some historical runs, and these differences are accounted for when interpreting wall-clock measurements. Configured training horizons are treated separately from completed rounds and selected checkpoints.

B.3 SVT Architecture and Initialization

Architecture. For Qwen3-4B, SVT attaches a private two-block value branch after the final policy representation. The branch is initialized from the actor's final two Transformer layers and adds its own normalization and scalar value head. Actor features are detached before entering the branch, so value gradients do not update the policy backbone. The Qwen3-30B-A3B configuration uses the same detached two-block design, attached after layer 48. Its value branch is initialized independently rather than transferred from the 4B model.

Value warmup. Before online training, we warm up the Qwen3-4B value branch with the actor frozen. The warmup set contains 8,192 prompts with four responses each, yielding 32,768 trajectories across all historical correctness levels. Training runs for 1,024 updates with a global batch size of 64, corresponding to two effective passes over the warmup data. The Transformer branch uses learning rate 2×10^{-5} and the scalar head uses 5×10^{-5}, with Adam $(\beta_1, \beta_2) = (0.9, 0.98)$, weight decay 0.1, and gradient clipping at 1.0. Response-token states are trained toward the terminal binary verifier outcome with mean squared error. The checkpoint with the lowest validation MSE initializes online training.

Online value optimization. During online training, SVT uses learning rate 2×10^{-5}, value clipping 0.2, and binary Monte Carlo targets. Each round performs eight value updates on 64 responses and four actor updates on 128 responses. Actor and value gradients are clipped separately at 1.0 and 5.0, and their optimizer states are maintained independently. Advantages and old policy probabilities are computed once per round and are not recomputed after intermediate value updates.

Ablation settings. We separately study branch depth, attachment layer, and the number of value updates. These experiments hold actor-update checkpoints fixed so that additional value updates change value-optimization cost rather than policy-update count. The frozen-actor attachment study follows the separate protocol described in Appendix C.1.

B.4 VGS Features and Online Selection

Selection features. VGS predicts the probability that a prompt will produce a mixed-outcome group, rather than predicting correctness alone. Let $p = h/4$, where $h \in \{0, \ldots, 4\}$ is the most recently observed number of correct responses for the prompt. When no history is available, we set $p = 0$ and include a missing-history indicator. Let v_0 denote the prompt-level value and $v_1, \ldots, v_4$ the four short-prefix values. The selector uses

$$f = \left[p,\, p^2,\, \mathbb{1}_{\text{known}},\, v_0,\, \bar{v},\, \sigma_v,\, \min_i v_i,\, \max_i v_i,\, \bar{v} - v_0 \right], \tag{8}$$

where $\bar{v}$ and σ_v are the mean and population standard deviation of the four prefix values. Features are standardized before being passed to a logistic predictor.

Ranking and continuation. VGS reserves 25% of the preferred pool for uniform sampling and fills the remaining positions by predicted mixed-group probability. Selected prefixes are continued

Table 4: **VGS sampling configuration.** Generation continues until 128 mixed-outcome groups are retained.

Setting	Non-thinking	Qwen3-4B thinking
Candidates / preferred pool	512 / 256	1,024 / 384
Prefixes per prompt / length	4 / up to 20 tokens	4 / up to 20 tokens
Uniform fraction	25%	25%
Initial full-generation batch	128 groups	256 groups
Refill batch	128 groups	128 groups
Target mixed groups	128	128
Calibration startup pool	128	256
Prefix continuation	Reuse sampled prefixes	Reuse sampled prefixes

directly into full responses rather than being regenerated, and their prefix tokens remain part of the training trajectories. Generation proceeds asynchronously until 128 mixed groups are retained. Pending requests are then cancelled. If the preferred pool is insufficient, additional distinct prompts are sampled in batches of 128.

For the non-thinking recipe, the first training round is used for calibration and ranked selection begins from the second round. The thinking configuration can additionally gate VGS according to recent response length so that prefix probing is enabled only when its overhead is likely to be amortized.

Online calibration. The selector is updated online from newly completed groups. A group receives label one exactly when its four responses contain both correct and incorrect outcomes. The logistic model is updated sequentially using a Gaussian approximation to the posterior while keeping feature normalization fixed. Only newly completed groups with available selector features contribute to the update, and labels observed in a round affect only later rounds. A held-out subset of prompts is excluded from fitting and used for online diagnostics. VGS is disabled during formal benchmark evaluation.

Selector variants. For analysis, we compare random selection, historical correctness alone, prompt-level value alone, and the full combination of history and prefix-value features. The fixed-response replay protocol used for these comparisons is described in Appendix C.4.

B.5 Formal Evaluation Protocol

Table 5: **Formal evaluation configuration.** Training rollouts use a separate temperature-1.0 sampling configuration.

Setting	Non-thinking	Thinking
Temperature / top-p	0.7 / 0.8	0.6 / 0.95
Top-k / min-p	20 / 0	20 / 0
Response / context limit	38,912 / 40,960	38,912 / 40,960
Context safety margin	64 tokens	64 tokens
AIME/HMMT samples	8, seeds 42–49	8, seeds 42–49
BeyondAIME samples	4, seeds 42–45	4, seeds 42–45
Prompt template	Qwen, thinking disabled	Qwen, thinking enabled
VGS / value reranking	Disabled	Disabled

Benchmarks and generation. The four AIME/HMMT benchmarks contain 30, 30, 33, and 30 problems, respectively, and BeyondAIME contains 100. Formal evaluation follows the benchmark definitions used in the main experiments. Prompts request a final answer in boxed form, with AIME additionally requiring an integer from 0 to 999. For each problem and repetition j, we use sampling

seed $42 + j$. The effective response budget is

$$\min(38912,\ 40960 - n_{\text{chat}} - 64)\,, \tag{9}$$

where n_{chat} is the number of tokens after applying the chat template.

Answer extraction and scoring. For AIME and HMMT, we use MathArena-compatible final-answer parsing. BeyondAIME uses robust integer-answer scoring. If the initial response cannot be parsed in the required format, we permit one formatting-only follow-up that requests a boxed final answer from the existing reasoning without asking the model to reason again. This retry uses the same sampling seed and does not count as an additional sample. Infrastructure failures are likewise not treated as additional responses.

For binary correctness c_{ij} over N problems and k sampled responses, we compute

$$\text{avg@}k = \frac{1}{Nk}\sum_{i=1}^{N}\sum_{j=1}^{k} c_{ij}, \qquad \text{pass@}k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\sum_{j=1}^{k} c_{ij} > 0\right]. \tag{10}$$

Scores are reported as percentages. The five-benchmark average assigns equal weight to each benchmark. Training-time heldout monitoring uses a separate 128-question set with avg@4 and is not part of the formal five-benchmark evaluation.

C Supporting Ablations

This section provides additional analyses of the Self-Valuing Transformer and Value-Guided Selection. These studies use protocols separate from the pilot experiments and focus on design choices rather than independent main results.

C.1 Self-Valuing Transformer

Attachment layer. We study where the value branch should attach to the policy backbone using a frozen actor. The dataset contains 80 training, 24 validation, and 24 test prompts, with four responses per prompt. Because this study informed the architecture used in later experiments, we treat it as development evidence rather than independent confirmation.

Branch depth and value updates. We compare one- and two-block value branches under different value-update budgets. All configurations are evaluated on the same 128-question heldout set with four responses per question after 360, 480, and 600 actor updates. Actor updates are matched across runs, while configurations with more value updates incur additional optimization cost. These comparisons use single training runs.

C.2 Calibration for the SVT Optimization Plot

Figure 5(a) reports the unweighted average over five benchmarks, using eight responses per problem on AIME/HMMT and four on BeyondAIME. Complete evaluations are available for the one-block branch with eight value updates after 360 actor updates, the one-block branch with sixteen value updates after 480 actor updates, and the two-block branch with sixteen value updates after 360 actor updates. Their five-benchmark averages are 44.23, 44.41, and 44.33, respectively, and these points use measured scores directly. For the two-block branch with eight value updates, complete fixed-seed evaluations after 320, 400, 440, and 520 actor updates serve as calibration anchors.

For each configuration, we compute the mean ratio between the five-benchmark average and held-out avg@4 at matching anchor checkpoints. Missing benchmark averages after 360, 480, or 600 actor updates are estimated by multiplying the corresponding heldout score by this configuration-specific ratio. This assumes that the ratio remains approximately stable across nearby checkpoints and does not replace a complete benchmark evaluation. Nine of the twelve plotted points are calibrated estimates. The same non-monotonic optimization pattern is visible in the original heldout scores without calibration.

C.3 Paired Sampling-Time Measurements

We measure sampling efficiency at two frozen Qwen3-4B non-thinking SVT checkpoints, after 320 and 480 actor updates, with no optimizer updates during measurement. Each checkpoint is evaluated in six blocks using seeds 20261001–20261006. The three selection arms are uniform selection, history-only selection, and history combined with 20-token prefix values. Their execution order covers all six permutations. Figure 5(b) reports the uniform and history-plus-prefix conditions, averaged over all six blocks at each checkpoint. These are repeated sampling trials rather than independent training seeds.

Matched protocol. Within each block, actor and value weights are frozen and all conditions begin from the same candidate set. Historical labels and selector priors are reset before each condition. Each candidate pool contains 512 prompts, with a preferred pool of 256. Complete groups are issued in batches of 128 until 128 mixed groups are retained. Refill demand therefore depends on the outcomes observed by each selection strategy. All runs use the same 32 H200 GPUs, FP8 actor and KV-cache serving, and tensor parallel size one. Prefix continuation, decoding, and verification follow Appendix B.4.

Timing protocol. Each condition begins with idle inference engines. A monotonic timer starts before sampling and ends only after all requests are completed or cancelled and the engines return to an idle state. Prefix time includes short-prefix generation and value scoring. The remaining rollout interval combines full generation, concurrent verification, filtering, and refill scheduling. Other time contains the remaining cleanup and postprocessing overhead. These components sum to total wall-clock time without double-counting overlapping request latency. Verification is not timed independently and is therefore included in the rollout interval.

Token accounting. Token counts include complete responses and discarded probing prefixes, divided by the 128 retained mixed groups. Retained prefixes are already included in complete-response lengths and are not counted twice. Tokens generated by unfinished requests are unavailable, so the reported counts are lower bounds.

C.4 Value-Guided Selection

Frozen-policy replay. We compare prompt-selection signals under a fixed policy using the same sampled responses for all selectors. The study contains 2,048 candidate prompts divided into four blocks of 512, with 256 prompts selected from each block. Ranked selectors retain a 25% uniform component for exploration. We compare random selection, historical outcomes, prompt-level values, prefix values, and combinations of history with prompt-level or prefix values. Prompts used for earlier calibration are excluded.

Metrics. Mixed-group yield is the fraction of selected prompts whose sampled responses contain both correct and incorrect outcomes. Tokens/mixed is the generated token count per retained mixed group. Token accounting includes completed responses and discarded probing prefixes, but excludes value-computation and scheduling overhead. AUC measures whether the selector predicts that a candidate prompt will produce a mixed group. Confidence intervals for gains over history are computed with paired stratified bootstrap resampling. When historical correctness features are hidden, the experiment measures the effect of missing history rather than out-of-distribution generalization.

Historical timing analyses. Some auxiliary runtime comparisons use historical runs rather than randomized forks from a shared checkpoint. Checkpoint state, response length, serving behavior, and optimization settings may therefore contribute to observed differences. We treat these measurements as descriptive rather than causal. Initial value-branch warmup is excluded from online timing, while incomplete prompt groups are excluded from completed-prompt counts even though they may consume computation. Prefix-generation time is reported separately where relevant and is not added again to total round time.