MOPD: Multi-Teacher On-Policy Distillation for Capability Integration in LLM Post-Training

Wenhan Ma[1*] **Jianyu Wei**[2] **Liang Zhao**[2] **Hailin Zhang**[2] **Bangjun Xiao**[1,2]
Lei Li[2,3] **Qibin Yang**[1,2] **Bofei Gao**[1,2] **Yudong Wang**[1,2]
Rang Li[1,2] **Jinhao Dong**[4,2] **Zhifang Sui**[1†] **Fuli Luo**[2†]

[1]Peking University [2]LLM Core, Xiaomi
[3]University of Hong Kong [4]Renmin University of China

Abstract

Modern large language models (LLMs) rely on reinforcement learning during post-training to push specific capabilities, yet integrating multiple capabilities into one model remains hard. Existing methods, such as Off-Policy Finetune and Mix-RL, are either inefficient or lose performance. In this work, we propose **Multi-teacher On-Policy Distillation** (MOPD), a post-training paradigm for combining the capabilities of multiple domain RL teachers: we first run per-domain specialised RL to obtain a set of domain teachers, then distill these teachers into the student on its own rollouts. This eliminates exposure bias and provides a dense optimization signal. On Qwen3-30B-A3B, MOPD outperforms Mix-RL, Cascade RL, Off-Policy Finetune, and Param-Merge baselines, inheriting nearly all of each teacher's capability. MOPD also enables parallel, independent development of domain teachers, removing the cross-domain coupling typical of multi-domain post-training. MOPD has been deployed in the post-training of MiMo-V2-Flash, an industrial-scale frontier model, demonstrating its practical value for capability integration in frontier-scale LLMs.

1 Introduction

Reinforcement learning (RL) has emerged as an important method for improving the capabilities of large language models (LLMs) (Ouyang et al., 2022; Schulman et al., 2017; Shao et al., 2024). Different task domains rely on different RL pipelines. Mathematical reasoning is now trained with verifiable-answer RL (Shao et al., 2024; Guo et al., 2025); software engineering with agent-style RL in executable sandboxes (Wei et al., 2025; Jain et al., 2025); instruction following and creative writing with rubric-based RL (Ouyang et al., 2022); and search-oriented agents with web-environment RL (Nakano et al., 2022; Jin et al., 2025). Each pipeline reliably improves the model's capability on its target domain. However, what we ultimately want is a single model that performs well across all of these domains. Yet building such a model remains an open problem in modern LLM post-training.

Prior approaches to this capability-integration problem fall into four families (Table 1). (1) Mix-RL (Yang et al., 2025; DeepSeek-AI et al., 2025) pools prompts from every domain into one dataset and runs joint RL, but cross-domain training signals interfere, producing the see-saw effect (Standley et al., 2020) and leaving the joint model below per-domain teachers. (2) Cascade RL (Wang et al., 2026) trains domains sequentially, so earlier-stage capabilities may decay as later stages progress and the long overall run amplifies stability risks. (3) Off-Policy Finetune (DeepSeek-AI et al., 2025) trains per-domain RL teachers and then fine-tunes

[*]Work done during an internship at Xiaomi.
[†]Corresponding authors: Fuli Luo (luofuli@pku.edu.cn), Zhifang Sui (szf@pku.edu.cn).

Reinforcement Learning from World Feedback (RLxF) Workshop, ICML 2026

Table 1: Capability-integration paradigms compared across three axes that matter in practice. MOPD is the only method that simultaneously attains dense optimization, on-policy training, and a parallelisable pipeline.

	Dense optimization	On-policy	Parallelisable
Param-Merge	×	—	✓
Off-Policy Finetune	✓	×	✓
Mix-RL	×	✓	×
Cascade RL	×	✓	×
MOPD (ours)	✓	✓	✓

the student on their rollouts via SFT; this off-policy supervision induces the canonical exposure bias (Ranzato et al., 2016). (4) Param-Merge (Wortsman et al., 2022; Ilharco et al., 2023) averages teacher weights or composes task vectors in weight space, but the fused model is typically unstable and fails to match all teachers' full capability. In short, these methods trade off learning efficiency, attainable peak, and training stability against one another, leaving no satisfactory middle ground.

To overcome these limitations, we propose **Multi-teacher On-Policy Distillation** (MOPD), a post-training paradigm that performs multi-teacher capability integration directly in policy space, rather than in weight space or in a static dataset space. MOPD first conducts specialised RL independently on each domain to obtain a collection of strong domain teachers. Then the capabilities of these teachers are distilled into a single student model via on-policy distillation (Hinton et al., 2015; Gu et al., 2026; Agarwal et al., 2024). This design has three structural benefits: (1) the training signal is dense and comes directly from each specialised teacher, so the student inherits the bulk of every teacher's capability; (2) training runs entirely on the student's own rollout distribution, which eliminates exposure bias by construction; (3) capability integration is realised in policy space through per-prompt routing rather than weight-space fusion, yielding a markedly more stable procedure.

We empirically validate MOPD on two base models of different scales and architectures. On Qwen3-30B-A3B (Yang et al., 2025), we compare Mix-RL, Cascade RL, Off-Policy Finetune, Param-Merge, and MOPD across three domains (Math, Instruction Following, and Software Engineering (Jimenez et al., 2024; OpenAI, 2024)); MOPD leads the strongest baseline by 5.5 points on the normalised score (0.937 vs. 0.882). We further deploy MOPD on the industrial-scale model MiMo-V2-Flash (Team et al., 2026), demonstrating its practical effectiveness in frontier-scale settings. Further analysis verifies the top-k distillation instantiation, confirms that same-origin teachers are critical for stable optimization, and demonstrates the benefits of multi-round iterative refinement.

Our contributions are summarised as follows. (i) We propose MOPD, a post-training paradigm that effects multi-teacher capability integration through dense, token-level supervision delivered on the student's own rollouts. (ii) We validate MOPD on Qwen3-30B-A3B, showing that it outperforms existing capability-integration baselines, and further apply it to the industrial-scale MiMo-V2-Flash to confirm practical viability.

2 Related Work

Reinforcement learning for LLM. In reinforcement learning with verifiable rewards (RLVR), PPO-style (Schulman et al., 2017) and GRPO-style (Shao et al., 2024) outcome-reward optimization has become standard practice. These RL pipelines span a diverse task landscape, including math reasoning (Guo et al., 2025; Shao et al., 2024), software engineering (Wei et al., 2025; Jain et al., 2025), instruction following and creative writing (Ouyang et al., 2022), and search agents (Nakano et al., 2022; Jin et al., 2025). Each such pipeline yields a strong domain expert, and consolidating them into a single model naturally raises the multi-domain / multi-task question. For this setting, prior work mainly takes two routes: mixed RL (Yang et al., 2025; DeepSeek-AI et al., 2025), which puts samples from different domains into the same training batch (each sample still uses its own per-domain reward and advantage), and cascade RL (Wang et al., 2026), which trains the domains one after another.

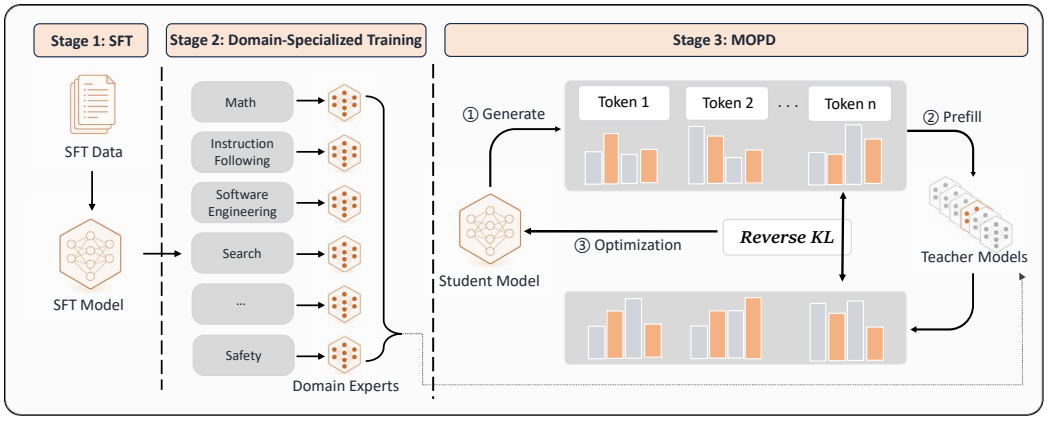

Figure 1: Overview of the three-stage MOPD pipeline. Stage 1 produces a shared SFT checkpoint; Stage 2 trains the per-domain teachers in parallel; Stage 3 repeatedly (i) samples a prompt from the domain mixture, (ii) routes it to the matching teacher, (iii) has the student roll out, (iv) prefills the teacher on the rollout to obtain token-level log-probabilities, and (v) updates the student using the teacher signal.

Distillation for LLMs. Classical distillation (Hinton et al., 2015; Kim and Rush, 2016) minimises a forward KL on a fixed corpus of teacher completions and is thus *off-policy*, so the student's inference-time trajectories can drift from the training distribution. On-policy distillation addresses this by training on the student's own rollouts with the teacher as a scoring signal, as in MiniLLM (Gu et al., 2026) (reverse KL) and teacher-as-reward variants (Agarwal et al., 2024; Meng et al., 2024). However, it is restricted to a single teacher and a single domain. MOPD keeps the "student samples, teacher scores" template and extends it to multiple, prompt-routed domain teachers, validated on production-scale base models.

Model merging. Model merging fuses multiple model checkpoints in weight space to obtain a combined model without additional training. Model Soups (Wortsman et al., 2022) averages the weights of independently fine-tuned models from the same initialisation. Task arithmetic (Ilharco et al., 2023) composes capabilities by adding and subtracting task vectors in weight space, while subsequent work mitigates parameter conflicts via TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), and AdaMerging (Yang et al., 2024).

3 Method

3.1 Pipeline

MOPD structures post-training as three sequential stages: general SFT, domain-specialised RL training, and a final MOPD pass that fuses the per-domain experts into a single model (Fig. 1).

- **Stage 1: General SFT.** We first fine-tune the base model on a broad corpus that covers all the base capabilities we care about, yielding an SFT checkpoint. This checkpoint serves as the shared initialisation for the domain-specialised RL training in Stage 2. And it is also used as the initialisation for the student of MOPD in Stage 3.
- **Stage 2: Domain-specialised RL training.** In this stage, for each domain d, we train a domain expert π_{ϕ_d} starting from the Stage-1 checkpoint. Each expert is trained with the RL recipe most natural for its domain, e.g., verifiable-answer RL for math and executable-sandbox agent RL for software engineering. The per-domain training are independent and can be executed fully in parallel.
- **Stage 3: MOPD.** The student is initialised from the Stage-1 SFT checkpoint, and the Stage-2 domain experts $\{\pi_{\phi_d}\}$ are frozen and form the teacher group; then we train the student on a multi-domain dataset, where each optimization step performs the following operations: (1) we sample a batch of prompts from the dataset; (2) the student generates a trajectory for each prompt and

records its per-token probability distribution along the trajectory; (3) each trajectory is dispatched to its corresponding domain teacher according to the task's domain, and then the teacher prefills on the trajectory to obtain its per-token probability distribution; (4) the student is updated by minimising the per-token reverse KL between the student and the domain teacher along the trajectory. After training, Stage 3 yields a single unified model that has acquired capabilities across all domains.

MOPD offers several structural advantages over existing capability-integration paradigms:

- **No exposure bias.** The student is trained on its own rollouts, so the training and inference state distributions match by construction.
- **Dense per-token supervision.** The teacher provides a probability distribution at every token of the trajectory. This is far denser than the trajectory-level reward used in standard RL, which lowers variance and improves sample efficiency.
- **Stable integration in policy space.** Capabilities are merged by routing each prompt to its matching teacher, not by averaging or task arithmetic on teacher parameters. This avoids the instability typical of weight-space fusion.
- **Modular and parallel teachers.** Teacher RL training runs fully in parallel, and each teacher can use its own hyperparameter configuration. An unexpected failure or hyperparameter adjustment in one teacher's RL training does not affect the training of other teachers.
- **Same-origin teacher stability.** Because each teacher is produced by domain-specific RL from the same SFT checkpoint that initialises the student, teacher and student share a closely aligned policy distribution. This ensures low initial KL divergence and smooth, stable optimization throughout distillation (§4.4.2).

3.2 Algorithm

In Stage 3, the student is optimised against a per-token reverse KL toward the dispatched teacher:

$$\mathcal{L}_{\text{rev-KL}} = \mathbb{E}_{x,\, y\sim\pi_\theta}\Big[\tfrac{1}{|y|}\sum_t \sum_v \pi_\theta(v)\log\tfrac{\pi_\theta(v)}{\pi_{\phi_d}(v)}\Big], \tag{1}$$

where π_{ϕ_d} is the teacher dispatched to prompt x, and $\pi_\theta(v)$, $\pi_{\phi_d}(v)$ are shorthand for $\pi_\theta(v \mid x, y_{<t})$, $\pi_{\phi_d}(v \mid x, y_{<t})$ (same shorthand applied below). We give two efficient implementations of this objective in the next two subsections.

3.2.1 Policy-gradient implementation

Following MiniLLM (Gu et al., 2026), we cast MOPD distillation as an RL process. The gradient of (1) is

$$\nabla_\theta \mathcal{L} = -\mathbb{E}_{x,y}\Big[\tfrac{1}{|y|}\sum_t \log\tfrac{\pi_{\phi_d}(y_t)}{\pi_\theta(y_t)}\nabla_\theta \log \pi_\theta(y_t)\Big], \tag{2}$$

which is exactly the policy-gradient form, with the teacher–student log-difference acting as a per-token *advantage*:

$$\hat{A}_{\text{MOPD},t} = \text{sg}[\,\log \pi_{\phi_d}(y_t) - \log \pi_\theta(y_t)\,]. \tag{3}$$

For training stability, we two-side-clip the advantage, $\hat{A}^{\text{clip}}_{\text{MOPD},t} = \text{clip}(\hat{A}_{\text{MOPD},t}, -A_{\max}, +A_{\max})$, giving the final loss

$$\mathcal{L}^{\text{PG}}_{\text{MOPD}}(\theta) = -\mathbb{E}_{x,y}\Big[\tfrac{1}{|y|}\sum_t \hat{A}^{\text{clip}}_{\text{MOPD},t}\,\log \pi_\theta(y_t)\Big]. \tag{4}$$

This form drops directly into existing PPO/GRPO training frameworks; the advantage computation is the only change.

3.2.2 Top-k distillation implementation

The policy-gradient form uses only the single sampled token at each rollout position. A lower-variance alternative is to distill on the teacher's top-k tokens (Peng et al., 2024), which exploits more of the

teacher's distribution while keeping the implementation simple. Let $\mathcal{T}_t^d = \mathrm{TopK}_k(\pi_{\phi_d}(\cdot \mid x, y_{<t}))$ be the top-k token set under the teacher at position t. We use the following distillation loss:

$$\mathcal{L}_{\text{MOPD}}^{\text{TopK}}(\theta) = \mathbb{E}_{x,y}\Big[\frac{1}{|y|}\sum_t \sum_{v\in\mathcal{T}_t^d}\big[\,\pi_\theta(v)\log\tfrac{\pi_\theta(v)}{\pi_{\phi_d}(v)} - \pi_\theta(v) + \pi_{\phi_d}(v)\big]\Big], \tag{5}$$

where $\pi_\theta(v)$ and $\pi_{\phi_d}(v)$ are both conditioned on $(x, y_{<t})$; we drop the conditioning for brevity.

We have added an extra $\pi_{\phi_d}(v) - \pi_\theta(v)$ term beyond the standard reverse KL. This added term corrects the bias introduced by top-k truncation, ensuring that the loss is minimised at $\pi_\theta = \pi_{\phi_d}$ on the top-k tokens; without it, the naive top-k-truncated reverse KL does not have this property.

Beyond correctness, the top-k form also reduces the infrastructure burden: it keeps the teacher-prefill payload small enough to be transmitted like a lightweight reward signal, in contrast to full-vocabulary distillation which requires shipping the entire vocabulary distribution per token.

3.3 Infrastructure

The additional operation in Stage 3 is the teacher prefill: every student rollout has to be passed through the dispatched teacher to obtain per-token log-probabilities (or top-k logits). A natural baseline is to fold teacher prefill into the RL training loop, but this complicates the infrastructure and adds extra serial latency. We observe that the teacher prefill computation has the same nature as reward computation in RL, so we instead deploy each domain teacher as a stand-alone prefill service sitting outside the RL trainer.

The runtime works as follows. The student sampler keeps generating rollouts; as soon as one sequence's rollout finishes, the trainer fires an asynchronous prefill request to the corresponding teacher service. The teacher service computes and returns the per-token log-probabilities. Because teacher prefill and other sequences' sampling overlap in time, the wall-clock cost of MOPD is dominated by sampling alone, and the teacher cost is essentially hidden behind student sampling. In our deployments, the teacher contributed nearly no measurable wall-clock overhead.

4 Experiments

We compare MOPD against four capability-integration paradigms on Qwen3-30B-A3B under controlled conditions (§4.1–4.2) and validate it on the industrial-scale frontier model MiMo-V2-Flash (§4.3). Finally, §4.4 presents our analysis.

4.1 Experimental setup

SFT model. Our starting checkpoint is Qwen3-30B-A3B-Base (Yang et al., 2025). To make it a strong starting point for RL, we fine-tune it on a broad-coverage corpus spanning all target domains. All baselines and MOPD start from this same SFT checkpoint.

Tasks and evaluation. Our tasks span three domains: math (evaluated on AIME25 and AIME26), instruction following (evaluated on IFBench (Pyatkin et al., 2025) and IFEval (Zhou et al., 2023)), and software engineering (evaluated on SWE-bench Verified (Jimenez et al., 2024)). Training data sources and hyperparameters are described in Appendix B.

Baselines. We compare five capability-integration paradigms: (i) Mix-RL, joint RL on a dataset that pools prompts from every domain; (ii) Cascade RL, per-domain RL applied sequentially with each stage initialised from the previous; (iii) Off-Policy Finetune, offline SFT of the student on rollouts produced by the Stage-2 teachers; (iv) Param-Merge, linear / task-arithmetic merge of the Stage-2 teachers; and (v) MOPD (ours).

Evaluation metric. The three domains differ substantially in absolute headroom, so a raw accuracy average over-weights the wider-headroom domains. To put the domains on a common scale we report a normalised score. Let $\mathcal{D}$ denote the set of evaluation domains. For each $d \in \mathcal{D}$, let s_d^{s} be the Stage-1 SFT student accuracy on that domain and s_d^{t} the corresponding Stage-2 per-domain specialist accuracy; for a method with domain accuracy s_d, the per-domain normalised

Table 2: Qwen3-30B-A3B: multi-domain capability integration. Per-benchmark columns are accuracy (%); higher is better. "RL Teacher" is the Stage-2 specialised RL teacher for that domain. The **Norm. score** column is the normalised score defined in §4.1; higher is better. Within the six integration methods we **bold** the column best and underline the runner-up.

	Math		Instruction Following		SWE	
Method	AIME25	AIME26	IFBench	IFEval	SWE-bench Verified	Norm. score
Student (SFT-only)	45.42	54.48	42.69	84.17	35.80	0.0000
RL Teacher	54.79	63.65	78.40	95.50	51.20	1.0000
Mix-RL	**52.71**	63.75	75.00	94.58	48.80	0.8818
Cascade RL	48.54	61.88	77.11	95.80	47.80	0.7752
Off-Policy Finetune	51.56	63.44	**80.95**	93.35	45.80	0.8241
Param-Merge (Avg.)	47.81	59.58	53.74	88.79	39.60	0.3280
Param-Merge (Task Arith.)	49.38	63.96	78.23	**95.81**	48.80	0.8574
MOPD (ours)	51.46	**65.31**	77.89	93.84	**50.40**	**0.9373**
Δ(MOPD − RL Teacher)	−3.33	+1.66	−0.51	−1.66	−0.80	−0.0627

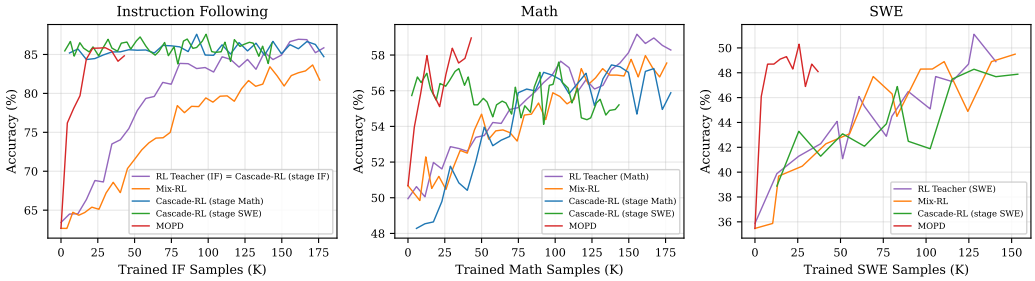

Figure 2: Training dynamics on Qwen3-30B-A3B. Each panel is a domain-level accuracy (%); the x-axis is cumulative training samples (K). Domain-RL (purple) is the single-domain specialist teacher for reference.

score is $\tilde{s}_d = (s_d - s_d^{\mathrm{s}})/(s_d^{\mathrm{t}} - s_d^{\mathrm{s}})$, which equals 0 at the Stage-1 student and 1 at the per-domain specialist teacher. The headline figure we report is the uniform average $\bar{\tilde{s}} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \tilde{s}_d$ across domains; values above 1 indicate improvement beyond the specialist teacher, values below 0 indicate a regression from the Stage-1 student.

4.2 Main results

Table 2 reports per-benchmark accuracy and the normalised score across all three domains. Figure 2 tracks domain-level accuracy against per-domain training samples. We make four observations.

- **Mix-RL, Cascade RL, and Off-Policy Finetune each leave domain-specific gaps, in different shapes.** The three methods reach normalised scores of 0.882, 0.775, and 0.824 respectively, similar in aggregate but with distinct per-domain weaknesses. Cascade RL (trained in order IF→Math→SWE) exhibits cross-domain interference: the first-trained IF domain closes 98% of the headroom, but Math (second stage) closes only 57%; Figure 2 further shows that Math accuracy degrades during the subsequent SWE stage. Off-Policy Finetune exceeds the teacher on IF (per-domain normalised score 1.01) yet closes only 65% on SWE, suggesting that offline imitation of teacher trajectories transfers unevenly across task types. Mix-RL is the most balanced baseline (per-domain range 0.064) but still trails MOPD by 5.5 points overall.
- **MOPD achieves the highest normalised score with the most uniform profile.** MOPD reaches a normalised score of 0.937, +0.055 over the next-best method (Mix-RL at 0.882). Its three per-domain normalised scores fall in $[0.91, 0.95]$, a range of only 0.044, the smallest of any method. No other approach closes the headroom this uniformly: Cascade RL spans 0.57–0.98 (range 0.41) and Off-Policy Finetune spans 0.65–1.01 (range 0.36).

Table 3: MiMo-V2-Flash: integration via the full MOPD pipeline. All teachers are RL-trained.

	AIME25	HMMT25	LCB	IFBench	SWE-Bench V.	τ^2-Bench	τ^2-Telecom
Student	89.3	76.9	77.5	55.4	67.8	75.9	92.7
Teacher	93.9	82.6	82.6	68.9	74.2	79.6	95.0
MOPD	94.1	84.4	83.2	66.7	73.4	80.3	95.3
Δ	$+0.2$	$+1.8$	$+0.6$	-2.2	-0.8	$+0.7$	$+0.3$

- **Param-Merge is highly sensitive to the merging recipe.** Linear averaging fails broadly (normalised score 0.328); Task Arithmetic recovers to 0.857, but its per-domain profile varies widely: it exceeds the teacher on IF (1.00) yet closes only 73% on Math. The outcome depends on both the merging coefficients and the benchmark, making parameter merging an unreliable integration tool.
- **MOPD is markedly more sample-efficient.** As Figure 2 shows (x-axis: per-domain samples consumed), MOPD reaches the teacher-level plateau on IF within $\sim$25K IF samples and on SWE within $\sim$30K SWE samples, whereas Mix-RL requires the full 150–180K sample budget *in each domain* to approach a comparable level. The dense per-token supervision from the teacher provides richer gradient signal per sample than trajectory-level RL reward, explaining the faster convergence.

4.3 Scaling to a 309B-parameter model

To confirm that MOPD carries over to industrial-scale frontier models, we apply the full three-stage pipeline to MiMo-V2-Flash. We use domain teachers covering Math, Code, IF, SWE, and Tool Use.

Table 3 presents the MOPD results on MiMo-V2-Flash. MOPD matches or exceeds the corresponding teacher on most benchmarks. The two regressions, IFBench (-2.2) and SWE-Bench Verified (-0.8), are modest relative to the gains on the remaining benchmarks.

4.4 Analysis

4.4.1 Policy-gradient and top-k distillation perform comparably

§3.2 introduced two instantiations of the MOPD distillation loss: the policy-gradient form (4) and the top-k form (5). We compare the two on Qwen3-30B-A3B under an otherwise identical pipeline (same teachers, data budget, and hyperparameters), with $k = 64$ for the top-k variant.

The top two rows of Table 4 report the comparison. Top-k distillation is comparable on Math and slightly worse on IF and SWE; the two forms reach overall similar capability, with normalised scores of 0.909 vs. 0.937. Figure 3 corroborates this: under same-origin teachers, both loss forms produce nearly overlapping training trajectories: math accuracy climbs smoothly, the per-token reverse KL decreases monotonically from an already low initial value (~ 0.04), and policy entropy remains stable around 0.30 throughout. When teacher and student share a closely related distribution, the student's rollouts concentrate in the teacher's high-probability region, so both gradient estimators receive similar informative signal and converge to comparable endpoints. This indicates that, with same-origin teachers, the policy-gradient form is already sufficiently stable; introducing top-k does not yield additional stability or variance reduction.

4.4.2 Same-origin teachers stabilize distillation

MOPD's domain teachers are obtained by RL training from the student. This means that teachers and the student share a similar policy distribution. Recent studies (Li et al., 2026; Ko et al., 2026) discuss the relationship between teacher–student policy divergence and on-policy distillation stability. A natural question arises: would replacing a teacher with a stronger but distributionally different external model yield better results? To test this, we design a controlled experiment: in the Math domain of MOPD, we replace the original Qwen3-30B-A3B Math RL teacher with Qwen3-235B-A22B, a significantly larger model with stronger math capability. All other settings (IF and SWE teachers, data, hyperparameters) remain unchanged.

The bottom two rows of Table 4 report the results. Despite Qwen3-235B-A22B's substantially stronger absolute math capability, the MOPD student's Math performance degrades under both loss variants. Figure 3 reveals the mechanism. The initial per-token KL is approximately $5\times$ higher than

Table 4: Distillation loss variants and teacher distribution alignment on Qwen3-30B-A3B. Per-benchmark columns are accuracy (%); the last column is the normalised score defined in §4.1. The top two rows use the RL teachers from the default MOPD pipeline; the bottom two rows replace the Math teacher with Qwen3-235B-A22B, a stronger but distributionally different model. Both loss variants use $k = 64$ for top-k.

		Math		Instruction Following		SWE	
Math Teacher	Loss Variant	AIME25	AIME26	IFBench	IFEval	SWE-bench Verified	Norm. score
RL Teacher	Policy gradient	51.46	65.31	77.89	93.84	50.40	0.9373
	Top-k distillation	51.77	64.79	75.85	93.07	50.20	0.9093
Qwen3-235B-A22B	Policy gradient	45.63	51.56	79.25	93.99	50.60	0.6003
	Top-k distillation	0.94	0.42	72.96	88.97	51.20	-1.1898

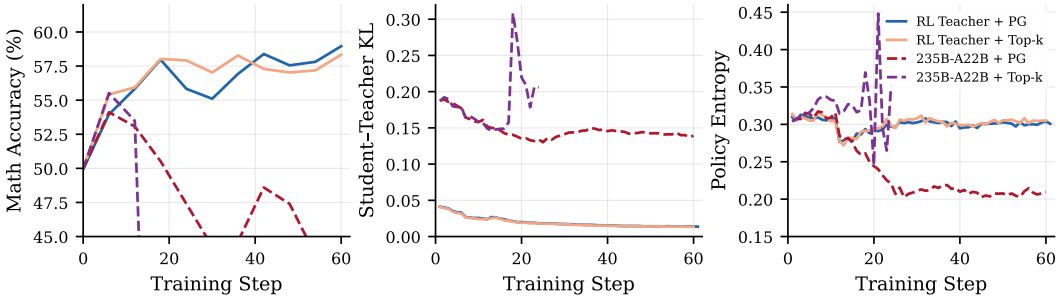

Figure 3: Training dynamics across the four runs in Table 4 on the Math domain of Qwen3-30B-A3B. **Left**: Math accuracy (%). **Middle**: per-token student–teacher reverse KL divergence. **Right**: student policy entropy. The two same-origin teacher runs (solid) show smooth, stable optimization; the two external-teacher runs (dashed) exhibit progressively unstable training, with the top-k variant collapsing catastrophically around step 18.

the same-origin setting ($\sim$0.19 vs. $\sim$0.04), quantitatively confirming the large distributional gap. Under the policy-gradient form, math accuracy degrades progressively while entropy contracts from 0.30 to 0.21; the student's policy narrows toward a single mode as it receives predominantly punitive gradient signal from the teacher's low-probability region. Under the top-k form, the collapse is even more severe: training diverges catastrophically around step 18, with wild oscillation in both KL and entropy, indicating a complete loss of optimization stability. These results underscore that MOPD's use of same-origin teachers is critical: the close distributional alignment ensures stable optimization throughout training, yielding reliable and consistent gains.

4.4.3 Multi-round student–teacher evolution

A single round of MOPD closes most of the student–teacher headroom on every domain, but Table 5 shows that headroom remains in the Math and IF domains. We propose using the post-MOPD model as a new student and repeating the procedure: retrain each per-domain teacher from this student, then perform another MOPD integration with the new teachers.

We validate one such follow-up round on Qwen3-30B-A3B. After the first round, the student becomes the initialisation for the next round's per-domain RL teachers, which drive the second MOPD pass. In this round, we apply RL training and MOPD distillation to the Math and IF domains only (for SWE, we perform neither RL training nor distillation).

We observe: (i) initialising the next round's per-domain RL from the Iter-1 student does produce stronger teachers: the Iter-2 RL Teacher reaches a normalised score of 1.030, +0.093 over Iter-1 MOPD; this confirms that per-domain capability headroom remains beyond what a single MOPD round extracts. (ii) Guided by the Iter-2 RL teachers, the Iter-2 student improves further on Math and IF, with its normalised score rising from 0.937 to 0.986 (+0.049), showing that the MOPD pipeline can continuously absorb capability from stronger teachers.

Table 5: Multi-round student–teacher evolution on Qwen3-30B-A3B. Per-benchmark columns are accuracy (%); the last column is the normalised score defined in §4.1. Iter-2 RL Teacher is the per-domain RL teacher initialised from the Iter-1 MOPD student; Iter-2 MOPD is the student produced from those teachers. No new SWE teacher was trained at Iter 2; the SWE scores reflect only indirect effects of Math/IF distillation.

	Math		Instruction Following		SWE	
Round	AIME25	AIME26	IFBench	IFEval	SWE-bench Verified	Norm. score
Iter 1 MOPD	51.46	65.31	77.89	93.84	50.40	0.937
Iter 2 RL Teacher	54.27	65.52	81.46	95.65	50.40	1.030
Iter 2 MOPD	53.44	64.90	79.76	95.44	50.20	0.986

5 Discussion

Beyond its empirical gains, MOPD structurally decouples the serial dependencies intrinsic to multi-domain post-training, separating capability production (Stage-2 per-domain RL) from capability integration (Stage-3 distillation). Three workflow consequences follow:

- **Parallel development.** Stage-2 teachers are mutually independent: each domain team iterates on its own rewards, sandboxes, and data pipelines concurrently, with no fixed ordering. Given sufficient compute, this directly raises the team's overall development throughput.
- **Recipe-level decoupling.** Each domain is free to choose its own RL recipe (algorithm, rollout procedure, reward function, hyperparameters, and so on) without worrying about conflicts with other domains or about its own optimization choices interfering with theirs.
- **Risk isolation.** RL training often requires iterative tuning of algorithms and hyperparameters (whether to push capability higher or simply to address stability issues), and each such adjustment forces a training restart. Under joint multi-domain RL, a restart sends the entire run back to the start; under MOPD's parallel teacher development, a restart is confined to the single affected domain and the other teachers continue undisturbed.

6 Conclusion

We proposed MOPD (Multi-Teacher On-Policy Distillation), a post-training paradigm that performs multi-domain capability integration in policy space: the student samples from its own rollouts, each prompt is routed to the corresponding frozen domain teacher, and the teacher provides a dense, token-level log-probability signal at every position. On Qwen3-30B-A3B, MOPD attains the best aggregate integration among Mix-RL, Cascade RL, Off-Policy Finetune, and Param-Merge, leading the next-best integration method by 5.5 normalised-score points while closing 91–95% of the student–teacher headroom on every domain; on MiMo-V2-Flash, the same recipe matches or exceeds the corresponding teacher on most benchmarks. Our analysis further shows that both policy-gradient and top-k loss forms perform comparably, that same-origin teachers are critical for stable optimization, and that the pipeline supports multi-round student–teacher evolution. Combined with the structural decoupling discussed in §5, we believe MOPD offers an engineering path for LLM post-training at scale that supports parallel development and rapid integration.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. *Preprint*, arXiv:2306.13649.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *Preprint*, arXiv:2502.17387.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2026. Minillm: On-policy distillation of large language models. *Preprint*, arXiv:2306.08543.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Preprint*, arXiv:2503.24290.

Hugging Face. 2025. Open R1: A fully open reproduction of DeepSeek-R1. `https://github.com/huggingface/open-r1`. Mixture-of-Thoughts dataset at `https://huggingface.co/datasets/open-r1/Mixture-of-Thoughts`.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. *Preprint*, arXiv:2212.04089.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. 2025. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *Preprint*, arXiv:2504.07164.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. *Preprint*, arXiv:1606.07947.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. 2026. Scaling reasoning efficiently via relaxed on-policy distillation. *Preprint*, arXiv:2603.11137.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. 2026. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *Preprint*, arXiv:2604.13016.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Preprint*, arXiv:2405.14734.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI. 2024. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/.

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

Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. 2024. Pre-training distillation for large language models: A design space exploration. *Preprint*, arXiv:2410.16215.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *Preprint*, arXiv:2507.02833.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *Preprint*, arXiv:1511.06732.

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

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

Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? *Preprint*, arXiv:1905.07553.

Core Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, Gang Xie, Hailin Zhang, Hanglong Lv, Hanyu Li, Heyu Chen, Hongshen Xu, Houbin Zhang, Huaqiu Liu, and 107 others. 2026. Mimo-v2-flash technical report. *Preprint*, arXiv:2601.02780.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2026. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *Preprint*, arXiv:2512.13607.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Preprint*, arXiv:2502.18449.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Preprint*, arXiv:2306.01708.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2024. Adamerging: Adaptive model merging for multi-task learning. *Preprint*, arXiv:2310.02575.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *Preprint*, arXiv:2311.03099.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

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

A Use of LLM

In this paper, the large language model was only used for polishing the text; it did not contribute to any research ideas or experimental results.

B Training Details

Training data. Our training tasks span three domains:

1. **Math**. SFT data come primarily from the math subset of the Mixture-of-Thoughts dataset (Hugging Face, 2025); RL training data are collected and filtered from many open-source datasets, including BigMath (Albalak et al., 2025) and ORZ (Hu et al., 2025). The maximum sequence length is 32,768.
2. **Instruction following**. SFT prompts are constructed following the method introduced by IF-Bench (Pyatkin et al., 2025) and distilled on gpt-oss-120b (OpenAI et al., 2025); RL training data are similarly synthesised following the IFBench recipe. The maximum sequence length is 32,768.
3. **Software engineering**. SFT data are distilled from R2E-Gym tasks via open-source models; RL training data use R2E-Gym-Lite (Jain et al., 2025). The maximum sequence length is 65,536 and the number of interaction turns is capped at 50.

Hyperparameters. For RL training, we use an on-policy GRPO (Shao et al., 2024) algorithm with Dynamic Sampling (Yu et al., 2025): data generated from each rollout is used for a single gradient update and then discarded. For per-domain RL, we set the learning rate to 3×10^{-6}. For Math and IF training, we set batch size (BS) to 144 with $N = 8$ rollouts per prompt, training approximately 175K sequences. For SWE, we set BS to 80 with $N = 8$, training approximately 150K sequences. Cascade RL uses the same configuration as per-domain RL, with domains trained in the order IF→Math→SWE. For Mix-RL, we set the learning rate to 4×10^{-6}, BS 256, $N = 8$, with a per-batch domain ratio of Math : IF : SWE $= 0.35 : 0.35 : 0.3$. For MOPD, we do not use Dynamic Sampling and set BS 2048, $N = 1$, with a per-batch domain ratio of Math : IF : SWE $= 0.35 : 0.35 : 0.3$. The default advantage clip maximum is 5 for the policy-gradient form; the default k is 64 for the top-k distillation form.

C Evaluation Details

We describe the evaluation protocol used for all reported results. For the two math benchmarks, AIME25 and AIME26, we sample 32 times per question and report the average accuracy (avg@32) to reduce the variance inherent to low-sample-count, high-difficulty math evaluation. For instruction following, we evaluate each question once on both IFBench and IFEval. For software engineering, we evaluate each task once on SWE-bench Verified.

For all benchmarks, decoding uses a sampling temperature of 1.0, with neither top-p nor top-k truncation applied.