BEYOND POLICY DISCREPANCY: CAPABILITY-AWARE MULTI-TEACHER ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-teacher on-policy distillation (MOPD) integrates domain specialists into a shared student, but determining how strongly each teacher should supervise remains challenging. Existing discrepancy-based allocation can assign strong supervision even when the student already matches or exceeds the teacher's task performance, since behavioral mismatch does not necessarily imply a remaining capability deficit. We propose Capability-Aware Multi-Teacher On-Policy Distillation (CA-MOPD), which combines specialist merging with capability-guided allocation to account for what the student has already learned. CA-MOPD first merges reinforcement-learning specialists into a strong student initialization, then uses verifier-estimated teacher–student capability gaps to adjust domain-level distillation strength. Supervision decreases as capability gaps close and increases when estimated student capability regresses, while dense token-level guidance remains provided by the teachers. Across six benchmarks spanning mathematics, code generation, and instruction following, CA-MOPD achieves an average score of 32.22, outperforming naive MOPD, Open-MOPD, and the merged initialization by 0.86, 0.83, and 0.74 points, respectively. Controlled comparisons further support the benefits of merged initialization and capability-gap guidance under matched training budgets.

1 INTRODUCTION

Reinforcement-learning (RL) post-training has become an effective way to develop language models with strong domain-specific capabilities, including mathematical reasoning, code generation, and instruction following (Shao et al., 2024; Pyatkin et al., 2026; Gao et al., 2026). These capabilities are often learned by independently trained specialists, making it costly to deploy and maintain one model per domain. Multi-teacher on-policy distillation (MOPD) offers a natural alternative by using domain-specific teachers to supervise trajectories generated by a shared student, thereby integrating multiple specialist capabilities into a single model (Agarwal et al., 2024; Ma et al., 2026).

Parameter merging offers another approach to integrating specialist capabilities into a single model by combining their parameters (Ilharco et al., 2022; Yadav et al., 2023). Parameter merging and MOPD are naturally complementary: merging can integrate specialist capabilities before distillation, potentially providing a stronger initialization for the student. However, the extent to which this initialization improves subsequent MOPD remains insufficiently understood.

A key challenge in MOPD is how to allocate optimization across teachers. Open-MOPD addresses this with token-share balancing, discrepancy-based allocation, and reward refresh (Gao et al., 2026), assigning larger weights to domains with greater teacher–student policy discrepancy. However, policy discrepancy measures behavioral mismatch rather than whether the teacher still has a task-level capability advantage. A student can remain distributionally different from a teacher while already matching or exceeding its task performance, especially after specialist merging.

Motivated by this distinction, we introduce **Capability-Aware Multi-Teacher On-Policy Distillation (CA-MOPD)**, which first merges RL specialists with Task Arithmetic (Ilharco et al., 2022) to provide a stronger student initialization for subsequent on-policy distillation and then allocates domain-level distillation strength according to verifier-estimated teacher–student capability gaps. Supervision decreases as the estimated gap closes, while a regression-aware retention mechanism increases

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.

it when previously acquired capability deteriorates. We retain token-share balancing to correct response-length imbalance.

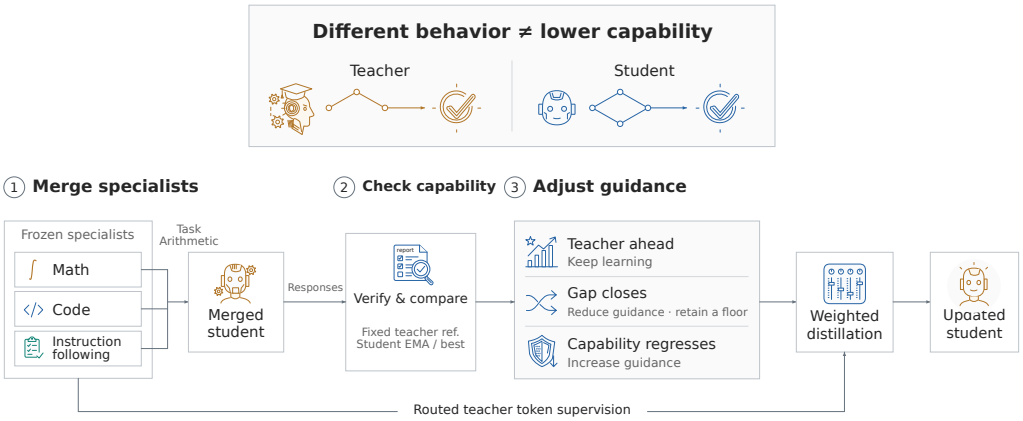

Figure 1: Overview of CA-MOPD. (1) Domain RL specialists sharing a MixSFT base are merged into a strong student initialization. (2) Student responses are verified to track capability relative to a fixed, calibrated teacher reference and the student's historical best. (3) Remaining capability headroom and regression feedback control domain weights, which combine with token balancing and global scaling to modulate dense teacher supervision.

Across six benchmarks spanning mathematics, code generation, and instruction following, CA-MOPD achieves an average score of 32.22, compared with 31.36 for naive MOPD, 31.39 for Open-MOPD, and 31.48 for the Task-Arithmetic initialization. The gains are concentrated in mathematics and instruction following, while code performance decreases relative to the merged initialization, reflecting a different cross-domain capability trade-off rather than uniform improvement.

Our main contributions are as follows:

- We show that initializing MOPD with merged RL specialists provides a stronger starting point than the shared SFT checkpoint and improves final aggregate performance under a fixed distillation budget.
- We distinguish teacher–student policy discrepancy from signed capability gap, showing that behavioral mismatch does not necessarily indicate remaining task capability to transfer.
- We propose **Capability-Aware Multi-Teacher On-Policy Distillation (CA-MOPD)**, which combines specialist merging, verifier-based capability estimation, dynamic domain-level allocation, and regression-aware retention.
- We show across three capability domains that CA-MOPD improves aggregate performance over naive MOPD, Open-MOPD, and the merged initialization, and analyze the resulting allocation dynamics.

2 Background and Motivation

2.1 Multi-Teacher On-Policy Distillation

Let $\mathcal{D}$ denote the domain set, π_{ϕ_d} the frozen specialist for domain d, and π_θ the shared student. Given a prompt $x \sim \mathcal{P}_d$, the student samples

$$y \sim \pi_{\theta_{\text{old}}}(\cdot \mid x), \tag{1}$$

and the domain teacher π_{ϕ_d} provides dense token-level supervision on this trajectory (Ma et al., 2026; Gao et al., 2026).

The OPD signal A_{itv} corresponds to candidate token v at response position t of example i. The exact student Top-K construction, candidate normalization, and clipped PPO objective are provided in Appendix A.2.

2.2 Discrepancy-Based Distillation Allocation

Domains can differ in response length and convergence rate even under balanced prompt sampling. Open-MOPD (Gao et al., 2026) addresses these differences through token-share balancing and discrepancy-based allocation.

Let $\mathcal{S}_{it}$ denote the candidate-token support used to compute A_{itv}, M_{it} the valid-response mask, and $N_d = \sum_{i:d(i)=d} \sum_t M_{it}$ the number of valid response tokens from domain d. For the mean statistic used in our implementation, the domain discrepancy is

$$m_d = \frac{1}{N_d} \sum_{i:d(i)=d} \sum_t M_{it} \left| \sum_{v \in \mathcal{S}_{it}} A_{itv} \right|. \tag{2}$$

Larger m_d indicates a larger token-level mismatch and therefore receives stronger supervision under discrepancy-following allocation.

2.3 Policy Discrepancy Is Not Capability Deficit

Let $V_d(x, y) \in [0, 1]$ be the domain verifier score for response y to prompt x. We define policy capability as

$$J_d(\pi) = \mathbb{E}_{x \sim \mathcal{P}_d,\, y \sim \pi(\cdot|x)} \left[V_d(x, y) \right], \tag{3}$$

and the signed teacher–student capability gap as

$$g_d(\theta) = J_d(\pi_{\phi_d}) - J_d(\pi_\theta). \tag{4}$$

Thus, $g_d > 0$ indicates a teacher capability advantage, while $g_d \leq 0$ indicates that the student matches or exceeds the teacher under the same domain distribution and verifier.

Policy discrepancy m_d measures behavioral mismatch without establishing a teacher capability advantage:

$$m_d > 0 \quad \not\Rightarrow \quad g_d(\theta) > 0. \tag{5}$$

This distinction is especially relevant after specialist merging, which can transfer task capabilities while leaving policy discrepancy nonzero.

3 Capability-Aware Multi-Teacher On-Policy Distillation

Figure 1 summarizes the CA-MOPD workflow. CA-MOPD implements the preceding principle in four steps. It first merges independently trained RL specialists into a strong multi-domain initialization, calibrates each teacher's task-level capability, tracks the student's corresponding capability from verifier-scored on-policy rollouts, and converts the remaining capability gap into a domain-level distillation factor. This factor is then combined with token-share balancing and applied to the common dense OPD objective.

3.1 Merged Initialization

We assume that all domain specialists are obtained by applying domain-specific RL to the same mixed-domain SFT checkpoint with parameters θ_0. Let ϕ_d denote the parameters of specialist d. Following Task Arithmetic (Ilharco et al., 2022), we initialize the student as

$$\theta_{\text{merge}} = \theta_0 + \sum_{d \in \mathcal{D}} \alpha_d (\phi_d - \theta_0), \qquad \alpha_d = 1. \tag{6}$$

The merge transfers part of the specialist updates before on-policy distillation and therefore changes the residual capability profile that distillation must address. CA-MOPD treats θ_{merge} as the starting point and allocates subsequent supervision according to the capability that remains to be transferred rather than assuming that every teacher retains a positive task-level advantage.

3.2 VERIFIER-CALIBRATED CAPABILITY TRACKING

Following Eq. (3), we estimate each teacher's capability once on a held-out calibration set $\mathcal{C}_d$:

$$T_d = \widehat{J}_{\mathcal{C}_d}(\pi_{\phi_d}). \tag{7}$$

The merged student is evaluated with the same protocol to initialize

$$C_d^{(0)} = \widehat{J}_{\mathcal{C}_d}(\pi_{\theta_{\text{merge}}}), \qquad C_d^{\star(0)} = C_d^{(0)}, \qquad H_d^{(0)} = [T_d - C_d^{(0)}]_+, \tag{8}$$

where $[z]_+ = \max(z, 0)$ and $C_d^\star$ records the best student capability estimate observed so far. The calibration split, sampling protocol, and score aggregation are detailed in Appendix C.1.

During training, CA-MOPD reuses verifier outcomes from the student's existing on-policy rollouts rather than generating an additional rollout for capability estimation. Let $Z_d^{(k)}$ denote the domain-level verifier observation at controller update k. We maintain an exponential moving average

$$C_d^{(k)} = \beta C_d^{(k-1)} + (1-\beta) Z_d^{(k)}, \tag{9}$$

and update the historical best as

$$C_d^{\star(k)} = \max\left\{ C_d^{\star(k-1)}, C_d^{(k)} \right\}. \tag{10}$$

The online signed gap and positive headroom are then

$$g_d^{(k)} = T_d - C_d^{(k)}, \qquad H_d^{(k)} = [g_d^{(k)}]_+. \tag{11}$$

Thus, T_d provides a fixed teacher reference, while $C_d^{(k)}$ tracks the student's current capability under the same verifier family.

3.3 CAPABILITY-AWARE DISTILLATION CONTROLLER

The controller captures two signals: the amount of teacher advantage that remains and whether previously acquired student capability has regressed.

Remaining capability headroom. For domains with non-negligible initial headroom, we measure the fraction of the original deficit that remains:

$$r_d^{(k)} = \frac{H_d^{(k)}}{H_d^{(0)}}. \tag{12}$$

The corresponding headroom factor is

$$f_{d,\text{head}}^{(k)} = \text{clip}\left(\left(r_d^{(k)} \right)^p, f_{\min}, f_{\max} \right). \tag{13}$$

As the estimated teacher advantage closes, this factor decreases; a positive floor keeps teacher feedback available instead of permanently disabling the domain. Numerical handling of near-zero initial headroom and all controller hyperparameters are given in Appendix A.1.

Regression-aware retention. Because continued optimization on other domains can reduce a previously acquired capability, we compare the current estimate with its historical best:

$$R_d^{(k)} = \left[C_d^{\star(k)} - C_d^{(k)} - \tau \right]_+. \tag{14}$$

When $R_d^{(k)} > 0$, a bounded guard factor $f_{d,\text{guard}}^{(k)}$ increases supervision according to the detected regression. The final capability factor is

$$f_d^{(k)} = \begin{cases} \max\left\{ f_{d,\text{head}}^{(k)}, f_{d,\text{guard}}^{(k)} \right\}, & R_d^{(k)} > 0, \\ f_{d,\text{head}}^{(k)}, & R_d^{(k)} = 0. \end{cases} \tag{15}$$

The exact guard mapping, warm-up, and controller constants are provided in Appendix A.1. The headroom term therefore allocates supervision toward capability that remains to be transferred, while the retention term responds when previously observed capability declines.

3.4 CAPABILITY-WEIGHTED OPD

Capability allocation does not replace token-share balancing: the former estimates how much task-level transfer remains useful, whereas the latter corrects unequal token contributions caused by response-length differences. Let $s_d^{(k)}$ be the observed response-token share of domain d and q_d its target share. We retain the Open-MOPD balancing factor

$$b_d^{(k)} = \frac{q_d}{s_d^{(k)}}. \tag{16}$$

The capability factor and token-balancing factor are combined into the actual per-token multiplier

$$w_d^{(k)} = F^{(k)} \frac{b_d^{(k)} f_d^{(k)}}{\sum_j s_j^{(k)} b_j^{(k)} f_j^{(k)}}, \qquad F^{(k)} = \max_j f_j^{(k)}. \tag{17}$$

For example i from domain $d(i)$, CA-MOPD scales the common dense OPD signal as

$$\widehat{A}_{itv} = \mathrm{sg}\left[w_{d(i)}^{(k)} A_{itv} \right]. \tag{18}$$

The resulting weighted signal is optimized with the token-normalized PPO objective (Schulman et al., 2017) used by the Open-MOPD implementation (Gao et al., 2026). The full candidate-level OPD signal, clipping rule, and optimization objective are given in Appendix A.2.

Overall, CA-MOPD leaves the token-level teacher supervision mechanism unchanged and modifies only how strongly each domain contributes to optimization. The teacher provides the local distillation direction, while verifier-estimated capability feedback determines the domain-level optimization pressure.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We use SmolLM3-3B (Bakouch et al., 2025) with the MixSFT checkpoint and three RL specialists released by Open-MOPD (Gao et al., 2026). The specialists cover mathematics, code generation, and instruction following and share the same MixSFT initialization. The main CA-MOPD model is initialized from the Task-Arithmetic merge of the three specialists and trained for 200 outer steps. We evaluate mathematics on AIME24 and AIME25, code generation on LiveCodeBench v5 and v6 (Jain et al., 2025), and instruction following on IFEval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2026). All six benchmark scores are reported as percentages, and their equal-weight mean is used as the aggregate score. Unless otherwise stated, comparisons use the same training budget and evaluation protocol. Full training, decoding, and benchmark configurations are provided in Table 6 and Appendix C.2.

4.2 OVERALL MULTI-DOMAIN PERFORMANCE

Table 1 shows that CA-MOPD achieves the highest six-benchmark average of 32.22, exceeding ParamMerge-TA, Naive MOPD, and Open-MOPD by 0.74, 0.86, and 0.83 points, respectively. It ranks first on both mathematics benchmarks and IFBench, with gains of 2.19 points on AIME24 and 3.34 points on IFBench over Open-MOPD. These results demonstrate the effectiveness of CA-MOPD in integrating specialist capabilities into a single model, achieving stronger aggregate performance than parameter merging alone and existing MOPD baselines.

4.3 SPECIALIST MERGING PROVIDES A STRONG INITIALIZATION FOR MOPD

We next examine whether specialist merging provides a stronger starting point for MOPD. To isolate the effect of initialization, we train CA-MOPD for the same 200-step budget from either MixSFT or ParamMerge-TA. Table 2 reports performance before and after CA-MOPD training.

ParamMerge-TA already provides a strong multi-domain initialization, improving the aggregate score from 24.92 to 31.48 before any on-policy distillation. Notably, this nearly matches the 31.53 obtained

Table 1: Overall performance across mathematics, code generation, and instruction following. All values are percentages. **Bold** indicates the highest score in each column among all single-model methods.

Method	AIME24	AIME25	LCBv5	LCBv6	IFEval	IFBench	Avg
MixSFT	14.58	17.14	14.49	19.89	68.76	14.67	24.92
RouteRL	23.23	24.27	23.11	22.06	71.90	22.00	31.10
ParamMerge-TA	23.80	23.23	**23.83**	**22.69**	73.01	22.33	31.48
Naive MOPD	22.92	24.79	21.92	22.34	74.86	21.33	31.36
Open-MOPD	22.76	25.00	22.99	21.66	**75.60**	20.33	31.39
CA-MOPD	**24.95**	**25.10**	22.75	21.60	75.23	**23.67**	**32.22**

Table 2: Effect of initialization under a fixed 200-step CA-MOPD budget. We report performance before and after training; **bold** marks the better post-training result for each metric.

Initialization	Stage	AIME24	AIME25	LCBv5	LCBv6	IFEval	IFBench	Avg
MixSFT	Initial	14.58	17.14	14.49	19.89	68.76	14.67	24.92
	+ CA-MOPD	21.93	**25.52**	**23.11**	**23.31**	72.64	22.67	31.53
ParamMerge-TA	Initial	23.80	23.23	23.83	22.69	73.01	22.33	31.48
	+ CA-MOPD	**24.95**	25.10	22.75	21.60	**75.23**	**23.67**	**32.22**

after training CA-MOPD from MixSFT for the full 200-step budget, suggesting that specialist merging can integrate a substantial portion of the target capabilities before MOPD begins.

With the same subsequent 200-step CA-MOPD budget, the merged initialization further improves the final average to 32.22. Although the gains are not uniform across domains, this result shows that merging and MOPD are complementary: merging provides a stronger capability-integrated starting point, while subsequent on-policy distillation refines the remaining capability gaps.

4.4 Ablation of Capability-Gap Guidance

We first examine whether capability gap and policy discrepancy provide equivalent signals for allocating distillation effort. Figure 2 shows that they exhibit substantially different training dynamics. In particular, a nonpositive EMA capability gap coexists with positive policy discrepancy at 170, 80, and 129 of the 200 recorded steps for mathematics, code, and instruction following, respectively. This behavior is especially common because the ParamMerge-TA initialization already approaches or exceeds the calibrated teacher capability in several domains, while substantial token-level discrepancy remains. Thus, policy discrepancy can persist even when little or no positive teacher capability advantage remains.

We next compare the step-50 CA-MOPD checkpoint with a distribution-discrepancy variant, both originating from MixSFT. The variant replaces the capability-headroom ratio with an EMA-smoothed discrepancy m_d from Eq. (2), normalized by its first-batch value, while retaining the capability-based regression guard, token balancing, warm-up, and clipping.

Table 3 shows that capability-gap guidance achieves a higher six-benchmark average (31.38 versus 31.09). It improves LCBv5, LCBv6, IFEval, and IFBench by 1.20, 0.29, 1.11, and 2.00 points, respectively, while distribution-discrepancy guidance performs better on both mathematics benchmarks. Together with the distinct signal dynamics in Figure 2, these results support the usefulness of capability-gap guidance for aggregate multi-domain performance, with gains in code generation and instruction following in this comparison.

4.5 Training Allocation Dynamics

Figure 3 shows how capability feedback modulates training allocation in the ParamMerge-initialized run. We report the weighted token share $a_d^{(k)} = s_d^{(k)} w_d^{(k)} / \sum_j s_j^{(k)} w_j^{(k)}$, which summarizes relative token weighting after response-length correction, but should not be interpreted as measured gradient

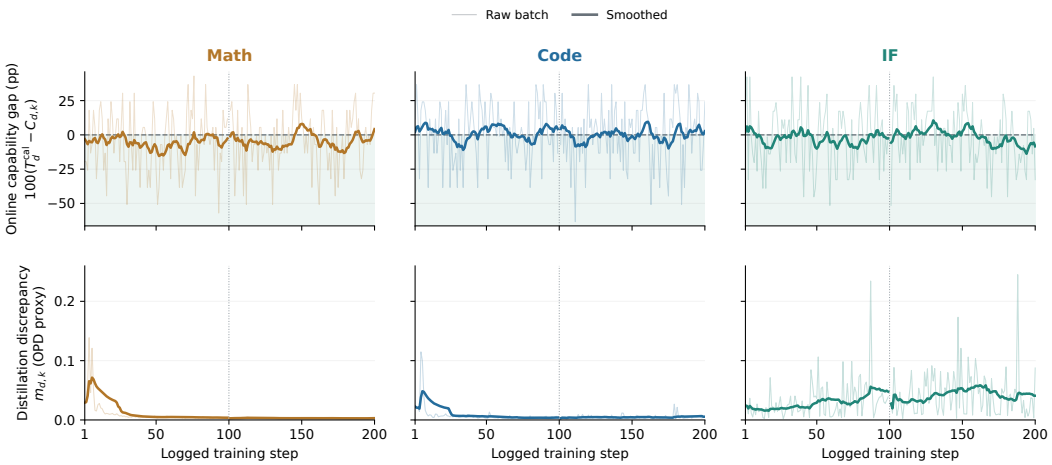

Figure 2: Capability-gap and discrepancy dynamics during 200 steps of ParamMerge-initialized CA-MOPD. Top: fixed calibrated teacher score minus student verifier score, in percentage points; faint lines use raw batch scores and bold lines use the logged EMA. Bottom: the OPD discrepancy proxy, shown as raw values and a trailing 20-step mean. Negative online gaps indicate that the student estimate exceeds the calibrated teacher reference. These are training-time diagnostics on changing prompts, not fixed-benchmark capability trajectories; IF uses the `content_ok` verifier.

Table 3: Step-50 comparison of capability-gap and distribution-discrepancy guidance from MixSFT. Both variants retain the capability-based regression guard. Scores are percentages and Avg is the equal-weight six-benchmark mean. **Bold** marks the higher score in each column.

Allocation signal	AIME24	AIME25	LCBv5	LCBv6	IFEval	IFBench	Avg
Capability gap	21.98	25.00	**21.74**	**22.46**	**74.12**	**23.00**	**31.38**
Distribution discrepancy	**24.06**	**25.73**	20.54	22.17	73.01	21.00	31.09

contribution. Table 4 relates these shares and regression-guard activity to domain-level performance changes over 200 training steps.

To isolate the guard's effect on the controller output, we reconstruct the headroom-only factor on the same recorded capability trajectory, preserving clipping and warm-up. The guard increases the final factor at 172, 92, and 107 of the 200 recorded steps for mathematics, code, and instruction following, respectively. This effect is most pronounced in mathematics, where the initially nonpositive teacher advantage would otherwise keep the headroom-only factor near its floor. Under the full controller, mathematics improves from 23.52 to 25.03, while instruction following improves from 47.67 to 49.45. These trajectories are consistent with the guard retaining useful supervision even when the estimated teacher advantage is small.

Allocation magnitude, however, does not directly predict downstream gain. Code receives the largest average weighted token share (40.13%) yet decreases by 1.08 points, reflecting domain-specific factors such as teacher-signal quality, cross-domain interference, and optimization dynamics. We therefore interpret allocation as a domain-conditioned response to the estimated capability state rather than as a universal predictor of downstream improvement. Importantly, the observed trajectories confirm that CA-MOPD produces substantially different supervision patterns across domains in response to their evolving capability states, which is precisely the behavior targeted by the proposed adaptive allocation mechanism.

4.6 Computational Cost

CA-MOPD estimates online capability from verifier outcomes on existing student rollouts and uses these estimates to reweight domain-level supervision, without requiring additional rollout generation. Table 5 reports the mean per-step runtime of the main training components over the 200 steps with

Table 4: Training allocation and domain-level outcomes for ParamMerge-initialized CA-MOPD. Share is the mean weighted token share over 200 recorded steps; guard steps count actual factor increases above the reconstructed headroom-only reference. Domain scores average the two corresponding benchmarks in Table 1; changes are computed before rounding, in percentage points. This is a descriptive comparison, not a component ablation.

Domain	Share (%)	Guard steps	Initial	Final	Δ
Mathematics	28.36	172/200	23.52	25.03	+1.51
Code	40.13	92/200	23.26	22.18	−1.08
Instruction following	31.51	107/200	47.67	49.45	+1.78

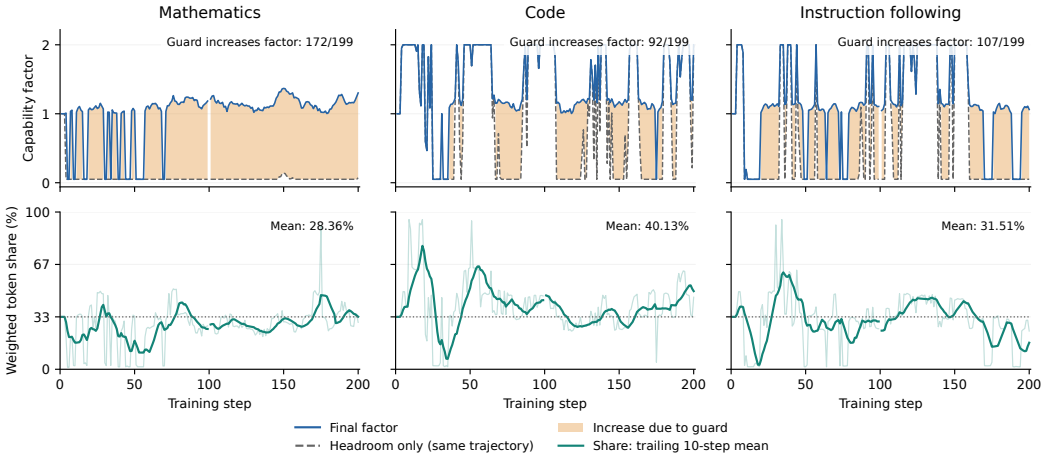

Figure 3: Training allocation during ParamMerge-initialized CA-MOPD. Top: logged final capability factors (blue) and reconstructed headroom-only factors on the same trajectory (dashed gray); orange shading marks increases due to the regression guard. Both factors include the same warm-up override. Count denominators indicate the 200-step training budget. Bottom: weighted token shares, with raw values in faint lines and trailing 10-step means in bold; dotted lines indicate equal shares. These are controller diagnostics on changing training batches, not held-out capability curves.

available timing records. Student rollout generation and actor optimization together account for 83.97% of the measured step time, while model forward passes for computing token probabilities used in distillation account for 12.62%. Online capability feedback takes 33.30 seconds per step, corresponding to 3.20% of the measured step time. These results indicate that online capability feedback constitutes a small fraction of the training runtime in this setting.

5 Related Work

On-policy and multi-teacher distillation. Knowledge distillation transfers teacher behavior to a student (Hinton et al., 2015; Kim & Rush, 2016), while GKD extends distillation to student-generated trajectories (Agarwal et al., 2024). MOPD further integrates multiple domain-specialized RL teachers into a shared student (Ma et al., 2026), with recent variants exploring richer teacher organizations (Wang et al., 2026; Yin et al., 2026). Our work focuses instead on how strongly each teacher should supervise once specialist capabilities have been partially integrated.

Adaptive allocation in multi-teacher distillation. Open-MOPD combines token-share balancing with discrepancy-based allocation and reward refresh (Gao et al., 2026), while D^3-MOPD adapts training allocation using distillation trajectories and convergence dynamics (Sun et al., 2026). These approaches primarily infer optimization demand from distillation-side signals. SPOT uses verifier-scored student continuations to calibrate token-level distillation targets (Qu et al., 2026). CA-MOPD

Table 5: Runtime breakdown of CA-MOPD training, averaged over the 200 steps with available timing records. Shares are relative to the total measured step time; the listed components account for 99.79% of this total.

Component	Mean time (s/step)	Share (%)
Student rollout generation	588.22	56.56
Model forward passes	131.19	12.62
Actor optimization	285.03	27.41
Online capability feedback	33.30	3.20

uses verifier outcomes to estimate signed domain-level capability gaps and adjust distillation strength while retaining the teacher-provided token-level signal.

Model merging. Weight-space merging integrates independently fine-tuned models without increasing inference-time model count, including Model Soups (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2022), and TIES-Merging (Yadav et al., 2023). Rather than introducing a new merging operator, we use Task Arithmetic to obtain a strong initialization and study how subsequent on-policy distillation can refine the remaining capability gaps.

6 Limitations

CA-MOPD depends on verifier scores to estimate capability, so its allocation quality is limited by the calibration distribution and verification procedure, while domain-level averages may hide example-level differences. The current controller also relies on smoothed point estimates and historical-best tracking rather than calibrated uncertainty, making small capability gaps potentially sensitive to noise. In addition, CA-MOPD continuously reweights domains but does not skip teacher computation, so it provides neither hard non-forgetting guarantees nor automatic compute savings. Finally, our evaluation is limited to one 3B model family, three domains, and known domain routing; broader generalization to larger models, heterogeneous teachers, more domains, and learned routing remains open.

7 Conclusion

In this work, we presented CA-MOPD, which combines specialist merging with capability-aware multi-teacher on-policy distillation. Our experiments show that initializing CA-MOPD with merged RL specialists provides a stronger starting point and improves final aggregate performance over initialization from the shared SFT checkpoint under the same training budget. Building on this initialization, CA-MOPD uses verifier-estimated teacher–student capability gaps and student regression signals to adapt domain-level supervision, accounting for capabilities already integrated into the student. Across six benchmarks spanning mathematics, code generation, and instruction following, the resulting approach achieves higher aggregate performance than naive MOPD, Open-MOPD, and the Task-Arithmetic merged initialization, although gains are not uniform across domains. These findings support combining parameter merging with subsequent on-policy distillation and highlight the importance of considering both the student's initial capabilities and the allocation of teacher supervision.

AI Use Statement

We used generative AI tools to assist with refining research hypotheses, providing feedback on methodology and experimental design, interpreting intermediate results, inspecting code and configurations, organizing literature, and drafting and editing the manuscript. All AI-assisted content was independently reviewed and verified by the authors, including citations, technical claims, code, configurations, and experimental results. The authors made all final research decisions and take full responsibility for the content of this work.

Ethics Statement

This work studies the integration of publicly available language-model checkpoints on mathematics, code, and instruction-following tasks. It does not involve human-subject experiments or the collection of private personal data. The resulting methods may inherit harmful behaviors or biases from their base models, teachers, training data, and verifiers. We therefore do not claim that improved benchmark integration implies broader safety or reliability, and recommend task-specific safety evaluation before deployment.

Reproducibility Statement

Appendix A specifies the capability controller, its hyperparameters, the capability-weighted OPD objective, and the training algorithm. Appendix B and Table 6 document the model family, merged initialization, hardware, training framework, sampling and optimization settings, training budget, and checkpoint schedule, together with the controller state and other training states required to resume training. Appendix C.1 describes the held-out calibration sets of 256 prompts per domain, four responses per prompt, capability-score aggregation, and reuse of training rollouts for online capability estimation. Appendix C.2 and Table 7 specify the six evaluation benchmarks, decoding settings, and scoring protocols, including average sampled accuracy for mathematics and code generation and strict prompt-level correctness for instruction following.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, et al. Smollm3: smol, multilingual, long-context reasoner. *Hugging Face Blog*, 2025.

Huan-ang Gao, Haohan Chi, Yong Yan, Shiyuan Feng, Hanlin Wu, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Open-mopd: Diagnosing and fixing capability imbalance in multi-teacher on-policy distillation. *arXiv preprint arXiv:2608.19098*, 2026.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, et al. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

Zikun Qu, Min Zhang, Mingze Kong, Zhiwei Shang, Yikun Ban, Shuang Qiu, and Zhongxiang Dai. Spot: Sparse probing and outcome calibration for on-policy distillation. *arXiv preprint arXiv:2608.04419*, 2026.

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

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

Zechen Sun, Zhiwei Zhang, Fei Zhao, Juntao Li, Mu Chuan, Huayu Deng, Guojian Zhan, Wenliang Chen, Yao Hu, and Min Zhang. D 3-mopd: Adaptive dynamic domain scheduling for efficient multi-teacher distillation. *arXiv preprint arXiv:2608.24987*, 2026.

Jianze Wang, Ying Liu, Jinlong Chen, Xuchun Hu, Qilong Zhang, Yu Cao, Jun Wang, Hua Yang, Yong Xie, and Qianglong Chen. Mad-opd: Breaking the ceiling in on-policy distillation via multi-agent debate. *arXiv preprint arXiv:2605.01347*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6672–6679, 2020.

Qixiang Yin, Huanjin Yao, Yuchen Cai, Jianghao Chen, Ziyi Wang, Min Yang, Fei Su, and Zhicheng Zhao. H-opd: Confidence aware heterogeneous multi-teacher multimodal on-policy distillation. *arXiv preprint arXiv:2607.02592*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Complete Method Details

This appendix provides the complete capability controller, optimization objective, and training procedure omitted from the main text for clarity.

A.1 Capability Controller

Recall that T_d denotes the fixed calibrated capability of teacher d, $C_d^{(k)}$ is the exponentially smoothed student capability estimate, and

$$H_d^{(k)} = [T_d - C_d^{(k)}]_+$$

denotes the remaining positive teacher advantage.

For domains with non-negligible initial headroom, the controller measures the fraction of the initial capability deficit that remains:

$$r_d^{(k)} = \frac{H_d^{(k)}}{H_d^{(0)}}, \qquad H_d^{(0)} > \epsilon. \tag{19}$$

When the initial headroom is close to zero, directly dividing by $H_d^{(0)}$ would make the controller sensitive to estimation noise. We therefore use

$$r_d^{(k)} = \begin{cases} H_d^{(k)}/H_d^{(0)}, & H_d^{(0)} > \epsilon, \\ 0, & H_d^{(0)} \le \epsilon \text{ and } H_d^{(k)} \le \epsilon, \\ H_d^{(k)}/B_d^{(k)}, & \text{otherwise}, \end{cases} \tag{20}$$

where

$$B_d^{(k)} = \max\left\{|T_d|, |C_d^{\star(k)}|, \epsilon\right\}. \tag{21}$$

The corresponding capability-headroom factor is

$$f_{d,\text{head}}^{(k)} = \text{clip}\left(\left(r_d^{(k)}\right)^p, f_{\min}, f_{\max}\right). \tag{22}$$

We use $p = 1$, $f_{\min} = 0.05$, and $f_{\max} = 2$. The positive floor prevents a domain from being completely removed from training when its estimated average capability deficit has closed.

Regression-aware retention. A small current teacher–student gap does not imply that the corresponding capability will remain stable while other domains continue updating the shared model. We therefore retain the historical-best student estimate $C_d^{\star(k)}$ and define

$$R_d^{(k)} = \left[C_d^{\star(k)} - C_d^{(k)} - \tau\right]_+, \tag{23}$$

where $\tau = 0.02$.

When regression is detected, the retention factor is

$$f_{d,\text{guard}}^{(k)} = \min\left\{f_{\max}, 1 + \lambda \frac{R_d^{(k)}}{B_d^{(k)}}\right\}, \tag{24}$$

with $\lambda = 1$.

The complete capability factor is

$$f_d^{(k)} = \begin{cases} \max\left\{f_{d,\text{head}}^{(k)}, f_{d,\text{guard}}^{(k)}\right\}, & R_d^{(k)} > 0, \\ f_{d,\text{head}}^{(k)}, & R_d^{(k)} = 0. \end{cases} \tag{25}$$

We override the controller with $f_d^{(k)} = 1$ during the first three controller updates to avoid reacting immediately to noisy capability estimates. The EMA coefficient is $\beta = 0.9$, and the target token shares are $q_d = 1/3$ for the three domains used in our experiments.

The historical-best mechanism should be interpreted as a bounded response to estimated regression rather than a hard non-forgetting guarantee. The controller continuously adjusts supervision and does not permanently disable or reactivate domains.

A.2 Capability-Weighted OPD Objective

All MOPD variants in our experiments use the same dense token-level teacher signal and differ in how domain-level supervision is weighted.

For response position t of example i, the candidate support $\mathcal{S}_{it}$ consists of the old student's Top-K predictions, with $K = 16$. The dense distillation signal follows Open-MOPD (Gao et al., 2026):

$$A_{itv} = \text{sg}\left[p_{itv}\left(\ell_{itv}^T - \ell_{itv}^{\text{old}}\right)\right], \qquad v \in \mathcal{S}_{it}. \tag{26}$$

CA-MOPD scales this signal using the effective domain multiplier $w_{d(i)}^{(k)}$:

$$\widehat{A}_{itv} = \text{sg}\left[w_{d(i)}^{(k)} A_{itv}\right]. \tag{27}$$

Let $s_d^{(k)}$ denote the observed valid-response-token share of domain d in the current rollout batch. The token-balancing factor is

$$b_d^{(k)} = \frac{q_d}{s_d^{(k)}}. \tag{28}$$

The final token multiplier is

$$w_d^{(k)} = F^{(k)} \frac{b_d^{(k)} f_d^{(k)}}{\sum_j s_j^{(k)} b_j^{(k)} f_j^{(k)}}, \qquad F^{(k)} = \max_j f_j^{(k)}. \tag{29}$$

This normalization satisfies

$$\sum_d s_d^{(k)} w_d^{(k)} = F^{(k)}. \tag{30}$$

Thus, $f_d^{(k)}$ and $w_d^{(k)}$ represent different quantities. The former captures capability-aware allocation, whereas the latter additionally incorporates token-share balancing and global scaling. Likewise, an effective weighted token share describes relative weighted-token contribution and should not be interpreted as a direct measurement of gradient-vector magnitude or parameter-space influence.

For optimization, let

$$\rho_{itv}(\theta) = \exp\left(\mathrm{clip}\left(\ell_{itv}^{\theta} - \ell_{itv}^{\mathrm{old}}, -20, 20\right)\right) \tag{31}$$

denote the candidate-level likelihood ratio.

We use the PPO clipped surrogate (Schulman et al., 2017) with the asymmetric clipping bounds of our Open-MOPD-based implementation (Gao et al., 2026):

$$P_{itv} = \max\left\{-\widehat{A}_{itv}\rho_{itv}, -\widehat{A}_{itv}\,\mathrm{clip}\left(\rho_{itv}, 0.8, 1.28\right)\right\}. \tag{32}$$

We apply dual clipping (Ye et al., 2020) with coefficient $c = 3$, giving the candidate loss

$$\ell_{itv} = \begin{cases} \min\left\{-3\widehat{A}_{itv}, P_{itv}\right\}, & \widehat{A}_{itv} < 0, \\ P_{itv}, & \text{otherwise.} \end{cases} \tag{33}$$

For an optimizer batch $\mathcal{B}$, the final token-normalized objective is

$$\mathcal{L}_{\text{CA-MOPD}}(\theta) = \frac{\sum_{i\in\mathcal{B}}\sum_t M_{it}\sum_{v\in\mathcal{S}_{it}}\ell_{itv}}{\sum_{i\in\mathcal{B}}\sum_t M_{it}}. \tag{34}$$

Entropy regularization and reference-policy KL regularization are disabled in our main configuration. The dense OPD objective, candidate support, and optimizer formulation are shared across the compared MOPD variants; CA-MOPD changes the domain-level weighting mechanism rather than the underlying token-level teacher signal.

A.3 Training Algorithm

Algorithm 1 summarizes the complete training procedure.

Algorithm 1: CA-MOPD training

Input: shared checkpoint θ_0; specialists $\{\phi_d\}_{d\in\mathcal{D}}$; calibration sets $\{\mathcal{C}_d\}$; training prompt distributions $\{\mathcal{P}_d\}$.

1. Merge specialists.
Construct

$$\theta_{\text{merge}} = \theta_0 + \sum_{d\in\mathcal{D}} (\phi_d - \theta_0),$$

and initialize the student with $\theta \leftarrow \theta_{\text{merge}}$.

2. Calibrate capability.
For each domain d:

$$T_d \leftarrow \widehat{J}_{\mathcal{C}_d}(\pi_{\phi_d}), \qquad C_d^{(0)} \leftarrow \widehat{J}_{\mathcal{C}_d}(\pi_\theta),$$
$$C_d^{\star(0)} \leftarrow C_d^{(0)}, \qquad H_d^{(0)} \leftarrow [T_d - C_d^{(0)}]_+.$$

3. Repeat for each outer training step k.
Sample prompts from all domains and generate student responses using $\pi_{\theta_{\text{old}}}$.
For each domain:

1. aggregate verifier scores into $Z_d^{(k)}$;
2. update
$$C_d^{(k)} = \beta C_d^{(k-1)} + (1-\beta) Z_d^{(k)};$$
3. update the historical best
$$C_d^{\star(k)} = \max\left\{C_d^{\star(k-1)}, C_d^{(k)}\right\};$$
4. compute capability headroom and regression;
5. obtain capability factor $f_d^{(k)}$;
6. compute observed token share $s_d^{(k)}$ and effective multiplier $w_d^{(k)}$.

Evaluate the corresponding teacher on the student's Top-16 candidate tokens, construct $\widehat{A}_{itv}$, and optimize Eq. (34).

Output: the final shared student π_θ.

B Implementation and Training Details

B.1 Training Configuration

All experiments use the SmolLM3-3B model family. The mathematics, code-generation, and instruction-following teachers are independently RL-trained specialists derived from the same mixed-domain SFT checkpoint. Table 6 summarizes the main training configuration.

B.2 Controller State and Checkpointing

The capability controller is stateful. For each domain, we store $C_d^{(k)}$, $C_d^{\star(k)}$, $H_d^{(0)}$, and the controller-update counter. Resuming training therefore requires restoring the controller state together with the model, optimizer, data-sampler, and rollout states. Restoring model parameters alone can lead to a different subsequent weighting trajectory even when the model checkpoint is identical.

Teacher references T_d and initial student estimates $C_d^{(0)}$ are tied to the corresponding calibration split and generation configuration. For reproducibility, we record the calibration split identifier together with the cached capability statistics.

Table 6: Training configuration used for the main CA-MOPD experiment.

Setting	Value
Student / teachers	SmolLM3-3B MixSFT / math, code, and IF RL specialists
Initialization	Task Arithmetic, $\alpha_d = 1$
Hardware	2× RTX PRO 5000 72GB
Precision	BF16
Training framework	VERL / FSDP with vLLM rollout
Maximum prompt length	2,048 tokens
Maximum training response length	16,384 tokens
Prompts per outer step	12 (4 per domain)
Responses per prompt	4
Training temperature	1.0
Training top-p	0.99
Training top-k	-1
Dense distillation support	old-student Top-16 candidate tokens
Optimizer	AdamW
Learning rate	1.5×10^{-6}
Adam β	$(0.9, 0.999)$
Weight decay	0.01
Gradient clipping	1.0
PPO clip range	$(0.2, 0.28)$
Dual-clip coefficient	3
Log-ratio clamp	$[-20, 20]$
PPO epochs	1
Mini-batch size	4
Micro-batch size	1
Loss aggregation	token mean
Entropy regularization	disabled
Reference KL	disabled
Outer training steps	200
Checkpoint steps	50, 100, 150, 200

C Capability Estimation and Evaluation Protocol

C.1 Capability Calibration

For each domain d, we construct a calibration set $\mathcal{C}_d$ containing 256 prompts. These prompts are excluded from subsequent CA-MOPD training.

For each calibration prompt x, both the frozen teacher and the initial student independently generate four responses using the same decoding protocol. Let $y_{x,1}, \ldots, y_{x,4}$ denote these responses. We first compute the prompt-level capability estimate

$$\widehat{J}_x(\pi) = \frac{1}{4} \sum_{r=1}^{4} V_d(x, y_{x,r}), \tag{35}$$

and then average equally across prompts:

$$\widehat{J}_{\mathcal{C}_d}(\pi) = \frac{1}{|\mathcal{C}_d|} \sum_{x \in \mathcal{C}_d} \widehat{J}_x(\pi). \tag{36}$$

This two-stage aggregation gives equal weight to every prompt regardless of the number of sampled responses.

The teacher reference

$$T_d = \widehat{J}_{\mathcal{C}_d}(\pi_{\phi_d}) \tag{37}$$

is computed once and remains fixed throughout training. The initial student capability

$$C_d^{(0)} = \widehat{J}_{\mathcal{C}_d}(\pi_{\theta_{\text{merge}}}) \tag{38}$$

Table 7: Benchmark-specific inference and evaluation settings. All evaluations use thinking mode. Top-$k = -1$ disables top-k filtering.

Domain	Benchmark	Temp.	Top-p	Top-k	Max Output Tokens	Max Sequence Length	Metric
Math	AIME24	0.6	0.95	-1	30,000	32,768	avg@64
Math	AIME25	0.6	0.95	-1	30,000	32,768	avg@64
Code	LCB v5	1.0	0.95	-1	30,000	32,768	avg@10
Code	LCB v6	1.0	0.95	-1	30,000	32,768	avg@10
IF	IFEval	1.0	0.95	-1	30,000	32,768	strict prompt
IF	IFBench	1.0	0.95	-1	30,000	32,768	strict prompt

initializes both the EMA state and the historical-best state.

During subsequent training, no additional student rollout is generated solely for capability estimation. Instead, the controller reuses verifier outcomes from responses already produced by the on-policy rollout stage.

Calibration scores and online capability estimates should therefore be interpreted as operational capability estimates under the specified prompt distribution and verifier, rather than as claims about task performance outside that evaluation setting.

C.2 Benchmark Evaluation

We evaluate mathematics using AIME24 and AIME25, code generation using LiveCodeBench v5 and v6, and instruction following using IFEval and IFBench. Table 7 summarizes the benchmark-specific evaluation configuration.

For AIME, we compute the verifier score of each independently generated response, average these scores within each problem, and then average equally across problems. We therefore report average sampled accuracy rather than majority-vote accuracy or pass@k.

For LiveCodeBench, we independently sample ten programs for each problem and evaluate each program against the benchmark tests. We first average the binary correctness scores over the ten samples for each problem, and then average equally across problems. We report this metric as avg@10.

IFEval and IFBench use strict prompt-level correctness: a prompt is counted as successful only when all required instructions are satisfied.