Open-MOPD: Diagnosing and Fixing Capability Imbalance in Multi-Teacher On-Policy Distillation

Anonymous authors
Paper under double-blind review

Abstract

Multi-teacher on-policy distillation (M-OPD) has emerged as a promising paradigm for consolidating domain-specialized reinforcement learning (RL) experts into a single generalist student via dense, token-level reward supervision. Despite its practical success, the optimization dynamics governing multi-teacher capability integration remain poorly understood, and open and rigorously reproducible recipes remain scarce. In this work, we construct an evaluation framework for multi-domain capability integration using SmolLM3-3B-Base. Our investigation reveals a pronounced *capability integration gap*: standard M-OPD recovers only 35.6% of the teacher–student capability gap. Our analysis shows that teacher disagreement might not be the major bottleneck; the main failure is a severe misallocation of the token-level optimization budget. We identify three separable contributors to this imbalance: structural sequence-length disparities across domains, different convergence rates across domains, and reward staleness caused by repeated minibatch updates on a shared rollout. To resolve these imbalances, we introduce Open-MOPD, a novel framework incorporating three mechanisms: token-share balancing, gap-aware dynamic budget allocation, and student reward refresh. Together, these mechanisms systematically restore cross-domain balance, raising the recovery rate from 35.6% to 83.4% of the teacher–student capability gap for a single student model. We open-source our end-to-end post-training recipe, training trajectories, and evaluation suites. The full M-OPD training and ablation experiments run on an $8\times$A100-80GB setup.

1 Introduction

Reinforcement learning (RL) (DeepSeek-AI et al., 2025; Zhang et al., 2025a) gives language models strong capabilities in specialized domains, but serving multiple expert models is often computationally expensive. Multi-teacher on-policy distillation (M-OPD) offers a practical solution: each input prompt is routed to a domain expert, which provides token-level reward supervision for responses generated by the student (Li et al., 2026; Ma et al., 2026a; Bercovich et al., 2025). However, the optimization dynamics of combining multiple teachers into a single student remain poorly understood, and the research community still lacks an open, reproducible recipe.

To study these dynamics, we build a multi-domain capability integration experimental framework using SmolLM3-3B-Base (Hugging Face, 2025). Our pipeline includes a mixed-domain SFT initialization, three domain-specific RL teachers, and mixed-domain M-OPD. The full M-OPD training recipe and its ablation experiments are reproducible on an $8\times$A100-80GB academic setup (Section 2.2). As shown in Table 1, we observe a clear *capability integration gap*. Naive M-OPD reaches an average score of 28.05, whereas distilling each domain independently (RouteOPD) reaches 31.55. In particular, short-response instruction-following (IF) tasks exhibit the largest gap, falling 6.16 points below their RouteOPD reference and plateauing earliest during training. Figure 1 shows that this gap is strongly asymmetric across domains.

When multiple teachers are merged into a single student, the central question is where the student's capability loss comes from. The most natural hypothesis is teacher disagreement: supervision signals from different teachers pull the student's weights in different directions, so capability integration across domains interferes with itself, making it difficult to reach the ideal upper bound (Wu et al., 2025; Ma et al., 2026b). However, our analysis and intervention experiments show that token-level

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.

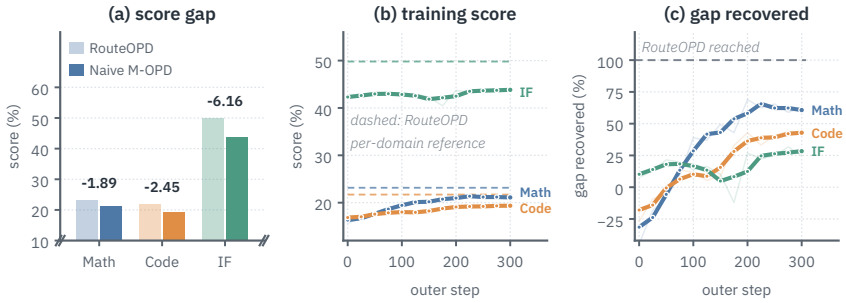

Figure 1: Naive M-OPD falls short of RouteOPD, especially on IF. (a) Domain scores and gaps; (b) training trajectories; (c) fraction of the RouteOPD gain recovered from mixed-domain SFT.

teacher disagreement might not be the primary bottleneck. Instead, intervening on disagreement tokens actually degrades training. These results motivate us to examine how the training budget is allocated across domains.

We dive into the training budget allocated across domains and diagnose three key factors that lead to an imbalanced distribution of the training signal. First, **differences in response length across domains skew which tokens participate in each update**. Second, **a mismatch between convergence rate and budget allocation** prevents high-potential domains from receiving more compute budgets. Third, under the inner-batch updates used to improve throughput, **small-batch updates leave the distillation reward stale** relative to the student's current policy. Together, these three imbalances keep training budget from flowing to the domains where it would do the most for overall capability recovery, and this is the optimization bottleneck of naive M-OPD.

To address these issues, we introduce Open-MOPD, a novel M-OPD framework with three targeted mechanisms, one for each source of imbalance. First, **token-share balancing** sets a target response-token share for each domain, flattening the cross-domain imbalance caused by differences in response length. Second, **gap-following allocation** shifts the training budget toward domains with a larger remaining teacher–student gap, directing more budget to domains where it is more valuable for overall capability recovery. Third, **student reward refresh** recomputes student-dependent probabilities before each inner update, keeping the distillation reward matched to the current policy at all times; we further show that this introduces no measurable additional compute overhead in the standard M-OPD pipeline. Together, this recipe raises the student model's recovery of the teacher–student capability gap from 35.6% to 83.4%.

2 Open-MOPD: An Open Recipe for Multi-Teacher Capability Integration

Our goal is to integrate the capabilities of several domain experts into one unified student model. Let the set of domains be $\mathcal{D}$. On our base model, $\mathcal{D} = \{\text{math}, \text{code}, \text{instruction_following (IF)}\}$. Each domain has an expert teacher π_{ϕ_d} trained with reinforcement learning, and the student model is π_θ. Each training sample carries a domain label $d(x)$, allowing a single student model to receive supervision signals from the corresponding domain expert. With the training framework fixed, Open-MOPD aims to identify a recipe that stably transfers the capabilities of all three experts into a single student model.

2.1 The Multi-Teacher On-Policy Distillation Objective

We train the student to match its domain teacher on student-generated prefixes. Let $\pi_d(v)$ and $\pi_\theta(v)$ denote teacher and student probabilities given $(x, y_{<t})$, with $d = d(x)$. Over the rollout student's top-k candidates $\mathcal{S}_t$ ($k = 16$), the token-level reward is $r_t(v) = \tilde{\pi}_\theta(v)[\log \pi_d(v) - \log \pi_\theta(v)]$. Each candidate reward serves as an advantage in the PPO-style surrogate (Schulman et al., 2017):

$$L_t = -\sum_{v \in \mathcal{S}_t} \min\{\rho_t(v) r_t(v),\ \text{clip}(\rho_t(v), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}})\, r_t(v)\}\,. \tag{1}$$

Table 1: **Integration gap on SmolLM3-3B.** Each domain reports sub-dataset scores and a macro-average; Total Score is the mean of the three. SFT rows 1–4 are successive checkpoints; single-domain RL and OPD report only their target domain. Gray-shaded *RouteRL* and *RouteOPD* pick a model by domain label at evaluation time. Recovery rate is $\frac{\text{Row Total}-\text{SFT}}{\text{RouteRL}-\text{SFT}}$ against π_{mixsft} (epoch 4). Within *Baselines* and *M-OPD*, bold and underline mark the best and second-best per column.

	Math (avg@64)			Code (avg@10)			IF				
Method	AIME24	AIME25	avg	LCBv5	LCBv6	avg	IFEval	IFB_{test}	avg	Total	Recovery rate
SmolLM3-3B-Base	2.08	1.72	1.90	3.41	6.17	4.79	16.08	13.00	14.54	7.08	—
SFT											
π_{mixsft} (e1)	12.24	16.61	14.43	13.29	17.20	15.25	64.70	16.00	40.35	23.34	—
π_{mixsft} (e2)	12.55	17.97	15.26	13.17	16.63	14.90	65.62	16.00	40.81	23.66	—
π_{mixsft} (e3)	15.10	19.53	17.32	14.97	18.57	16.77	66.36	18.00	42.18	25.42	—
π_{mixsft} (e4)	15.63	20.26	17.95	15.99	19.20	17.60	66.91	16.00	41.46	25.67	—
RL											
$\pi_{\phi_{\text{math}}}$	23.65	24.84	24.24	—	—	—	—	—	—	—	—
$\pi_{\phi_{\text{code}}}$	—	—	—	22.16	21.31	21.73	—	—	—	—	—
$\pi_{\phi_{\text{IF}}}$	—	—	—	—	—	—	74.49	27.67	51.08	—	—
RouteRL	23.65	24.84	24.24	22.16	21.31	21.73	74.49	27.67	51.08	32.35	100%
OPD											
Math-OPD	22.34	23.96	23.15	—	—	—	—	—	—	—	—
Code-OPD	—	—	—	22.28	21.14	21.71	—	—	—	—	—
IF-OPD	—	—	—	—	—	—	75.60	24.00	49.80	—	—
RouteOPD	22.34	23.96	23.15	22.28	21.14	21.71	75.60	24.00	49.80	31.55	88.0%
Baselines											
RFT	**22.97**	**23.91**	**23.44**	18.98	19.43	19.21	55.08	18.67	36.87	26.51	12.6%
π_{mixrl}	21.15	22.14	21.64	16.59	20.97	18.78	70.24	<u>22.67</u>	46.45	28.96	49.3%
ParamMerge-Avg	18.91	20.99	19.95	18.38	21.20	19.79	70.06	17.67	43.86	27.87	32.9%
ParamMerge-TA	21.93	22.76	22.34	**21.74**	**23.14**	**22.44**	<u>71.53</u>	21.67	<u>46.60</u>	<u>30.46</u>	<u>71.7%</u>
M-OPD											
Naive M-OPD	20.92	21.60	21.26	17.53	20.99	19.26	68.61	18.67	43.64	28.05	35.6%
Open-MOPD (Ours)	<u>21.98</u>	<u>22.86</u>	<u>22.42</u>	<u>20.84</u>	<u>22.63</u>	<u>21.73</u>	**74.49**	**24.67**	**49.58**	**31.24**	**83.4%**

Here $\tilde{\pi}_\theta$ is normalized over $\mathcal{S}_t$, and $\rho_t(v) = \pi_\theta(v)/\pi_{\theta_{\text{old}}}(v)$. Appendix A.4 gives gradient handling and the dual-clipping extension used in implementation.

2.2 Settings

The End-to-End Training Recipe. The Open-MOPD framework consists of three sequential stages. First, we build a shared checkpoint via **mixed-domain SFT**, with math, code, and instruction following (IF) as representative multi-domain tasks. Next, we train a **domain-specific RL** teacher for each target domain. Finally, we distill all three teachers into a unified student model using **naive M-OPD**. We use **SmolLM3-3B-Base** as our base model, requiring only an 8×A100-80GB setup to reproduce the full M-OPD training and ablations. *Considerations behind the base model choice are in Appendix A.1, and detailed training and evaluation setups are provided in Appendix A.*

Evaluation Setup. We use the RL experts for the three domains to form *RouteRL*, and the students produced by single-teacher OPD to form *RouteOPD*, as references for measuring how much of the capability gained from SFT to RL can be recovered. The performance delta between a unified M-OPD model and *RouteOPD* reflects the *integration gap*. Table 1 summarizes the performance of all baseline models and multi-domain capacity merging methods.

3 Diagnosing the Multi-Teacher Integration Gap

The recipe in Section 2 provides an open baseline for multi-teacher OPD. It combines mixed-domain SFT, one teacher for each domain and on-policy distillation on the student's own responses. This setup gives all domains the correct teacher signal while training one shared student. **However, this baseline recovers only 35.6% of the score improvement from mixed-domain SFT to RouteRL.** Where does the capability gap come from?

3.1 Analyzing teacher conflict

We first investigate whether teacher disagreement accounts for the integration gap (Wu et al., 2025; Ma et al., 2026b; Yu et al., 2026b). We measure sampled-token teacher disagreement using the spread of teacher log-probabilities on the same prefix. For a student-sampled token y_t, we compute the largest minus the smallest log-probability assigned by the teachers:

$$c_t = \max_{d \in \mathcal{D}} \log \pi_{\phi_d}(y_t \mid x, y_{<t}) - \min_{d \in \mathcal{D}} \log \pi_{\phi_d}(y_t \mid x, y_{<t}). \tag{2}$$

Here c_t denotes the degree of teacher conflict at position t, d indexes a domain in $\mathcal{D}$, and π_{ϕ_d} is the teacher policy for domain d. If teacher conflict were a systematic bottleneck for capability merging, we would expect a large number of tokens with large c_t. To further analyze this hypothesis, we also design two intervention mechanisms for teacher conflict: **conflict masking** and **teacher consensus**. Conflict masking removes the top 1%, 5%, or 20% of tokens ranked by teacher spread from the distillation loss; teacher consensus uses a threshold of 1 nat to distinguish disagreement from conflict, using the average log-probability of the three teachers on tokens with $c_t \leq 1$, while retaining the probability of the corresponding prompt teacher on tokens with $c_t > 1$. The two mechanisms respectively aim to choose whether to "question" or "trust" the probability of the corresponding domain teacher when teachers disagree.

Results: Figure 2 reveals three phenomena. **First, disagreement is low and stays stable.** The batch-mean teacher spread averages 0.126 nats and stays below 0.27 nats throughout training. **Second, the number of high-disagreement tokens is small.** Only 0.62% of all tokens exceed 1 nat, with domain-specific rates of 3.9% for IF and 0.31% for math. **Third, the intervention mechanisms instead reduce training effectiveness.** Although conflict tokens account for only a small fraction, under both conflict masking and teacher consensus, the total multi-teacher evaluation score drops significantly. In total, all four tested interventions reduce the total score compared to the baseline.

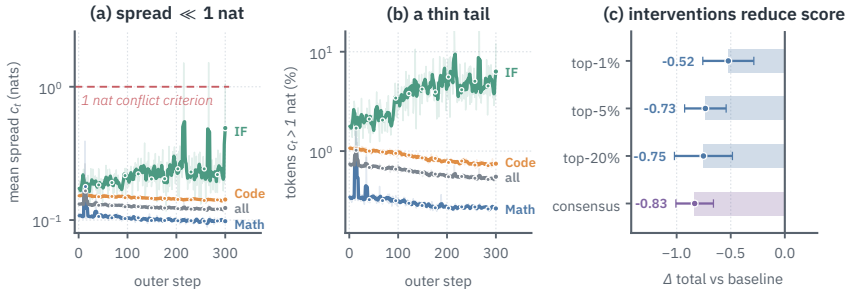

Figure 2: **Sampled-token disagreement interventions do not improve capability integration.** We report the spread trajectories, the fraction of tokens exceeding 1 nat, and the downstream impact of conflict-mitigation interventions.

Through the analysis above, token-level teacher disagreement does not appear to be the major bottleneck. Detailed intervention settings and measurements are provided in Appendix B.2. We thus shift our focus to how the training signal is shared across the three domains (Liang et al., 2025).

3.2 Measuring training imbalance across domains

By tracking the training signal across domains, we find that inter-domain training imbalance might be a larger source of the total capability-merging gap than teacher disagreement. We measure three sources of training imbalance: **token counts across domains, gaps to teachers, and outdated rewards during inner updates.**

Token imbalance. Although we give each domain a relatively uniform sampling rate, we find that **the token counts across domains differ greatly:** as shown in Figure 3(a), even when the prompt sampling ratios for math, code, and IF stay fairly balanced, the resulting token shares differ sharply, with IF receiving only 0.99% of the token share. We find that this imbalance is caused by the vastly different response lengths of the three task types: instruction-following responses are far shorter than

those for math and code (Figure 3(b)). While a simple fix is to increase the prompt sampling ratio of the low-share IF task, the required unnormalized sampling weight is 33.6 for IF and about 0.67 for each other domain (Figure 3(c)). After normalization, this concentrates prompts on IF, reducing prompt diversity and long-response training for math and code.

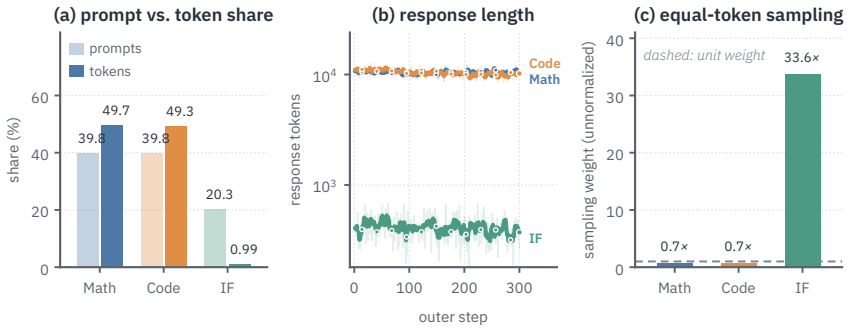

Figure 3: **Prompt share and token share differ sharply.** (a) Prompt and token shares; (b) response length; (c) Unnormalized sampling weights $(1/3)/s_d^{\text{tok}}$ for equal token shares, assuming unchanged response lengths; the dashed line marks unit weight.

Different convergence rates. We further find that token-count imbalance is not the only source of training imbalance, since an equal token share does not imply an equal reward signal. The update budget across domains is also affected by **differences in average reward.** To formalize these cross-domain reward disparities, let s_d^{tok} denote domain d's share of response tokens in the current batch (Figure 3(a)); the contribution of a domain then approximately satisfies:

$$B_d \propto s_d^{\text{tok}}\, \bar{m}_d, \qquad \bar{m}_d = \mathbb{E}_{t\in d}[|r_t|]. \tag{3}$$

Here $\bar{m}_d$ is domain d's average reward-signal magnitude in the current batch. The token share itself is $s_d^{\text{tok}} = n_d L_d / \sum_{j\in\mathcal{D}} n_j L_j$, where n_d is the number of prompts in domain d and L_d is its average response length.

Figure 4 reveals three phenomena. **First, the reward signal magnitudes differ significantly across domains.** In Figure 4(a), the initial signal difference between instruction following and math is as high as $4.9\times$. **Second, reward magnitudes decrease at different rates across domains**. Figure 4(b) highlights that the gap in instruction following shrinks by $2.4\times$ over the course of training, while code shrinks by only $1.9\times$. **Third, even if the token-count imbalance is resolved, the effective budget still drifts as training progresses.** As shown in Figure 4(c), although the weighted token shares are balanced, the budget share of code becomes dominant in late training.

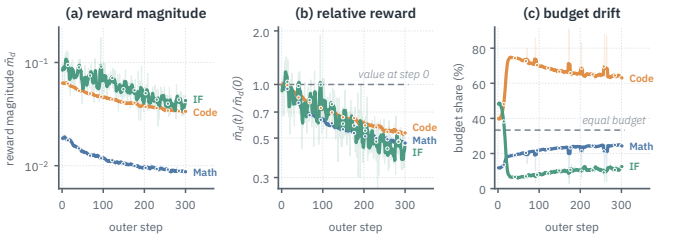

Figure 4: **The effective budget drifts with distillation progress.** (a) per-token reward magnitude $\bar{m}_d$ (log axis); (b) each domain's magnitude normalized to its own initial value, representing the remaining teacher–student gap; (c) budget share on a trajectory where $w_d^{\text{share}} s_d^{\text{tok}} = 1/3$, so the effective budget reduces to $B_d \propto \bar{m}_d$ and its drift follows the reward magnitudes.

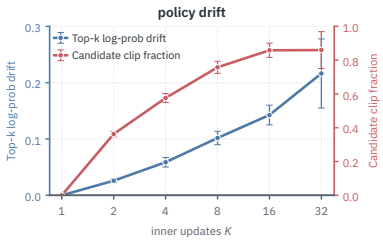

Figure 5: **Policy drift within a rollout batch.** Blue/left: top-k log-prob drift; red/right: candidate clip fraction. Means $\pm$ one standard error over available measurements through outer step 24.

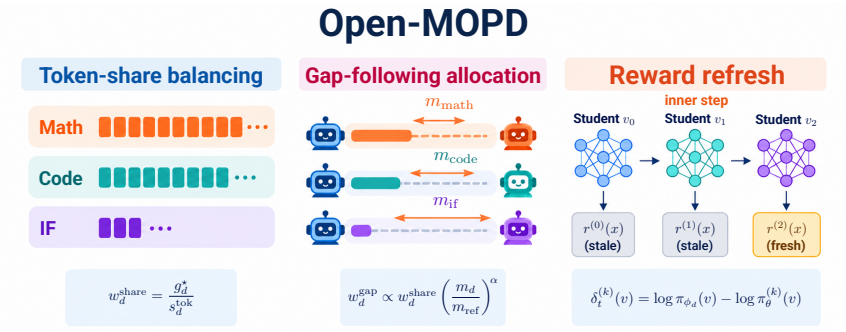

Figure 6: **Overview of Open-MOPD.** *Left:* token-share balancing weights each domain by its response-token share s_d^{tok}, so that its share of the loss stops following its response length. *Middle:* gap-following allocation scales the budget by the remaining teacher–student gap, for which the reward magnitude m_d is the proxy. *Right:* reward refresh recomputes the student-dependent reward against the current student at every inner step k.

Reward staleness. Throughput optimization introduces a third source of imbalance (Fu et al., 2025; Zhong et al., 2025; Sheng et al., 2025). To amortize generation cost, practical training typically performs K inner updates from one large rollout batch (Kang et al., 2026; Li et al., 2025a). Here the teacher model is fixed, but the student model has already changed after the first inner update; if subsequent updates still use the log-probabilities $\log \pi_\theta(v \mid x, y_{<t})$ and weights $\tilde{\pi}_\theta(v \mid x, y_{<t})$ $(v \in \mathcal{S}_t)$ from rollout, then the student-dependent part in Equation 9 becomes inconsistent with the current policy.

We track this deviation with the top-k log-probability drift defined in Appendix B.4. Across the configurations in Figure 5, this statistic grows from 0 at $K = 1$ to 0.216 at $K = 32$, and the candidate clip fraction reaches 86%. **Consequently, the student-dependent rewards become outdated in later minibatches.**

The above results systematically analyze the imbalance phenomena in multi-teacher OPD. The interventions in Section 3.1 provide no evidence that reducing sampled-token teacher disagreement improves capability integration. We further diagnose the inter-domain update budget and deviation, and identify three factors that cause the capability merging problem: imbalance in domain token counts, imbalance in average rewards across domains, and bias from reward staleness in inner-batch updates.

4 From Diagnosis to Method: Open-MOPD

Guided by the preceding diagnosis, we introduce three mechanisms to address training imbalance and improve the use of the distillation budget: **token-share balancing**, **gap-following allocation**, and **reward refresh**. We combine these mechanisms into **Open-MOPD**, a framework that retains the teacher routing and on-policy distillation objective of naive M-OPD. Figure 6 shows the complete pipeline.

4.1 Token-Share Balancing

Token-share balancing weights each domain's token losses within the batch-wide token average according to its current token share (Liang et al., 2025). Let $g_d^\star$ be the target domain share, with $\sum_d g_d^\star = 1$, and let s_d^{tok} be the response-token share in the current batch. We set

$$w_d^{\text{share}} = \frac{g_d^\star}{s_d^{\text{tok}}}. \tag{4}$$

The effective weighted share is then

$$\frac{w_d^{\text{share}} s_d^{\text{tok}}}{\sum_j w_j^{\text{share}} s_j^{\text{tok}}} = g_d^\star.$$

We use equal target shares, $g^\star = (1/3, 1/3, 1/3)$, as our main recipe. Using the token shares in Figure 3, this gives weights of 0.67 for math, 0.68 for code, and 33.6 for IF. Thus, IF tokens get about 50 times as much weight to make up for the 25-fold length difference, making the weighted shares equal. Compared to directly increasing the prompt ratio for IF, this preserves the prompt diversity of math and code.

Token-share balancing changes the loss contribution, but not the prompt sampling or the reward magnitude of an individual token. We keep the reward magnitude because it provides information about the remaining teacher–student gap.

4.2 Gap-Following Allocation

Token-share balancing reduces loss imbalance across domains, but its target share is fixed in advance. **Because domains train at different speeds, forcing equal shares is less effective than assigning a larger budget to domains that currently lag behind**. We use m_d as an online proxy for the remaining distillation signal. Here m_d is the mean absolute token-level reward for domain d in the current batch. Appendix A.5 specifies its computation and update schedule.

We initially tried an inverse factor, $m_d^{-\alpha}$, to give more budget to domains with smaller rewards. **A smaller reward often indicates that the student is already closer to its teacher.** For example, between outer steps 1 and 74 of the inverse-rule run, IF's m_d decreased by a factor of 35.3 and its weight rose from 26.7 to 90.7, over-allocating budget to a nearly learned task.

We therefore allocate the budget according to the reward-magnitude proxy:

$$\tilde{w}_d = w_d^{\text{share}} \,\text{Clamp}\left(\left(\frac{m_d}{m_{\text{ref}}}\right)^{\alpha}, 0.05, 20\right), \qquad w_d^{\text{gap}} = \frac{\tilde{w}_d}{\sum_j \tilde{w}_j s_j^{\text{tok}}}. \tag{5}$$

Here m_{ref} is the share-correction-weighted mean of the domain reward magnitudes. The exponent $\alpha > 0$ controls allocation sensitivity. A larger m_d gives a larger gap factor and more budget, while a decreasing m_d reduces the assigned budget accordingly. Clipping prevents extreme changes, and final normalization keeps the token-weighted mean weight at one.

4.3 Reward Refresh

The first two mechanisms allocate the budget across domains. **Reward refresh addresses staleness within a rollout** (Kang et al., 2026; Li et al., 2025a; Heo et al., 2026). During the K inner updates, the student changes while the sampled responses remain fixed. We recompute the reward using the current student before each inner update:

$$\begin{aligned} \delta_t^{(k)}(v) &= \text{sg}\Big[\log \pi_{\phi_{d(x)}}(v \mid x, y_{<t}) - \log \pi_\theta^{(k)}(v \mid x, y_{<t})\Big], \\ r_t^{(k)}(v) &= \text{sg}\Big[\delta_t^{(k)}(v)\, \tilde{\pi}_\theta^{(k)}(v \mid x, y_{<t})\Big], \qquad v \in \mathcal{S}_t. \end{aligned} \tag{6}$$

Here $\tilde{\pi}_\theta^{(k)}$ is normalized over $\mathcal{S}_t$. Each $r_t^{(k)}(v)$ serves as the corresponding candidate-token advantage; sg detaches the reward. This reuses the student forward pass already required by PPO and adds no teacher forward pass. The resulting reward tracks the student throughout the inner updates. Appendix A.5 specifies the gradient handling and caching schedule; Algorithm 1 gives the complete training loop.

5 Experiments

5.1 Main Results

We use the experimental setup described in Section 2, with the inner-update and batch settings shown in Table 2. Table 2 shows that Open-MOPD raises the capability recovered relative to *RouteRL*, compared to SFT, from 35.6% to 83.4%, lifting the total score by 3.19 points over Naive M-OPD.

Tracing where this gain comes from, we find modest improvements in math and code, with the largest contribution coming from IF, indicating that Open-MOPD directs more training budget to the domain where naive M-OPD was most imbalanced.

Table 2: **From Naive M-OPD to the full recipe.** The sequence shows the mechanism additions and the switch from $K = 1$ to $K = 4$. The three mechanism columns indicate whether token-share balancing, gap-following allocation, and reward refresh are enabled. K is the number of inner updates per rollout batch; at $K = 4$, each rollout contains 1,024 prompts, with 256 per minibatch.

	Mechanism				Evaluation scores				
Configuration	share	gap	refresh	K	Math	Code	IF	Total	Δ
Naive M-OPD				1	21.26	19.26	43.64	28.05	—
+share	✓			1	20.55	19.57	47.53	29.22	+1.17
+gap	✓	✓		1	21.00	19.31	49.50	29.94	+1.89
+K=4	✓	✓		4	23.05	21.07	47.16	30.43	+2.38
Open-MOPD	✓	✓	✓	4	22.42	21.73	49.58	**31.24**	**+3.19**

5.2 Ablation Study

To isolate the contribution of each intervention, we evaluate token-share balancing, gap-following allocation, and reward refresh incrementally and in combination (Table 2). Together, these mechanisms address the three sources of training imbalance identified in Section 3.2.

Token-Share Balancing. Token-share balancing maintains a constant weighted token share ($w_d^{\text{share}} s_d^{\text{tok}} = 33.33\%$) across the entire training trajectory. **Without it, the default sampler directs nearly 99% of gradient tokens to math and code, leaving only $\sim 1\%$ for IF.** By correcting this token-count imbalance, it improves the update budget for instruction following, which previously had a significantly lower token share, with only modest changes in math and code performance. Token-share balancing raises the overall score by **1.17 points**.

Gap-Following Allocation. We analyze how gap-following allocation dynamically adjusts domain weights as teacher–student gaps change. In Figure 7(a), we use token-share balancing to keep the weighted token share among domains at $1/3$, while gap-following allocation assigns more update budget to domains with larger gaps. Under this mechanism, the budget share of math grows, while that of IF drops, and code receives the highest average budget share. **This precisely addresses the problem shown in Figure 4(b), giving larger update budgets to domains that shrink slowly and thereby effectively alleviating the imbalance in update speed.** At $K = 1$, incorporating this mechanism gives an additional $+0.72$ points. Conversely, using an inverse reward-magnitude factor mistakenly gives priority to domains close to their teachers and therefore collapses early in our inverse-rule run (Figure 7(b)).

Reward Refresh. Reward refresh updates the student-dependent part of the dense reward before every inner update, while computing the teacher term only once per rollout. **Because it reuses the student forward pass required by PPO, it introduces no extra teacher forward passes.** Experiments show that reward refresh further improves training effectiveness without introducing identifiable extra time overhead (Figure 8). Adding reward refresh on top of the first two interventions contributes a further $+0.81$ points and completes the full Open-MOPD recipe.

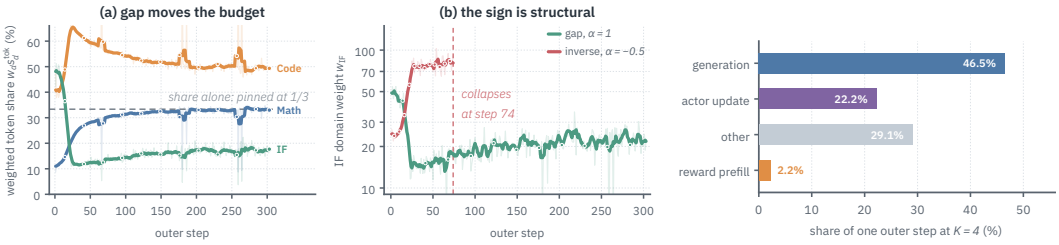

Figure 7: **Gap-following allocation dynamically changes the budget trajectory.** (a) weighted token share $w_d s_d^{\text{tok}}$, static at $1/3$ under token-share balancing alone (dashed line) and shifting toward the domain with the larger gap once gap-following is added; (b) IF domain weight under α=1 versus the inverse formulation $\alpha = -0.5$, which collapses training at step 74.

Figure 8: **Average training cost composition of one outer step at K=4.** The dense-reward computation accounts for only 2.2%. Reward refresh updates reward without changing total throughput.

6 Related Work

6.1 Knowledge Distillation

From Off-Policy to On-Policy Distillation. Knowledge distillation trains a student to match a teacher's output distribution (Hinton et al., 2015). For sequence models, the teacher usually generates responses in advance, and the student learns from this fixed corpus (Kim & Rush, 2016). The student therefore trains on teacher-generated prefixes but is evaluated on prefixes generated by its own policy. This distribution shift creates the exposure-bias problem (Ranzato et al., 2015; Bengio et al., 2015), and a large capacity gap between the teacher and the student can make distillation difficult (Cho & Hariharan, 2019; Mirzadeh et al., 2019; Li et al., 2025b). On-policy distillation evaluates the teacher on responses sampled by the student, so the teacher provides feedback on prefixes that the student actually visits (Gu et al., 2023; Agarwal et al., 2023). Recent OPD work studies generalized knowledge distillation (GKD), divergence choices, sampling rules, and token-level training signals (Ko et al., 2024; 2025; Wen et al., 2023). In particular, *Rethinking On-Policy Distillation* analyzes the phenomenology, mechanism, and recipe of OPD (Li et al., 2026).

Multi-teacher OPD. Multi-teacher OPD extends OPD to several domain teachers. Existing recipes route each student response to its domain teacher and distill token-level feedback into one student (Xiaomi LLM-Core Team et al., 2026; Ma et al., 2026a). Larger systems use teachers defined by domain, reasoning effort, or model scale (Yang et al., 2026; Bai et al., 2026; DeepSeek-AI et al., 2026; Kimi Team, 2026). Open-MOPD studies how one student should share its training budget across these teachers.

6.2 Integrating Domain Experts

Integration in data space. A common way to combine domains is to train one model on mixed-domain data. Qwen3 uses general-domain RL to improve a wide range of tasks (Yang et al., 2025). Other work studies sequential domain training or data-mixture optimization, including Nemotron-Cascade, DoReMi, and MoDoMoDo (Wang et al., 2025; Xie et al., 2023; Liang et al., 2025). Equal task sampling still does not guarantee equal training because tasks can produce different response lengths, reward magnitudes, and gradient scales (Wu et al., 2025). Open-MOPD studies these effects in multi-teacher OPD.

Integration in weight space and module space. Another route combines experts after separate training. Weight-space methods merge task-specific parameter changes, while modular methods retain separate experts and combine them at inference time (Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2023; Li et al., 2022; Sukhbaatar et al., 2024; Zhang et al., 2025b). These methods either merge specialists or keep them as separate modules. Open-MOPD instead trains one shared student from their on-policy outputs and focuses on the unequal training budgets received by different domains.

7 Conclusion

We build from scratch a fully open multi-teacher on-policy distillation pipeline—mixed-domain SFT, three domain RL teachers, multi-teacher OPD—and use it to ask what stops the capabilities of three experts from being merged into one model. We **track the reward and share bottlenecks** during training in depth, and identify the bottleneck as the allocation of the optimization budget. It has three separable contributors: **unequal token counts, unequal reward magnitudes, and rewards left stale by repeated inner updates.** The three mechanisms of **Open-MOPD** answer them one for one, and together close the gap to RouteOPD from 3.50 points to 0.31, raising recovery of the teacher–student capability gap **from** 35.6% **to** 83.4%**.** We release the complete recipe and the mechanism implementations to support reproducible follow-up work.

AI Use Statement

The authors used generative AI tools during the preparation of this paper. First, to aid or polish writing: AI assistance was used to improve clarity, fix grammar, and tighten phrasing across the draft, and to help typeset and format tables, figures, and equations in LaTeX. Second, for retrieval and discovery: AI tools helped search for and surface related work, including papers on multi-teacher distillation, on-policy distillation, and reinforcement learning post-training, which the authors then read, evaluated, and cited where relevant. Third, to draft sections of the paper: portions of the exposition, in particular early drafts of the diagnosis and ablation sections, were first drafted with AI assistance from the authors' experimental logs, plots, and notes, and then revised by the authors. AI assistance also supported interpretation of diagnostics, mathematical exposition, and consistency checks between the manuscript and code. The authors are responsible for verifying all AI-assisted text, claims, figures, and code against the underlying evidence and take full responsibility for the accuracy and originality of the submission.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. arXiv preprint arXiv:2306.13649, 2023. URL https://arxiv.org/abs/2306.13649.

Lei Bai et al. Scaling the Horizon, Not the Parameters: Reaching Trillion-Parameter Performance with a 35B Agent. arXiv preprint arXiv:2606.30616, 2026. URL https://arxiv.org/abs/2606.30616.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. arXiv preprint arXiv:1506.03099, 2015. URL https://arxiv.org/abs/1506.03099.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, et al. Llama-Nemotron: Efficient Reasoning Models. arXiv preprint arXiv:2505.00949, 2025. URL https://arxiv.org/abs/2505.00949.

Jang Hyun Cho and Bharath Hariharan. On the Efficacy of Knowledge Distillation. arXiv preprint arXiv:1910.01348, 2019. URL https://arxiv.org/abs/1910.01348.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948, 2025. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, et al. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. arXiv preprint arXiv:2606.19348, 2026. URL https://arxiv.org/abs/2606.19348.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, et al. AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning. arXiv preprint arXiv:2505.24298, 2025. URL https://arxiv.org/abs/2505.24298.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-Policy Distillation of Large Language Models. arXiv preprint arXiv:2306.08543, 2023. URL https://arxiv.org/abs/2306.08543.

Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-Policy Delta Distillation. arXiv preprint arXiv:2607.15161, 2026. URL https://arxiv.org/abs/2607.15161.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531, 2015. URL https://arxiv.org/abs/1503.02531.

Hugging Face. SmolLM3: Smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

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. URL https://arxiv.org/abs/2212.04089.

Yuxuan Jiang, Runchao Li, Shubhashis Roy Dipta, Dawei Li, and Zhao Yang. Cornerstones or Stumbling Blocks? Deciphering the Rock Tokens in On-Policy Distillation. arXiv preprint arXiv:2605.09253, 2026. URL https://arxiv.org/abs/2605.09253.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjun Kang, Sanghyun Park, Donghoon Kim, Minjae Lee, Minseo Kim, et al. AsyncOPD: How Stale Can On-Policy Distillation Be? arXiv preprint arXiv:2606.24143, 2026. URL https://arxiv.org/abs/2606.24143.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. arXiv preprint arXiv:1606.07947, 2016. URL https://arxiv.org/abs/1606.07947.

Kimi Team. Kimi K3: Open Frontier Intelligence. arXiv preprint arXiv:2607.24653, 2026. URL https://arxiv.org/abs/2607.24653.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards Streamlined Distillation for Large Language Models. arXiv preprint arXiv:2402.03898, 2024. URL https://arxiv.org/abs/2402.03898.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A Contrastive Approach Boosts the Distillation of LLMs. arXiv preprint arXiv:2503.07067, 2025. URL https://arxiv.org/abs/2503.07067.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-Train-Merge: Embarrassingly Parallel Training of Expert Language Models. arXiv preprint arXiv:2208.03306, 2022. URL https://arxiv.org/abs/2208.03306.

Xiaocan Li, Shiliang Wu, and Zheng Shen. A-3PO: Accelerating Asynchronous LLM Training with Staleness-aware Proximal Policy Approximation. arXiv preprint arXiv:2512.06547, 2025a. URL https://arxiv.org/abs/2512.06547.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, et al. Rethinking On-Policy Distillation of Large Language Models: Phenomenology, Mechanism, and Recipe. arXiv preprint arXiv:2604.13016, 2026. URL https://arxiv.org/abs/2604.13016.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small Models Struggle to Learn from Strong Reasoners. arXiv preprint arXiv:2502.12143, 2025b. URL https://arxiv.org/abs/2502.12143.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. MoDoMoDo: Multi-Domain Data Mixtures for Multimodal LLM Reinforcement Learning. arXiv preprint arXiv:2505.24871, 2025. URL https://arxiv.org/abs/2505.24871.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, et al. MOPD: Multi-Teacher On-Policy Distillation for Capability Integration in LLM Post-Training. arXiv preprint arXiv:2606.30406, 2026a. URL https://arxiv.org/abs/2606.30406.

Yingzi Ma, Zichen Zhu, Ming Jiang, and Chaowei Xiao. One Student, Many Teachers: Multi-Task On-Policy Distillation via Soft-Prompt Privileged Context. arXiv preprint arXiv:2607.18293, 2026b. URL https://arxiv.org/abs/2607.18293.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved Knowledge Distillation via Teacher Assistant. arXiv preprint arXiv:1902.03393, 2019. URL https://arxiv.org/abs/1902.03393.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence Level Training with Recurrent Neural Networks. arXiv preprint arXiv:1511.06732, 2015. URL https://arxiv.org/abs/1511.06732.

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.

Jiabin Shen, Guang Chen, and Chengjun Mao. When Top-K Misses the Decision: Tool-Call Drift in Multi-Teacher On-Policy Distillation. arXiv preprint arXiv:2607.07050, 2026. URL https://arxiv.org/abs/2607.07050.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. arXiv preprint arXiv:2409.19256, 2024. URL https://arxiv.org/abs/2409.19256.

Guangming Sheng, Yuxuan Tong, Borui Wan, Wang Zhang, Chaobo Jia, Xibin Wu, Yuqi Wu, Xiang Li, et al. Laminar: A Scalable Asynchronous RL Post-Training Framework. arXiv preprint arXiv:2510.12633, 2025. URL https://arxiv.org/abs/2510.12633.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, et al. Branch-Train-MiX: Mixing Expert LLMs into a Mixture-of-Experts LLM. arXiv preprint arXiv:2403.07816, 2024. URL https://arxiv.org/abs/2403.07816.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, et al. Nemotron-Cascade: Scaling Cascaded Reinforcement Learning for General-Purpose Reasoning Models. arXiv preprint arXiv:2512.13607, 2025. URL https://arxiv.org/abs/2512.13607.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-Divergence Minimization for Sequence-Level Knowledge Distillation. arXiv preprint arXiv:2307.15190, 2023. URL https://arxiv.org/abs/2307.15190.

Runzhe Wu, Ankur Samanta, Ayush Jain, Scott Fujimoto, Jeongyeol Kwon, Ben Kretzu, Youliang Yu, Kaveh Hassani, Boris Vidolov, and Yonathan Efroni. Imbalanced Gradients in RL Post-Training of Multi-Task LLMs. arXiv preprint arXiv:2510.19178, 2025. URL https://arxiv.org/abs/2510.19178.

Xiaomi LLM-Core Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, et al. MiMo-V2-Flash Technical Report. arXiv preprint arXiv:2601.02780, 2026. URL https://arxiv.org/abs/2601.02780.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining. arXiv preprint arXiv:2305.10429, 2023. URL https://arxiv.org/abs/2305.10429.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Beyond GRPO and On-Policy Distillation: An Empirical Sparse-to-Dense Reward Principle for Language-Model Post-Training. arXiv preprint arXiv:2605.12483, 2026. URL https://arxiv.org/abs/2605.12483.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. arXiv preprint arXiv:2306.01708, 2023. URL https://arxiv.org/abs/2306.01708.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade 2: Post-Training LLMs with Cascade RL and Multi-Domain On-Policy Distillation. arXiv preprint arXiv:2603.19220, 2026. URL https://arxiv.org/abs/2603.19220.

Guo Yu, Wenlin Liu, Yulan Hu, Hao-Xuan Ma, Jun-Peng Jiang, and Han-Jia Ye. Dense Supervision, Sparse Updates: On the Sparsity and Geometry of On-Policy Distillation. arXiv preprint arXiv:2606.13657, 2026a. URL https://arxiv.org/abs/2606.13657.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. arXiv preprint arXiv:2311.03099, 2023. URL https://arxiv.org/abs/2311.03099.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, 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.

Zichao Yu, Chengzhi Yu, Shengze Xu, Yujin Han, Bingqing Jiang, Xu Wang, and Difan Zou. Mismatch Matters: On-Policy Distillation Beyond Token Agreement. arXiv preprint arXiv:2608.09836, 2026b. URL https://arxiv.org/abs/2608.09836.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, et al. A Survey of Reinforcement Learning for Large Reasoning Models. arXiv preprint arXiv:2509.08827, 2025a. URL https://arxiv.org/abs/2509.08827.

Qizhen Zhang, Prajjwal Bhargava, Chloe Bi, Chris X. Cai, Jakob Foerster, Jeremy Fu, Punit Singh Koura, Ruan Silva, Sheng Shen, Emily Dinan, Suchin Gururangan, and Mike Lewis. BTS: Harmonizing Specialized Experts into a Generalist LLM. arXiv preprint arXiv:2502.00075, 2025b. URL https://arxiv.org/abs/2502.00075.

Yinmin Zhong, Zili Zhang, Xiaoniu Song, Hanpeng Hu, Chao Jin, Bingyang Wu, Nuo Chen, Yukun Chen, et al. StreamRL: Scalable, Heterogeneous, and Elastic RL for LLMs with Disaggregated Stream Generation. arXiv preprint arXiv:2504.15930, 2025. URL https://arxiv.org/abs/2504.15930.

A Constructing the Open-MOPD Recipe

In this section, we describe how we arrived at the final Open-MOPD recipe. We first screen on math SFT for a base model that stably produces reasoning trajectories, and then extend the selected base model to a mixed-domain SFT over math, code and instruction following.

A.1 Base Model Choice

OpenR1 math comparison between 1.7B and 7B. Early screening applied the same 93,733 OpenR1 math examples to Qwen3-1.7B-Base and Qwen2.5-7B-Base separately. Both runs use a 32,768-token SFT length limit, a global batch of 128 and a learning rate of 4×10^{-5}, with 732 steps per epoch. Table 5 reports the complete curve over the first five epochs.

The issue with Qwen3-1.7B-Base is not insufficient generation length: under the 31K budget, 69.17–80.42% of the samples still hit the length limit at all five checkpoints, and only 20.00–31.25% of the responses contain a closing `</think>` tag; adding epochs does not produce a monotone improvement either, and AIME24 stays below 7.1%. This means that a large fraction of the trajectories seen by a subsequent RL or OPD stage contain only a reasoning prefix and never reach a final answer. A drop in task score cannot then be attributed uniquely to the reward, the routing or cross-domain interference, because the base model and the data mixture themselves have not yet learned to terminate reasoning trajectories reliably.

The same OpenR1 data shows the opposite trend on Qwen2.5-7B-Base: AIME24 rises from 16.25% to 31.25% and the truncation rate falls from 42.92% to 12.92%. This rules out the explanation that "OpenR1 trajectories are inherently too long, so any base model would truncate", and it shows that enlarging the capacity of the base model does recover trajectory closure. Qwen2.5-7B-Base therefore satisfies the capability condition, but it puts the student, the three RL teachers and every mechanism ablation of the full recipe at 7B scale; for an open research baseline that has to be run repeatedly, this cost is too high.

Choosing SmolLM3-3B-Base. The final recipe selects SmolLM3-3B-Base as a compromise between these two extremes. Four epochs of mixed-domain SFT over the three domains give π_{mixsft}: OpenR1-Math-93k, the OCR-50k subset sampled from the full OpenCodeReasoning set, and Instruction-Nemotron contribute roughly 37.3%, 28.1% and 34.6% of the training response tokens respectively. The per-domain sample counts are balanced by response token count, so that the 820K short IF answers do not overwhelm the smaller number of longer math and code trajectories.

Table 3 summarizes the selection evidence for the three candidate base models. The table identifies 3B as an operating point bracketed by two failure boundaries: going down to 1.7B, generation is dominated by unclosed reasoning and truncation, and mechanism attribution is no longer unique; going up to 7B, the single model is stronger, but the end-to-end multi-teacher recipe and its repeated ablations exceed the resource boundary we set for a community baseline. SmolLM3-3B-Base retains enough trajectory capacity while still allowing the final Open-MOPD training to run on a single 8×A100-80GB node.

Table 3: Base-model choice for Open-MOPD. The table summarizes directly the experimental results used when constructing the recipe. The truncation evidence is reported in a dedicated row below each base model.

Base model	SFT data	Performance
Qwen3-1.7B-Base	OpenR1-Math-93k	AIME24 best 7.08
Truncation evidence: 69.17–80.42%; capacity and trajectory closure insufficient.		
SmolLM3-3B-Base	Three-domain mixSFT	AIME24 15.63
Truncation evidence: 15.2%; chosen as the base model of the recipe.		
Qwen2.5-7B-Base	OpenR1-Math-93k	AIME24 31.25
Truncation evidence: 12.92%; capable enough, but a full multi-teacher ablation is too costly.		

A.2 Pipeline summary

Table 4 summarizes the base model, the data, and the evaluation settings of the recipe.

Table 4: **The Open-MOPD pipeline.** Starting from a mixed-domain SFT checkpoint (π_{mixsft}), domain-specific RL produces three teacher models π_{ϕ_d}. Multi-teacher OPD then trains the student model π_θ from the initial checkpoint. All datasets are publicly released for end-to-end reproducibility.

Stage	Data	Key settings and products
Base model	SmolLM3-3B-Base	3B; 1.7B fails to close long trajectories, while 7B is too costly for repeated ablations (Appendix A.1)
I Mixed-domain SFT	Math: OpenR1-Math-93k Code: OCR-50k (sampled from the full set) IF: Instruction-Nemotron	Four epochs; sequence limit 32,768; approximately balanced by response tokens (37/28/35) product π_{mixsft}
II Domain RL teachers	Math: DAPO-Math-17k Code: DeepCoder-24k (LCB-decontaminated) IF: Nemotron-IF-RL-46k	Independent RL per domain, no domain mixing; all forked from π_{mixsft} products $\pi_{\phi_{\text{math}}}$, $\pi_{\phi_{\text{code}}}$, $\pi_{\phi_{\text{IF}}}$
III Multi-teacher OPD	The union of the three domains' RL prompts above; each prompt is hard-routed to its teacher by domain label	Response limit 16K for math/code and 2K for IF; top-k=16; the student is initialized from π_{mixsft} product π_θ
Evaluation	Math: AIME24/ AIME25 Code: LiveCodeBench v5/v6 IF: IFEval+ $\text{IFBench}_{\text{test}}$	Math: n=64, temperature 0.6 (avg@64); Code: n=10, temperature 1.0 (avg@10); IF: n=1 (avg@1); averaged evenly across the datasets within a domain first, then macro-averaged over the three domains

A.3 Stage Hyperparameters

This section lists all hyperparameters of the three stages of the recipe, taken from the training configurations used in the experiments. Table 6 gives the mixed-domain SFT and the three domain RL teachers, and Table 7 gives single-domain OPD (the RouteOPD reference of the main text) and multi-teacher OPD. Items not listed use the verl defaults (Sheng et al., 2024).

Mixed-domain SFT and domain RL teachers. The three teachers share one GRPO configuration and differ in data, sequence length, rollout group size, number of steps, and world size: IF answers are short, so it uses a smaller response limit, smaller groups and more steps; math and code use a 30,000-token response limit to accommodate the full reasoning chain. All three disable the KL penalty and apply group filtering on accuracy (the samples of one prompt are discarded if they are all correct or all wrong, with at most 8 generation batches resampled, following the dynamic-sampling trick proposed by DAPO (Yu et al., 2025)), so that the gradient comes only from groups that discriminate.

Single-domain OPD and multi-teacher OPD. The single-domain and multi-teacher runs share the optimizer, learning rate, and PPO clipping settings. Table 7 lists their sequence limits and mechanism settings. The dense reward is computed on the student top-k (k=16), and the ratio of the training-batch size to the minibatch size is K=4, which determines the number of inner updates. The multi-teacher column additionally enables the domain sampler and the three mechanism switches: token-share balancing sets the target weighted token share, gap-following allocation adjusts the domain weights using the reward-magnitude proxy, and reward refresh recomputes the dense reward within every inner update; the table gives their values in the final recipe.

Table 5: Base-model screening under the same OpenR1-Math-93k SFT. Each cell is AIME24 accuracy / generation truncation rate (%). Qwen3-1.7B-Base mainly generates long, unclosed reasoning trajectories at every observed checkpoint; Qwen2.5-7B-Base instead raises accuracy and lowers truncation together as SFT proceeds.

Epoch	Step	Qwen3-1.7B-Base	Qwen2.5-7B-Base
1	732	4.58 / 75.42	16.25 / 42.92
2	1464	**7.08** / 72.50	20.83 / 35.42
3	2196	6.67 / 80.42	30.83 / 22.08
4	2928	5.83 / **69.17**	30.42 / 19.58
5	3660	6.67 / 69.58	**31.25** / **12.92**

Table 6: Hyperparameters of stages one and two. SFT is standard supervised fine-tuning; the three RL teachers are all GRPO, each initialized from π_{mixsft} and trained only on the verifiable reward of its own domain. World size is given as number of nodes $\times$ GPUs per node, all A100-80GB. "—" means the item does not apply to that stage.

	Mixed-domain SFT	RL: math	RL: code	RL: IF
World size	8×8	8×8	8×8	4×8
Starting point	SmolLM3-3B-Base		π_{mixsft}	
Algorithm	Cross-entropy		GRPO	
train batch size	128	128	128	128
mini batch size	—	32	32	32
Learning rate	4×10^{-5}		1×10^{-6}	
lr schedule	cosine, warmup 3%		constant, warmup 10 steps	
grad clip	0.2		1.0	
Prompt limit	—	1,024	2,048	2,048
Response limit	32,768 (total length)	30,000	30,000	2,048
rollout n	—	16	16	8
Temperature	—		1.0	
epochs / steps	4 epochs	1,000 steps	1,000 steps	3,000 steps
$\text{clip}_{\text{low}}/\text{clip}_{\text{high}}$	—		0.2 / 0.25	
KL penalty	—		disabled	
entropy coeff	—		0	
group filtering	—		by acc, at most 8 generation batches	

A.4 Detailed Distillation Objective

For a prompt x, the student first samples a response $y \sim \pi_\theta(\cdot \mid x)$ from the current policy. The teacher evaluates student rollouts via a single prefill pass to generate the teacher distribution. We denote the routed teacher as $\pi_{\phi_{d(x)}}$. At position t, we take the top-k token set of the student distribution (Shen et al., 2026; Jiang et al., 2026)

$$\mathcal{S}_t = \text{TopK}_k\big(\pi_{\theta_{\text{old}}}(\cdot \mid x, y_{<t})\big), \tag{7}$$

with $k = 16$ in our recipe. The `only_stu` reward branch uses all candidates in this set; the separate `top_p_intersec` option is not applied here. For each $v \in \mathcal{S}_t$, we first define the teacher–student log-probability difference

$$\delta_t(v) = \text{sg}\Big[\log \pi_{\phi_{d(x)}}(v \mid x, y_{<t}) - \log \pi_\theta(v \mid x, y_{<t})\Big], \tag{8}$$

where $\text{sg}[\cdot]$ denotes stop-gradient; this is the *token-level advantage signal* of dense distillation. We weight each $\delta_t(v)$ by the student probability normalized over $\mathcal{S}_t$ to obtain a *candidate-token reward* (Xu et al., 2026; Yu et al., 2026a)

$$r_t(v) = \text{sg}[\delta_t(v) \cdot \tilde{\pi}_\theta(v \mid x, y_{<t})], \qquad \tilde{\pi}_\theta(v \mid x, y_{<t}) = \text{softmax}_{u \in \mathcal{S}_t}\big[\log \pi_\theta(u \mid x, y_{<t})\big](v), \tag{9}$$

The entire candidate-token reward, including its student-probability weight, is detached from the computation graph. Tokens outside $\mathcal{S}_t$ contribute zero. For reward-magnitude statistics, we summa-

Table 7: Hyperparameters of stage three. The left columns are the three single-domain OPD runs (the RouteOPD reference of the main text) and the right column is multi-teacher OPD. The table reports the shared optimization settings and the sequence limits for each configuration; the token-share balancing / gap-following allocation / reward refresh rows are the values of our mechanisms in the final recipe.

	Single-domain OPD (RouteOPD reference)			Multi-teacher OPD
	math	code	IF	Open-MOPD
World size	1×8	1×8	1×8	1×8
Teacher	$\pi_{\phi_{\text{math}}}$	$\pi_{\phi_{\text{code}}}$	$\pi_{\phi_{\text{IF}}}$	3-teacher hard routing
Student starting point	π_{mixsft}			
train batch size	1,024			
mini batch size	256 (i.e. K=4)			
Learning rate	1.5×10^{-6}, constant			
$\text{clip}_{\text{low}}/\text{clip}_{\text{high}}$	0.2 / 0.28			
KL penalty	disabled			
Advantage estimation	candidate-token rewards used directly as advantages			
Prompt limit	1,024	2,048	2,048	2,048
Response limit	16,384	16,384	1,024	16K (math/code) / 2K (IF)
Total steps	600			
Reward form	single-teacher dense reward			multi-teacher hard-routed dense reward
top-k	16, from the rollout student distribution (`only_stu`)			
Reward weighting	weighted by student probability $\tilde{\pi}_\theta$ (Eq. 9)			
share target $g^\star$	—	—	—	$(1/3, 1/3, 1/3)$
gap-following α	—	—	—	1.0 (m_d/m_{ref} forward)
reward refresh	enabled			enabled

rize the candidates at each response token by

$$r_t = \sum_{v \in \mathcal{S}_t} r_t(v). \tag{10}$$

The candidate-token rewards $r_t(v)$ serve as advantages in Eq. 1, with $\rho_t(v) = \pi_\theta(v \mid x, y_{<t})/\pi_{\theta_{\text{old}}}(v \mid x, y_{<t})$. Clipping is applied to each candidate before summing its loss over $\mathcal{S}_t$. The bounds are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. The denominator remains fixed throughout the inner updates; gradients flow only through current-student probabilities in the ratios.

Dual clipping and loss aggregation. The implementation additionally applies dual clipping to negative advantages. Writing $q_t(v)$ for the candidate loss inside the sum in Eq. 1, including its minus sign, the implemented candidate loss is

$$\ell_t(v) = \begin{cases} \min\{q_t(v), -c\, r_t(v)\}, & r_t(v) < 0, \\ q_t(v), & r_t(v) \geq 0, \end{cases} \qquad c = 3 \text{ (implementation default).} \tag{11}$$

For numerical stability, the log-ratio is clipped to $[-20, 20]$ before exponentiation. Candidate losses are summed, then averaged over valid response tokens with domain weights w_d. The positive domain weight can equivalently multiply each candidate advantage before clipping. Padding and prompt tokens are excluded.

A.5 Mechanism Implementation Details

Reward-magnitude statistics and domain weights. The reward-magnitude statistic m_d is recomputed from the current rollout batch. For outer step n, let $\mathcal{T}_{d,n}$ contain the valid response-token positions of domain d in that batch. Using the detached rollout-time token-level summary $r_{b,t}^{\text{roll}}$ from Eq. 10, we compute

$$m_{d,n} = \frac{1}{|\mathcal{T}_{d,n}|} \sum_{(b,t) \in \mathcal{T}_{d,n}} \left| r_{b,t}^{\text{roll}} \right|. \tag{12}$$

Thus, candidate-token rewards are summed before taking the absolute value; padding and prompt tokens are excluded. There is no EMA, temporal averaging window, or warm-up initialization: the first batch uses its own mean. We compute the domain weights once per outer step and hold them fixed during its inner updates, including when reward refresh is enabled. Below we suppress the outer-step index n.

Let $\mathcal{D}_n$ be the domains with valid response tokens in the current batch. The reference is the share-correction-weighted mean

$$m_{\text{ref}} = \frac{\sum_{d \in \mathcal{D}_n} w_d^{\text{share}} m_d}{\sum_{d \in \mathcal{D}_n} w_d^{\text{share}}}. \tag{13}$$

Target shares are renormalized over $\mathcal{D}_n$, and all domain sums in the weighting rule use this set. For numerical stability, both magnitudes in the ratio are floored at 10^{-12}. All statistics and domain weights are detached.

Reward refresh and gradient handling. The candidate set $\mathcal{S}_t$ itself is fixed at its rollout-time value throughout the K inner updates. We compute the routed teacher's log-probabilities for the sampled responses once, and recompute the student-dependent term with the current student before each inner update, as in Eq. 6.

The stop-gradient covers both the log-probability difference and the softmax weight. The refreshed reward is a fixed advantage for that inner update; only the current-policy ratio in Eq. 1 carries gradients, while its rollout-policy denominator stays fixed. At $K = 1$, Eqs. 9 and 6 give the same reward.

A.6 The Complete Algorithm

Algorithm 1 summarizes the complete training loop. Teacher log-probabilities are computed once for each rollout, while the student-dependent reward is recomputed for each inner minibatch.

Algorithm 1 Open-MOPD training loop

Require: Mixed-data sampler, student π_θ, teachers $\{\pi_{\phi_d}\}_{d\in\mathcal{D}}$, inner-step count K
Ensure: Updated student π_θ

for each training step t **do**
  Snapshot $\theta_{\text{old}} \leftarrow \theta$ and sample rollout batch $\mathcal{R}_t$.
  Cache rollout candidate sets and their student log-probabilities.
  Compute teacher log-probabilities $\ell^{(b)}_{\phi,d(x^{(b)})}$.
  Compute response-token shares s_d^{tok} and weights w_d^{share} using Equation 4.
  Compute detached rollout rewards and current-batch means m_d using Equation 12.
  Compute m_{ref} using Equation 13 and gap weights w_d using Equation 5; hold w_d fixed within this rollout.
  for each inner minibatch $\mathcal{M}_{t,k}$, $k = 0, \ldots, K-1$ **do**
    Recompute current-student log-probabilities and refresh the detached $r_t^{(k)}(v)$ using Equation 6.
    Update θ with the PPO objective using candidate advantages $w_d r_t^{(k)}(v)$.
  end for
end for

B Additional Diagnosis and Mechanism Details

B.1 Additional training-imbalance measurements

Token imbalance across domains. Figure 3 shows that the prompt and token shares are completely decoupled along the whole trajectory. The prompt share is fixed by the sampler at 39.8%/39.8%/20.3%, whereas the token share is 49.7%/49.3%/0.99% (Figure 3(a)). Across all 300 steps, the token share of IF never exceeds 1.65% and never falls below 0.44%, so this is not an incidental property of a particular batch.

This discrepancy is explained by response length (Figure 3(b)): the average response length of math and code is about 10,500 tokens, while that of IF is only 409 tokens, a more-than-25-fold difference. The response-token share is therefore almost entirely determined by length. This also shows the cost of equalizing token shares through sampling. With unchanged response lengths, the unnormalized weights are $a_d = (1/3)/s_d^{\text{tok}}$, including $a_{\text{IF}} = 33.6$. If p_d denotes the original prompt share, the new share is $p'_d = p_d a_d / \sum_j p_j a_j$. Thus, 33.6 is the IF sampling weight; normalization determines its actual prompt share. This concentrates a fixed prompt batch on IF, leaving fewer math and code prompts for long-chain supervision. Figure 3 is in Section 3.2.

Reward magnitudes also affect the update budget. Figure 4(a) and Equation 3 show how the reward magnitude changes during training. Early in training, $\bar{m}_d$ is 0.019 for math, 0.063 for code, and 0.091 for IF. The largest value is 4.9 times the smallest. We use $\bar{m}_d$ as a proxy for the remaining distillation signal. Its observed decrease differs across domains. Along the same trajectory, IF shrinks by 2.4 times, math by 2.1 times, and code by 1.9 times (Figure 4(b)). Here $\bar{m}_d$ is the average per-token reward magnitude for domain d. It is the expectation of $|r_t|$ over the tokens in that domain and enters the update-budget proxy in Equation 3.

The effect of the reward magnitude becomes clear after the token shares are balanced. When $w_d^{\text{share}} s_d^{\text{tok}} = 1/3$ for every domain, the training contribution of each domain depends mainly on $\bar{m}_d$ in Equation 3. Within 25 steps, the budget share of IF drops from 48.7% to around 9%, ending at 11.4%, while that of code rises from 39.6% to 63.8% (Figure 4(c)). Thus, balancing the token counts does not keep the training contributions balanced in later steps because the reward magnitudes differ across domains. Figure 4 is in Section 3.2.

Table 8: **Summary of the three Open-MOPD mechanisms.** Each mechanism modifies a different part of the training computation.

Mechanism	Measurement	Intervention
token-share balancing	Response-token share s_d^{tok} in the current batch	Set $w_d^{\text{share}} = g_d^\star / s_d^{\text{tok}}$ to equalize the domains' token shares in the loss.
gap-following allocation	Current-rollout mean reward magnitude m_d for each domain	Multiply the share weight by the clipped factor $(m_d/m_{\text{ref}})^\alpha$, giving more weight to domains with a larger current reward signal.
reward refresh	Student log-probabilities at each inner update	Recompute the student-dependent reward using the teacher log-probabilities computed for the rollout.

Multiple inner updates make the reward stale. To reduce the cost of generation, practical training usually performs K inner updates on one large rollout batch. The teacher remains fixed, while the student changes after the first inner update. If later updates still use the student probabilities computed during rollout, $\log \pi_\theta(v \mid x, y_{<t})$ and $\tilde{\pi}_\theta(v \mid x, y_{<t})$ $(v \in \mathcal{S}_t)$, then the student-dependent part of Equation 9 is inconsistent with the current policy.

Figure 5 reports top-k log-probability drift between the rollout and current policies, as defined in Appendix B.4. Across the plotted configurations, it rises from 0 at $K = 1$ to 0.059 at $K = 4$ and 0.216 at $K = 32$ (blue curve, left axis). The candidate clip fraction rises from 0 to 0.86 (red curve, right axis). These measurements show that candidate probabilities change during repeated updates, motivating recomputation of the student-dependent reward. Figure 5 is in Section 3.2.

Reward refresh introduces no measurable timing overhead. Reward refresh addresses the staleness above by reusing the actor forward already required by PPO, so it adds no extra teacher forward passes (Section 5.2). We verify this directly with a with/without-refresh timing comparison. On steady training steps at $K{=}4$ (steps 25–199, excluding warm-up and validation steps; medians over 168 paired steps), the dense-reward prefill takes **27.3 s with reward refresh enabled and 27.8 s without it**, i.e. 2.10% and 2.12% of one outer step (Figure 8). The full outer-step wallclock is **1298 s with refresh and 1313 s without it**, a -1.2% difference that is negative in sign and smaller than the step-to-step variation, so it cannot be attributed to reward refresh. This follows directly from what refresh changes: without it, the dense-reward prefill runs once over the whole rollout batch; with it, the prefill runs once per inner minibatch instead. Since the K minibatches partition the same rollout batch, every token is prefilled exactly once per outer step under either scheme — refresh changes only the student parameters at which that prefill is evaluated, not how many times it runs.

B.2 Teacher-conflict interventions

We tested whether directly changing high-conflict tokens improves training. We used two interventions. A conflict mask removes the top 1%, 5%, or 20% of tokens by c_t from the distillation loss, using thresholds of 0.83, 0.49, and 0.22 nat. We rescaled the remaining weights to keep the total loss scale fixed. A consensus target replaces the hard-routed teacher target with the average of the three teachers' log-probabilities when $c_t \leq 1$; it keeps the routed teacher when $c_t > 1$ (purple points in Figure 2(c)).

Under the same settings, the three conflict masks reduce the total score by 0.52, 0.73, and 0.75 points, respectively, relative to the baseline. The consensus target reduces it by 0.83 points. High-conflict tokens may be irrelevant noise, or they may carry domain-specific information. Token-level disagreement alone cannot tell these cases apart. This may explain why removing or replacing such tokens hurts performance.

B.3 Summary of Open-MOPD mechanisms

Table 8 summarizes the three mechanisms and the quantities they modify. Together, they change token weighting, domain weighting, and reward evaluation in the shared student.

B.4 Within-Batch Policy-Drift Diagnostics

Figure 5 reads the implementation's `actor/ppo_kl` and `actor/pg_clipfrac` metrics. In the top-k branch, the former averages $D_t = -\sum_{v \in \mathcal{S}_t} \text{clip}(\log \pi_\theta(v) - \log \pi_{\theta_{\text{old}}}(v), -20, 20)$ over valid response positions. This is an unweighted sum of candidate log-probability changes; it has no probability weighting and is not a KL divergence. The clip fraction averages the indicator $-\text{clip}(\rho_t(v), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})r_t(v) > -\rho_t(v)r_t(v)$ over valid response positions and candidates. It records when ordinary PPO clipping changes the candidate loss, before the additional dual clip.

Each point summarizes available logged measurements through outer step 24; its error bar is the sample standard deviation divided by the square root of the measurement count. The sweep includes refresh-enabled configurations at $K = 2, 32$ and stale-reward configurations at $K = 4, 8, 16$; at $K = 1$ the initial reward is current. These measurements describe within-batch drift across these configurations. The controlled refresh comparison is the $K = 4$ ablation in Table 2.

B.5 Additional ablation trajectories

Figure 7 and Figure 8 are in Section 5.2 and Section 5.2, respectively.

Gap-following allocation shifts the token-share curve. From the first 25 steps to the [200, 300] window, the weighted token share $w_d s_d^{\text{tok}}$ moves with the teacher–student gap (Figure 7(a)): the math curve rises from 15.1% to 32.6%, and the IF curve contracts from 34.5% to 17.1%. In a separate middle window, outer steps [25, 175], code has a mean weighted token share of 55.4%; its peak over the full trajectory is 87.6%. The token-share-balancing-only diagnostic run provides a complementary view of the allocation signal: over its middle window, the per-domain reward magnitudes are 0.028 for code, 0.008 for math, and 0.003 for IF. Their ordering motivates assigning more budget to code when gap-following allocation is enabled. The combined run computes its weights from its own current-batch statistics using the reference in Eq. 13.

Reversing the gap factor collapses training. We also test the inverse formulation, which uses $\tilde{w}_d = w_d^{\text{share}} / m_d^{0.5}$ so that a *smaller* reward magnitude receives *more* budget (Figure 7(b)). This division branch uses the same final token-weight normalization, with no gap-factor clamp; it skips the division when $m_d \leq 10^{-8}$. Under this reversed rule, the mean IF domain weight rises from 24.4 over outer steps 1–5 to 80.9 over steps 55–74. The mean reward magnitude $\bar{m}_d$ falls from 0.08047 to 0.002475 over these respective windows, a factor of 32.5. The run collapses at step 74. The endpoint comparison in Section 4.2 uses individual steps 1 and 74 from this same run; the figures here average the early and late windows. This is the over-allocation failure mode already motivated in Section 4.2: as the IF reward signal shrinks, the inverse rule keeps diverting more budget to that domain.