SWIFT MOPD: A BETTER BALANCED METHOD FOR PARAMETER AND DATA ALLOCATION IN MULTI TEACHER ON POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-Teacher On-Policy Distillation (MOPD) aims to transfer complementary capabilities from multiple domain-specialized teachers into a single student policy, but directly applying OPD from a generic base model can lead to unstable optimization and uneven absorption of teacher knowledge across domains. In this work, we first analyze the difficulty of transferring multiple domain experts into a single student and show that different domains exhibit different fitting difficulty, which implies that multi-teacher distillation should carefully balance both data allocation and parameter allocation across teachers. Building on this analysis, we propose Swift-MOPD, a framework that assigns the initial merging coefficients according to per-task fitting difficulty, so that the student starts from a policy that preserves the capabilities of the harder-to-fit domains, and dynamically allocates more training data to under-fitted tasks during distillation. This enables a better initialization and substantially faster fitting of the student. Through comprehensive experiments on MOPD across four distinct domains, we derive two main findings: ❶ initializing the merged weights according to task difficulty coefficients effectively preserves the performance on harder-to-distill domains, approaching the level of the corresponding teachers; ❷ under the same training budget, while vanilla MOPD suffers from uneven training across domains, Swift-MOPD consistently recovers the performance of all domain experts. We hope our work offers new insights for future research on MOPD.

1 INTRODUCTION

Recently, on-policy distillation (OPD) (Agarwal et al., 2024b; Gu et al., 2024a; Lu & Lab, 2025; Jin et al., 2026; Xing et al., 2026; Hao et al., 2026; Xiao et al., 2026b; Li et al., 2026a) has emerged as an effective paradigm for transferring capabilities from a teacher model to a student policy, by aligning the student with teacher token distributions on the student's own generated trajectories. Compared with off-policy distillation on a fixed dataset (Hinton et al., 2015; Kim & Rush, 2016; Gu et al., 2023; Xu et al., 2024), OPD mitigates the train–test mismatch and provides dense token-level supervision, improving calibration and post-training efficiency (Ko et al., 2024; 2025; Shenfeld et al., 2026). Building on this success, multi-teacher on-policy distillation (MOPD) (Ma et al., 2026; Yin et al., 2026; Lian et al., 2026; Song & Zheng, 2026; Thinking-Space, 2026b) has been proposed to consolidate multiple domain-specialized teachers into a single deployable student.

However, extending OPD to the multi-teacher setting is far from trivial. MOPD has become a popular recipe for *model consolidation*: a single base model is specialized into multiple domain experts via domain-wise post-training, and their capabilities are then merged back into one deployable model. Yet recent analyses (Li et al., 2026b; Fang et al., 2026; Thinking-Space, 2026a) reveal that such consolidation yields a mixture-dependent seesaw among expert capabilities, making MOPD highly sensitive to training-data composition. While existing methods (Ma et al., 2026; Yin et al., 2026; Wang et al., 2026) have advanced the integration of teacher capabilities through improved objectives and supervision signals, a complementary design choice remains overlooked: **the initialization of the student policy**. All existing approaches start from a base model, overlooking initialization as a design choice. We argue that a well-initialized student can preserve harder-to-distill capabilities from the beginning.

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.

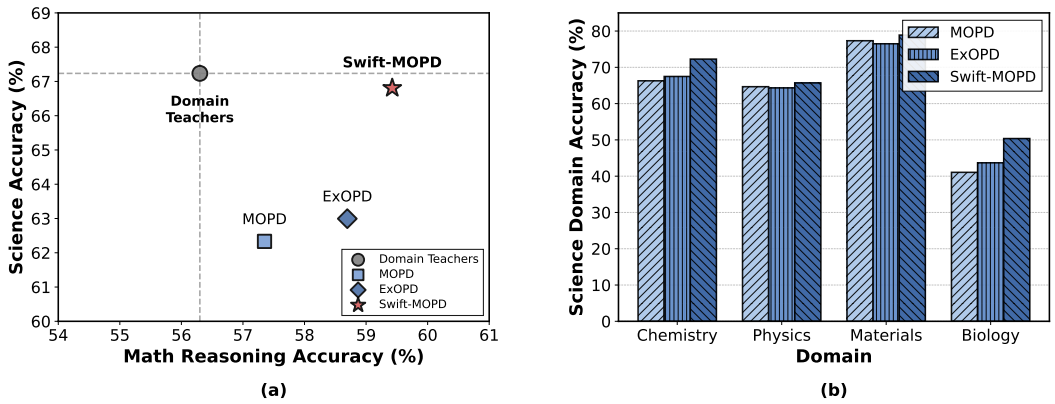

Figure 1: (a) **Multi-teacher distillation results under the four-teacher setting (Code, Math, Science, and planning experts) , shown on Math and Science.** The gray dashed lines mark the performance of domain-specialized teachers. `Swift-MOPD` recovers performance closest to the domain teachers. (b) **Per-domain performance on the four Science sub-tasks under the same four-teacher setting**. `Swift-MOPD` achieves the most balanced recovery across all four domains, with the largest gain on the hardest domain (Biology).

Specifically, we first analyze MOPD under uniform data allocation and find that harder-to-distill domains suffer from incomplete fitting due to imbalanced learning progress across tasks. We then propose `Swift-MOPD`, which initializes the student by merging domain-specialized teachers with weights proportional to per-task fitting difficulty, thereby preserving the performance of harder-to-fit domains from the start. This better initialization enables us to allocate more training data to under-fitted experts during distillation, achieving faster and more balanced MOPD. We evaluate `Swift-MOPD` on four domains—Math, Code, Science, and Planning—under a four-teacher distillation setting. As shown in fig. 1 and experiments, `Swift-MOPD` recovers performance closest to the domain teachers, while `vanilla MOPD` and `ExOPD` lag behind on the harder-to-distill Science domain, confirming that difficulty-aware initialization enables more balanced capability integration. The key contributions of our paper are as follows:

- We identify initialization as a blind spot in MOPD through a controlled analysis showing that uniform data allocation leads to incomplete fitting of harder-to-distill domains.
- We provide insights on how to initialize the student policy and allocate training data across teachers in MOPD, demonstrating that difficulty-aware initialization and dynamic data reallocation are critical for balanced capability integration.
- We validate `Swift-MOPD` on 4 different domains, demonstrating that `Swift-MOPD` recovers performance closest to domain teachers and achieves the largest gains on the hardest sub-tasks. Quantitatively, Swift-MOPD attains a normalized recovery rate of 1.0185.

2 Methodology

2.1 Preliminary

On-Policy Distillation. On-Policy Distillation (OPD) (Agarwal et al., 2024a; Gu et al., 2024b; Lu & Lab, 2025) distills knowledge from a teacher model using trajectories sampled from the current student policy. Different from off-policy distillation, where supervision is provided on a fixed dataset, OPD evaluates the teacher on student-generated responses and therefore better matches the distribution encountered by the student during optimization. Given an input $\boldsymbol{x} \sim \mathcal{D}$ and a response $\boldsymbol{y} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})$, the student is optimized by minimizing the reverse KL divergence to the teacher policy π^*:

$$\mathcal{J}_{\mathrm{OPD}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \left[\mathcal{D}_{\mathrm{KL}} \left(\pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \,\|\, \pi^*(\boldsymbol{y} \mid \boldsymbol{x}) \right) \right]. \tag{1}$$

This objective provides dense token-level supervision on the student's own trajectories, making OPD an effective post-training paradigm for transferring teacher behavior to the student policy.

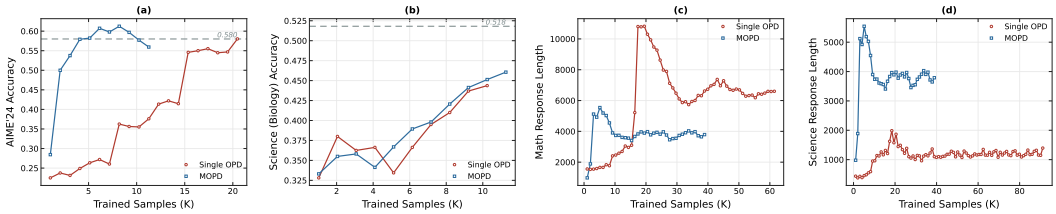

Figure 2: (a)–(b) **Validation accuracy during distillation under the four-teacher setting , comparing single-teacher OPD and MOPD on Math and Science.** The gray dashed lines mark the performance of domain-specialized teachers. (c)–(d) **Dynamic of the student response length throughout training on Math and Science.**

Multi-Teacher On-Policy Distillation. While standard OPD assumes a single teacher, practical post-training often involves multiple specialist teachers, each optimized for a different domain or capability, such as mathematical reasoning, code generation, scientific QA, or planning. Directly distilling from a single teacher can bias the student toward one domain and fail to exploit the complementary strengths of other teachers. To address this, Multi-Teacher OPD extends the OPD objective by assigning each training sample to a corresponding domain teacher. Let $\{\pi_k^*\}_{k=1}^K$ denote a set of specialist teachers, and let $\tau(\boldsymbol{x}) \in \{1, \ldots, K\}$ be a routing function that selects the teacher associated with input $\boldsymbol{x}$. The multi-teacher objective can be written as:

$$\mathcal{J}_{\text{MOPD}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \left[\mathcal{D}_{\text{KL}} \left(\pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \,\middle\|\, \pi^*_{\tau(\boldsymbol{x})}(\boldsymbol{y} \mid \boldsymbol{x}) \right) \right]. \tag{2}$$

In this formulation, the student still learns from its own on-policy trajectories, but the teacher signal is selected according to the domain of each sample. This enables the student to absorb heterogeneous capabilities from multiple expert policies under the distillation process.

2.2 The Characteristics of Fitting Difficulty in Teacher Training and Student Distillation

Prior studies on OPD mainly focus on enabling students to learn from teachers more efficiently, through better objectives, training signals and so on. In this work, we revisit MOPD from a different perspective and ask:

> *Is there a correlation between the fitting difficulty required to train a teacher model and the difficulty for a student model to distill toward that teacher's performance level?*

Answering this question is important because it provides a principled way to allocate training data, balance teacher contributions, and design distillation strategies in the MOPD process.

Pilot Experiment and Metrics. We study four representative domains: **Math**, **Code**, **Science**, and **Planning**. For each domain, we train a specialized teacher and then separately distill a student from that teacher. We primarily conduct experiments using the Qwen3-4B-Non-Thinking (Yang et al., 2025) and following the same training and testing set from `EXOPD` (Yang et al., 2026) and `OPSD` (Hübotter et al., 2026a). Detailed training hyper-parameters are in Appendix A.

Observation. We then track the accuracy and the student response length throughout training, and compare single-teacher OPD against MOPD under an identical per-domain data budget . The two differ only in batch composition: each training batch of single-teacher OPD consists purely of samples from one domain, whereas an MOPD batch mixes samples from all domains. From these training dynamics, we draw two takeaways as follows.

Takeaway 1

Mixed training converges faster and yields more concise responses than separate single-teacher distillation.

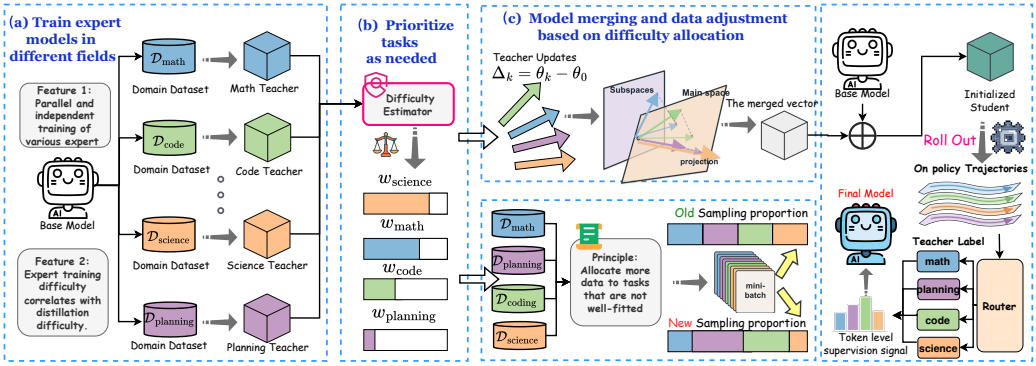

Figure 3: **Overview of the `Swift-MOPD` pipeline.** Domain expert teachers are first trained independently (a), and difficulty estimators assigns per-domain weights (b). Guided by this difficulty, `Swift-MOPD` proceeds in two steps: (i) *difficulty-aware initialization*, which merges teacher updates into a better starting point for the student; and (ii) *difficulty-aware data allocation*, which shifts the sampling proportion toward under-fitted domains (c).

Although trained on a mixed pool of all four domains, MOPD matches the teacher's performance on AIME'24 with substantially fewer mixed samples, whereas single-teacher OPD improves far more gradually and only reaches 0.58 after consuming over 20K samples (fig. 2a). Meanwhile, the student response length under MOPD stabilizes early at 3.7K tokens, while single-teacher OPD inflates to over 10K tokens before settling (fig. 2c).

Takeaway 2

Uniform data allocation yields uneven recovery across domains.

Under a uniform allocation, every domain receives an identical share of each batch, yet the resulting recovery is far from uniform. As shown in fig. 2a, Math is fitted well, with MOPD matching the teacher's 0.58 using only a fraction of the budget. In contrast, the Science domain—which itself spans heterogeneous sub-tasks (Chemistry, Physics, Materials, Biology) that are inherently harder to fit—recovers much more slowly: Biology climbs only from 0.33 to 0.46, trailing the teacher's 0.518 ceiling by $5.8\% \downarrow$ even after the full budget (fig. 2b). These domains are thus harder to fit, and uniform allocation only widens the gap.

2.3 Swift-MOPD: Better Initialization and Data Allocation

The above analysis shows that uniform allocation under-fits harder domains. A natural remedy is to re-weight the batch composition during training, but this requires periodically extracting checkpoints and re-evaluating each domain to adjust the data ratio. We instead ask whether the problem can be defused **before** training begins: if the student is initialized from a point that already preserves the harder domains' capabilities, a simple uniform allocation no longer starves them. Building on this insight, we propose `Swift-MOPD`, a difficulty-aware framework for MOPD. Since different teachers are not equally easy to obtain, `Swift-MOPD` leverages this teacher-side fitting difficulty to guide both **parameter-space initialization** and **data-space allocation**, as shown in figure 3.

2.3.1 Difficulty-Aware Initialization via Subspace Projection

The first step of `Swift-MOPD` constructs a better starting point for the student by merging the domain teachers in parameter space. The central design is to *protect the hardest-to-distill domain* while letting the easier ones contribute only insofar.

Main and auxiliary spaces. We designate the teacher of the hardest-to-fit domain as the *subject*, whose task vector spans the protected *main space*, and treat the remaining teachers as *auxiliaries*. Here, hardness is measured by the number of optimization steps each RL expert requires to reach its peak performance: the domain whose teacher converges slowest is regarded as the hardest to distill.

This division is not fixed: the choice of which domain occupies the main space can be adjusted flexibly according to the target applications and the domains one wishes to prioritize.

Task vectors. Let θ_0 be the base model and θ_k the k-th teacher. Each teacher induces a task vector $\Delta_k = \theta_k - \theta_0$. Collecting the subject vector Δ_s and the auxiliary vectors $\{\Delta_a\}_{a\in\mathcal{A}}$:

$$\Delta_s = \theta_{\text{subject}} - \theta_0, \qquad \Delta_a = \theta_a - \theta_0, \;\; a \in \mathcal{A}, \tag{3}$$

where $\mathcal{A}$ indexes the set of auxiliary teachers, Δ_s spans the protected main space, and each Δ_a must be reconciled with it before merging.

Subspace projection. Rather than adding each auxiliary vector , we project it onto the main space spanned by the subject vector(s),

$$\widehat{\Delta}_a = \underset{\Delta\in\text{span}(\{\Delta_s\})}{\arg\min} \left\|\Delta_a - \Delta\right\|_2^2 + \lambda\|\Delta\|_2^2, \tag{4}$$

where the ridge coefficient λ controls the strength of the projection. This keeps only the component of Δ_a aligned with the main space and discards directions orthogonal to it; we further clip the component of $\widehat{\Delta}_a$ that is anti-aligned with the subject anchor, so that no auxiliary update ever *subtracts* from the protected domain.

Merging. The initialized student combines the subject vector with the projected vectors:

$$\theta_{\text{init}} = \theta_0 + \Delta_s + \alpha \sum_{a\in\mathcal{A}} w_a \widehat{\Delta}_a, \tag{5}$$

where $\alpha < 1$ scales down the auxiliary contributions, and w_a are the per-teacher weights, normalized to sum to one. Embedding the hardest domain into θ_{init} thus alleviates the uneven recovery.

2.3.2 Difficulty-Aware Data Allocation

The second step of `Swift-MOPD` reallocates the training data to complement the initialization. Since the main space has already been preserved in θ_{init}, it requires only limited data to be maintained during distillation, whereas the auxiliary domains—which enter the initialization only through their projected, down-scaled components—remain under-fitted and call for additional supervision.

Formally, within each batch the subject domain is sampled with probability proportional to γ, while each auxiliary domain is sampled with probability proportional to 1,

$$p_s = \frac{\gamma}{\gamma + |\mathcal{A}|}, \qquad p_a = \frac{1}{\gamma + |\mathcal{A}|} \quad (a \in \mathcal{A}), \qquad \gamma < 1, \tag{6}$$

where $\gamma < 1$ is a single hyper-parameter that down-weights the main space. We set $\gamma = 0.75$ throughout our experiments. Together with the difficulty-aware initialization, this directs the learning budget toward the domains that need it most, while retaining the hardest domain's capability.

3 Experiments and Analysis

In this section, we conduct experiments to address the following research questions:

- **RQ1:** Does difficulty-aware model merging provide a better initialization for subsequent training?
- **RQ2:** Does our data allocation accelerate convergence compared to uniform allocation?
- **RQ3:** Does `Swift-MOPD` outperform standard OPD across different domains?

3.1 Experimental Settings

Training Datasets. For math, we construct the RL data by filtering DeepMath (He et al., 2025) to 57K samples at difficulty level 6 or above; for code, we adopt the 25K-sample Eurus-RL-Code (Cui et al., 2025) dataset. For scientific reasoning, we use the reasoning subsets (L3) of SciKnowEval (Feng et al., 2024a) covering Chemistry, Physics, Biology, and Materials, and randomly hold out 80% of the samples for training. For planning, we train on Sudoku-Bench (Seely et al., 2025)

Table 1: Performance of the domain teachers, the base student, and the merged initialization (θ_{init}). Merging alone recovers the main-space domains almost for free before any distillation, while also improving the auxiliary domains. Subscripts denote the gain of merging over the base student. Plan./Chem./Phys./Bio./Mat. denote Planning, Chemistry, Physics, Biology, and Materials.

	Math		Code	Plan.	Science			
Method	AIME24	AIME25	LCB	Sudoku	Chem.	Phys.	Bio.	Mat.
Teacher	58.0	54.6	27.3	86.60	72.14	65.31	51.81	79.68
Student	21.5	21.9	17.9	28.80	42.50	59.53	30.81	60.87
Merged (θ_{init})	$25.42_{+3.9}$	$25.31_{+3.4}$	$18.7_{+0.8}$	$57.60_{+28.8}$	$72.02_{+29.5}$	$66.45_{+6.9}$	$50.69_{+19.9}$	$79.55_{+18.7}$

using the same training split as (Wang et al., 2025). Applying RL to the base model on each domain separately yields domain teachers. The distillation stage reuses the same data as RL.

Training Settings. All models in our experiments are built upon Qwen3-4B-Non-Thinking (Yang et al., 2025): the student starts from this base model, while the domain teachers are specialized from the same base model via RL on domain-specific data. We first train domain teachers via Group Relative Policy Optimization (GRPO) (Shao et al., 2024), using a binary outcome reward: a response receives 1.0 if its final answer is correct for math reasoning, or if it passes all unit tests for code generation, and 0.0 otherwise. Full GRPO hyper-parameters are listed in Appendix A. To mitigate the mismatch between training and inference, we apply token-level rollout correction (Liu et al., 2025). All experiments are built upon the `verl` (Sheng et al., 2024) framework.

Evaluation. We evaluate on four domains. For math reasoning, we use two competition-level benchmarks, AIME24 (AI-MO, 2024) and AIME25 (OpenCompass, 2025), sampling 32 solutions per problem and reporting average accuracy. For code generation, we adopt LiveCodeBench (v6, February∼May 2025) (Jain et al., 2024), sampling 4 solutions per problem. For scientific reasoning, we report pass@1 on the 20% held-out split of the SciKnowEval L3 subsets. For planning, we use the Sudoku-Bench evaluation split of (Wang et al., 2025). Unless otherwise stated, we set the temperature to 1.0, top-p to 1.0, and the maximum generation length to 16,384; math answers are checked with the rule-based `Math-Verify` verifier.[1] Since the four domains differ substantially in absolute headroom, raw accuracy averages over-weight the wider-headroom domains. Following (Ma et al., 2026), we therefore report a normalised *recovery rate* as the headline metric. Let $\mathcal{D}$ be the set of evaluation benchmarks, s_d^{s} the accuracy of the base student and s_d^{t} that of the domain teacher on benchmark $d \in \mathcal{D}$. For a method with accuracy s_d, the per-benchmark recovery rate is

$$\tilde{s}_d = \frac{s_d - s_d^{\text{s}}}{s_d^{\text{t}} - s_d^{\text{s}}}, \tag{7}$$

which equals 0 at the base student and 1 at the domain teacher. The headline figure is the uniform average across benchmarks,

$$\bar{\tilde{s}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \tilde{s}_d, \tag{8}$$

where values above 1 indicate improvement beyond the domain teacher and values below 0 regression below the base student.

3.2 The "Free Lunch" of Difficulty Perception Merging (RQ1)

[Observation 1] **Merging alone provides a strong initialization at zero training cost.** As shown in table 1, the merged model retains the main-space domains almost entirely, recovering Chemistry to 72.02 against the teacher's 72.14 while lifting Biology from 30.81 to 50.69, a striking $+19.9\uparrow$ that brings it within 1.1 of the teacher. Physics and Materials tell the same story, reaching 66.45 and 79.55 to match or even surpass the teacher level. The auxiliary domains benefit as well, though more modestly: AIME24 and AIME25 improve by $+3.9\uparrow$ and $+3.4\uparrow$ respectively, and LCB edges up to 18.7. Remarkably, all of this is achieved before a single distillation step. With the hardest domain already secured in θ_{init}, the student starts from a favorable point, freeing the subsequent training to concentrate its budget on the domains that still lag behind.

[1] https://github.com/huggingface/Math-Verify

3.3 TRAINING DYNAMICS OF SWIFT-MOPD (RQ2)

We next examine the early training dynamics of `Swift-MOPD` against MOPD and ExOPD. We track validation accuracy during the first ten gradient steps to assess how quickly each method recovers the performance (fig. 4).

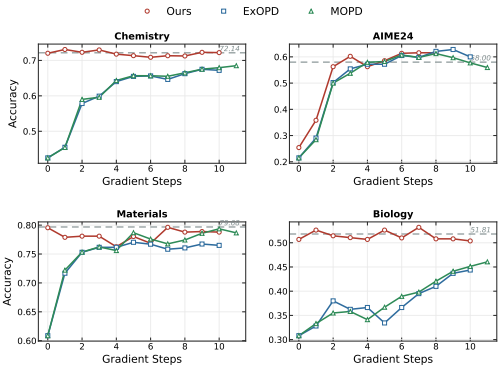

Figure 4: Training dynamics across Math and Science domains under the four-teacher setting. Dashed lines denote teacher performance. `Swift-MOPD` recovers performance within the initial training steps, whereas MOPD and ExOPD remain below the teachers on hard sub-tasks.

[Observation 2] **Swift-MOPD rapidly recovers teacher-level performance without sacrificing individual domains.** During the first few training steps, Chemistry consistently remains within $0.71 \sim 0.73$, closely matching the teacher score of 0.72, while Materials stays near 0.79, effectively retaining the 0.80 teacher-level performance. Biology likewise remains stable around 0.51, close to the teacher's 0.52, despite receiving a deliberately reduced sampling share. Notably, this stability on the protected main-space domains is achieved from the very first step, indicating that the merged initialization already carries over most of the teacher capabilities before any distillation begins. Meanwhile, on the auxiliary Math domain, AIME24 rises rapidly from 0.36 to 0.62 within only eight gradient steps and surpasses the teacher score of 0.58. Overall, the initialization acts as a strong prior that preserves the hard domains, while the difficulty-aware allocation directs the learning budget toward the other domains.

[Observation 3] **The baselines recover more slowly and remain incomplete on difficult domains.** On Biology, `MOPD` and `EXOPD` reach only 0.46 and 0.44 after ten steps, respectively, leaving a gap of $6 \sim 8$ points to the 0.52 teacher. Their Chemistry curves also begin near 0.45 and climb gradually, whereas `Swift-MOPD` maintains teacher-level performance from the initial steps. A similar trend holds on Materials, where the baselines require several steps to approach the teacher while `Swift-MOPD` stays flat near 0.79 throughout. The gap is especially pronounced in the first five steps, suggesting that the baselines spend their early budget re-learning capabilities that `Swift-MOPD` already inherits from the merged initialization. These results show that the difficulty-aware initialization and allocation jointly enable rapid and balanced domain recovery.

3.4 MAIN RESULTS RQ3

We compare five approaches to multi-domain capability integration: (i) `Mix RL`, which trains the student with RL on a single mixture of prompts drawn uniformly from all domains; (ii) `Cascade RL`, which applies domain-wise RL in sequence, warm-starting each stage from the checkpoint of the previous one; (iii) `Param-Merge (Avg.)`, which directly averages the weights of the domain teachers into a single model without further training; (iv) `MOPD`, which employs linear *reward interpolation* to accelerate OPD convergence; and (v) `EXOPD`, which instead adopts *reward extrapolation* (coefficient 1.25 following the original paper) to further expedite OPD convergence.

[Observation 4] **Superior and balanced cross-domain performance.** As shown in Table 2, `Swift-MOPD` achieves the highest normalised recovery rate of **1.0185**, exceeding the aggregate teacher reference of 1.0000 and outperforming the strongest baseline, `Mix-RL`, by 0.0802. Across the eight benchmarks, `Swift-MOPD` ranks first on five, ties for first on Sudoku, and ranks second on both LCB and Physics, demonstrating consistently strong performance rather than gains concentrated in a single domain. It surpasses the corresponding teachers on AIME24, AIME25, Sudoku, Chemistry, and Physics, while remaining within 1.44 points of the teachers on all other benchmarks. Notably, simply attaching vanilla `MOPD` to a linearly merged initialization already lifts the recovery rate to 0.9306, confirming that parameter-space merging provides a useful head start; however, its uniform merging weights fail to protect the hardest domains, leaving Biology and Materials well below their teachers. In contrast, `Swift-MOPD`'s difficulty-aware merging closes these gaps, suggesting that a well-initialized student can serve as a better pivot for cross-domain transfer than any individual teacher, and that modeling fitting difficulty turns merging into a principled initialization and distillation into a cooperative rather than competing process.

Table 2: Performance across math, code, planning, and scientific reasoning. Norm. Score denotes the average recovery rate relative to the student and the domain teachers. The best and second-best results among integration methods are boldfaced and underlined, respectively; the student and teacher references are excluded from ranking.

	Math		Code	Plan.	Science				Overall
Method	AIME24	AIME25	LCB	Sud.	Chem.	Phys.	Bio.	Mat.	Norm. Score
Student	21.50	21.90	17.90	28.80	42.50	59.53	30.81	60.87	0.0000
Teacher	58.00	54.60	27.30	86.60	72.14	65.31	51.81	79.68	1.0000
`Linear Merge`	33.23	27.29	18.70	81.80	53.96	63.52	34.25	72.44	0.4179
`Mix-RL`	59.39	55.10	**27.30**	88.00	65.88	**67.15**	41.75	75.93	<u>0.9383</u>
`Cascade RL`	54.68	51.25	24.30	86.40	66.26	63.33	40.68	76.07	0.7776
`MOPD`	60.60	54.10	26.17	87.00	66.28	64.66	41.06	77.33	0.8745
`EXOPD`	<u>62.08</u>	<u>55.31</u>	<u>27.14</u>	**88.20**	67.48	64.33	43.68	76.49	0.9076
`Linear Merge + MOPD`	60.31	54.12	<u>27.14</u>	87.80	<u>70.22</u>	64.49	<u>45.50</u>	<u>77.79</u>	0.9306
`Swift-MOPD`	**62.81**	**56.04**	<u>27.14</u>	**88.20**	**72.27**	<u>65.70</u>	**50.37**	**78.89**	**1.0185**
Δ(`Swift-MOPD` − Teacher)	+4.81	+1.44	−0.16	+1.60	+0.13	+0.39	−1.44	−0.79	—

Table 3: Combining `Swift-MOPD` with existing MOPD improvements.

	Math		Code	Plan.	Science				Overall
Method	AIME24	AIME25	LCB	Sud.	Chem.	Phys.	Bio.	Mat.	Norm. Score
`Swift-MOPD`	**62.81**	<u>56.04</u>	<u>27.14</u>	<u>88.20</u>	<u>72.27</u>	<u>65.70</u>	<u>50.37</u>	<u>78.89</u>	<u>1.0185</u>
`EXOPD`	<u>62.08</u>	55.31	<u>27.14</u>	<u>88.20</u>	67.48	64.33	43.68	76.49	0.9076
`Swift-MOPD + EXOPD`	61.56	**58.54**	**28.00**	**88.80**	**72.91**	**67.38**	**50.81**	**79.16**	**1.0799**

[Observation 5] **Swift-MOPD amplifies existing OPD improvements.** Since difficulty-aware initialization and data allocation operate on the starting point and batch composition rather than the distillation objective, they should stack with objective-level advances. Equipping `Swift-MOPD` with the reward extrapolation of `EXOPD` (coefficient reduced from 1.25 to 1.1, as the original proves too aggressive under our initialization) confirms this: as shown in Table 3 and Figure 5, the combined `Swift-MOPD + EXOPD` achieves the best results on seven of the eight benchmarks and raises the normalized recovery rate to **1.0799**. `Swift-MOPD` thus does not compete with existing OPD improvements but compounds with them, offering a general foundation for stronger MOPD.

4 Related Work

On-Policy Distillation. OPD reduces the train–test mismatch in autoregressive distillation by rolling out the student and aligning it with teacher token distributions on student-generated trajectories (Agarwal et al., 2024b; Gu et al., 2024a). Compared with off-policy distillation, OPD provides dense on-policy supervision and has shown improved calibration, exposure-bias reduction, and post-training efficiency (Ko et al., 2024; 2025). Recent studies improve OPD via entropy-aware, relaxed, trust-region, reward-extrapolated, hybrid, and offline objectives (Ko et al., 2026; Jin et al., 2026; Xing et al., 2026; Yang et al., 2026; Zhu et al., 2026; Wu et al., 2026), and extend it to cross-tokenizer, black-box, and self-distillation settings (Niu et al., 2026; Ye et al., 2025; Hübotter et al., 2026b; Penaloza et al., 2026; Zhao et al., 2026; Kaur et al., 2026).

Multi-Teacher On-Policy Distillation. Multi-Teacher On-Policy Distillation (MOPD) (Yin et al., 2026; Lian et al., 2026; Song & Zheng, 2026) can be viewed as a behavior-space approach to model consolidation: multiple specialized teachers supervise the student on its own trajectories, allowing their capabilities to be merged into a single deployable checkpoint. This is especially useful for integrating independently trained domain or RL-specialized models, while avoiding ensemble deployment and parameter-interference issues in direct model merging (Ilharco et al., 2022b; Yadav

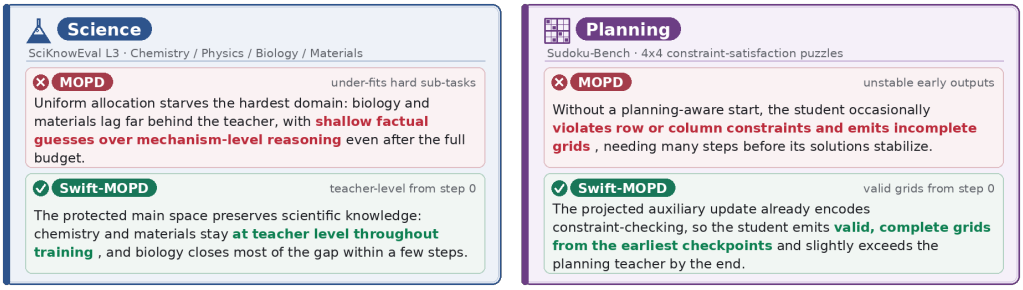

Figure 5: Qualitative comparison between `MOPD` and `Swift-MOPD`.

et al., 2023; Yu et al., 2023). Recent MOPD work explicitly targets capability integration in LLM post-training (Ma et al., 2026; Xiao et al., 2026a; Xu et al., 2026); related methods use multi-agent teacher debate or reward-extrapolated OPD to overcome single-teacher limits and combine domain expertise (Wang et al., 2026; Yang et al., 2026). Compared with parameter merging and LoRA composition (Wu et al., 2024; Feng et al., 2024b), MOPD merges models through token-level behavior supervision under the student's on-policy distribution.

Model Merging. Model merging offers a practical means of combining the capabilities of multiple models without revisiting their raw training data or incurring the cost of further large-scale optimization (Yang et al., 2024a; Cheng et al., 2025; Akiba et al., 2025). Existing methods commonly merge fine-tuned models through weight-space operations, including task-vector arithmetic (Ilharco et al., 2022a), parameter averaging and interpolation (Wortsman et al., 2022), or alignment-aware fusion schemes that reduce inter-model conflicts (Stoica et al., 2023; Zhang et al., 2024). More recent work also explores adaptive merging through learned weighting, representation-level combination, or routing (Yu et al., 2024; Yang et al., 2024b). The common goal is to consolidate complementary strengths into a single model with balanced and strong performance across capabilities. In contrast, we use merging to build a difficulty-aware initialization from domain-specialized RL teachers, preparing the student for subsequent multi-teacher on-policy distillation.

5 Conclusion and Discussion

In this work, we revisited MOPD from the perspective of student initialization and data allocation. Our analysis revealed that different domains exhibit distinct fitting difficulties, and that uniform data allocation under vanilla MOPD systematically under-fits harder-to-distill domains, leading to imbalanced capability integration. Building on this insight, we proposed Swift-MOPD, a difficulty-aware framework that initializes the student by merging teacher task vectors with weights proportional to per-task fitting difficulty, and dynamically reallocates training data toward under-fitted domains during distillation. Experiments across Math, Code, Planning, and Science under a four-teacher setting demonstrate that Swift-MOPD achieves a normalized recovery of 1.0185, surpassing the average domain-teacher performance, and that combining it with reward extrapolation further raises the score to 1.0799, confirming its role as a plug-and-play foundation for existing OPD variants.

6 Limitations

`Swift-MOPD` requires all teachers and the student to share the same architecture, since the merging operates in weight space. We note, however, that this is precisely the dominant setting in practice: as domain experts are increasingly obtained by fine-tuning a shared foundation model, weight-space merging applies without any architectural surgery, and our four-teacher study suggests the framework scales naturally as more experts are added to the pool. Besides, our study is limited to text-only models. That said, nothing in the framework is specific to text: task vectors and difficulty-aware allocation are defined over parameters and batch composition, and both carry over to multimodal experts fine-tuned from a shared vision–language backbone.

Ethics Statement

This work studies multi-teacher on-policy distillation for capability integration in large language models. All models are trained and evaluated on publicly available benchmarks (DeepMath, Eurus-RL-Code, SciKnowEval, and Sudoku-Bench) that contain no private or personally identifiable information.

Reproducibility Statement

We document the base model, training datasets, domain-teacher RL hyper-parameters (Tables 4–6), distillation configurations (Table 7), prompt templates for all domains, and the difficulty-aware merging and allocation procedure in the appendix. All experiments are built upon the publicly available `verl` framework, and evaluation protocols follow the original benchmarks.

AI Use Statement

Generative models were used as experimental subjects (domain teachers and students) as described in the methodology. The authors additionally used AI-based assistants for language polishing of the manuscript. The authors designed the method, experiments, and analyses, and remain fully responsible for the content and claims of this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024a.

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

AI-MO. Aime 2024. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024.

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, 2025.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, and Feng Zhao. Flow-OPD: On-policy distillation for flow matching models. *arXiv preprint arXiv:2605.08063*, 2026.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024a.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning for large language models. In *Proceedings of COLING*, 2024b.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023. Early version of MiniLLM focusing on off-policy distillation.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2024a.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Dongze Hao, Zhiwei Jin, Chen Chen, and Haonan Lu. Stabilizing on-policy distillation for MLLM reasoning with global normalization. *arXiv preprint arXiv:2606.09091*, 2026.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026a. URL https://arxiv.org/abs/2601.20802.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026b.

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*, 2022a.

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*, 2022b.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of EMNLP*, 2016.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

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.

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

Boyan Li, Bingsen Chen, Chenghao Yang, Ping Nie, Chen Zhao, and Xi Ye. Sequential beats joint: On the interplay between on-policy distillation and RLVR. *arXiv preprint arXiv:2609.04108*, 2026a.

Zhaoyi Li, Deyang Kong, Yuan Wei, Evan Yang, Ranran Shen, Mahardika Krisna Ihsani, Ming Yang, Wei Zhang, Chuan Hao, Jian Yang, Ran Tao, Bryan Dai, Shikun Zhang, Wei Ye, Ying Wei, and Defu Lian. Every coin has two sides: On the dual nature of generalization in on-policy distillation of large language models. *arXiv preprint arXiv:2608.16647*, 2026b.

Niu Lian, Alan Chen, Zhehao Yu, Chengzhen Duan, Fazhan Liu, Hui Liu, Pei Fu, Jian Luan, Yaowei Wang, Shu-Tao Xia, et al. Ui-mopd: Multi-platform on-policy distillation for continual gui agent learning. *arXiv preprint arXiv:2607.04425*, 2026.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025. URL https://richardli.xyz/rl-collapse.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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

Yifan Niu, Han Xiao, Dongyi Liu, Zelong Wang, Dihong Gong, Yasheng Wang, and Jia Li. Breaking the tokenizer barrier: On-policy distillation across model families. *arXiv preprint arXiv:2606.09456*, 2026.

OpenCompass. Aime 2025. https://huggingface.co/datasets/opencompass/AIME2025, 2025.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Jeffrey Seely, Yuki Imajuku, Tianyu Zhao, Edoardo Cetin, and Llion Jones. Sudoku-bench: Evaluating creative reasoning with sudoku variants. *arXiv preprint arXiv:2505.16135*, 2025.

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

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.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

Thinking-Space. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026a.

Thinking-Space. Rethinking on-policy distillation of large language models II: One training example. *arXiv preprint arXiv:2609.04172*, 2026b.

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

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models, 2025. URL https://arxiv.org/abs/2509.06949.

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

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

Yecheng Wu, Song Han, and Hai Cai. Lightning opd: Efficient post-training for large reasoning models with offline on-policy distillation. *arXiv preprint arXiv:2604.13010*, 2026.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026a.

Jinwei Xiao et al. Finding the evidence: Discovering decision-supporting tokens for on-policy reasoning distillation. *arXiv preprint arXiv:2606.22830*, 2026b.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 2024a.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. *arXiv preprint arXiv:2402.02705*, 2024b.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026.

Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025.

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

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.

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. In *Forty-first International Conference on Machine Learning*, 2024.

Frederic Z Zhang, Paul Albert, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. Knowledge composition using task vectors with learned anisotropic scaling. *Advances in Neural Information Processing Systems*, 37:67319–67354, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Wenhong Zhu, Ruobing Xie, Rui Wang, and Pengfei Liu. Hybrid policy distillation for llms. *arXiv preprint arXiv:2604.20244*, 2026.

Table 4: GRPO hyper-parameters for RL on the math and science teachers.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	500
KL Coefficient	0.0

Table 5: GRPO hyper-parameters for RL on the code teacher.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	8192
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	300
KL Coefficient	0.0

Table 6: GRPO hyper-parameters for RL on the planning teacher.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	100
KL Coefficient	0.0

Table 7: Distillation hyper-parameters of `Swift-MOPD` (following the `Ex-OPD` configuration).

Hyper-parameter	Value
Batch Size	1024
Rollout n	1
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-5}

A Detailed Training Settings

Domain teacher training. The GRPO hyper-parameters for training the domain teachers are summarized in Table 4 (math and science), Table 5 (code), and Table 6 (planning). All teachers are trained from the same base model, Qwen3-4B-Non-Thinking, with a binary outcome reward. The math and science teachers share the same configuration with 500 optimization steps, while the code teacher uses a shorter maximum response length of 8,192 and is trained for 300 steps; the planning teacher is trained for 100 steps, as Sudoku performance saturates early.

Multi-teacher distillation. The distillation hyper-parameters of `Swift-MOPD` are listed in Table 7, which follow the configuration of `Ex-OPD` for a fair comparison. Unless otherwise stated, all distillation experiments run for 10 gradient steps under the four-teacher setting, and the sampling proportion of the protected main-space domain is down-weighted by $\gamma = 0.75$ as described in Section 2.3. The reward extrapolation coefficient in the combined `MOPD + Ex-OPD` variant is set to 1.1 instead of the original 1.25, since the latter proves too aggressive under our difficulty-aware initialization. All experiments are built upon the `verl` framework, and token-level rollout correction is applied to mitigate the mismatch between training and inference.

B Prompt Templates

The prompt templates used for training and evaluation in each domain are shown below.

Training and Evaluation Prompt Template for Math Reasoning

< |im_start| >user
{question}
Please reason step by step, and put your final answer within \boxed{}.< |im_end| >
< |im_start| >assistant

Training and Evaluation Prompt Template for Code Generation

< |im_start| >user
{question}
Write Python code to solve the problem. Present the code in
```python
Your code
```
at the end.
You need to think first then write the Python code.< |im_end| >
< |im_start| >assistant

Training and Evaluation Prompt Template for Scientific Reasoning

< |im_start| >user
Given a question and four options, please select the right answer. Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>

For the answer, only output the letter corresponding to the correct option (A, B, C, or D), and nothing else. Do not restate the answer text. For example, if the answer is "A", just output:
<answer>
A
</answer>

{question} < |im_end| >
< |im_start| >assistant

Training and Evaluation Prompt Template for Planning

< |im_start| >user
Please solve the following 4x4 Sudoku puzzle. The puzzle is provided as a 16-character string reading left-to-right, top-to-bottom, where '0' represents empty cells.

Rules:
- Fill empty cells with digits 1-4
- Each row must contain digits 1-4 exactly once
- Each column must contain digits 1-4 exactly once
- Each 2x2 box must contain digits 1-4 exactly once

Important: Your solution must be a COMPLETE 16-character string with only the digits 1-4, representing your final solved grid.

Respond in this exact format:
<reasoning>
...
</reasoning>
<answer>

</answer>

Solve the following Sudoku puzzle: {puzzle} < |im_end| >
< |im_start| >assistant