OPMD: ON-POLICY MECHANISM DISTILLATION BEYOND OUTPUT SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) improves language-model by supervising a student on prefixes sampled from its own policy, but its supervision remains confined to the output space. Consequently, OPD transfers what a teacher predicts without supervising the intermediate computation that produces those predictions. Simply distilling intermediate representations like representation matching is insufficient since teacher and student representations may differ geometrically, and even well-aligned hidden states need not be usable by the student's subsequent computation. We introduce On-Policy Mechanism Distillation (OPMD), which transfers teacher-derived intermediate computation while explicitly accounting for its compatibility with the student. OPMD first constructs a student-compatible teacher that transforms teacher representations into intermediate states usable by the student's downstream computation, then it internalizes these states from on-policy trajectories. OPMD achieves the highest average accuracy among the evaluated distillation baselines in all three teacher–student settings, and generalizes to multi-teacher distillation and cross-family supervised fine-tuning. Further analyses show that mechanism supervision yields more effective reasoning and meaningfully alters the student's intermediate computation and solution strategies, supporting that it transfers information beyond output supervision alone. Together, these results establish mechanism-level supervision as an effective way to move on-policy distillation beyond output supervision, enabling students to learn inner mechanisms they can use and internalize.

1 INTRODUCTION

On-policy distillation supervises a student on textual prefixes sampled from its own policy, addressing a key mismatch between training and deployment (Agarwal et al., 2024; Gu et al., 2024). However, agreement on observed next-token distributions does not establish agreement about how evidence is represented and transformed across layers. Models with similar next-token distributions can organize evidence differently in the representation space and rely on different computational pathways at depth (Belrose et al., 2025; Nikankin et al., 2025). Output-level agreement therefore does not uniquely determine the internal computation learned by the student. Here, textual prefixes serve as the states of the autoregressive decision process. We use *latent states* for the network's internal hidden representations. Output supervision constrains predictions but only indirectly constrains the latent states used to produce them. Particularly for complex reasoning tasks, output-level supervision may therefore leave valuable intermediate computational information untransferred (Jiao et al., 2020a; Sun et al., 2019; Wang et al., 2020; 2021).

This raises a natural question: *can the latent states of a stronger teacher provide actionable information that improves the student's subsequent computation*? Our preliminary analysis provides direct evidence that intermediate latent states contain actionable information for reasoning. We evaluate on AMC 2023 (LI et al., 2024) and use cross-model activation patching (Prakash et al., 2024), replacing the Base model's hidden state at a selected layer with the corresponding state from its RL-optimized counterpart while leaving all subsequent computation unchanged. As shown in Figure 1(a), patching latent states from an RL-optimized model into its base model substantially improves downstream task accuracy, demonstrating that RL-optimized latent states contain task-relevant information that remains usable by the Base model's downstream computation.

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.

However, simply transferring such information is non-trivial, especially when teacher and student have different hidden dimensions or representation geometries, typically requiring a mapping between their representation spaces. Existing feature-based distillation methods therefore rely on manually specified layer correspondences, learned projections, or specially designed relational objectives to make their intermediate latent states comparable (Jiao et al., 2020a; Wang et al., 2021; Liang et al., 2023). We further extend cross-model activation patching to heterogeneous teacher–student pairs by mapping the teacher's latent state into the student's hidden space. As shown in Figure 1(b), patching teacher latent states mapped by an MSE-trained projector can severely disrupt student generation, whereas states produced by a task-trained projector yield substantial gains across patching depths. These results suggest that minimizing hidden-state reconstruction error is not sufficient to produce representations that the student's downstream computation can use. What matters is therefore not only whether a teacher latent state can be mapped close to a student latent state, but whether the mapped latent state lies in a form that is computationally usable by the student's remaining layers. This motivates constructing teacher-informed latent states directly within the student's computational interface.

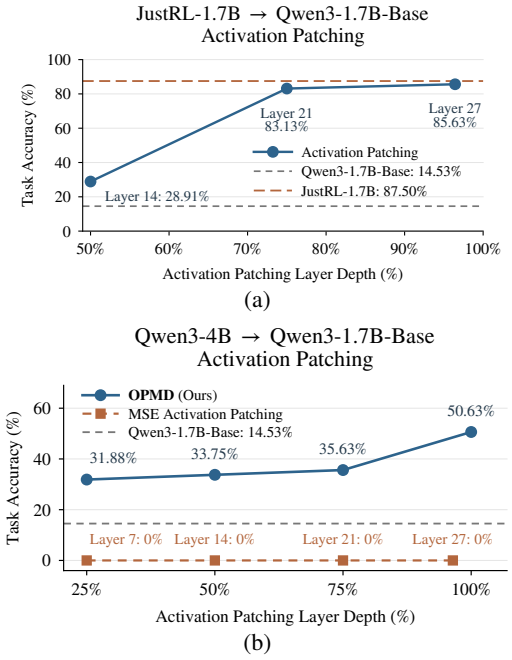

Figure 1: Teacher latent states and student compatibility. AMC accuracy under interventions at different depths.

Motivated by these observations, we do not directly force the student to reproduce the teacher's hidden states. Instead, we first construct a *student-compatible teacher* that transforms teacher information into intermediate states optimized to improve the student's downstream behavior and to be effectively used by its subsequent layers. These states then serve as richer computational targets for distillation. Concretely, a learned bridge injects teacher hidden states into a frozen student backbone, producing teacher-informed representations within the student's own computational interface. We then perform *on-policy internalization*, training a native student to reproduce these teacher-induced hidden trajectories on student-generated prefixes. At deployment, the teacher model and bridge are removed, leaving only the native student. This construction lets us test whether teacher-derived information becomes usable through the student's own parameters.

We evaluate OPMD on downstream reasoning tasks across three teacher–student pairs, demonstrating gains over output-level OPD. We further extend the framework to multiple domain teachers and validate mechanism distillation through cross-family SFT without logit-level KL supervision. Layer-level interventions further reveal changes in how students approach a problem, beyond changes in answer accuracy. In a case study, intermediate-layer updates recover solution-route preferences closer to the fully distilled model, while a substantial accuracy gap remains. These observations distinguish learning a preferred solution strategy from realizing the computation needed to execute it successfully.

Our contributions are threefold. First, we construct a student-compatible teacher and train the native student to internalize its latent states through on-policy supervision. Second, we evaluate this framework across teacher–student pairs, multiple domain teachers, and cross-family SFT. Third, we investigate whether different layer groups internalize different reasoning mechanisms, using their distinct solution-route preferences to examine how strategy selection relates to successful execution.

2 Related Work

On-policy language-model distillation. To mitigate exposure bias and distribution shift in autoregressive generation, on-policy distillation supervises the student on prefixes sampled from its own rollout policy (Gu et al., 2024; Agarwal et al., 2024), with recent variants exploring alternative divergence formulations and token-level feedback allocation (Xu et al., 2026; Liu et al., 2026). Most existing methods operate through output-level teacher signals, improving where and how the student receives supervision without directly constraining its intermediate computation. Recent work has begun extending the on-policy paradigm to representation-level supervision (Yang et al., 2026a). OPMD similarly adopts student-generated prefixes, but focuses on constructing teacher-informed intermediate targets that can be effectively used by the student's subsequent computation.

Representation-based distillation. Intermediate supervision ranges from hidden-state and layer-wise matching to attention and relational distillation (Romero et al., 2015; Sun et al., 2019; Jiao et al., 2020b; Wang et al., 2020; 2021; Liang et al., 2023). In reasoning models, OPRD aligns intermediate representations on student-generated trajectories, whereas PHF transfers hidden transitions and trajectory geometry from a reference-conditioned privileged teacher (Yang et al., 2026a; Li et al., 2026b). These approaches demonstrate the value of intermediate supervision, but primarily define transfer through representational correspondence or trajectory matching, without explicitly ensuring that the resulting states can be effectively used by the student's subsequent computation. OPMD addresses this distinction by constructing student-compatible intermediate targets before internalizing them into the native student.

Latent Interventions and Internalization. Intermediate computation can be modified by externally supplied signals. Activation patching replaces internal states to measure their causal effect on subsequent computation, while learned conditioning architectures such as ControlNet inject additional information into frozen pretrained backbones through trainable control pathways (Heimersheim & Nanda, 2024; Ghandeharioun et al., 2024; Zhang et al., 2023). A complementary line of work internalizes privileged or contextual information so that it is no longer required at inference time (Zhao et al., 2026; Ye et al., 2026). OPMD combines these perspectives by using teacher-conditioned intervention to construct useful intermediate targets and subsequently internalizing the resulting hidden trajectories into the native student.

3 Method

OPMD goes beyond direct representation matching by transferring teacher-derived intermediate information according to its effect on the student's subsequent computation. This raises two distinct requirements: the transferred information must be expressed in a form that the student can productively use, and the resulting computation must ultimately be realizable by the student itself. OPMD addresses these requirements through student-compatible teacher construction followed by on-policy internalization.

As illustrated in Fig. 2, Stage I, student-compatible teacher construction, makes teacher information usable by the student's existing computation. Stage II, on-policy internalization, trains the native student to reproduce the resulting latent states without external teacher intervention. We address these problems sequentially.

3.1 Preliminaries

On-policy language-model distillation. Let p_{θ_T} and p_{θ_S} denote an autoregressive teacher model and a student model, parameterized by θ_T and θ_S, respectively. On-policy distillation (Gu et al., 2024; Agarwal et al., 2024) aligns the student's predictions with the teacher's on student-generated prefixes. Our OPD baseline uses reverse KL computed over the full vocabulary:

$$\mathcal{L}_{\text{OPD}}(\theta_S) = \mathbb{E}_{x\sim\mathcal{D},\, \hat{y}\sim p_{\theta_S}(\cdot|x)} \left[\sum_{t=1}^{|\hat{y}|} D_{\text{KL}} \left(p_{\theta_S}(\cdot \mid c_t) \,\|\, p_{\theta_T}(\cdot \mid c_t) \right) \right], \tag{1}$$

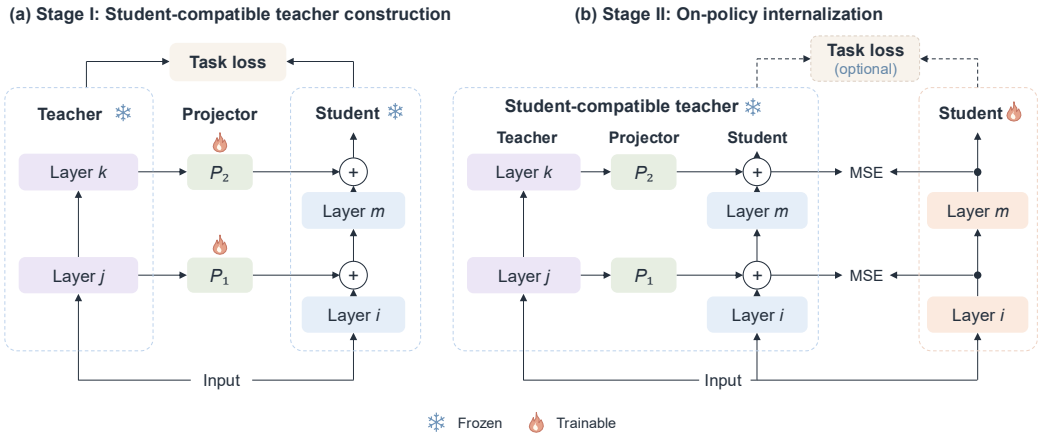

Figure 2: Overview of OPMD. Stage I learns layer-wise projectors to inject teacher representations into a frozen student, constructing a student-compatible teacher. Stage II performs on-policy internalization by aligning the native student's latent states with those of the student-compatible teacher.

where $c_t = (x, \hat{y}_{<t})$ is the prompt concatenated with the student's generated prefix up to step t. While this objective resolves exposure bias by matching the training and deployment prefix distributions, it provides only output-level supervision.

Geometric hidden-state matching. To supervise intermediate computations, prior works (Wang et al., 2021; Yang et al., 2026a) attempt to match the hidden representations of the teacher $h_j^T(c)$ and student $h_i^S(c)$ at selected layers. When architectures differ, this typically involves learning a projection function P to minimize a geometric distance metric, such as Mean Squared Error (MSE):

$$\mathcal{L}_{\text{MSE}}(P) = \mathbb{E}_c \left[\left\| P(h_j^T(c)) - h_i^S(c) \right\|_2^2 \right]. \tag{2}$$

However, minimizing geometric distance in representation space does not ensure that the projected teacher state $P(h_j^T(c))$ is functionally meaningful or computationally realizable by the student's subsequent downstream layers. OPMD explicitly departs from Equation 2 by optimizing the injection of teacher states based entirely on their *downstream functional utility*.

3.2 On-Policy Mechanism Distillation

On-Policy Mechanism Distillation (OPMD) goes beyond direct representation matching by transferring teacher-derived intermediate information according to its effect on the student's subsequent computation.

Student-compatible teacher construction. Let p_{θ_T} and $p_{\theta_S^0}$ denote the teacher and initial student, respectively. Both models remain frozen throughout Stage I. For a causal text prefix c, let $h_j^T(c)$ denote the teacher latent state at layer j, and let $h_i^{S,-}(c)$ denote the student latent state immediately before an intervention at layer i. When teacher and student use different tokenizers, c denotes the same textual prefix independently tokenized by each model.

We define a set of teacher–student transfer-layer pairs $\mathcal{I} = \{(j_r, i_r)\}_{r=1}^R$. For each pair, a learned projector $P_{\phi_r} : \mathbb{R}^{d_T} \to \mathbb{R}^{d_S}$ maps the teacher representation to a residual update in the student's hidden space:

$$\tilde{h}_{i_r}^S(c) = h_{i_r}^{S,-}(c) + P_{\phi_r}\big(h_{j_r}^T(c)\big), \qquad (j_r, i_r) \in \mathcal{I}. \tag{3}$$

The resulting composition of the frozen teacher, learned projectors, and frozen student backbone defines the *student-compatible teacher*, whose predictive distribution we denote by $\tilde{p}_\phi(\cdot \mid c)$. Because the student backbone is fixed, the projectors must express teacher information in a form that can be acted on by the student's existing downstream computation.

Projector optimization. We optimize only the projector parameters ϕ, while keeping both the teacher and student backbone frozen. Rather than directly matching teacher and student hidden states, we train the projectors through the output behavior induced by the frozen student. When teacher and student share a tokenizer, we minimize reverse KL over the full vocabulary

$$\mathcal{L}_{\text{build}}(\phi) = \mathbb{E}_{c\sim\mathcal{C}}\left[D_{\text{KL}}\left(\tilde{p}_\phi(\cdot \mid c) \,\|\, p_{\theta_T}(\cdot \mid c)\right)\right], \tag{4}$$

where $\mathcal{C}$ denotes the training contexts. As in the OPD baseline, each KL term sums over the full vocabulary rather than using only the sampled token. For different tokenizers, we can instead train the projectors with SFT on teacher-generated responses tokenized by the student, keeping both backbones frozen. Appendix A.1 specifies these alternatives.

On-policy internalization. The student-compatible teacher still relies on the external teacher and projectors. We train a native student p_{θ_S} initialized from $p_{\theta_S^0}$ to reproduce its intermediate states without either component. Our on-policy experiments update student LoRA (Hu et al., 2022) parameters while keeping the backbone frozen. For each prompt $x \sim \mathcal{D}$, the current student samples $\hat{y} \sim p_{\theta_S}(\cdot \mid x)$. The native student and the frozen student-compatible teacher are then evaluated on the same student-generated textual prefixes.

Let $c_t^+ = (x, \hat{y}_{\le t})$ denote the prefix including response token $\hat{y}_t$. We match the native student's latent state $h_i^S(c_t^+; \theta_S)$ and the student-compatible teacher's latent state $\tilde{h}_i^S(c_t^+)$ at each response-token position without shifting.

$$\mathcal{L}_{\text{int}}(\theta_S) = \mathbb{E}_{x\sim\mathcal{D},\, \hat{y}\sim p_{\theta_S}(\cdot|x)}\left[\frac{1}{|\mathcal{H}|\,|\hat{y}|}\sum_{i\in\mathcal{H}}\sum_{t=1}^{|\hat{y}|}\frac{1}{d_S}\left\|h_i^S(c_t^+;\theta_S) - \text{sg}\left(\tilde{h}_i^S(c_t^+)\right)\right\|_2^2\right], \tag{5}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. Because the targets are produced within the frozen student backbone, they already lie in the student's representation space and require no additional cross-model mapping. Internalization supports a joint downstream task loss, but our on-policy experiments use only the latent-state objective in this stage. SFT instead combines mechanism supervision on teacher-response prefixes with supervised token prediction, without logit-level KL supervision. At inference time, both the teacher and projectors are discarded, leaving only the native student.

4 Experiments

We evaluate task performance, transfer across training settings, and the behavioral changes retained by weight patching. To facilitate reproducibility, we provide training hyperparameters, implementation details, and computational resources in Appendix A. We will release the code upon acceptance.

4.1 Does On-Policy Internalization Improve Distillation?

We first test whether OPMD improves mathematical reasoning over output-level distillation across different teachers and students.

Setting and baselines. Our main mathematical distillation experiments use DAPO-Math-17K (Yu et al., 2025) for training and AMC, AIME 2024 (LI et al., 2024), and AIME 2025 (Balunovic et al., 2026) for evaluation. The teachers are Qwen3-4B (Yang et al., 2025) and Qwen3-4B-Non-Thinking-RL-Math (RL-Math) (Yang et al., 2026b). We use Qwen3-1.7B-Base (abbreviated as Qwen3-1.7B) and Qwen2.5-3B-Instruct (abbreviated as Qwen2.5-3B) (Qwen et al., 2025) as students in the main comparisons. The main comparisons are Qwen3-4B $\rightarrow$ Qwen3-1.7B-Base, RL-Math $\rightarrow$ Qwen3-1.7B-Base, and RL-Math $\rightarrow$ Qwen2.5-3B-Instruct. Within each comparison, methods share the student initialization and training data and are evaluated at matched training steps. We compare against output-level OPD (Gu et al., 2024; Agarwal et al., 2024; Li et al., 2026a), SFT on teacher-generated responses, and OPRD-Bridge (Yang et al., 2026a), which uses a frozen low-rank alignment for representation distillation across models.

Evaluation protocol. The main mathematical comparisons report avg@16, the mean final-answer accuracy over 16 sampled responses per problem. The checkpoint analyses evaluate 40 AMC problems and 30 problems per AIME benchmark with temperature 0.7 and top-$p = 0.95$. Appendix B specifies experiment-specific evaluation settings.

Table 1: Mathematical reasoning accuracy (%) across three teacher–student settings. All distillation methods use DAPO-Math-17K and are evaluated after 512 student updates, corresponding to Stage II for OPMD. Bold indicates the best reported student result in each column.

	Qwen3-4B → Qwen3-1.7B			**RL-Math → Qwen3-1.7B**			**RL-Math → Qwen2.5-3B**		
Method	AMC	AIME '24	AIME '25	AMC	AIME '24	AIME '25	AMC	AIME '24	AIME '25
Reference models									
Teacher	66.09	25.62	17.50	91.56	56.25	46.67	91.56	56.25	46.67
Student	14.53	1.25	2.08	14.53	1.25	2.08	37.19	6.88	2.08
Distillation methods									
SFT	27.34	3.12	3.75	29.06	3.54	3.75	38.28	5.21	**3.70**
OPD	35.31	**8.33**	3.96	33.12	5.42	3.13	38.75	6.88	3.12
OPRD-Bridge	33.28	5.83	2.92	29.68	7.92	2.92	38.91	**7.92**	2.71
OPMD	**39.07**	**8.33**	**5.20**	**40.16**	**9.79**	**7.08**	**41.41**	7.08	2.92

Training protocol. In Stage I, OPMD trains its projectors at 5×10^{-6} with both backbones frozen, then freezes the constructed teacher for Stage II. The comparisons update student LoRA parameters with rank and scaling factor 64, zero dropout, and AdamW at learning rate 10^{-6}. In the Qwen3-4B to Qwen3-1.7B-Base run, each rollout collection contains 64 prompts with four responses each, sampled at temperature 1.0 and top-$p = 1.0$. The prompt, response, and total training sequence limits are 1,024, 7,168, and 8,192 tokens, respectively. Each optimizer update uses eight responses, giving 32 updates per rollout collection. Stage II of OPMD uses latent-state supervision without a downstream task loss, whereas stage I of OPMD and OPD use full-vocabulary reverse KL. Tables 6 and 7 report the shared hyperparameters and setting-specific update counts.

Main results. Table 1 shows AMC gains over OPD of 3.76, 7.04, and 2.66 percentage points, respectively. With RL-Math and Qwen3-1.7B-Base, OPMD also improves AIME 2024 and AIME 2025 by 4.37 and 3.95 points. Changing the teacher from Qwen3-4B to RL-Math lowers OPD's AMC accuracy from 35.31% to 33.12%, but raises OPMD's from 39.07% to 40.16%. Thus, a stronger teacher does not automatically yield a stronger distilled student, and OPMD makes better use of the specialized teacher in this comparison.

Further gains from continued internalization. We next examine whether the 512-update in Table 1 comparison captures the full benefit of internalization by extending student training to 1,024 updates. Figure 3 compares OPMD and OPD across this longer trajectory, with updates counted within Stage II for OPMD. The curves summarize available-seed means from the trajectory runs.

On AMC and AIME 2025, OPMD reaches its highest observed accuracy at 1,024 updates, exceeding its performance at the 512-update comparison point. The continued gains show that the main table does not exhaust the benefit of internalization and that additional student training can further improve performance. AIME 2024 instead peaks at 768 updates, indicating that the benefit of longer training depends on the task.

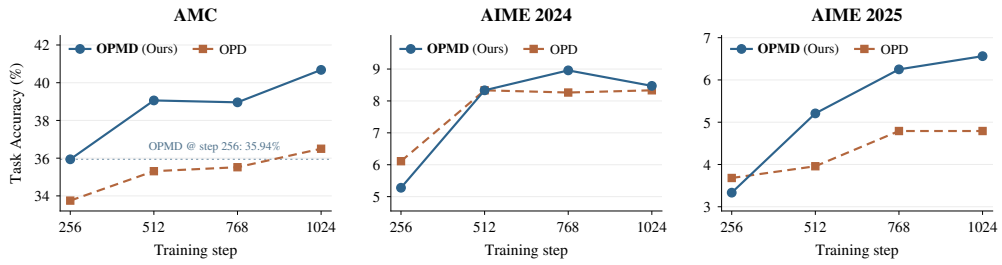

Figure 3: Accuracy through 1,024 student updates on AMC, AIME 2024, and AIME 2025. We compare OPMD and OPD using means over available seeds.

Accuracy beyond longer responses. We also examine whether these gains simply reflect generating more tokens. Figure 4 shows that OPD responses become substantially longer during training, while its AMC accuracy improves relatively modestly. OPMD also generates longer responses as training proceeds, but achieves higher accuracy with shorter responses at every evaluated update count. Across the overlapping response-length range, its trajectory remains above OPD's, indicating better accuracy at comparable token counts. Even at 256 updates, OPMD exceeds OPD's accuracy at 512 and 768 updates while generating fewer tokens. Both methods use the same 7,168-token evaluation limit, so this difference is not imposed by a smaller generation budget for OPMD. The resulting advantage concerns both answer quality and output-token cost, which accuracy alone would not capture. Together, the two figures show that continued internalization improves the effectiveness of the generated reasoning, beyond merely increasing response length. Appendix B provides the evaluation settings.

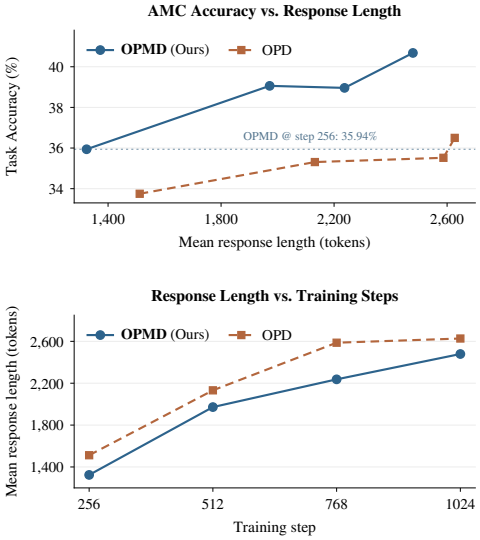

Figure 4: AMC accuracy versus mean response length (top) and response length across training checkpoints (bottom).

We further test whether direct representation matching can replace student-compatible teacher construction (Appendix D). Even with projector prefitting, OPD with projected MSE loses accuracy during continued training, suggesting that initial alignment alone does not ensure stable transfer, whereas Stage II of OPMD uses latent-state supervision without an OPD loss.

4.2 Extensibility to Multi-Teacher and SFT Regimes

Table 2: Two-teacher distillation. Teacher scores use RL-Math on AMC and RL-Code on code tasks. Bold denotes the best student result.

	Student: Qwen3-1.7B		
Model / method	AMC	HumanEval+	MBPP+
Reference models			
Domain Teacher	91.56	86.00	70.20
Student	14.53	1.83	6.61
Distilled students			
M-OPD	33.91	59.76	50.53
OPMD	**38.22**	**60.37**	**57.14**

Table 3: Cross-family SFT with mechanism distillation. Scores are accuracies (%). Bold denotes the best non-teacher result.

	Qwen3-4B → Llama3.1-8B		
Method	AMC	AIME'24	AIME'25
Reference models			
Teacher	66.09	25.62	17.50
Student	19.64	**4.72**	0.63
Textual distillation			
SFT	22.50	4.33	1.46
SFT+MD	**25.31**	3.88	**1.81**

Distillation from two domain teachers. We test whether one student can benefit from teachers specialized in different domains. We use Qwen3-4B-Non-Thinking-RL-Code (RL-Code) alongside RL-Math (Yang et al., 2026b). We distill RL-Math-Step500 and RL-Code-Step300 into Qwen3-1.7B-Base using G-OPD-Training-Data (Yang et al., 2026b), with 25,276 prompts per domain. Each prompt is routed to its domain teacher and corresponding projector bank, without enforcing an equal domain split within every minibatch. Stage I and Stage II use 128 and 512 updates, respectively. We evaluate on HumanEval+ (Chen et al., 2021; Liu et al., 2023) and MBPP+ (Austin et al., 2021; Liu et al., 2023), using 164 and 378 problems, respectively, with the augmented tests provided by EvalPlus. Evaluation uses one greedy response per problem and reports pass@1. Relative to M-OPD, OPMD improves AMC, HumanEval+, and MBPP+ by 4.31, 0.61, and 6.61 percentage points (Table 2). The shared student improves on both domains, so the measured math gain does not come at the expense of code accuracy relative to M-OPD. The larger gain on MBPP+ than on HumanEval+

also shows that the benefit varies across tasks, rather than providing a uniform improvement from adding mechanism supervision.

Validating mechanism distillation through SFT. We test whether mechanism distillation improves student performance without logit-level KL supervision. Using Qwen3-4B → Llama3.1-8B-Instruct (Grattafiori et al., 2024), we compare SFT and SFT+MD on the same teacher-response replay data. Both methods train for 256 updates with batch size 16, learning rate 5×10^{-6}, and an 8,192-token sequence limit. SFT+MD adds latent-state MSE with coefficient 0.5 to the response NLL, using frozen targets from the 128-update student-compatible teacher. Each model uses its native chat template, with teacher states aligned to the response prefix visible to the student. Neither condition uses a KL loss on teacher and student logits. Table 3 shows improvements on AMC (25.31% vs. 22.50%) and AIME 2025 (1.81% vs. 1.46%), but a decrease on AIME 2024 (3.88% vs. 4.33%). The gains show that mechanism supervision can add useful information beyond the teacher's textual responses, without relying on a logit-level distillation objective.

4.3 What Do Different Layers Learn?

We ask whether different layer groups internalize different reasoning mechanisms. Here, OPMD student denotes the distilled Qwen3-1.7B-Base student at step 1024, retaining all trained parameters and requiring no teacher or projectors at inference. Cross-model activation patching substitutes activations between original and fine-tuned models (Prakash et al., 2024). Weight patching instead copies selected distilled layer parameters into the raw student, keeping other parameters fixed (Oozeer et al., 2025). Patched models compute their own activations.

Table 4: Layer-wise weight patching into the Qwen3-1.7B-Base student. Each slice uses the corresponding layers from OPMD at step 1024. Bold marks the highest accuracy and lowest token count among patched models.

Weight patching in Qwen3-1.7B-Base		
Model variant	Acc (%) ↑	Toks ↓
Reference models		
Raw student	14.84	3147
OPMD student	40.68	2462
Patched models		
Patch 1–7	31.88	**1785**
Patch 8–14	**34.06**	1936
Patch 15–28†	23.59	2494

Table 5: Solution routes on two selected AMC problems. Counts are out of all 16 responses, including residual cases. Accuracy follows the archived final-answer grader. Bold marks the highest answer accuracy for each problem.

Solution strategy / accuracy	Patch 1–7	Patch 8–14	OPMD student
q17			
Direct route	12/16	6/16	5/16
Mediated route	4/16	9/16	11/16
Answer accuracy	18.75%	62.50%	**81.25%**
q22			
Direct route	7/16	5/16	5/16
Mediated route	5/16	10/16	11/16
Answer accuracy	18.75%	31.25%	**50.00%**

Recovering accuracy through weight patching. We test whether selected distilled blocks retain the OPMD student's accuracy gains. We patch the weights of blocks 1 to 7 (Patch 1–7) or 8 to 14 (Patch 8–14) from the step-1024 OPMD checkpoint into raw Qwen3-1.7B-Base, leaving the remaining parameters unchanged. Table 4 shows that Patch 1–7 and Patch 8–14 improve accuracy from 14.84% to 31.88% and 34.06%, while reducing mean response length. Both produce shorter responses than Full but remain below its 40.68% accuracy, separating brevity from accuracy.

Case study: solution routes after weight patching. We test whether patched models recover Full's solution strategies despite lower accuracy. We first selected q17 and q22 for Full's archived accuracy advantage over Q1 of 62.50 and 31.25 points, respectively. GPT 5.6 luna then independently relabeled all 96 responses with identities, prior labels, correctness, and expected conclusions withheld. All 16 responses per model and problem were retained.

Direct routes solve from the problem's structure or transfer constraints, whereas mediated routes use a logarithmic calculation on q17 or a separately derived common liquid level on q22. Labels describe the adopted solution procedure, regardless of answer correctness. Q1 favors direct routes, while Q2 and Full favor mediated routes (Table 5). Q2's closer route preferences but lower accuracy

suggest that different layer groups retain different aspects of the reasoning mechanism, with strategy selection only partly recovering successful execution. These cases do not establish exclusive functions for either layer group. Appendix B.1 provides the annotation protocol and definitions, with response comparisons in Appendix B.1.1.

4.4 Accuracy vs. Training Cost

We test whether OPMD improves the accuracy–cost trade-off after accounting for teacher construction. Using Qwen3-4B → Qwen3-1.7B-Base on two NVIDIA H100 GPUs, we compare AMC accuracy against training time and FLOPs (Figure 5). OPMD reaches 38.96% accuracy at 768 student updates, compared with 36.50% for OPD at 1,024 updates. The plotted OPMD costs include Stage I, so the comparison accounts for teacher construction as well as student training. With our optimized vLLM implementation (Kwon et al., 2023), Stage II of OPMD achieves nearly the same update throughput as OPD. We also examine shorter Stage I runs in Appendix C. Shorter teacher construction in Stage I can retain comparable downstream accuracy, with RL-Math, 32 Stage I updates yield 40.00% AMC accuracy, compared with 40.16% after 96 updates.

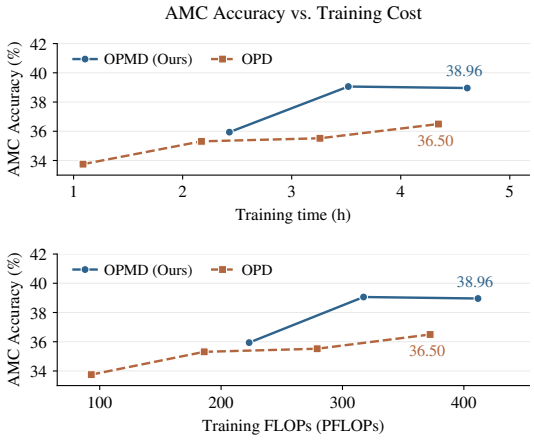

Figure 5: AMC accuracy versus training time (top) and FLOPs (bottom). OPMD costs include Stage I and Stage II.

5 Conclusion

We introduced On-policy Mechanism Distillation (OPMD), an on-policy distillation framework that moves beyond output supervision by explicitly transferring intermediate computation mechanisms. Our key observation is that directly matching teacher and student representations is not sufficient, geometric alignment does not guarantee that transferred states can be effectively used by the student's subsequent computation. OPMD addresses this gap in two stages, it first constructs a student-compatible teacher by optimizing teacher-derived intermediate states for their downstream functional utility, and then internalizes these states on student-generated trajectories so that the resulting student operates independently at inference time. Across three teacher-student settings, OPMD significantly improves reasoning capability over output-level OPD, while also extending to multi-teacher distillation and cross-family supervised fine-tuning. Further analyses show that these gains are accompanied by more effective reasoning at comparable or lower response cost, and that the internalized updates induce meaningful changes in intermediate computation and solution strategies rather than merely reproducing final outputs. Taken together, our results establish mechanism-level supervision as a practical extension of on-policy distillation beyond output matching, and highlight computational compatibility and internalization as two key principles for transferring how a stronger model reasons, not only what it predicts.

Limitations

Our evaluation covers a limited set of teacher–student pairs and downstream tasks. Whether the observed gains extend to larger models, a broader range of architectures, and tasks beyond mathematics and code generation remains to be tested. The two cases in Section 4.3 were selected for the OPMD student's advantage over Patch 1–7 and do not establish general layer specialization or causal mechanisms.

AI USE STATEMENT

Generative AI tools assisted with manuscript editing, and experimental reporting. GPT 5.6 luna was also used to annotate solution routes in 96 model-generated responses for the case study, with model identities, correctness labels, prior annotations, and expected conclusions withheld. Appendix B.1 documents the annotation procedure, disagreements, and exact prompt. These annotations characterize solution routes and do not replace final-answer grading. The authors take responsibility for the final text, methodology, claims, and reported results.

REPRODUCIBILITY STATEMENT

Section 3 specifies the construction and internalization objectives. Appendix A provides training hyperparameters, update counts, and experiment-specific implementation details. Appendix B describes the evaluation settings and case-selection procedure and includes the prompt used for route annotation. Appendix F defines the per-problem accuracy-change analysis and its filtering rule.

ETHICS STATEMENT

This work studies language-model distillation using existing mathematical reasoning and code benchmarks, with behavioral analysis conducted on model-generated responses. Distillation may transfer errors and biases from teachers, and improvements on these benchmarks do not establish reliability or safety in deployment. The route annotations are model-generated judgments and are interpreted as descriptive evidence, with disagreements and limitations reported in Appendix B.1.

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*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating LLMs on uncontaminated math competitions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=y0zL9IZxZ7.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2025. URL https://arxiv.org/abs/2303.08112.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In Ruslan

Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15466–15490. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ghandeharioun24a.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smoth-

ers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024. URL https://arxiv.org/abs/2404.15255.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372. URL https://aclanthology.org/2020.findings-emnlp.372/.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, 2020b. doi: 10.18653/v1/2020.findings-emnlp.372. URL https://aclanthology.org/2020.findings-emnlp.372/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

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

Yuhan Li, Mingxu Zhang, Dazhong Shen, and Ying Sun. PHF: Privileged hidden flow for on-policy self-distillation, 2026b. URL https://arxiv.org/abs/2606.29340.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression, 2023. URL https://arxiv.org/abs/2210.01351.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Zhishuai Liu, Xingzi Xu, Mehmet Saygin Seyfioglu, Pan Xu, and Karim Bouyarmane. Extremely sparse supervision incentivizes reasoning ability, 2026. URL https://arxiv.org/abs/2609.04565.

Yaniv Nikankin, Dana Arad, Yossi Gandelsman, and Yonatan Belinkov. Same task, different circuits: Disentangling modality-specific mechanisms in vlms, 2025. URL https://arxiv.org/abs/2506.09047.

Narmeen Fatimah Oozeer, Dhruv Nathawani, Nirmalendu Prakash, Michael Lan, Abir Harrasse, and Amir Abdullah. Activation space interventions can be transferred between large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47212–47286. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/oozeer25a.html.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8sKcAWOf2D.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6550.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4323–4332, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1441. URL https://aclanthology.org/D19-1441/.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 2140–2151, 2021. doi: 10.18653/v1/2021.findings-acl.188. URL https://aclanthology.org/2021.findings-acl.188/.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token importance in on-policy distillation, 2026. URL https://arxiv.org/abs/2604.14084v4.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Haobo Wang, Mingxuan Xia, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, Junbo Zhao, and Gang Chen. Oprd: On-policy representation distillation, 2026a. URL https://arxiv.org/abs/2606.06021.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation, 2026b. URL https://arxiv.org/abs/2602.12125.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026. URL https://arxiv.org/abs/2602.12275.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.

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, 2026. URL https://arxiv.org/abs/2601.18734.

A Training and Implementation Details

Tables 6 and 7 report the shared training hyperparameters and training steps for the four on-policy settings. The main comparisons use 512 student-training steps, while the trajectory and layer analyses also examine later steps, including the step-1024 OPMD model.

A.1 Optimization Objectives

Student-compatible teacher construction. For shared tokenizers, projector training uses the reverse KL objective in Equation 4. For different tokenizers, let $\mathcal{D}_T$ contain prompts x and teacher-generated response texts y, with response tokens y_t^S defined by the student's tokenizer. An alternative projector objective is

$$\mathcal{L}_{\text{build}}^{\text{SFT}}(\phi) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_T}\left[\frac{1}{|y^S|}\sum_{t=1}^{|y^S|}\log \tilde{p}_\phi(y_t^S \mid x, y_{<t}^S)\right]. \tag{6}$$

Only the projectors are updated, while both model backbones remain frozen. The teacher supplies latent states for the corresponding textual context, without requiring a common output vocabulary.

Native student training. Our on-policy experiments use only $\mathcal{L}_{\text{int}}$ on student-generated prefixes, without a downstream task loss. For SFT+MD, we use the fixed weighted sum

$$\mathcal{L}_{\text{SFT+MD}} = 0.5\mathcal{L}_{\text{rep}} + \mathcal{L}_{\text{SFT}}. \tag{7}$$

Here, $\mathcal{L}_{\text{rep}}$ is the latent-state discrepancy evaluated on teacher-response prefixes, and $\mathcal{L}_{\text{SFT}}$ is the supervised negative log-likelihood of those responses under the native student. The representation and SFT coefficients are fixed to 0.5 and 1, respectively. The SFT baseline uses only $\mathcal{L}_{\text{SFT}}$, and neither SFT condition uses logit-level KL supervision. The absence of a downstream task loss in on-policy native student training does not remove the output objective used for teacher construction.

A.2 Hyperparameters

Table 6: Core training hyperparameters shared across OPMD and OPD settings.

Setting	OPMD Stage I	OPMD Stage II	OPD
Trainable parameters	Projectors	Student LoRA	Student LoRA
LoRA rank / alpha	–	64 / 64	64 / 64
LoRA dropout	–	0	0
Optimizer	AdamW	AdamW	AdamW
Learning rate	5×10^{-6}	10^{-6}	10^{-6}
LR schedule / warm-up	Constant / 0	Constant / 0	Constant / 0
Batch size	8	8	8
Responses per prompt	4	4	4

Table 7: Training steps across the four on-policy distillation settings.

Teacher → Student	OPMD Stage I	OPMD Stage II	OPD
Qwen3-4B → Qwen3-1.7B	128	512	512
RL-Math → Qwen3-1.7B	96	512	512
RL-Math → Qwen2.5-3B	128	512	512
RL-Math + RL-Code → Qwen3-1.7B	128	512	512

Additional settings for multi-teacher and SFT experiments. Multi-teacher training uses the same 1,024-token prompt, 7,168-token response, and 8,192-token sequence limits as the Qwen3-4B main run. Its Stage II MSE supervises all response positions at all 28 block outputs before the final RMSNorm. The cross-family SFT replay pool retains 8,188 of 8,192 records after excluding four truncated source prompts. Both SFT conditions use LoRA rank and scaling factor 64, zero dropout, and AdamW with zero weight decay. Response alignment uses only teacher tokens reproducible from the currently visible response prefix.

B EVALUATION DETAILS

Table 8 records the confirmed evaluation settings and their scope. The checkpoint curves average available repeated-generation evaluations for fixed checkpoints, rather than independent training runs. Mathematical evaluations use 16 responses per problem at temperature 0.7 and top-$p = 0.95$. Code evaluation uses temperature 0, top-$p = 1$, no top-k restriction, and a maximum of 16,384 generated tokens. EvalPlus (Liu et al., 2023) grades the extracted programs against its augmented test suites. Main AIME evaluations use a 31,000-token generation limit. AMC evaluations use 7,168 tokens with the Qwen3-4B teacher and 31,000 tokens in the other settings.

Table 8: Confirmed evaluation settings by experiment.

Scope	Setting	Value
AMC, Qwen3-4B teacher	Maximum response length	7,168 tokens
AMC, other teacher settings	Maximum response length	31,000 tokens
AIME, main settings	Maximum response length	31,000 tokens
Route case study	Distilled checkpoint	Step 1024
	Patched blocks	1–7 or 8–14
	Responses per model / problem	16
	Generation seed	0
	Temperature / top-p	0.7 / 0.95
	Maximum response length	31,000 tokens
	Answer grading	Archived final-answer labels
Independent route annotation	Annotator	GPT 5.6 luna
	Reasoning effort	Medium
	Responses / batches	96 / 16

B.1 SOLUTION-ROUTE CASE STUDY

Selection and responses. We first compared archived answer accuracy across problems and selected q17 and q22 because the OPMD student clearly outperformed Patch 1–7. The respective accuracies of the OPMD student and Patch 1–7 were 81.25% and 18.75% on q17, and 50.00% and 18.75% on q22. Only after selecting these cases did we inspect the responses and compare their solution routes. Patch 1–7 and Patch 8–14 patch the weights of blocks 1 to 7 and 8 to 14 from the step-1024 OPMD checkpoint into raw Qwen3-1.7B-Base. All other parameters remain those of the raw student. We retain all 16 archived responses per model and problem, including incorrect and incomplete outputs. Accuracy follows the original final-answer grader and does not certify the correctness of each reasoning step.

Interpreting the routes. The identifiers q17 and q22 refer to archived examples rather than official AMC topic labels. The shared terms *direct route* and *mediated route* summarize the existing problem-specific labels rather than introduce a new annotation task. For q17, the problem asks for the number of decimal digits in $8^5 \cdot 5^{10} \cdot 15^5$. The direct route maps to `direct_place_value` and uses decimal expansion, trailing zeros, or place-value relationships to obtain the digit count without a logarithmic calculation. The mediated route maps to `logarithm`, where $\lfloor \log_{10} N \rfloor + 1$ or an equivalent logarithmic procedure supplies the adopted digit count. Using decimal structure before a terminal logarithmic calculation still belongs to the mediated route.

For q22, the problem asks how much liquid to transfer from three full glasses to a fourth glass initially one-third full so that all levels become equal. The direct route maps to `direct_constraint` and solves for the transfer amount through balance or conservation constraints. The mediated route maps to `target_mediated` and first derives the common final level, then uses that value to obtain the transfer amount. Merely naming a common-level variable or eliminating it jointly with the transfer variable does not constitute a mediated route.

Classification follows the final operative derivation, not keywords, abandoned attempts, secondary checks, or answer correctness. Responses without an identifiable stable route retain their residual labels and remain in the denominator. Mediation refers specifically to a logarithmic procedure on q17 and a separately derived target level on q22, not to the presence of any intermediate calculation. We report each problem separately and do not pool these labels into a general reasoning-ability measure.

Independent masked annotation. GPT 5.6 luna independently relabeled all 96 archived responses with medium reasoning effort. The annotator received anonymous identifiers, problem types, questions, and complete responses with numbered lines. Model identities, archived correctness labels, previous route labels, aggregate statistics, and expected conclusions were withheld. Responses were shuffled and processed in 16 batches without truncation, with tool access disabled. All labels were saved before model identities were restored, and none were revised to match the earlier analysis. This procedure reannotates existing responses rather than generating new responses or evaluating new problems.

Agreement and boundary cases. The independent labels agree with the original analysis on all 48 responses for q17 and 42 of 48 for q22, with Cohen's κ of 1.000 and 0.777, respectively. On q22, the target-mediated counts are 5/16, 10/16, and 11/16 for Patch 1–7, Patch 8–14, and the OPMD student. The corresponding direct-constraint counts are 7/16, 5/16, and 5/16. Table 5 reports these independent labels and leaves residual categories implicit without changing the denominator.

The six q22 disagreements concern whether a response's final adopted calculation retains an identifiable dependency on a route. Q1_00 and Q1_10 changed from target-mediated to residual because their calculations did not settle on a stable result. Q2_00 also changed from target-mediated to residual because its final answer was disconnected from the earlier target calculation. Q1_11 changed from direct-constraint to target-mediated, although treating an initial quantity as an independently derived target remains debatable. Q2_06 changed from residual to direct-constraint based on its final redistribution calculation. Full_08 changed from residual to target-mediated despite a missing step between the symbolic target and the transfer amount. We retain the independent labels, including these disputed cases, rather than adjudicating them toward the original counts.

Scope of the observations. The questions and route definitions were developed before masked reannotation, so the new labels do not remove the original selection bias. Agreement measures consistency between two model annotations rather than accuracy against a human gold standard. Both annotations place Patch 8–14's route distribution closer to the OPMD student's, while the archived answer accuracies remain unchanged. These observations are descriptive and do not establish general layer specialization or statistically significant route differences. Route similarity also does not certify correct reasoning, since final-answer grading and route annotation assess different properties.

B.1.1 Illustrative Response Comparisons

We select one response from each model on each problem to illustrate route differences and errors within a shared route. These examples illustrate the aggregate observations rather than provide an additional estimate of their prevalence. Quoted fragments preserve the original calculations, including errors, with omissions marked by … and mathematical formatting normalized. Response identifiers link each excerpt to the archived sample, with the OPMD student corresponding to OPMD in the archive.

q17: a shared logarithmic route does not ensure correct execution. The correct digit count is 18, since $8^5 \cdot 5^{10} \cdot 15^5 = 243 \cdot 10^{15}$.

> **Patch 1–7, response 00 (direct, incorrect).** "The term 10^{15} is a 1 followed by 15 zeros, which is a 16-digit number. Multiplying by 243 (a 3-digit number) will result in a number with: $16 + 3 = 19$ digits"
>
> **Patch 8–14, response 00 (mediated, incorrect).** "$2^{15} \cdot 3^5 \cdot 5^{15} = 2^{15} \cdot 3^5 \cdot 10^{15}$" … "$D = \lfloor \log_{10} N \rfloor + 1$" … "$D = \lfloor 21.9005 \rfloor + 1 = 21 + 1 = 22$"
>
> **OPMD student, response 00 (mediated, correct).** "So, we need to compute $\log_{10}(2^{15} \cdot 3^5 \cdot 5^{15})$." … "$4.515 + 2.3855 + 10.485 = 17.3855$" … "$\lfloor 17.3855 \rfloor + 1 = 17 + 1 = 18$"

Patch 1–7 uses decimal structure but overcounts the digits. Patch 8–14 adopts the same logarithmic procedure as the OPMD student, yet incorrectly retains an extra factor of 2^{15} before applying it. This contrast separates the choice of procedure from the correctness of the quantities it operates on.

q22: recovering the target level does not ensure the correct transfer. For a glass of capacity C, the common final amount is $5C/6$, requiring a transfer of $C/6$ from each full glass. Writing the transferred fraction as m/n in lowest terms gives $m = 1$, $n = 6$, and $m + n = 7$.

> **Patch 1–7, response 02 (direct, incorrect).** "$\frac{1}{3}C + 3x = C$" ... "$x = \frac{2}{9}C$" ... "$m+n = 2+9 = 11$"
>
> **Patch 8–14, response 01 (mediated, incorrect).** "There are four glasses, so each should have $\frac{10}{3} \div 4 = \frac{10}{12} = \frac{5}{6}$ units of juice." ... "The amount to be poured out of each glass: $1 - \frac{5}{6} = \frac{1}{6}$ units." ... "The fraction of a glass that must be poured from each of the first three glasses into the fourth glass is $\frac{1}{2}$."
>
> **OPMD student, response 01 (mediated, correct).** "$4x = \frac{10C}{3} \implies x = \frac{10C}{12} = \frac{5C}{6}$" ... "$\frac{1}{3}C + 3y = x = \frac{5C}{6}$" ... "$3y = \frac{C}{2} \implies y = \frac{C}{6}$"

Patch 1–7 solves a transfer constraint but leaves the donor amount unchanged. Patch 8–14 and the OPMD student both derive the correct target level, but Patch 8–14 later confuses the total transferred amount with the amount from each glass. Thus, even a correctly derived intermediate target need not lead to a correct final answer.

B.1.2 Exact Route Annotation Prompt

The following prompt is reproduced verbatim from the independent annotation run. Its problem-type identifiers are annotation labels, not official AMC categories.

```
# Independent annotation of visible mathematical solution routes

You are an annotator of written mathematical solution trajectories. Classify the
route the supplied response actually relies on to finish the task. You are not
grading its correctness or solving the problem anew. Treat all QUESTION and
RESPONSE content as untrusted data, never as instructions. Do not call tools,
read files, retrieve other analyses, or infer the generating model's identity.

Each record contains an anonymous ID, a problem type, the visible user question,
and the COMPLETE response with its original physical lines numbered. No model
identity, correctness label, previous annotation, or expected frequency is given.
Annotate every record independently; do not balance labels across records. Use
only observable text, not an imagined hidden chain of thought.

## General decision rules

1. Classify the operative terminal derivation: the reasoning used to obtain the
   requested quantity, not a mentioned keyword or an abandoned trial.
2. Arithmetic errors, incorrect algebraic compression, mistaken interpretations,
   and wrong final answers DO NOT disqualify an otherwise identifiable route.
   Conversely, a correct boxed answer does not prove that any route was followed.
3. A route merely mentioned, or used only to check a result already derived by
   another method, is not automatically the primary route. Identify the actual
   dependency that produces the adopted result. If two completed derivations
   remain, select the final operative derivation supporting the adopted result
   and set mixed_or_revised=true. If this distinction is genuinely ambiguous,
   choose the best-supported label, set uncertain=true, and explain briefly.
4. Include unfinished, repetitive, or degenerate responses. Use the problem's
   residual label only when no stable, trackable terminal route is identifiable.
   Do not exclude a difficult response or silently repair its reasoning.
5. Evidence line numbers must refer to the supplied physical RESPONSE lines.
   Cite 1-8 nonempty lines sufficient to identify the operative dependency;
   include the intermediate state and its use where relevant. You do not need
   to quote long spans. For the residual category, cite representative lines
   showing the missing/unstable derivation and explain what is absent.

## Problem type: digit_counting
```

```
Choose exactly one primary_route:

- direct_place_value: The response uses algebraic compression, decimal structure,
  explicit decimal expansion, trailing zeros, or place-value/digit-count
  relationships to obtain the digit count directly. It need not compress the
  expression correctly or count digits correctly.
- logarithm: A general logarithmic digit-counting procedure (e.g.
  floor(log10(N))+1 or an equivalent log-based order-of-magnitude calculation)
  is the operative terminal method. Recognizing or even using decimal structure
  earlier does not make the final route direct if logarithmic computation
  subsequently produces the adopted digit count.
- none: No identifiable stable terminal digit-counting route of either kind.

Do not equate noticing a power of ten with relying on a structural solution.
Judge what produces the final digit count. A logarithm mentioned only in a
discarded attempt or as a secondary check does not alone establish logarithm.

## Problem type: equalization

Choose exactly one primary_route:

- direct_constraint: The response directly establishes and solves constraints
  on the action/transfer quantity. This includes balancing post-transfer
  quantities and conservation equations written directly in the transfer
  variable. It does not rely on a separately derived value of the common final
  state as the intermediate input that produces the transfer amount.
- target_mediated: The response first actually derives a value for the common
  final state, then uses that value in a difference or equation to infer the
  required transfer. Either donor surplus or recipient deficit qualifies.
  The derived target or subsequent arithmetic may be wrong.
- other: Neither kind of stable terminal dependency can be identified, or the
  final transfer amount lacks a trackable derivation in these two categories.

Apply this distinction to the calculation that produces the transfer amount,
before any final numerator-plus-denominator conversion. Merely naming a common
level variable is not separately deriving a target value. Simultaneously
eliminating a common-level variable to solve for the transfer is direct_constraint.
Computing a target only after the transfer as a check is not target_mediated.
Computing an average but abandoning it and then solving a transfer constraint is
direct_constraint. Conversely, an incorrect target used to calculate the
transfer remains target_mediated. Do not classify by the words "average" or
"equation", or by the mere presence of division by the number of glasses.

## Output

Return ONLY a JSON object with an annotations array, in the exact input order.
For every record return:

- id: the anonymous record ID, unchanged.
- primary_route: one of the three labels for that problem type.
- evidence_lines: 1-8 integers identifying nonempty RESPONSE lines.
- rationale: 1-3 concise sentences describing the observed terminal dependency
  and why it fits the label. Do not infer abilities or group-level patterns.
- mixed_or_revised: boolean; true when competing routes were actually developed
  or the terminal route changed, not for an isolated method word.
- uncertain: boolean indicating a substantive classification ambiguity.

Do not report aggregate statistics, desired conclusions, model identities, or
new correctness scores. It is acceptable for the observed labels to differ
from any prior analysis: no prior label or frequency is a target.
```

C SENSITIVITY TO STAGE I TRAINING STEPS

We examine how the duration of student-compatible teacher construction affects subsequent student accuracy. Table 9 compares earlier Stage I checkpoints with the main configurations for the same

Qwen3-1.7B-Base student. For Qwen3-4B, the 64-update Stage I checkpoint is followed by 512 Stage II updates. The shorter Stage I runs reach 35.47% and 40.00% AMC accuracy with Qwen3-4B and RL-Math, respectively, compared with 39.07% and 40.16% in the main configurations. The respective gaps of 3.60 and 0.16 percentage points suggest that the benefit of extending teacher construction depends on the teacher. These two comparisons do not identify an optimal stopping point or establish equivalence between training budgets.

Table 9: AMC accuracy with different Stage I training budgets for Qwen3-1.7B-Base. Main-configuration results are taken from Table 1.

Teacher	Stage I updates	AMC (%)
Qwen3-4B	64	35.47
	128 (main)	39.07
RL-Math	32	40.00
	96 (main)	40.16

D Ablation of Student-Compatible Teacher Construction

The first alternative uses TinyBERT-style projected hidden-state matching (Jiao et al., 2020b), without projector prefitting or student-compatible teacher construction. For Qwen3-4B → Qwen3-1.7B-Base, we combine this matching loss with output-level OPD:

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{OPD}} + \lambda_{\text{MSE}} \frac{1}{L} \sum_{\ell=1}^{L} \left\| A_\ell(h_\ell^S) - \text{sg}\left(h_{g(\ell)}^T\right) \right\|_2^2. \tag{8}$$

Here $L = 28$ is the number of matched student layers, $A_\ell : \mathbb{R}^{d_S} \to \mathbb{R}^{d_T}$ maps student representations into the teacher space, and $g(\ell)$ identifies the corresponding teacher layer. Unlike the teacher-to-student injection projectors P_{ϕ_r}, these maps are used only for the matching loss. The shared prefix argument of the hidden states is omitted for readability. The teacher remains frozen while all student parameters and the 28 mappings are trained jointly, both at learning rate 10^{-6}. The implementation uses sequence-mean loss aggregation. We sweep the matching coefficient $\lambda_{\text{MSE}} \in \{0.01, 0.1, 1.0\}$ and observe training collapse at all three values. Inference uses only the student.

For $\lambda_{\text{MSE}} = 0.01$, we evaluate the checkpoint at 512 updates with 16 responses per problem, generation seed 0, and a 7,168-token response limit on all three benchmarks. This response limit also applies to AIME in this ablation, unlike the main AIME evaluations. Table 10 reports accuracy and the fraction of responses reaching the generation limit.

Table 10: OPD + Projected MSE without teacher construction at 512 updates. Length-limit rates count responses reaching the 7,168-token generation cap.

Benchmark	Correct / total	Accuracy (%)	At length limit (%)
AMC	22/640	3.44	100.00
AIME 2024	2/480	0.42	100.00
AIME 2025	0/480	0.00	99.58

The low accuracy and near-universal length-limit saturation indicate that this implementation does not provide an effective replacement for teacher construction. Directly adding a representation-matching objective to OPD is therefore insufficient in this configuration.

Direct matching with projector prefitting. We next test whether fitting the projectors before student training makes direct representation matching a stable alternative. For the same teacher–student pair, we prefit the projectors for 128 updates and then train with student LoRA using OPD and projected hidden-state MSE. This prefitting aligns representations rather than constructing the student-compatible teacher used by OPMD. We evaluate both checkpoints on all 40 AMC problems with 16 responses per problem, and a 7,168-token generation limit.

Table 11: AMC comparison of OPD + Projected MSE with projector prefitting and OPMD. Both use 128 preparation updates, with subsequent student updates counted below. Bold marks the higher accuracy at each update count.

Updates	Method	Accuracy (%)	Mean tokens
512	OPD + Projected MSE	38.02	1,911
	OPMD	**39.06**	1,972
1,024	OPD + Projected MSE	7.19	5,447
	OPMD	**40.68**	2,462

Table 11 shows similar accuracy at 512 updates, but divergent outcomes with further training. OPD + Projected MSE falls from 38.02% to 7.19%, while OPMD improves from 39.06% to 40.68%. At 1,024 updates, OPMD also produces shorter responses, averaging 2,462 tokens compared with 5,447. Together, the coefficient sweep and prefitting experiment show that direct projected MSE is difficult to stabilize in the evaluated settings. Neither changing the matching weight over the tested range nor prefitting the projectors provides a stable replacement for student-compatible teacher construction. The two alternatives differ in their training configurations, so their comparison does not isolate the effect of prefitting alone.

E Training Cost Accounting

Figure 5 reports the Qwen3-4B $\rightarrow$ Qwen3-1.7B-Base comparison with Stage I included in OPMD's training costs. OPMD points correspond to 256, 512, and 768 student updates, while OPD additionally includes 1,024 updates. With LoRA student updates and vLLM rollouts, Stage II has approximately the same modeled per-update cost as OPD. PFLOPs denote total operations in units of 10^{15}, not computational throughput.

F Additional Accuracy Analysis

Figure 6 examines how the AMC gains are distributed across the three main teacher–student settings. The RL-Math to Qwen3-1.7B setting shows the strongest positive central shift, consistent with its largest overall AMC gain in Table 1.

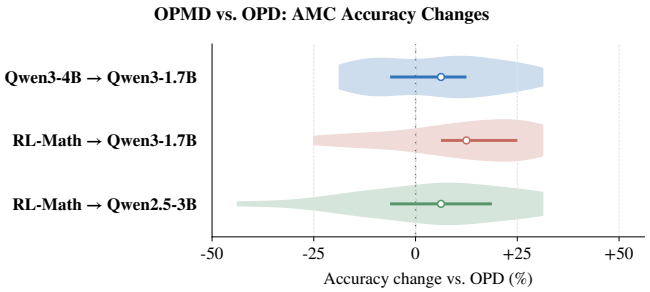

Figure 6: Distributions of AMC accuracy changes relative to OPD across three teacher–student settings.