RETHINKING ON-POLICY DISTILLATION OF LARGE LANGUAGE MODELS: ONE TRAINING EXAMPLE

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) trains on student rollouts with dense teacher supervision, but the role of training data remains unclear. We find that a single query supports hundreds of improving updates and recovers most of full-data OPD's gain across domains and model families. We explain this result through the states visited during training and the rate at which the student aligns with the teacher. We measure *state coverage*: the fraction of the states full-data OPD visits that a query set's rollouts reach. One query reaches 71.5%, mostly within 100 steps. Adding semantically distinct queries raises coverage and validation accuracy together, until 16 queries reach 98.9% and match full-data training. Yet alignment slows at a similar pace whether OPD trains on one query or the whole dataset, and even a fixed set of states takes hundreds of steps to absorb. OPD is therefore data-overfed but algorithm-starved. Rollouts quickly expose broad supervision early, while the student absorbs it increasingly slowly. As a further stress test, content-light templates and off-domain WildChat queries also approach the real-query baseline. These findings suggest that training queries matter through the teacher-supervised states their rollouts induce, while learning progress depends on how efficiently the student absorbs that supervision.

1 INTRODUCTION

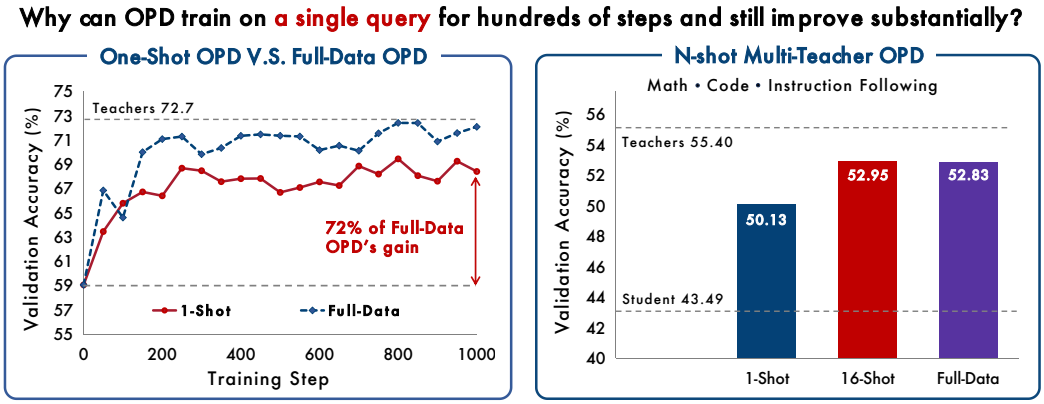

Figure 1: A single query recovers most of full-data OPD's gain on mathematical reasoning (left), and the effect holds across domains and model families. Under multi-teacher OPD, 16 queries per domain match full-data training (right).

On-policy distillation (OPD) is used alongside supervised fine-tuning and reinforcement learning in Qwen3, MiMo, GLM-5, DeepSeek-V4, and Kimi K3 (Yang et al., 2025; Xiao et al., 2026; Zeng et al., 2026; Xu et al., 2026; Team et al., 2026). The student generates rollouts and receives teacher supervision at each visited prefix (Gu et al., 2024; Agarwal et al., 2024), yielding dense token-level targets rather than the trajectory-level rewards of reinforcement learning with verifiable rewards.

Existing analyses primarily explain OPD through its objective and training dynamics (Li et al., 2026; Fu et al., 2026; Cai et al., 2026; Zhu et al., 2026). Yet they leave open how training data shapes

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.

OPD and how data and the distillation algorithm interact. Closing this gap is essential to a complete understanding of OPD. In RLVR, Wang et al. (2026a) investigate the role of training data through an extreme controlled experiment: training on a single example. We bring the same experimental lens to OPD by training on a single query, a setting we call *one-shot OPD*. Figure 1 shows that, despite training repeatedly on a single query, the student continues to improve for hundreds of steps and recovers most of full-data OPD's gain. The phenomenon holds across four task domains and three model families, even on queries the student never solves. We therefore devote this work to answering a single question: *Why can OPD, when trained on just one example,* ***keep learning for so long*** *and* ***improve by so much****?*

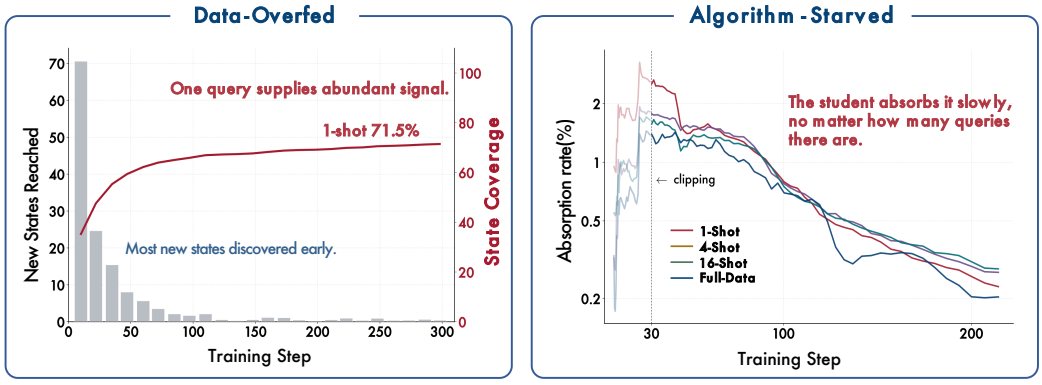

Figure 2: **OPD is data-overfed but algorithm-starved. Data:** one query already covers most of the states full-data OPD visits. **Algorithm:** the student absorbs an ever smaller proportion of the remaining teacher–student gap, no matter how many queries it trains on.

Answering this question requires explaining both why a single query supplies enough supervision for substantial gains and why the algorithm takes so many updates to absorb it. Figure 2 summarizes these two perspectives: **OPD is data-overfed but algorithm-starved**. From the *data perspective*, a query acts on the student through the *states* its rollouts reach, each state being a prefix $s = (x, y_{<i})$ at which the teacher supplies token-level supervision, so a query set is worth the part of that space it covers. We quantify this by *state coverage*: we group the states full-data OPD visits into semantic clusters and report the fraction a setting's rollouts reach. A single query already supplies enough states on its own, covering 71.5% of what full-data OPD visits, and 16 semantically diverse queries cover 98.9% and match full-data training (Section 4). This is why so little data improves the student by so much. From the *algorithm perspective*, what falls as training proceeds is the *absorption rate*, the proportion of the remaining teacher–student gap that one update closes, and it declines in much the same way on one query as on all 17k (Section 5), so the pace of a run is a property of OPD rather than of the training set. In other words, the supervision a single query supplies remains largely unexploited, rate-limited by how quickly an on-policy student can absorb it.

The finding that a few diverse queries can match full-data training also extends to multi-teacher OPD (Xiao et al., 2026; Xu et al., 2026): 16 semantically diverse queries per domain match full-data MOPD (Appendix D). Taking the data side to its extreme, we further show that even a training input that states no problem can remain effective: off-domain WildChat prompts and content-free templates drive OPD nearly as effectively as the full training set (Section 6.1). Therefore, an input is useful largely because it starts the student reasoning. We also compare OPD with RLVR on the same query: over 1000 steps, OPD gains more than twice as much validation accuracy, retaining token-level supervision after outcome variation diminishes (Section 6.2). These findings motivate selecting inputs by the states they induce and improving the step efficiency of supervision absorption.

2 Preliminaries

2.1 Notation

Let $x \sim \mathcal{D}$ be a training query, $y = (y_1, \ldots, y_L)$ a sampled response, and $s_i = (x, y_{<i})$ the *state* at token position i. A *trajectory* is (x, y). Student π_θ and teacher π_T define next-token distributions over a shared vocabulary; t indexes optimization steps and θ_t the student parameters.

2.2 On-Policy Distillation

On-policy distillation (OPD) aligns the student to the teacher on states sampled from the student (Agarwal et al., 2024). In the main setting, we use the sampled-token teacher–student advantage

$$A_i^{\text{sample}} = \log \pi_T(y_i \mid s_i) - \log \pi_\theta(y_i \mid s_i), \tag{1}$$

where y_i is the token sampled by the student. More generally, we may use a top-k approximation, with $\mathcal{V}_i^{(k)} = \mathrm{TopK}(\pi_\theta(\cdot \mid s_i), k)$ and $\tilde{\pi}_\theta^{(k)}$ denoting the student distribution renormalized over this set:

$$A_i^{(k)} = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta}\left[\sum_{w\in\mathcal{V}_i^{(k)}} \tilde{\pi}_\theta^{(k)}(w \mid s_i)[\log \pi_T(w \mid s_i) - \log \pi_\theta(w \mid s_i)]\right]. \tag{2}$$

The top-k form averages the same log-ratio over the student's k most probable tokens, while the sampled-token form evaluates it on the token generated by the student.

2.3 Metrics

Recovery Ratio. To compare gains across domains and model pairs, let M_0, M_T, and M_t be the evaluation scores of the initial student, teacher, and student at step t. The student–teacher gap recovery ratio is $\frac{M_t - M_0}{M_T - M_0} \times 100\%$. With full-data OPD as the reference, the full-data recovery ratio is $\frac{M_t - M_0}{M_F - M_0} \times 100\%$, where M_F is the full-data score at the same step. Thus, 0% indicates no gain and 100% matches the corresponding reference.

Dynamic Metrics. Following Li et al. (2026), we track three dynamic diagnostics on the top-k sets $S_t^\theta(s) = \mathrm{TopK}(\pi_{\theta_t}(\cdot \mid s), k)$ and $S^T(s) = \mathrm{TopK}(\pi_T(\cdot \mid s), k)$. Specifically, these are the *Overlap Ratio*, *Overlap-Token Advantage*, and *Abs Entropy Gap*. For non-padding response positions $\mathcal{T}_t$, the overlap ratio is

$$O_t^{(k)} = \frac{1}{|\mathcal{T}_t|} \sum_{i\in\mathcal{T}_t} \frac{|S_t^\theta(s_i) \cap S^T(s_i)|}{k}. \tag{3}$$

For the overlap-token advantage, let $I_i = S_t^\theta(s_i) \cap S^T(s_i)$ and let $\bar{\pi}_\theta$ and $\bar{\pi}_T$ denote the student and teacher distributions renormalized over I_i. We then average their log-probability difference over the shared tokens:

$$A_t^{\text{ov}} = \frac{1}{|\mathcal{T}_t|} \sum_{i\in\mathcal{T}_t} \frac{1}{|I_i|} \sum_{w\in I_i} [\log \bar{\pi}_T(w \mid s_i) - \log \bar{\pi}_\theta(w \mid s_i)]. \tag{4}$$

Finally, letting $\bar{H}_t(\pi)$ denote the mean next-token entropy over the response positions $\mathcal{T}_t$, we report the absolute entropy gap $\Delta H_t = |\bar{H}_t(\pi_{\theta_t}) - \bar{H}_t(\pi_T)|$.

3 The One-Shot Phenomenon

This section presents the experimental setup and results of One-Shot OPD. We investigate *how much of full-data OPD's gain a single query recovers, and how robust that gain is*.

Takeaways

One query recovers most of full-data OPD's gain, even when never solved. The effect extends across three model families and four task domains.

3.1 Experimental Setup

Models and Datasets. Table 1 lists the student–teacher pairs used across four domains and three model families. The instruction-following teacher is post-trained in this work; Appendix A.1 gives the training details. For training data, we use DAPO-Math-17K (Yu et al., 2026) for mathematical reasoning, Open-R1 Codeforces (Penedo et al., 2025) for code generation, UltraData-SFT-2605 (subset) (Wang et al., 2026c) for instruction following, and xLAM-function-calling-60K (Liu et al., 2024) for agentic tool use. For one-shot training in the math domain, we use easy, medium, and hard

Table 1: Student–teacher pairs used in our experiments.

Setting	Student	Teacher
Math	DeepSeek-R1-Distill-Qwen-1.5B (R1-Distill-1.5B)(Guo et al., 2025)	JustRL-DeepSeek-1.5B (JustRL-1.5B)(He et al., 2025b)
	Llama-3.2-3B-Instruct (Llama-3B-It)(Grattafiori et al., 2024)	GT-Llama-3.2-3B-Instruct-MATH (GT-Llama-3B-Math)(Zhang et al., 2026b)
	OLMo-3-7B-Instruct-DPO (OLMo-7B-It-DPO)(Olmo et al., 2025)	OLMo-3-7B-Instruct (OLMo-7B-It)(Olmo et al., 2025)
Code	DeepSeek-R1-Distill-Qwen-1.5B (R1-Distill-1.5B)(Guo et al., 2025)	Nemotron-Research-Reasoning-Qwen-1.5B-v2-RLVE (Nemotron-1.5B)(Zeng et al., 2025)
Instructionfollowing	DeepSeek-R1-Distill-Qwen-1.5B (R1-Distill-1.5B)(Guo et al., 2025)	UltraData-IF-1.5B (ours)
Agentictool use	Qwen2.5-Coder-1.5B-Instruct (Qwen-Coder-1.5B)(Hui et al., 2024)	Hammer2.1-1.5b (Hammer-1.5B)(Lin et al., 2024)

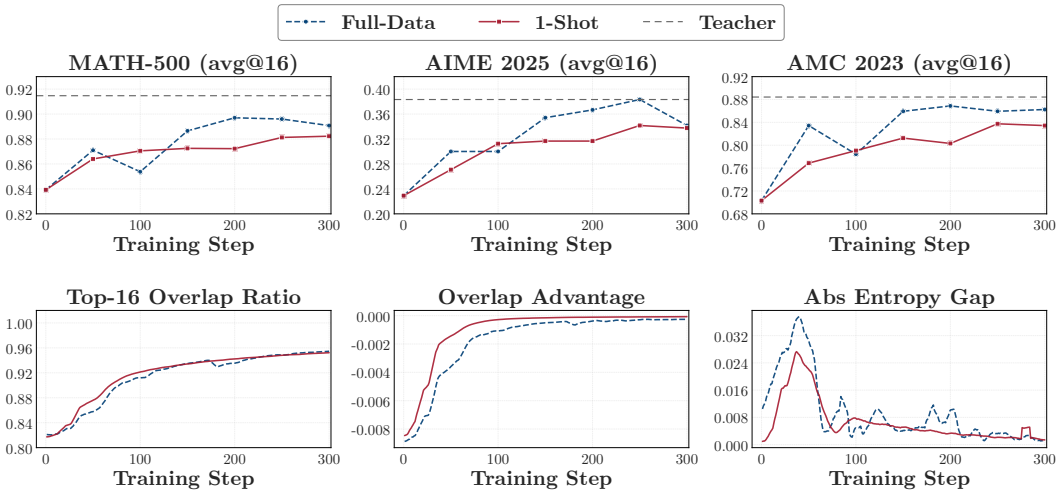

Figure 3: Validation accuracy and training dynamics of one-shot OPD on mathematical reasoning.

mathematical queries with initial pass rates of $8/8$, $4/8$, and $0/8$, respectively, and sample one query at random in each other domain.

Training. We implement OPD in VeRL (Sheng et al., 2025). The mathematical-reasoning runs use the top-k objective in Eq. 2 with $k = 16$, while the other domains use the sampled-token objective in Eq. 1. Each update uses 64 rollouts; we use AdamW with learning rate 10^{-6}, rollout temperature 1.0, and gradient clip norm 1.0. Remaining hyperparameters are listed in Table 3.

Evaluation. We report avg@16 on MATH-500 (Hendrycks et al., 2021), AMC 2023 (Li et al., 2024), and AIME 2025 (Balunovic et al., 2025). Other domains use LiveCodeBench v6 (Jain et al., 2025) (avg@3), Multi-IF (He et al., 2024) (third-turn, eight-language mean), and BFCL v3 (Patil et al., 2025) (avg@8). Response caps for mathematics, code generation, instruction following, and agentic tool use are 31,744, 65,536, 16,384 per turn, and 4,096 tokens, respectively.

3.2 Experimental Results

One-shot OPD recovers most of full-data OPD's gain. Figure 3 shows that one-shot OPD improves mathematical reasoning on all three benchmarks, approaching full-data OPD. At step 300, one-shot and full-data OPD reach 68.5 and 69.8 average accuracy, corresponding to 69% teacher–student gap recovery and 87% recovery of the full-data gain. Beyond step 300, the two curves remain within about 3 points, with full-data recovery ranging from 62% to 89%. At step 1000, one-shot OPD reaches 68.4 versus 72.1 for full-data OPD and recovers 72% of its gain (Figure 1). The training dynamics show the same alignment pattern: the one-shot top-16 overlap ratio reaches the full-data

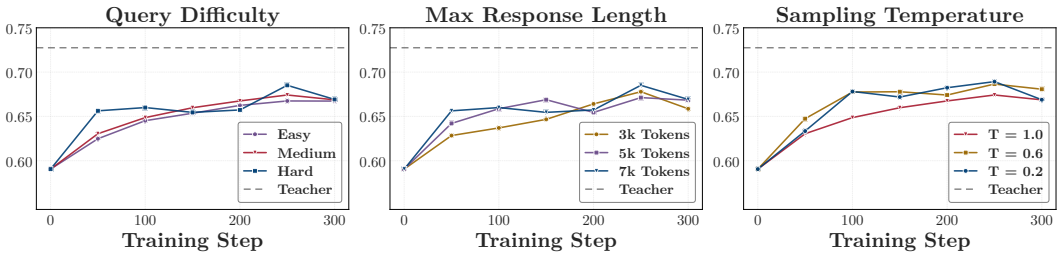

Figure 4: Validation accuracy of one-shot OPD under varying query difficulty, response-length cap on the hard query, and sampling temperature on the medium query.

level, the overlap-token advantage approaches zero, and the absolute entropy gap nearly closes. Despite training only on a single mathematical query, one-shot OPD does not degrade out-of-domain performance (Appendix A.2).

The one-shot effect is robust. Figure 4 shows that the gain is robust to the query and the rollout conditions. Easy, medium, and hard queries produce comparable gains even though the easy query is solved on nearly every rollout and the hard query is never solved (training-time pass rates in Appendix A.1), and tightening the response-length cap or lowering the rollout temperature preserves the improvement. The effect is likewise not specific to one model family or task domain: Appendix A.3 shows that it holds across three student–teacher families and extends beyond mathematics to code generation, instruction following, and agentic tool use.

4 Data Perspective: Abundant States

Section 3 showed that one query robustly recovers most of full-data OPD's gain. We now ask *why a single query goes so far, and why a few diverse queries suffice to match full-data OPD*.[1]

Takeaways

What OPD consumes is states rather than queries. One query's rollouts already reach 71.5% state coverage of full-data OPD, and 16 semantically diverse queries fill in most of the remainder, reaching 98.9% and matching full-data training, including in multi-teacher OPD.

4.1 State Coverage of One-Shot OPD

Setup. With 64 rollouts per update, one query yields tens of thousands of teacher-supervised states. We hypothesize that these rollouts cover a broad part of the state space visited by full-data OPD, supplying enough supervision to recover most of its gain. To test this, we measure *state coverage* in three steps: (1) **representation**: each state is represented by its *teacher signature*, the teacher's final-layer hidden vector at the state's last token; (2) **reference space**: full-data DAPO-Math-17K rollouts contribute 8 uniformly spaced states each, and a 3:2 rollout split defines disjoint reference and held-out pools; (3) **clusters**: PCA and K-means partition the reference pool into $K = 200$ clusters, fitted once and weighted equally. *State coverage* is the fraction of these clusters that a setting's states touch. Comparisons match rollout budgets over 300 steps; held-out full data empirically reaches 100%. Appendix B.1 details sampling and robustness to K, reference sets, and state positions.

One-shot OPD reaches 71.5% of the state space. Figure 5 shows that one-shot OPD reaches 71.5% state coverage by step 300. Most of this coverage appears early: the run reaches 65.9% by step 100, then adds only 5.6 points over the next 200 steps. The same run raises validation accuracy from 59.1 to 66.9, compared with 70.8 for full-data OPD. One-shot OPD is therefore small in query count, yet broad in the supervision it generates.

[1] Sections 4–5 use sampled-token advantages (Eq. 1), unlike Section 3's top-k math runs. This gives slightly different accuracies; within-section comparisons use the same objective.

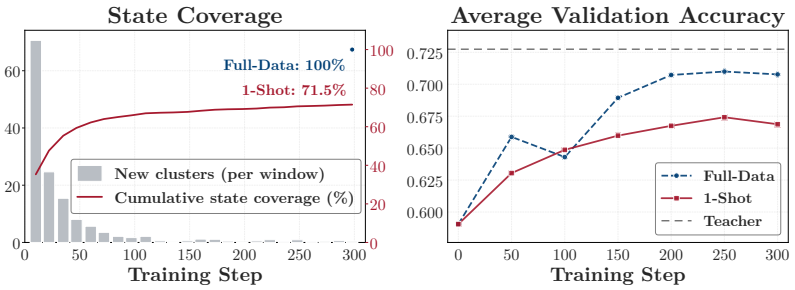

Figure 5: State coverage and validation accuracy of one-shot OPD on the medium query. Full-data OPD provides the coverage reference and validation baseline.

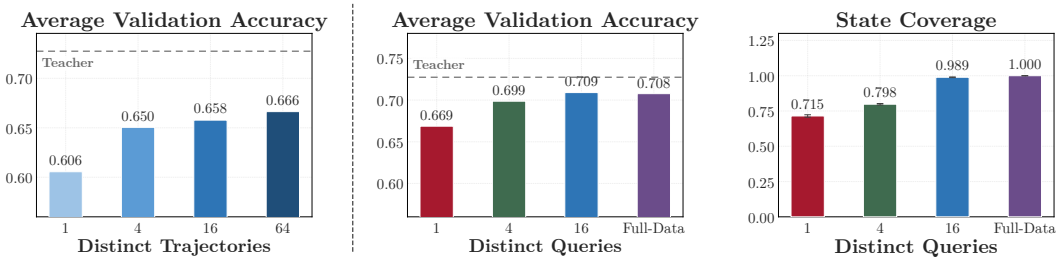

Figure 6: Validation accuracy and state coverage under increasing response diversity (off-policy) and query diversity (on-policy).

4.2 Diversity Expands State Coverage

To distinguish added states' benefits from coverage growing through learning, we control response and query diversity.

Setup. (i) Response diversity (off-policy): from 64 initial-student trajectories on the medium query, we repeat nested subsets of $1/4/16/64$ to fill batches of 64. States stay fixed, and query, batch size, and update budget are controlled. **(ii) Query diversity (on-policy):** we compare 1/4/16-shot and full-data OPD, selecting one query per BGE-M3 (Chen et al., 2024a) semantic cluster. We measure on-policy coverage and accuracy, and average off-policy accuracy over five checkpoints at steps 300–500; full-data coverage uses the held-out split.

Diverse queries expand state coverage. Figure 6 shows off-policy accuracy rising with trajectory diversity. On-policy, 16 queries reach 98.9% coverage, up from one query's 71.5%, and match full-data OPD. At a fixed count of 16, sampling from 16 semantic clusters improves coverage and accuracy over sampling from one; changing query order has little effect (Appendix B.2).

Extension to multi-teacher OPD. With domain-specific teachers for math, code, and instruction following, increasing from 1 to 16 diverse queries per domain raises five-benchmark average accuracy from 50.1 to 52.9, matching full-data MOPD's 52.8 over 300 steps (Figure 1, right; Appendix D).

5 Algorithm Perspective: Slow Alignment

Section 4 showed that one query quickly reaches broad state coverage. We now ask *why learning from those states still takes hundreds of steps*.

Takeaways

Alignment slows over time, insensitive to query count. Even a fixed set of states supports learning over hundreds of steps.

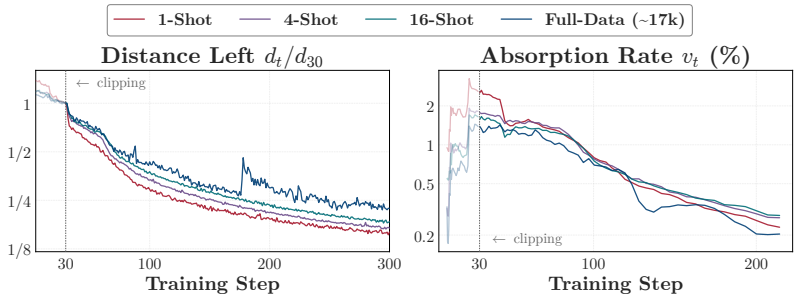

Figure 7: Teacher–student alignment for 1, 4, 16, and all DAPO-Math-17K queries. Distance (left) and absorption rate (right) are both on logarithmic vertical axes against linear steps.

5.1 Alignment Metrics

Distance d_t. The average magnitude of the sampled-token advantage over the visited positions $\mathcal{T}_t$:

$$d_t = \frac{1}{|\mathcal{T}_t|} \sum_{i \in \mathcal{T}_t} \left| \log \pi_T(y_i \mid s_i) - \log \pi_{\theta_t}(y_i \mid s_i) \right|. \tag{5}$$

Absorption rate v_t. The proportion of remaining distance absorbed per update:

$$v_t = \frac{d_t - d_{t+1}}{d_t}. \tag{6}$$

We track both metrics from step 30, after gradient clipping no longer binds, and report each run against its own value at that step, the *distance left* d_t/d_{30}.[2]

5.2 Alignment Slows Similarly Across Data

Setup. We track d_t and v_t for OPD trained on 1, 4, 16, and all DAPO-Math-17k queries, over the same 300 steps and under the mathematical configuration of Section 3.1.

Alignment slows similarly across query counts. Figure 7 shows that the distance falls for the whole run, so alignment is never stuck; what slows is the absorption rate, each update absorbing less of what is left than the last, which is why a run takes hundreds of steps rather than tens. Beyond this, each run removes 78% to 84% of its step-30 distance by step 300, and all four slow by a similar factor during training (Appendix C.1 gives the per-run numbers and the conventions these readings are taken under). One query therefore does not make alignment slower or faster: the pace is a property of OPD rather than of the training set.

Two natural candidates remain: that the run is sustained by the steady arrival of new states, or that its pace is set by the optimizer's step size. Neither holds: a run trained on a fixed set of states still slows just as much (Appendix C.2), and the learning rate changes how many steps that takes, but not the way the distance falls (Appendix C.3).

6 Discussion

The mechanism above shifts attention from prompts to the reasoning-state regions they induce: an input helps when the student's rollouts reach regions the teacher can supervise. Stating a problem is just one way to get there. This raises two questions: how much task content the input needs, and why one-shot OPD and RLVR differ. We study both via content-light and off-domain inputs, then compare with one-shot RLVR.

[2]To reduce noise late in training, v_t is estimated over the geometric window $[t/1.4,\ 1.4t]$, which is why it stops short of step 300. Appendix C.1 provides the estimators and supporting evidence.

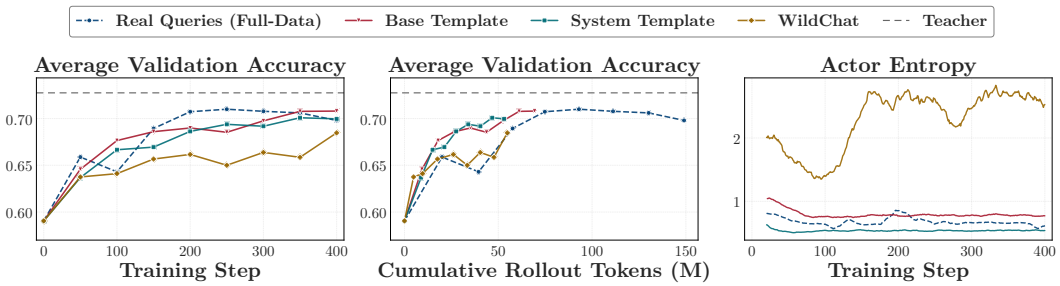

Figure 8: Validation accuracy and actor entropy for mathematics OPD trained on the full training set, content-light templates, or general-domain WildChat queries.

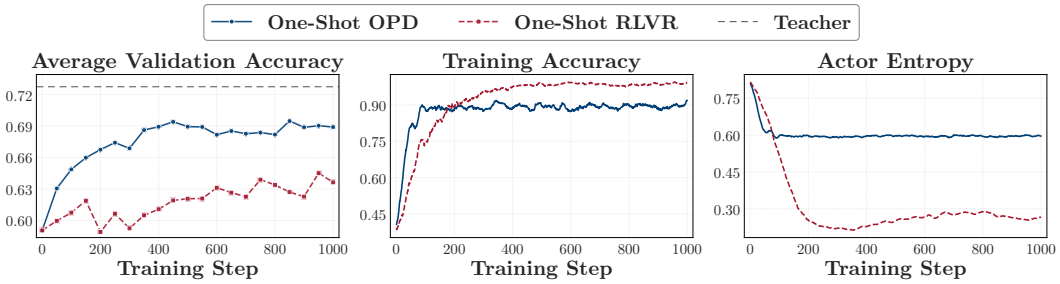

Figure 9: Validation accuracy and training dynamics on the same query. Both methods use 64 rollouts per step: RLVR uses 8 groups of 8 with GRPO outcome rewards; OPD uses 64×1 with per-token teacher targets. RLVR additionally uses a KL penalty with coefficient 0.001. Training accuracy measures the fraction of rollouts solving the query.

6.1 Inputs as State Generators

We stress-test this prediction with three conditions: (i) an empty user turn ending in <think>, (ii) the same template with a simple domain system prompt (cases in Figure 19), and (iii) general-conversation queries from WildChat (Zhao et al., 2024). WildChat is predominantly English, with roughly 0.17% math-related and 2.63% code-related queries.

Content-light and off-domain inputs can still induce useful states. Figure 8 shows that all three conditions track full-data OPD's 59.1-to-69.8 accuracy gain using one-third to one-half of its rollout tokens; code results are similar (Appendix E.1), and explicit domain content changes final accuracy by about one point. However, immediately closing thinking (<think>\n </think>\n) yields short meta-level replies unsuitable for training. Thus, task content is not generally dispensable: induced states and teacher targets also determine an input's usefulness.

6.2 Relation to One-Shot RLVR

Figure 9 compares OPD and one-shot RLVR (Wang et al., 2026a) with the same medium query, initial student, and learning rate.

Useful inputs differ across objectives. RLVR needs verifiable outcomes with within-group variation to yield useful advantages. OPD learns from teacher–student gaps at visited states: its inputs need not have verifiable answers or a particular difficulty, but state coverage still matters.

OPD extracts more signal from the same query. Both methods draw fresh rollouts. Over 1000 steps, OPD closes 72% of its teacher gap and gains over twice as much validation accuracy as RLVR. Its advantage persists at matched rollout-token budgets, excluding teacher computation.

As RLVR solves the query, outcome variation diminishes; unanimous groups provide no GRPO advantage. OPD retains local teacher–student gaps after solving it, although their absorption slows. The bottlenecks thus differ: limited outcome variation versus slow absorption of dense supervision. OPD targets a teacher distribution, whereas RLVR is not tied to one.

7 Related Work

On-policy distillation. OPD supervises student rollouts with a teacher (Song & Zheng, 2026). MiniLLM (Gu et al., 2024) used reverse KL and policy gradient; GKD (Agarwal et al., 2024) generalized the divergences and on-/off-policy mixtures; Yang et al. (2026) cast OPD as dense KL-constrained RL. Frontier post-training stacks use OPD (Yang et al., 2025; Zeng et al., 2026; Xiao et al., 2026; Xu et al., 2026; Blakeman et al., 2026), including multi-teacher routing to domain specialists (Appendix D). Algorithmic analyses characterize its early-locked, low-rank update subspace (Cai et al., 2026; Shen et al., 2026) and token-level policy-gradient form (Ko et al., 2026), alongside distinctions between dense SFT updates and localized RL subnetworks (Mukherjee et al., 2026; Zhu et al., 2025). Other work examines shared thinking patterns and teacher strength (Li et al., 2026), token learnability (Armandpour et al., 2026; Wang et al., 2026b), and failure modes and remedies (Zhu et al., 2026; Fu et al., 2026). These accounts take the training set as given. We vary the query count and identify the student's slow absorption of teacher supervision as the bottleneck.

Data-efficient reasoning post-training. Under sparse outcome-reward RL, one example can improve a model for over a thousand steps (Wang et al., 2026a), while LIMR (Li et al., 2025) reduces query count by ranking learning impact. For supervised fine-tuning, LIMA (Zhou et al., 2023), LIMO (Ye et al., 2025), and s1 (Muennighoff et al., 2025) achieve competitive alignment or reasoning with roughly a thousand curated examples. Other work schedules OPD supervision over reasoning prefixes (Zhang et al., 2026a) or selects subsets by influence or quality (Xia et al., 2024; Chen et al., 2024b; He et al., 2025a;c). Whereas RL relies on outcome rewards and SFT on reference solutions, OPD provides dense supervision even on incorrect rollouts (Armandpour et al., 2026). Our one-shot results include queries the student never solves (Section 3), shifting attention from signal availability to how quickly the student absorbs it (Section 5.2).

Synthetic and unsupervised data. Self-Instruct (Wang et al., 2023) and Evol-Instruct (Xu et al., 2024) synthesize instructions, while Magpie (Xu et al., 2025) generates instruction–response pairs from a chat-template prefix. Label-free RL (He et al., 2026) uses confidence or entropy (Zhao et al., 2026b; Agarwal et al., 2026), rollout majority agreement (Zuo et al., 2026), or rewards from unlabeled corpora (Dong et al., 2025); other approaches let models propose their own tasks (Zhao et al., 2026a; Zweiger et al., 2026). These methods relax requirements on rewards or query sources. Our template- and WildChat-OPD results (Section 6.1) show that off-domain chat logs and content-free templates work nearly as well as curated problems: queries place the student in useful reasoning states, where the teacher supplies dense supervision.

8 Conclusion, Limitations, and Future Work

Conclusion. Motivated by one-shot RLVR (Wang et al., 2026a), we ask whether a single query is also enough for OPD, and find that it is: one-shot OPD keeps improving for hundreds of updates and recovers most of full-data OPD's gain. OPD is thus data-overfed but algorithm-starved. A query acts through the states its rollouts reach; one query already covers most of the full-data state space, and sixteen diverse queries match full-data training. What limits a run is the absorption rate, which falls at the same pace on one query as on all 17k. Queries therefore matter through the teacher-supervised states they induce, shifting data design from collecting problems to selecting queries by the states they reach.

Limitations. State coverage is a proxy relative to full-data rollouts; it weights clusters equally, ignoring visitation frequency and remaining teacher signal. The cause of the falling absorption rate remains open.

Future work. Three directions follow: estimating state coverage without a full-data reference run, improving step efficiency through constrained batch reuse or teacher-signal weighting, and extending the setting to more teachers and domains, larger students, and long contexts.

AI USE STATEMENT

In this work, we used large language models for code debugging and writing assistance during manuscript preparation. We did not use generative AI tools to generate experimental results, fabricate data, or produce the core technical contributions of this paper. The research design, experimental implementation, data analysis, and all scientific claims are our own. All AI-assisted text was reviewed and verified by us, and we take full responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects research, does not release new datasets containing personal or sensitive information, and does not present methodologies with foreseeable harmful applications. All experimental results are based on publicly available datasets and open-source models. The training data and evaluation benchmarks used in this work are either released under permissive licenses or commonly used in academic research, spanning mathematical reasoning, code generation, instruction following, and agentic tool-use tasks.

REPRODUCIBILITY STATEMENT

We provide the complete experimental code as anonymized supplementary material, including the training recipes for GRPO, OPD (full, one-shot, and template modes), and MOPD, the dataset processing scripts, and the evaluation scripts. A README documents the environment, the available configurations, and the list of hyperparameters. In the paper, Section 3.1 describes the training and evaluation protocols; the student–teacher pairs and training datasets are listed in Table 1, and the one-shot queries in Table 2. The instruction-following teacher, the only model we post-train rather than download, is trained from a publicly available base model using the procedure described in Appendix A.1; hyperparameters for OPD and MOPD are given in Tables 3 and 9. The recovery and alignment metrics are defined in Section 2, the state-coverage construction and its robustness checks in Appendix B.1, and the absorption-rate estimator and the conventions under which it is read in Appendix C.1. All runs use a single 8-GPU node.

REFERENCES

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

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *Advances in Neural Information Processing Systems*, 38:107150–107180, 2026.

Mohammadreza Armandpour, Fatih Ilhan, David Harrison, Ajay Jaiswal, Duc NM Hoang, Fartash Faghri, Yizhe Zhang, Minsik Cho, and Mehrdad Farajtabar. Unmasking on-policy distillation: Where it helps, where it hurts, and why. *arXiv preprint arXiv:2605.10889*, 2026.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2025.

Aaron Blakeman, Aaron Thomas, Aastha Jhunjhunwala, Abhibha Gupta, Abhinav Khattar, Adam Rajfer, Adi Renduchintala, Adil Asif, Aditya Vavre, Adriana Flores Miranda, et al. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2606.15007*, 2026.

Yuchen Cai, Ding Cao, Liang Lin, Chunxi Luo, Xin Xu, Kai Yang, Weijie Liu, Saiyong Yang, Tianxiang Zhao, Guangzhong Sun, et al. Learning to foresee: Unveiling the unlocking efficiency of on-policy distillation. *arXiv preprint arXiv:2605.11739*, 2026.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 4(5), 2024a.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. In *International Conference on Learning Representations*, volume 2024, pp. 34767–34797, 2024b.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bingxiang He, Ning Ding, Cheng Qian, Jia Deng, Ganqu Cui, Lifan Yuan, Haiwen Hong, Huan-ang Gao, Longtao Huang, Hui Xue, et al. The right time matters: Data arrangement affects zero-shot generalization in instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 222–243, 2025a.

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, et al. Justrl: Scaling a 1.5 b llm with a simple rl recipe. *arXiv preprint arXiv:2512.16649*, 2025b.

Bingxiang He, Wenbin Zhang, Jiaxi Song, Cheng Qian, Zixuan Fu, Bowen Sun, Ning Ding, Haiwen Hong, Longtao Huang, Hui Xue, et al. Air: A systematic analysis of annotations, instructions, and response pairs in preference dataset. *arXiv preprint arXiv:2504.03612*, 2025c.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, et al. How far can unsupervised rlvr scale llm training? In *International Conference on Learning Representations*, volume 2026, pp. 14823–14865, 2026.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

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. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 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.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025.

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

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*, 2024.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482, 2024.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level. *Notion Blog*, 1, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *Advances in Neural Information Processing Systems*, 38:132119–132138, 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

OpenBMB. Ultradata-sft-2605, 2026. URL `https://huggingface.co/datasets/openbmb/UltraData-SFT-2605`.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces. `https://huggingface.co/datasets/open-r1/codeforces`, 2025.

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

Zhennan Shen, Yanshu Li, Qingyu Yin, Chak Tou Leong, Zhilin Wang, Yanxu Chen, Rongduo Han, Sunbowen Lee, and Yi R Fung. On the geometry of on-policy distillation. *arXiv preprint arXiv:2606.07082*, 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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

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

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *Advances in Neural Information Processing Systems*, 38: 122721–122764, 2026a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Yuanyi Wang, Su Lu, Yanggan Gu, Pengkai Wang, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. Not all disagreement is learnable: Token teachability in on-policy distillation. *arXiv preprint arXiv:2605.26844*, 2026b.

Yudong Wang, Zixuan Fu, Hengyu Zhao, Chen Zhao, Chuyue Zhou, Xinle Lin, Hongya Lyu, Shuaikang Xue, Yi Yi, Yingjiao Wang, et al. Data science and technology towards AGI part I: Tiered data management. *arXiv preprint arXiv:2602.09003*, 2026c.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

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*, 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.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *International Conference on Learning Representations*, volume 2024, pp. 30745–30766, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. In *International Conference on Learning Representations*, volume 2025, pp. 76346–76382, 2025.

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.

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.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, et al. Rlve: Scaling up reinforcement learning for language models with adaptive verifiable environments. *arXiv preprint arXiv:2511.07317*, 2025.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D Lyng, Sanjit Singh Batra, and Robert E Tillman. Fast and effective on-policy distillation from reasoning prefixes. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25553–25569, 2026a.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Xuan Li, Xiao Feng, Jiangchao Yao, Bo Han, et al. Co-rewarding: Stable self-supervised rl for eliciting reasoning in large language models. In *International Conference on Learning Representations*, volume 2026, pp. 82520–82558, 2026b.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026a.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In *International Conference on Learning Representations*, volume 2024, pp. 34590–34605, 2024.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations*, volume 2026, pp. 2548–2581, 2026b.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, et al. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*, 2025.

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

Adam Zweiger, Jyo Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *Advances in Neural Information Processing Systems*, 38:74084–74115, 2026.

Table 2: Mathematical queries used in the query-difficulty experiment, labelled by the student's pass rate before OPD training.

Difficulty	ID	Query
Easy	`t59`	In triangle ABC, let D be the foot of the altitude from A. Suppose that $AD = 4$, $BD = 3$, $CD = 2$, and AB is extended past B to a point E such that $BE = 5$. Determine the value of CE^2.
Medium	`t22`	Let S be the set of triples (a, b, c) of non-negative integers such that $a + b + c$ is even. Determine the value of $\sum_{(a,b,c)\in S} \frac{1}{2^a 3^b 5^c}$. This sum can be expressed as $\frac{m}{n}$, where m and n are relatively prime positive integers. Compute $m + n$.
Hard	`t56`	Byan is playing a game called "raven, raven, falcon" with his three friends. His friends sit in a circle, and Byan walks clockwise around them, tapping each friend he passes on the head and saying either "raven" or "falcon," each with probability $\frac{1}{2}$. The game ends when Byan has said "falcon" twice. The probability that one of his friends will be called a "falcon" twice can be expressed as $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $100m + n$.



Figure 10: Fraction of the 64 rollouts per step that solve the training query itself, for the three queries of Table 2.

A Details for Section 3

A.1 Experimental Setup and Training Queries

Instruction-following setup. Unlike the other primary-domain teachers, the instruction-following teacher is trained specifically for our experiments. Starting from DeepSeek-R1-Distill-Qwen-1.5B, we post-train the model on synthetic instruction-following tasks with programmatically checkable constraints. We use GRPO with rule-based outcome rewards that indicate whether the response satisfies the target-turn constraints. Training samples four completions per prompt at temperature 0.9, uses a maximum prompt length of 4,096 tokens and a maximum response length of 16,384 tokens, and optimizes with AdamW at learning rate 1×10^{-6}, train batch size 16, and clip ratio 0.2. The step-600 checkpoint is used as the instruction-following teacher, improving IFBench (Pyatkin et al., 2026) from 14.33 to 20.65 and Multi-IF final-turn accuracy from 20.84 to 28.58.

For this domain, "full-data" denotes the subset sampled from UltraData-SFT-2605 (OpenBMB, 2026) rather than the complete dataset, and the one-shot query is drawn at random from that subset.

Mathematical queries for one-shot OPD (Table 2, Figure 10). We sample 8 rollouts from the untrained student for each of the first 64 queries of DAPO-Math-17K and assign difficulty by the resulting pass rate, which selects `t59` (8/8), `t22` (4/8), and `t56` (0/8) as the easy, medium, and hard queries. How often each is solved during training follows that assignment: the easy query stays near 1.0 throughout, the medium one rises from 0.36 to about 0.89, and the hard one is never solved in 300 steps, yet all three produce comparable benchmark gains in Figure 4.

Table 3: Default hyperparameters for OPD.

Item	Value
Rollout batch size	64
Mini batch size	64
Responses per prompt	1
KL coefficient	0.0
LogProb top-K	0 (sampled token only, default) / 16 (top-16 tokens)
Top-K source	student (only when top-K is enabled)
Loss aggregation	token-mean
Training temperature	1.0
Top-p	1.0
Optimizer	AdamW (β_1=0.9, β_2=0.999, wd = 0.01)
Learning rate	1e-6
Gradient clip norm	1.0
Max prompt length	1024 (math); 4096 (code, IF, agentic)
Max response length	7168 (math, code, IF); 2048 (agentic)

Table 4: Out-of-domain evaluation of models trained on mathematical reasoning.

Model	LCB v6 avg@3	Multi-IF turn-3
R1-Distill-1.5B (student)	21.96	21.67
+ 1-shot step300	23.55	23.32
+ 1-shot step1000	24.75	23.27
JustRL-1.5B (teacher)	28.74	25.11

Default hyperparameters for OPD (Table 3). Unless otherwise noted, all experiments use these defaults. The top-K row covers both objectives: the mathematical one-shot runs of Section 3.2 use $K = 16$, and all other experiments use the sampled-token objective.

A.2 Out-of-Domain Capability Preservation

One-shot OPD applies hundreds of updates to a single training query with a KL coefficient of zero, so a natural concern is that the student overfits to that query and loses capability elsewhere. We test this directly by evaluating checkpoints trained on mathematical reasoning (DAPO-Math-17K) on code generation (LiveCodeBench v6) and instruction following (Multi-IF), two domains that never appear in training.

Table 4 shows that neither checkpoint degrades on either benchmark. Both improve on the base model, on code generation by 1.6 points at step 300 and 2.8 points at step 1000, and on instruction following by 1.7 and 1.6 points respectively. The hundreds of updates spent on one mathematical query therefore do not come at the cost of out-of-domain capability: the student generalizes rather than overfitting to the training query, and on both benchmarks the effect is positive. Both checkpoints also remain below the teacher on these benchmarks, so the improvement is partial rather than a recovery of the full gap.

A.3 Generality Across Model Families and Task Domains

The one-shot effect is robust across model families. Figure 11 shows that one-shot OPD improves mathematical reasoning across all three student–teacher families. At the final checkpoint, validation accuracies averaged over MATH-500 and AMC 2023 increase from 77.1 to 85.5 for R1-Distill-1.5B, from 28.2 to 40.2 for Llama-3B-It, and from 70.8 to 82.4 for OLMo-7B-It-DPO. Unlike the three-benchmark average used for the main mathematical results, this comparison excludes AIME 2025 because the Llama pair scores near zero on it.

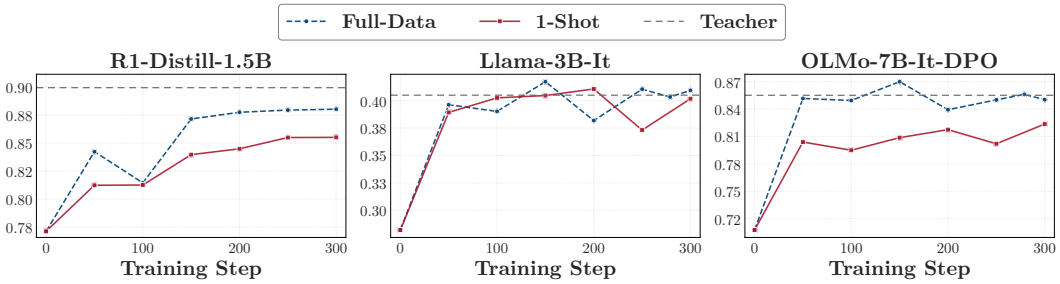

Figure 11: Validation accuracy of one-shot OPD across three student–teacher families, averaged over MATH-500 and AMC 2023. AIME 2025 is excluded because the Llama pair scores near zero on it.

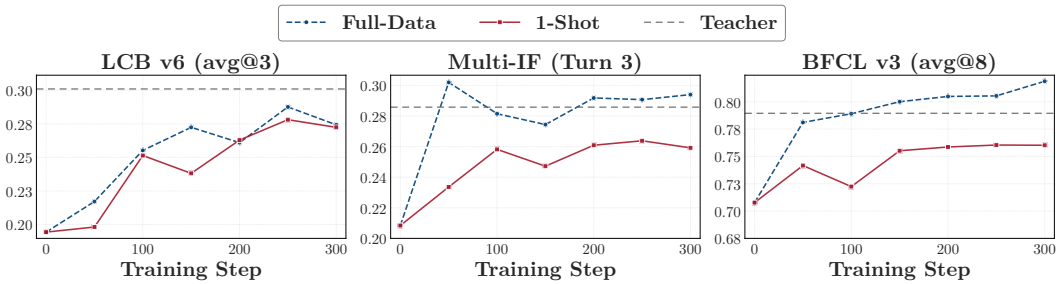

Figure 12: Validation accuracy of one-shot OPD across code generation (LiveCodeBench v6), instruction following (Multi-IF), and agentic tool use (BFCL v3).

The one-shot effect extends beyond mathematical reasoning. Figure 12 shows that one-shot OPD recovers 73%, 66%, and 64% of the teacher–student gaps on code generation, instruction following, and agentic tool use, respectively, under the evaluation protocols in Section 3.1.

B Details for Section 4

B.1 Training-State Coverage

Construction. Rollouts are collected by optimization step rather than at random, so that every stage of training is equally represented and the budget is comparable across settings, which run for the same number of steps: at each step from 1 to 500 of the full-data run we take 5 rollouts, three building the clusters and two forming full data (held-out), for 12,000 reference and 8,000 held-out states. PCA reduces these to 50 dimensions, retaining 71.9% of the variance, before K-means. Every evaluated setting is then measured on that same schedule, drawing 2 of the 5 rollouts at each step and accumulating over the run, so the step-300 budget of 600 rollouts matches the held-out split. We repeat the draw 30 times; bands and error bars are one standard deviation over these repetitions, which reflects rollout subsampling rather than variation across trained models.

16-shot is close to full data, and this closeness is stable across K, reference construction, and state positions (Tables 5 and 6, Figure 13). Coverage depends on how finely the reference states are clustered and on which states define the clusters, so we vary both. Table 5 repeats the analysis at $K \in \{50, 100, 200, 500\}$: finer clusters lower absolute coverage, but 16-shot remains close to full data in every case. Table 6 rebuilds the clusters from the union of 2,000 rollouts per setting instead of from full-data states alone; the gap between 16-shot and full data persists, so the closeness is not an artifact of the reference construction. Because states from all settings help define these pooled clusters, pooled coverage is not a recovery measure, and its rows are to be read only against each other. Figure 13 varies the positions read from each rollout, comparing the final non-padding token and fixed absolute positions 2,000 and 4,000 against the 8 relative positions of the main analysis. The 16-shot–full gap stays small across all four variants. The fixed-position variants land close to the main construction, whereas the final token gives markedly lower coverage when there are few

Table 5: Full-data-anchored training-state coverage at step 300 under different cluster counts K.

K	1-shot	4-shot	16-shot	Full data (held-out)
50	0.755	0.829	1.000	1.000
100	0.727	0.803	1.000	1.000
200	0.715	0.798	0.989	1.000
500	0.564	0.695	0.904	0.956

Table 6: Training-state coverage under alternative cluster constructions, at $K = 200$.

Reference construction	1-shot	4-shot	16-shot	Full data (held-out)
Full-data anchored	0.663	0.773	0.953	0.998
Equal-size pooled	0.750	0.845	0.958	0.966

queries, which is what one expects of a position governed by response length and termination; we therefore read 8 relative positions, which sample throughout a response of any length.

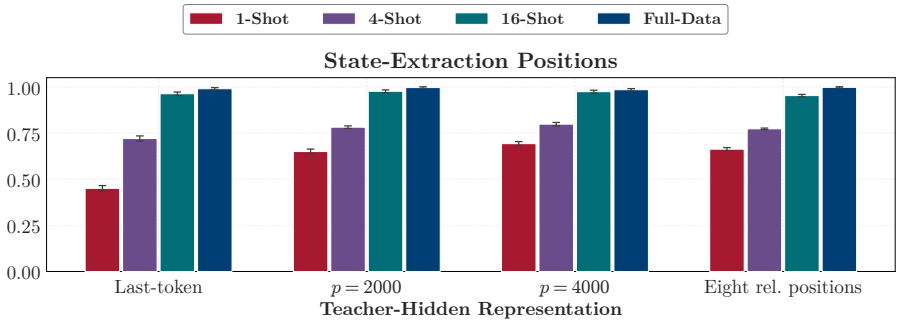

Figure 13: Training-state coverage under alternative state-extraction positions, at $K = 200$. Error bars show one standard deviation over rollout subsamples.

One-shot has a heavier tail of distant states, but is not separable from full data (Table 7). Nearest-centroid assignment places a state in some cluster however far it lies from every full-data state, so coverage alone cannot tell a setting that fills the reference space thinly from one that leaves it. We therefore also measure, for each state, the mean cosine distance to its ten nearest reference states in the unreduced 1536-dimensional space, and call a state off-reference when that distance exceeds the 95th percentile of the held-out full-data states. One-shot OPD exceeds the threshold more often than the other settings, so its distances have a heavier upper tail. But an AUROC of 0.552 against held-out full-data states is close to chance, so one-shot states are not separable from full-data states as a whole.

Table 7: Reference-distance diagnostics across query-set sizes. AUROC compares each setting with the held-out full-data states. The last row is the complete full-data pool, including the rollouts that built the clusters, so its exceedance rate need not equal the 5% the threshold is set at.

Setting	Exceedance (%)	Mean distance	AUROC
1-shot	17.2	0.283	0.552
4-shot	4.4	0.218	0.477
16-shot	3.8	0.231	0.512
Full data (all)	4.0	0.215	0.476

B.2 Controls on the Training Data

Diversity in 16-shot OPD (Figure 14). Every query on the ladder of Section 4.2 represents a different semantic cluster, so it raises diversity and count together. Holding the count at 16 and drawing the second set from a single cluster separates them: the 16 cluster representatives reach 70.9 validation accuracy at step 300 against 69.9 for the single-cluster set, which is about what 4-shot OPD reaches with a quarter as many queries, and their state coverage separates the same way and does so early, 98.9% against a single-cluster set that flattens near 76.8% within 100 steps. Queries from one cluster keep returning the student to the part of the state space it has already seen.

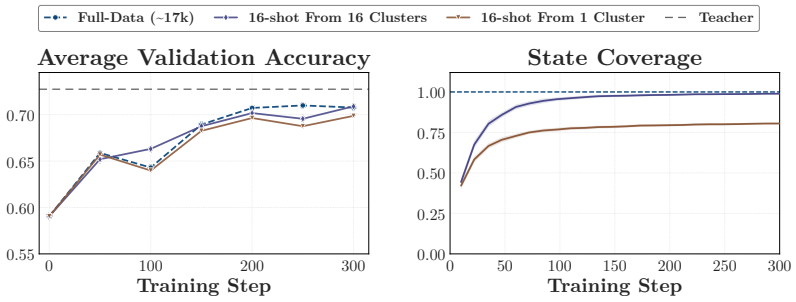

Figure 14: Two 16-shot query sets that differ only in diversity, against full-data OPD.

Query scheduling in 2-shot OPD (Figure 15). Holding a two-query set fixed and varying only when each query is shown, the sequential schedule trains on the hard query for the first 300 steps of a 600-step budget and on the medium query for the rest, while the mixed schedule puts 32 copies of each in every batch. At step 600 they reach 68.8 and 69.1, both above the 67.4 of the one-shot baseline they share, so ordering a fixed query set barely matters next to what the second query itself buys. These runs train longer than those of Section 4.2 and take the hard query as their baseline, so neither their horizon nor their baseline is interchangeable with Figure 6.

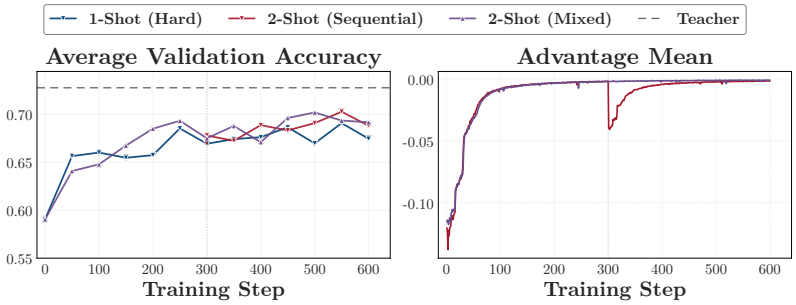

Figure 15: Effect of query scheduling in 2-shot OPD. The vertical dotted line marks the query switch in the sequential schedule.

C Details for Section 5

C.1 Estimating the Absorption Rate and the Measurement Window

Alignment metrics. Section 5.2 measures under three conventions. Why each one:

- $[t/1.4, 1.4t]$. A difference between adjacent steps is mostly noise once d moves slowly, so v_t is read as the slope of $\log d$ across the endpoints of this window, using the median d near each. Only centers whose window fits inside the measured range are plotted, which is why the main text reads the rate at steps 50 and 200 rather than at step 300.

- $\boldsymbol{d_t/d_{30}}$. Each run measures its distance on the states it visits and the four sit at different levels by step 30, so plotting each against its own value there lets the curves be compared by shape rather than by level.
- **Start at step** 30**.** Gradient clipping binds until then, and a clipped update is not the update the objective asked for: v_t inside that window measures what one *fixed-size* step buys rather than what one step buys, which is why the measured rate rises there rather than falls.

Per-run numbers and conventions. Table 8 collects the readings Section 5.2 quotes from this range: the proportion of the step-30 distance still left at step 300, and the factor by which the absorption rate falls between steps 50 and 200.

Table 8: Alignment dynamics over steps 30–300 for on-policy OPD, at the default learning rate.

Setting (queries)	1	4	16	Full data
d_{300}/d_{30}	0.16	0.17	0.18	0.22
v_{50}/v_{200}	5.6	5.2	4.2	6.4

C.2 Ablation on Fixed States

On-policy training continually produces new states as the student changes. To test whether this supply is required for prolonged learning, we hold the training states fixed.

Setup. We compare two one-shot runs that differ only in where their training states come from. The on-policy run draws fresh rollouts from the current student at every step. The off-policy run reuses the 64 trajectories of Section 4.2, sampled once from the initial student, for every update, so its training states stay fixed while the student changes.

Even fixed states support hundreds of steps of learning. Figure 16 shows that the off-policy run gains validation accuracy for about 200 steps before plateauing. Its top-16 token overlap rises and its overlap-token advantage approaches zero over hundreds of steps as well. A fixed set of states therefore supports prolonged learning, so continued state acquisition is not required to explain the run's duration. This finding concerns learning duration and does not imply that fresh states cannot improve final validation accuracy.

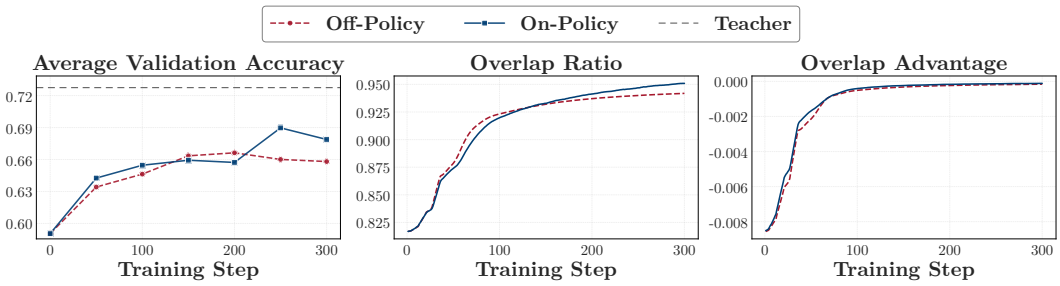

Figure 16: Validation accuracy and alignment metrics for the on-policy and off-policy one-shot runs.

C.3 The Learning Rate Rescales the Step Axis

We repeat on-policy one-shot OPD at half, the default, and twice the default learning rate, holding every other setting fixed, and read all three over the same 300 steps as Section 5.2.

The learning rate rescales the step axis and leaves the slowdown untouched. A larger learning rate reaches any given distance sooner, the three curves separated by roughly the ratio of the learning rates (Figure 17). Changing the horizontal axis from the step t to $u = (\mathrm{lr}/10^{-6})\, t$ brings them onto one curve: the spread between the three runs narrows from about $2.2\times$ to about $1.1\times$. The learning

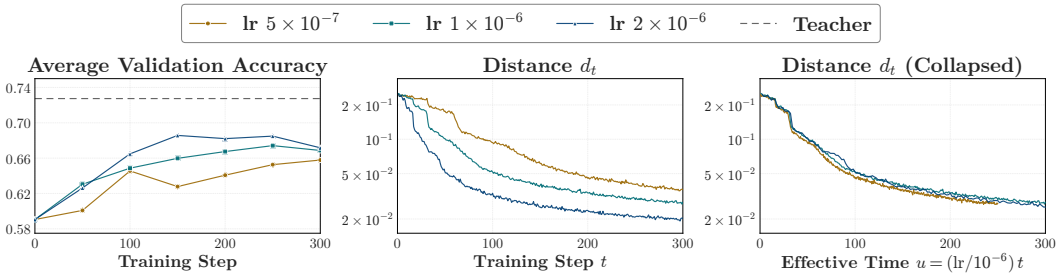

Figure 17: Validation accuracy and teacher–student distance for one-shot OPD at half, one, and twice the default learning rate. Distance is on a logarithmic vertical axis against linear time.

rate therefore sets only how fast a run moves along that one curve, not the shape of the curve itself. It bends flatter as it goes, just as the four query settings of Section 5.2 do against the step count. A proportionally larger learning rate therefore shortens the run proportionally, over the fourfold range we tested, without reaching whatever makes alignment slow in the first place.

D One-Shot Extends to Multi-Teacher OPD

Section 4.2 showed that a small, semantically diverse query set can match full-data OPD within one domain. Modern post-training pipelines often use *multi-teacher on-policy distillation* (MOPD) (Xiao et al., 2026; Xu et al., 2026), where one student is trained on several domains in a single run and each query is routed to its domain teacher. This section asks *whether the same query-diversity result carries over: can a small, diverse query set for each domain match full-data MOPD?*

Takeaways

16 semantically diverse queries within each domain are sufficient to match full-data MOPD, which reaches nearly the same average validation accuracy as separate full-data OPD.

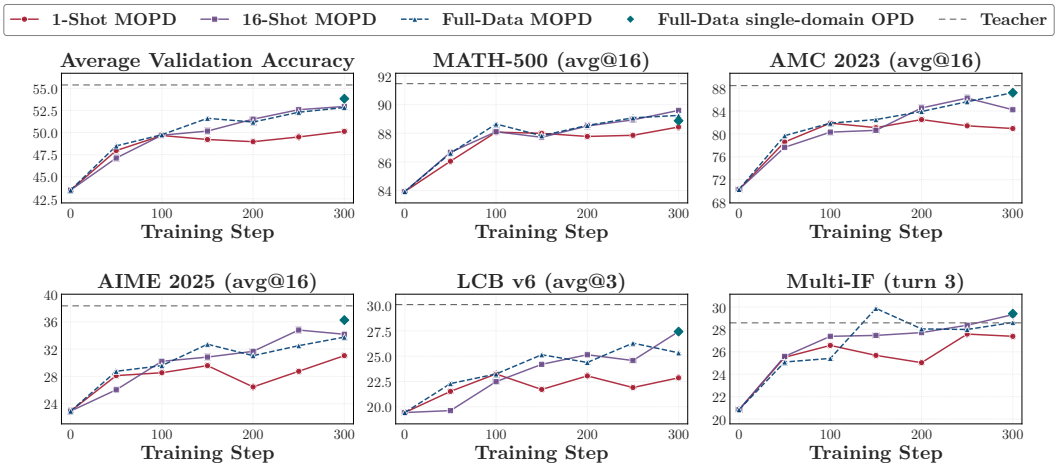

Figure 18: Validation accuracy of one-shot, 16-shot, and full-data MOPD.

Setup. We train one R1-Distill-1.5B student with MOPD on mathematical reasoning, code generation, and instruction following. The domain teachers from Section 3.1 are JustRL-1.5B for mathematics, Nemotron-1.5B for code, and UltraData-IF-1.5B for instruction following. We exclude agentic tool use because it uses a different student, Qwen-Coder-1.5B. The three settings are one-shot, 16-shot, and full-data MOPD. As in Section 4.2, the 16-shot setting selects one representative from each of 16 BGE-M3 (Chen et al., 2024a) semantic clusters fitted separately within each domain. All settings run for 300 steps with the hyperparameters in Table 9.

Full-data MOPD matches separate full-data OPD. At step 300, full-data MOPD raises the average validation accuracy from 43.5 to 52.8, recovering 79% of the teacher–student gap. The three corresponding full-data OPD runs reach 53.8. This 1.0-point difference shows that MOPD remains effective when all three domains are trained in a single run and establishes full-data MOPD as the reference for the query-diversity comparison below.

16 diverse queries per domain match full-data MOPD. Increasing the query count from 1 to 16 per domain raises the average validation accuracy from 50.1 to 52.9 at step 300. Full-data MOPD reaches 52.8, so 16-shot MOPD recovers 101% of full-data MOPD's gain. The result is not driven by one domain: calculated separately, the corresponding proportions are 93% for mathematics, 136% for code generation, and 109% for instruction following. Because 16-shot and full-data MOPD use the same rollout and optimization budgets, expanding each domain's query set beyond 16 diverse queries produces no additional average gain in this setting.

Together, these results are consistent with a state-space account for MOPD: semantically diverse queries can induce complementary states within each domain, allowing 16 queries per domain to match full-data MOPD.

Hyperparameters. Table 9 shows the default hyperparameters for MOPD.

Table 9: Default hyperparameters for MOPD (multi-teacher on-policy distillation).

Item	Value
Number of teachers	3 (math / code / IF)
LogProb top-k	0
KL coefficient	0.0
Loss aggregation	seq-mean-token-mean
Rollout batch size	64
Mini batch size	64
Responses per prompt	1
Training temperature	1.0
Top-p	1.0
Optimizer	AdamW (β_1=0.9, β_2=0.999, wd = 0.01)
Learning rate	1e-6
Gradient clipping	1.0
Max prompt length	4096
Max response length	7680

E Details for Section 6

E.1 Template and WildChat OPD in Code Generation

Section 6.1 reports the four input conditions in mathematics. Here we report the same four in code generation.

Setup. The student is R1-Distill-1.5B, the teacher DeepCoder-1.5B-Preview (Luo et al., 2025), and the baseline full-data OPD on TACO (Li et al., 2023). The base template, the system template, and the WildChat queries are exactly those of Section 6.1; the system template's one-line instruction is the competitive-programming instruction of Figure 19b. We evaluate on LiveCodeBench v6 under avg@3.

(a) Base template

```
<|begin_of_sentence|><|User|><think>\n
```

(b) System template — Code

```
<|begin_of_sentence|>Solve competitive
programming problems.  Read from
stdin, write to
stdout.<|User|><think>\n
```

Figure 19: The two content-light templates used as OPD training inputs.

Results. Figure 20 shows the four conditions ending within 1.7 points of one another, and the token accounting separating them much further than the step accounting does: over the same 500 steps the baseline spends 277M rollout tokens against 18M for the base template, with the other two in between. Actor entropy orders the conditions exactly as it does in mathematics, with WildChat highest and the system template lowest. The one difference is that no condition falls behind here, whereas WildChat costs about a point in mathematics.

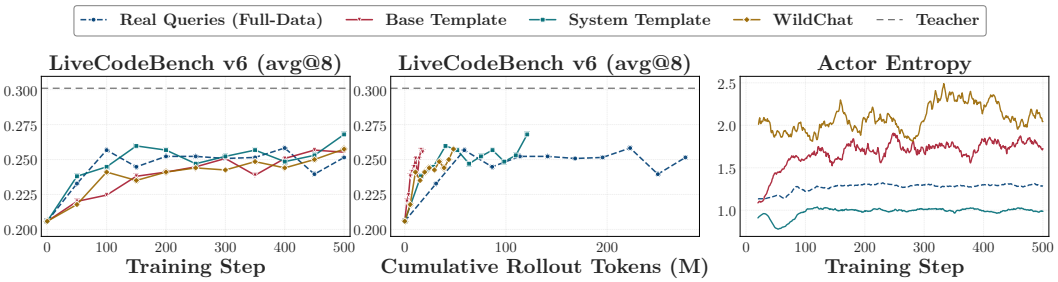

Figure 20: Validation accuracy and actor entropy for code-generation OPD.