ACTIVATION-CONDITIONED SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy self-distillation uses a model as its own teacher to provide dense supervision for reasoning, often through reference-solution conditioning. Providing privileged information does not by itself ensure effective token-level supervision throughout long responses. We introduce Activation-Conditioned Self-Distillation (ACSD), which extracts a steering vector by contrasting activations of self-generated trajectories that reach verified correct answers within a generation budget with those of all remaining trajectories. A frozen copy of the base model applies this vector at each prediction position, and the student learns from its next-token distributions on student-generated prefixes. Outcome verification is used for direction construction and calibration; distillation requires neither problem-specific reference text nor teacher parameter updates. The distilled student is used alone at inference. On each of five models, ACSD achieves the highest mean accuracy over four mathematical benchmarks among the evaluated methods. On DeepSeek-R1-0528-Qwen3-8B, mean mathematical accuracy reaches 71.9% and LiveCodeBench v6 pass@12 reaches 70.9%, compared with 69.0% and 66.3% for the reference-conditioned OPSD baseline. Contrasts among correct trajectories also support distillation, and extracted directions can be reused across mathematical training datasets. On fixed student trajectories, ACSD maintains more stable late-position logit-update magnitudes than OPSD.

1 INTRODUCTION

Post-training has substantially advanced the reasoning capabilities of language models, particularly in mathematics and coding. Supervised fine-tuning (SFT) exposes models to high-quality solution demonstrations, while reinforcement learning with verifiable rewards (RLVR) trains on the model's own responses using correctness signals. Group Relative Policy Optimization (GRPO; Shao et al., 2024), for instance, samples multiple responses to the same problem and assigns each an outcome reward based on final-answer correctness. However, this reward lacks fine-grained feedback on intermediate reasoning steps, and sampling multiple long responses significantly increases training cost. On-policy distillation (OPD) addresses this by employing a strong teacher model to provide dense, token-level supervision over the student's own responses: the student first attempts a problem, and the teacher then computes a next-token distribution at every position, conditioned on the problem and the student's preceding tokens. The student learns by matching these distributions along its own reasoning trajectory (Agarwal et al., 2024).

However, OPD commonly relies on a stronger external teacher. Recent self-distillation methods instead derive a teacher from the model itself, giving it additional context that the student does not receive. OPSD conditions a frozen copy of the initial model on a correct answer or reference solution, while SDFT conditions the teacher on expert demonstrations. Both use the model's in-context learning ability to turn this privileged context into next-token targets along the student's own responses (Zhao et al., 2026; Shenfeld et al., 2026). Other methods obtain conditioning information from the model's attempts and external feedback. SDPO conditions its teacher on feedback, such as execution errors, or successful attempts at the same problem (Hübotter et al., 2026). SD-Zero trains a reviser using binary correctness feedback and then distills its next-token predictions into a generator (He et al., 2026). RLCSD contrasts teacher predictions conditioned on correct and incorrect solutions and uses this contrast to shape reward-based GRPO updates (Pan et al., 2026).

Providing privileged information does not by itself ensure effective token-level supervision throughout a long response. In our fixed-trajectory analysis of OPSD, the reference-conditioned teacher's

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

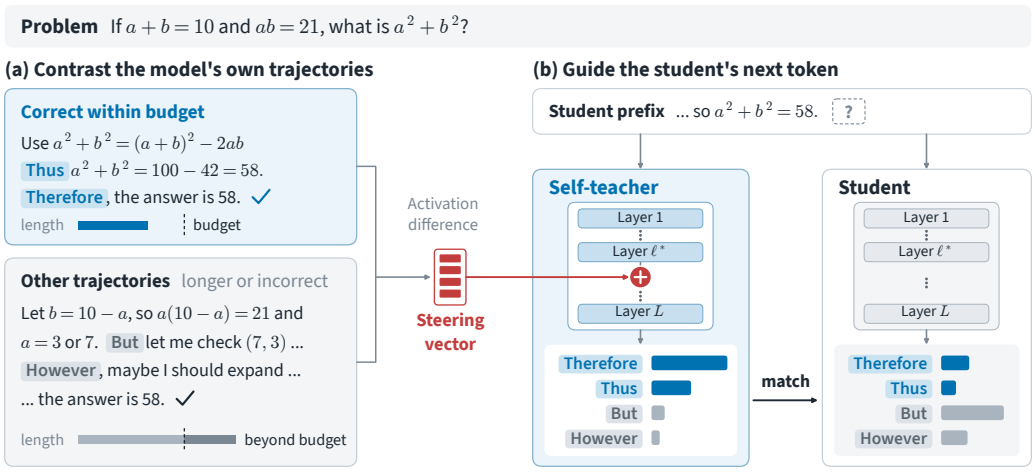

Figure 1: Overview of ACSD. **(a)** We extract a steering vector from the activation contrast between the model's own trajectories that are correct within a generation budget and all remaining trajectories. **(b)** The vector steers hidden states at layer ℓ^* to form the self-teacher. Teacher and student process the same problem and student-generated prefix, and the student learns by matching the teacher's next-token distribution.

update magnitudes decrease as the student's prefix grows (Figure 4(a)), consistent with OPSD's discussion of weaker late-token penalties (Zhao et al., 2026). This observation motivates testing whether conditioning the teacher's hidden states at every prediction position can sustain useful token-level supervision as the student's response grows longer. The model's own attempts provide a candidate source of such conditioning: for the same problem, some reach the correct answer within a generation budget, others require longer reasoning, and still others fail. These attempts provide contrasts in both verified outcome and response length, including differences among successful trajectories that correctness labels alone do not capture. Contrasting their activations offers a way to encode these differences as a reusable condition, which the teacher combines with the current student prefix when predicting each next token. We investigate whether this construction provides useful supervision throughout long responses.

We propose Activation-Conditioned Self-Distillation (ACSD), illustrated in Figure 1. ACSD applies the resulting *steering vector* to a frozen copy of the base model, which receives no reference answer or solution during distillation. At each prediction position, the teacher adds this vector to its current hidden state, combining the trajectory contrast with the problem and student-generated prefix to produce a next-token distribution. The student matches these distributions on its own responses. Like OPSD, ACSD samples one response per training problem during distillation, whereas GRPO and RLCSD use groups of responses for their reward-based updates. Its additional preparation consists of collecting and verifying trajectories, extracting the steering vector, and selecting the layer ℓ and strength α through recovery on a calibration pool. The distilled student is used alone at inference, without activation interventions. Among the evaluated methods, ACSD achieves the highest mean accuracy across four mathematical benchmarks on each of five models and the highest LiveCodeBench v6 pass@k at all evaluated k values on DeepSeek-R1-0528-Qwen3-8B (Table 1). Our main contributions are as follows:

- **We propose ACSD, an on-policy self-distillation method that constructs a teacher without reference solutions or additional teacher training.** We extract a steering vector from the model's own trajectory contrasts and apply it to a frozen teacher, which provides dense supervision on student-generated responses.
- **We analyze the role of teacher conditioning over long reasoning trajectories and examine the basis and reusability of the steering vector.** On fixed student trajectories, ACSD maintains more stable update magnitudes near the end of long responses than reference-conditioned OPSD. Contrasting shorter and longer correct trajectories also yields distillation gains, and the extracted directions transfer across math training datasets and difficulty levels.

- **We validate the method on mathematical reasoning and code generation benchmarks.** On DeepSeek-R1-0528-Qwen3-8B, ACSD achieves improvements of 4.3% in average accuracy across four mathematical reasoning benchmarks and 5.2% in LiveCodeBench v6 pass@12 over the base model.

2 PRELIMINARIES

2.1 ON-POLICY DISTILLATION

Let x denote a problem and $y = (y_1, \ldots, y_{|y|})$ a response sampled from a student model π_θ. At prediction position t, the prefix $y_{<t}$ consists of the student's preceding response tokens. In on-policy distillation, a teacher computes a next-token distribution conditioned on x and $y_{<t}$, and the student learns to match this distribution over the vocabulary (Agarwal et al., 2024). Thus, the teacher supplies a target at each position along the student's own response, including prefixes that differ from those the teacher would generate. Subsequent responses are sampled from the updated student; this use of the current student's trajectories makes the training on-policy. In self-distillation, the teacher is derived from the model being trained. Its predictions can be conditioned differently from the student's; for example, OPSD gives the teacher a reference answer or solution that the student does not receive (Zhao et al., 2026).

2.2 ACTIVATION STEERING

As a transformer processes a problem and response prefix, its layers compute hidden-state vectors, or *activations*, representing the current context. The *residual stream* is the representation passed between transformer layers. Let $\boldsymbol{h}_t^{(\ell)}$ denote the residual-stream state at layer ℓ used to predict token y_t. Activation steering modifies this state during a forward pass while keeping the model parameters fixed (Turner et al., 2023; Rimsky et al., 2024). An additive intervention shifts the state along a *steering vector* $\boldsymbol{d}^{(\ell)}$; the remaining layers process the modified state to produce the next-token distribution. The vector specifies a direction in activation space, and its scaling determines the size of the intervention.

A common way to construct a steering vector is to contrast mean activations from two groups of examples. For example, Contrastive Activation Addition (CAA) uses paired prompts with contrasting answers (Rimsky et al., 2024), while reasoning interventions can contrast activations associated with annotated reasoning behaviors (Venhoff et al., 2025). The groups, extraction positions, and normalization rule determine the resulting direction. A fixed vector can be reused across contexts, but its effect on the output distribution depends on the hidden state and the subsequent network computation. Applying steering at successive prediction positions lets the intervention influence predictions throughout a response. The trajectory groups, normalization, intervention scaling, and calibration used by ACSD are specified in Section 3.

3 ACTIVATION-CONDITIONED SELF-DISTILLATION

Let π_0 denote the base model, π_θ the student, and $\mathcal{D}$ the set of distillation problems. ACSD uses a steered, frozen copy of π_0 to supervise π_θ on its own trajectories.

3.1 MOTIVATION: FROM TRAJECTORY CONTRASTS TO A SELF-TEACHER

A reasoning model can generate different trajectories for the same problem: one attempt reaches a correct answer within a moderate number of tokens, another succeeds only after much longer reasoning, and others fail. The successful attempts demonstrate reasoning that the model can already perform and provide a source of comparisons for learning from its own behavior. Figure 2 examines four sampled responses per problem on 512 discovery problems for each model. Many problems have a correct response that finishes well before the largest generation budget, although allowing longer completions increases coverage. Even among problems with multiple correct responses, the shortest and longest successful trajectories can differ by thousands of tokens. Correctness labels alone therefore do not capture the variation among successful attempts. A generation budget pro-

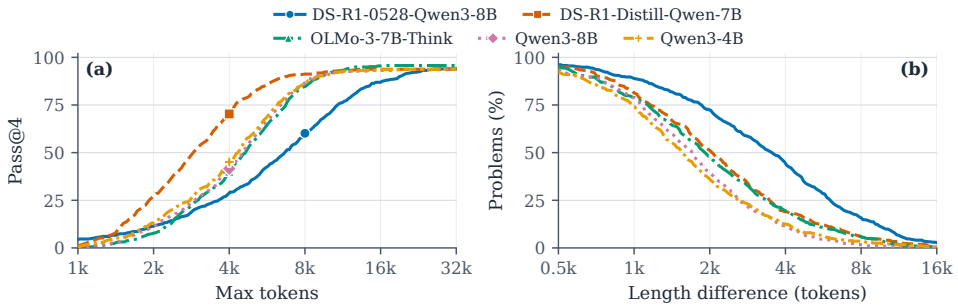

Figure 2: Variation among four discovery responses per problem, with 512 problems per model. (a) Fraction of problems with at least one correct response completed within the generation budget, computed cumulatively from the stored responses (Pass@4). Markers indicate ACSD's budget B. (b) Fraction of eligible problems whose longest and shortest correct responses differ by at least the stated number of tokens. Eligibility requires at least two correct responses; eligible counts are 466, 469, 483, 476, and 476 in legend order.

vides an additional criterion: we contrast trajectories that reach a verified correct answer within the budget with all remaining trajectories, including longer correct attempts.

To turn this trajectory-level contrast into token-level supervision, we need a way to incorporate it into the teacher's next-token predictions. Activation steering provides such a mechanism: the difference between the two groups' mean activations defines a direction that can be applied to the teacher's hidden states. The weaker late-position updates observed in our OPSD analysis (Figure 4(a)) motivate applying this condition at every prediction position, so that trajectory-derived information is introduced as the student prefix grows. At each position, the teacher processes the problem and the student's preceding tokens; adding the direction to the resulting hidden state allows the remaining network to combine the trajectory contrast with the current reasoning context. The same vector can therefore influence predictions throughout a response while producing context-dependent changes in the next-token distribution. Whether those changes provide useful supervision is an empirical question.

Before using this construction for distillation, we probe whether the intervention helps the model complete prefixes from contrast-group trajectories on problems held out from direction extraction (the calibration pool). On DeepSeek-R1-0528-Qwen3-8B, the selected configuration increases the fraction of verified correct continuations from 26.6% to 66.0% under matched sampling settings and continuation budgets (Appendix A.2). This result shows that the extracted contrast can improve continuation success in this setting, motivating its use to condition a self-teacher. We construct that teacher from a frozen copy of the base model: the student generates its own response, and the steered teacher supplies a next-token distribution at each position along that response (Figure 1b). The student learns from these distributions and is used alone at inference time. We evaluate the teacher's usefulness through the distilled student's performance, since improved recovery on the calibration pool does not by itself establish better distillation targets. The following subsections specify the construction, configuration selection, and student training.

3.2 Self-Teacher Construction

Trajectory collection and labeling. For each discovery problem x_i, we sample reasoning trajectories $r_i \sim \pi_0(\cdot \mid x_i)$. Let $V(x_i, r_i) \in \{0, 1\}$ indicate whether the final answer passes verification and $|r_i|$ the number of generated tokens. Using a generation budget B, we assign the label

$$u_i = \mathbf{1}\{V(x_i, r_i) = 1 \text{ and } |r_i| \leq B\}. \tag{1}$$

Trajectories with $u_i = 1$ form the positive group. All remaining trajectories form the contrast group, including longer correct responses, incorrect responses, and unfinished responses. The model-specific values of B are listed in Table 3.

Steering vector extraction. At each candidate layer ℓ, we collect activations from the first $\min(|r_i|, B)$ generated tokens of each trajectory. Thus, B serves both as the labeling budget and

as the activation-extraction cap. Let $\mathcal{H}_+^{(\ell)}$ and $\mathcal{H}_-^{(\ell)}$ denote the collections of activations from the two groups. We normalize each activation before computing the group means:

$$\mu_\pm^{(\ell)} = \frac{1}{|\mathcal{H}_\pm^{(\ell)}|} \sum_{\boldsymbol{h} \in \mathcal{H}_\pm^{(\ell)}} \frac{\boldsymbol{h}}{\|\boldsymbol{h}\|_2}. \tag{2}$$

The mean pools individual token activations. Each contributes a unit vector, while each trajectory contributes $\min(|r_i|, B)$ vectors; normalizing activations therefore does not equalize trajectory weights. The steering direction is the normalized difference between these means,

$$\boldsymbol{d}^{(\ell)} = \frac{\mu_+^{(\ell)} - \mu_-^{(\ell)}}{\|\mu_+^{(\ell)} - \mu_-^{(\ell)}\|_2}. \tag{3}$$

This unit normalization separates the direction from the intervention strength, which we calibrate below.

Teacher intervention. A frozen copy of π_0 processes the problem and the student's preceding tokens. At each response-prediction position t, we modify its residual state at layer ℓ as

$$\widetilde{\boldsymbol{h}}_t^{(\ell)} = \boldsymbol{h}_t^{(\ell)} + \alpha \|\boldsymbol{h}_t^{(\ell)}\|_2 \boldsymbol{d}^{(\ell)}, \tag{4}$$

where α scales the intervention relative to the current hidden-state norm. The remaining layers process the modified state to produce the next-token distribution. Steering begins at the final prompt state, which predicts the first response token, and continues at subsequent response-prediction positions. The direction is fixed across positions; its effect on predictions depends on the current problem and student prefix.

3.3 Configuration Selection

We select the intervention layer ℓ^* and strength α on the calibration pool. For fixed prefixes from contrast-group trajectories, we sample continuations with and without steering under the same sampling settings and continuation budget. *Recovery* is the fraction of continuations whose final answers pass verification; prefix construction and evaluation details are given in Appendix A.2. After selection, we set $\boldsymbol{d} = \boldsymbol{d}^{(\ell^*)}$ and denote the resulting teacher by $\widetilde{\pi}_0$. Its parameters, direction, layer, and strength remain fixed throughout distillation. Teacher construction requires verified self-generated trajectories and forward passes through the base model, with no reference solution or teacher parameter updates.

3.4 On-Policy Student Training

We initialize $\pi_\theta \leftarrow \pi_0$. For each training problem $x \sim \mathcal{D}$, the student generates a response $y = (y_1, \ldots, y_{|y|}) \sim \pi_\theta(\cdot \mid x)$. At position t, both models process the problem x and the student's preceding tokens $y_{<t}$ to compute logits $\boldsymbol{z}_t^S$ and $\boldsymbol{z}_t^T$, with the teacher applying the hidden-state intervention. At a shared distillation temperature $T_\mathrm{d} > 0$, their next-token distributions are

$$p_t^S = \mathrm{softmax}(\boldsymbol{z}_t^S / T_\mathrm{d}), \qquad p_t^T = \mathrm{softmax}(\boldsymbol{z}_t^T / T_\mathrm{d}). \tag{5}$$

At each position, the student learns the teacher's probability assignment over the full vocabulary. For each entry v in vocabulary $\mathcal{V}$, its contribution to the forward KL divergence from teacher to student is

$$\ell_{t,v} = p_t^T(v) \log \frac{p_t^T(v)}{p_t^S(v)}. \tag{6}$$

Following the per-token pointwise clipping of Zhao et al. (2026), we cap each entry's contribution at $\tau > 0$ to limit the influence of individual vocabulary items. The training objective first clips and sums over the vocabulary at each position, then averages over positions:

$$\mathcal{L}_{\mathrm{ACSD}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} \sum_{v \in \mathcal{V}} \min(\ell_{t,v}, \tau) \right]. \tag{7}$$

Table 1: Mathematical reasoning (Avg@12, %) and, bottom right, LiveCodeBench v6 code generation (pass@k, %). Bold marks the best result per column within a model; averages use unrounded scores.

Method	DS-R1-0528-Qwen3-8B					DS-R1-Distill-Qwen-7B					OLMo-3-7B-Think				
	AIME 24	AIME 25	HMMT 25	AIME 26	Avg.	AIME 24	AIME 25	HMMT 25	AIME 26	Avg.	AIME 24	AIME 25	HMMT 25	AIME 26	Avg.
Base	78.3	70.6	49.7	71.7	67.6	52.5	37.8	25.0	46.1	40.4	72.5	68.3	44.4	71.1	64.1
SFT	77.2	71.4	50.0	74.2	68.2	52.2	38.6	25.0	45.6	40.3	71.4	63.1	40.3	63.6	59.6
GRPO	79.2	73.9	**55.8**	74.2	70.8	56.1	41.4	26.4	48.3	43.1	74.7	68.6	46.9	71.4	65.4
OPSD	78.9	71.4	51.1	74.7	69.0	**56.4**	40.0	25.6	**51.4**	43.3	75.8	68.1	46.7	71.4	65.5
RLCSD	77.8	73.3	53.6	71.1	69.0	55.8	41.1	26.4	49.2	43.1	75.6	67.8	47.8	**72.5**	65.9
ACSD	**80.8**	**75.0**	55.3	**76.7**	**71.9**	**56.4**	**41.7**	**27.8**	50.3	**44.0**	**76.1**	**68.9**	**48.1**	71.9	**66.3**

Method	Qwen3-8B					Qwen3-4B					DS-R1-0528-Qwen3-8B			
	AIME 24	AIME 25	HMMT 25	AIME 26	Avg.	AIME 24	AIME 25	HMMT 25	AIME 26	Avg.	*LiveCodeBench v6 (code)*			
											pass@1	pass@5	pass@8	pass@12
Base	75.8	68.1	44.7	64.4	63.3	72.2	68.1	43.3	65.6	62.3	49.6	61.7	64.1	65.7
SFT	73.6	61.1	38.9	65.8	59.9	71.7	66.9	41.1	64.4	61.0	48.5	60.7	63.3	65.1
GRPO	77.2	70.6	47.8	70.0	66.4	**76.1**	68.9	46.4	67.2	64.7	48.7	61.0	64.4	67.4
OPSD	77.5	**71.7**	43.9	70.8	66.0	75.6	**69.7**	46.1	69.4	65.2	49.6	60.8	63.8	66.3
RLCSD	76.7	69.2	45.8	70.0	65.4	75.8	66.4	46.1	65.8	63.5	49.7	63.0	66.1	68.6
ACSD	**79.7**	**71.7**	**49.2**	**71.4**	**68.0**	75.6	68.9	**47.2**	**69.7**	**65.3**	**51.0**	**65.0**	**68.4**	**70.9**

Each update treats the sampled token sequence and the teacher's distributions as constants; gradients flow only through the student distribution p_t^S. Subsequent rollouts are generated by the updated student. At inference time, only the student model is used. Hyperparameter values (T_{d}, τ, and other training settings) are given in Appendix A. Algorithm 1 summarizes the procedure.

The distillation stage uses one student response per training problem, as in OPSD. GRPO and RLCSD instead generate groups of responses for their outcome-based updates. ACSD's additional preparation consists of rollout collection, activation extraction, and recovery on the calibration pool to select ℓ and α; these computations are performed before student training.

4 Experiments

4.1 Experimental Setup

Models and Data. Mathematical reasoning experiments cover DeepSeek-R1-0528-Qwen3-8B, DeepSeek-R1-Distill-Qwen-7B, OLMo-3-7B-Think, Qwen3-8B, and Qwen3-4B (DeepSeek-AI et al., 2025; Team Olmo et al., 2025; Yang et al., 2025); code generation, ablations, and further analyses use DeepSeek-R1-0528-Qwen3-8B. The main mathematical comparisons use the OpenThoughts-Math-30K split released with OPSD (Zhao et al., 2026). For these comparisons, ACSD extracts each direction from 2,048 trajectories on 512 problems and uses a separate calibration pool for recovery evaluation. Code experiments train on Codeforces, with directions extracted from a separate pool of verifier-labeled Codeforces trajectories. Direction-transfer experiments use DeepMath subsets of different difficulty levels (He et al., 2025).

Baselines and Training. Both tasks compare ACSD with the base model, SFT on reference trajectories, GRPO with binary outcome rewards, OPSD with a reference-conditioned self-teacher, and RLCSD combining contrastive self-distillation with reinforcement learning (Shao et al., 2024; Zhao et al., 2026; Pan et al., 2026). For mathematics, SFT, GRPO, OPSD, and ACSD use the same framework and LoRA for 200 updates; RLCSD uses its released implementation for 450 updates. ACSD and OPSD use a 4,096-token training window; GRPO and RLCSD use 16,384 tokens. ACSD and OPSD share the forward KL objective with pointwise clipping over vocabulary entries. Full settings appear in Appendix A.

Evaluation. We evaluate mathematical reasoning on AIME24, AIME25, HMMT25, and AIME26. For each problem, we sample 12 responses and report their average correctness (Avg@12), together

with the average over the four benchmarks. For code generation, we sample 12 responses per problem on the 175 problems in LiveCodeBench v6 and report pass@1, pass@5, pass@8, and pass@12 (Jain et al., 2025). Within each model, all methods share the evaluation generation budget, sampling parameters, answer parser, and verifier. ACSD evaluates the distilled student.

4.2 Main Results

Mathematical Reasoning. The strongest baseline varies across the five models in Table 1, while ACSD achieves the highest average accuracy on each. On DeepSeek-R1-0528-Qwen3-8B, average accuracy increases from 67.6 to 71.9, compared with 70.8 for GRPO and 69.0 for both OPSD and RLCSD. On Qwen3-8B, ACSD reaches 68.0, compared with 63.3 for the base model and 66.4 for GRPO. ACSD also consistently outperforms OPSD, which shares its distillation objective and training window.

Extension to Code Generation. To evaluate ACSD beyond mathematical reasoning, we construct a self-teacher from Codeforces trajectories and test code generation on LiveCodeBench v6. ACSD achieves the highest scores across all four metrics in the code block of Table 1, increasing pass@1 from 49.6 to 51.0 and pass@12 from 65.7 to 70.9. GRPO and OPSD improve pass@12 while leaving pass@1 at or below the base model, whereas ACSD improves both single-sample accuracy and the probability of solving a problem with multiple samples.

Training Cost. On Codeforces, ACSD and OPSD use 59.93 and 61.82 H100 GPU-hours for 200 updates; GRPO and RLCSD use 101.21 and 642.46 GPU-hours for 450 updates under their own protocols, so ACSD obtains its stronger code results at lower training cost. ACSD additionally spends 10.85 GPU-hours generating the discovery pool, 2.09 extracting directions at five layers, and 6.17 generating the calibration pool, 79.04 GPU-hours in total with training, before layer and strength selection; a compact search over a middle layer and three or four strengths adds to this cost.

4.3 Ablation Studies

Trajectory Contrasts. ACSD groups trajectories by correctness and generation budget. To examine these choices, we first contrast correct trajectories completed within and beyond the budget. This direction improves the student throughout training (Figure 3(a)). These gains show that differences among successful trajectories with identical correctness labels can provide useful supervision, supporting the motivation in Section 3.1. Full ACSD reaches a higher peak accuracy, suggesting that retaining incorrect and unfinished trajectories adds complementary information. We also contrast correct and incorrect trajectories matched by length. This direction improves accuracy initially, then falls below the base model. Together, these results support contrasting successful with unsuccessful trajectories, with the budget further separating trajectories that reach the same correct outcome.

Direction Perturbations. We replace the ACSD direction with a random unit vector, a coordinate-shuffled vector, or its reverse. Shuffling preserves the coordinate values and norm; reversing preserves the axis and changes its sign. All three perturbations lead to declining accuracy as training continues, while the original direction consistently improves the student (Figure 3(b)). The gains thus depend on the extracted direction and its orientation (construction details in Appendix A.3).

4.4 Token-Level Supervision from the Teacher

The direction ablations establish that activation differences between trajectories can support self-distillation. To examine how these differences affect teacher supervision, we fix the student and 512 of its reasoning trajectories, split evenly into discovery and held-out sets. Both teachers compute next-token distributions on the same problems and student prefixes: ACSD applies the direction to hidden states, and OPSD adds a reference solution to the teacher context. We call the negative gradient of the clipped distillation loss with respect to student logits the *logit update*. Positive and negative coordinates indicate upward and downward updates in logit space. All results below use the 256 held-out trajectories.

Update Magnitude over Long Trajectories. We group prefixes by absolute token position and compute the root-mean-square norm of the logit-update vectors in each group. In Figure 4(a), ACSD's update magnitude increases early and then remains relatively stable, while OPSD's de-

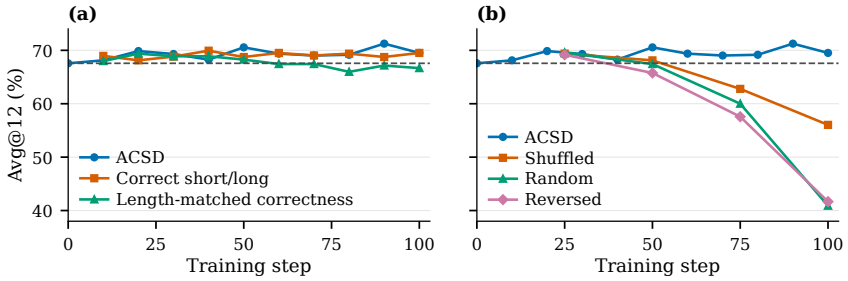

Figure 3: Direction ablations on DeepSeek-R1-0528-Qwen3-8B: mean Avg@12 over four mathematical benchmarks during the first 100 updates. (a) Trajectory contrasts. (b) Random, shuffled, and reversed directions. Controls share the layer and strength of the ACSD run in Table 1; dashed lines mark the base model.

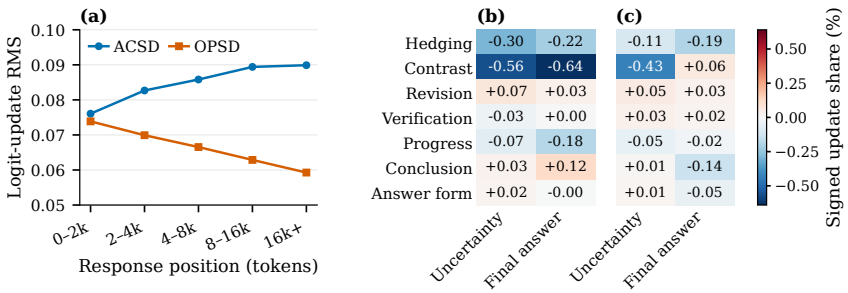

Figure 4: Token-level updates on 256 held-out DeepSeek-R1-0528-Qwen3-8B trajectories under the training clipping rule. (a) Root-mean-square logit-update magnitude by response position. (b) ACSD and (c) OPSD signed updates for uncertainty estimation and final answers, as a percentage of the role's total absolute update; both teachers share prefixes, marker groups, and color scale.

creases as the prefix grows. OPSD has approximately 97% of ACSD's magnitude in the first 2k tokens and 66% beyond 16k tokens, so ACSD maintains more stable update magnitudes toward the end of long trajectories. A training-window experiment also finds that ACSD improves over the base model at every tested window from 1k to 16k tokens; full curves appear in Appendix A.5.

Updates across Reasoning Roles. We next examine updates in different reasoning contexts. A language-model annotator assigns spans to seven roles, including deduction, uncertainty estimation, backtracking, and final answers, using the response text and role definitions. We define seven marker groups on the discovery set. Within each role, we sum the signed updates for each marker group and normalize by the total absolute update over the vocabulary. Figure 4(b–c) shows uncertainty estimation and final answers, using the same marker groups for both teachers.

During uncertainty estimation, ACSD assigns negative updates to hedging markers such as *probably* and *perhaps* and contrast markers such as *But* and *However*, while assigning positive updates to revision markers such as *wait* and *Actually*. In final-answer spans, ACSD assigns negative updates to contrast markers and positive updates to conclusion markers such as *Therefore* and *Thus*; OPSD shows the opposite signs for these two groups. ACSD's update size also depends on the role: the conclusion-marker update is larger in final-answer spans than during uncertainty estimation, so a fixed vector yields role-dependent supervision. Full role heatmaps, annotation details, and robustness checks using an expanded marker lexicon, leave-one-marker-out variants, and matched random tokens appear in Appendix A.4.

4.5 The Effect of Direction Extraction Data

Direction Transfer. We vary the extraction problems to examine the effect of the discovery distribution (Table 2(a)). We first retain the OpenThoughts-Math-30K direction and distill on hard DeepMath problems (Level 7–10), reaching 70.1 Avg@12 compared with the base model's 67.6. We then extract directions from easy DeepMath problems (difficulty at most 3) and hard problems, and distill on the same hard set with the model, layer, strength, and optimizer fixed. These directions

reach 70.6 and 71.0, respectively, showing that trajectory contrasts from easy problems can support self-distillation on harder problems.

Discovery-Set Size. We further reduce the discovery set from 512 to 16 problems using nested subsets, with four trajectories per problem. Table 2(b) reports the extraction budget, direction similarity, and distillation result for each subset. Every setting improves over the base model. A direction extracted from just 64 trajectories on 16 problems reaches 70.8 after distillation, compared with 67.6 for the base model, so a small set of the model's own trajectories suffices to construct an effective self-teacher (details in Appendix A.6).

Table 2: Direction extraction data on DeepSeek-R1-0528-Qwen3-8B (Avg@12, %, over four mathematical benchmarks). (a) Direction transfer across extraction and training data. (b) Discovery-set size, with four trajectories per problem. Cosine is measured against the layer-18 direction of the shaded main setting (512 OpenThoughts-Math-30K problems, $B = 8{,}192$).

(a) Direction transfer

Direction data	Training data	Cosine	Avg@12
Base model		—	67.6
OpenThoughts	OpenThoughts	1.000	**71.9**
OpenThoughts	DeepMath (7–10)	1.000	70.1
DeepMath (≤ 3)	DeepMath (7–10)	0.949	70.6
DeepMath (7–10)	DeepMath (7–10)	0.711	71.0

(b) Discovery-set size

Problems	B	Cosine	Avg@12
512	8,192	1.000	71.9
256	9,216	0.951	70.4
128	8,192	0.954	**72.1**
64	9,216	0.935	71.5
32	9,216	0.841	71.0
16	8,192	0.824	70.8

5 Related Work

On-Policy Distillation and Self-Distillation. MiniLLM uses reverse KL (Gu et al., 2024), and DistiLLM combines skew KL with adaptive reuse of student-generated responses (Ko et al., 2024). OPSD conditions the teacher on a reference solution (Zhao et al., 2026), while SD-Zero trains a reviser using correctness feedback and distills its predictions (He et al., 2026). RLCSD contrasts correct and incorrect sibling rollouts to modulate the GRPO advantage (Pan et al., 2026); OGLS-SD derives teacher-logit adjustments from successful and failed rollout contexts (Yang et al., 2026). Other variants control reference exposure, learn soft prompts, or remove reference-induced log-probability terms (Han et al., 2026; Ma et al., 2026; Shen et al., 2026).

Activation Steering for Reasoning. Representation engineering studies how activation directions encode and control model behavior (Zou et al., 2023); inference-time intervention improves truthfulness by shifting selected attention-head activations (Li et al., 2023). Hidden-state probes also predict the correctness of intermediate answers (Zhang et al., 2025). Reasoning-specific directions capture execution and reflection or behaviors such as uncertainty estimation, backtracking, and example testing (Chen et al., 2025; Venhoff et al., 2025); SEAL and Efficiency Steering apply them during decoding (Chen et al., 2025; Zhao et al., 2025), and S3-CoT uses a direction to collect variable-length trajectories for supervised fine-tuning (Du et al., 2026). SafeSteer constructs a refusal teacher from activation differences between harmful and harmless instructions, then restricts reverse-KL distillation to a global set of selected safety tokens (Li et al., 2026). ACSD constructs its direction from verifier-labeled reasoning rollouts and provides token-level supervision throughout complete student-generated rollouts on reasoning tasks.

6 Discussion

ACSD shows how trajectory-level verification can support token-level supervision through activation differences. The correct short–long ablation shows that useful supervision can come from differences among successful trajectories that share a correctness label, with the generation budget selecting this contrast; recovery evaluation shows that the intervention improves continuation success under a fixed budget, and distillation retains the gains in the student's parameters. The construction fixes one direction, layer, and strength per training setting. Adapting the intervention to the prefix, evaluating code generation on further model families, and defining contrasts from richer execution feedback are natural extensions.

AI USE STATEMENT

In this work, we used generative AI tools to implement and execute parts of the experimental pipeline, specifically to write and debug the job scripts used for training, sampling, and scoring, and to monitor these runs, and for data annotation, specifically to assign reasoning roles to spans of model-generated text in the token-level analysis (Appendix A.4). We have not used generative AI tools to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide ingredients for proofs or to write proofs, to propose or refine hypotheses, to design or provide feedback on the research methodology or experiments, or to interpret results; translation is not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability, to create and edit software code, and to search for and identify relevant literature. We have reviewed all AI-assisted work: AI-edited text was checked by the authors for meaning and accuracy; AI-assisted code was reviewed and tested by the authors, and the experiment runs were verified against their logs and outputs; the annotation protocol and its robustness checks are documented in the appendix; every reference found with AI tools was read by the authors before being cited. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI tools.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or newly collected data. All experiments use publicly released open-weight models and public datasets: the OpenThoughts-Math-30K split released with OPSD, DeepMath, Codeforces problems from the Open-R1 release, and the AIME, HMMT, and LiveCodeBench benchmarks, each used under its published terms. The method trains a model on its own generated trajectories and does not add capabilities beyond those of the underlying base models; its potential for misuse is therefore that of the base models themselves. The role annotation in the token-level analysis (Appendix A.4) was performed by a language model on model-generated text only. The authors are not aware of conflicts of interest related to this work.

REPRODUCIBILITY STATEMENT

The method is specified in Section 3: trajectory labeling and direction extraction in Section 3.2, layer and strength selection in Section 3.3, and the training objective in Section 3.4, with the full procedure in Algorithm 1. Appendix A lists the per-model direction settings (Table 3), the optimization hyperparameters, the baseline configurations, the evaluation sampling settings (Table 4), and the dataset identifiers; the prefix construction for recovery evaluation is described in Appendix A.2, the token-level analysis protocol in Appendix A.4, and the direction controls, training-window, transfer, discovery-set-size, and inference-time steering experiments in Appendices A.3 and A.5 to A.7. All datasets, benchmarks, and models are publicly available, and random seeds are fixed as stated in the appendix. An anonymized repository at `https://anonymous.4open.science/r/Activation_Condition_Self_Distillation-EF27` contains the training and evaluation code, the extracted steering directions, the fixed prefixes used for recovery evaluation, and launch scripts that reproduce the DeepSeek-R1-0528-Qwen3-8B results for mathematics and code, the recovery scan, and the direction controls.

REFERENCES

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

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. SEAL: Steerable reasoning calibration of large language models for free. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=klPszYDIRT`.

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

Yanrui Du, Sendong Zhao, Yibo Gao, Danyang Zhao, Qika Lin, Ming Ma, Jiayun Li, Yi Jiang, Kai He, Qianyi Xu, Bing Qin, and Mengling Feng. S3-cot: Self-sampled succinct reasoning enables efficient chain-of-thought llms. *arXiv preprint arXiv:2602.01982*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ac015d409635f196f9e3e9dcfb9a94e-Paper-Conference.pdf.

Zihao Han, Tiangang Zhang, Huaibin Wang, and Yilun Sun. Adaptive teacher exposure for self-distillation in llm reasoning. *arXiv preprint arXiv:2605.11458*, 2026.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-distillation zero: Self-revision turns binary rewards into dense supervision. *arXiv preprint arXiv:2604.12002*, 2026.

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

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

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*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24872–24895. PMLR, 2024. URL https://proceedings.mlr.press/v235/ko24c.html.

Hao Li, Jingkun An, Zijun Song, Pengyu Zhu, Rui Li, Hao Wang, Wendi Feng, Yesheng Liu, Lijun Li, Jin-Ge Yao, and Lei Sha. SafeSteer: Localized on-policy distillation for efficient safety alignment. *arXiv preprint arXiv:2606.02530*, 2026. doi: 10.48550/arXiv.2606.02530. URL https://arxiv.org/abs/2606.02530.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

Yingzi Ma, Zichen Zhu, Ming Jiang, and Chaowei Xiao. One student, many teachers: Multi-task on-policy distillation via soft-prompt privileged context. *arXiv preprint arXiv:2607.18293*, 2026.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. Rlcsd: Reinforcement learning with contrastive on-policy self-distillation. *arXiv preprint arXiv:2606.11709*, 2026.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

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

Zhanming Shen, Jintao Tong, Shaotian Yan, Chen Shen, Hao Chen, Wentao Ye, Xiaomeng Hu, Rui Miao, Haobo Wang, Junbo Zhao, Gang Chen, and Jieping Ye. Purified OPSD: On-policy self-distillation without losing how to think. *arXiv preprint arXiv:2607.02234*, 2026.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026. URL https://arxiv.org/abs/2601.19897.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. doi: 10.48550/arXiv.2512.13961.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=OwhVWNOBcz.

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. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388.

Yuxiao Yang, Xiaoyun Wang, and Weitong Zhang. Ogls-sd: On-policy self-distillation with outcome-guided logit steering for llm reasoning. *arXiv preprint arXiv:2605.12400*, 2026.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=O6I0Av7683.

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. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

Weixiang Zhao, Jiahe Guo, Yang Deng, Xingyu Sui, Yulin Hu, Yanyan Zhao, Wanxiang Che, Bing Qin, Tat-Seng Chua, and Ting Liu. Exploring and exploiting the inherent efficiency within large reasoning models for self-guided efficiency enhancement. *arXiv preprint arXiv:2506.15647*, 2025.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023. URL https://arxiv.org/abs/2310.01405.

A Experimental Details and Supplementary Results

A.1 Experimental Setup

For the results in Table 1, we follow the checkpoint selection protocol of OPSD (Zhao et al., 2026).

Direction Extraction. For each model, we select 512 problems from the OpenThoughts-Math-30K split released with OPSD and sample four closed-book trajectories per problem, yielding 2,048 trajectories. The split is available as `siyanzhao/Openthoughts_math_30k_opsd`. We extract the last balanced `\boxed{}` expression from each response and compare it with the dataset answer using `math_verify`. When symbolic parsing fails, we use normalized exact matching. Trajectories with verified final answers and at most B generated tokens enter the positive group; the rest enter the contrast group.

The budget for each model is determined from the analysis in Section 3.1. We extract activations from the first $\min(|r_i|, B)$ generated tokens of each trajectory, normalize each activation to unit norm, and pool them by group. Normalizing the difference between the two group means gives the direction at that layer. Table 3 gives the settings for the main mathematical reasoning experiments. Layer indices are zero-based.

Table 3: ACSD settings for the main mathematical reasoning experiments. B is used for trajectory labeling and activation extraction; α is the intervention strength relative to the current hidden-state norm.

Model	B	Layer	α	Training window
DS-R1-0528-Qwen3-8B	8,192	18	0.30	4,096
DS-R1-Distill-Qwen-7B	4,096	14	0.30	4,096
OLMo-3-7B-Think	4,096	16	0.10	4,096
Qwen3-8B	4,096	13	0.05	4,096
Qwen3-4B	4,096	13	0.05	4,096

Optimization. In the main mathematical reasoning experiments, SFT, GRPO, OPSD, and ACSD use the public OPSD framework. We train for 200 updates on eight H100 GPUs, saving a LoRA adapter every 25 updates. We use AdamW with learning rate 5×10^{-6}, linear decay, effective batch size 32, and maximum gradient norm 0.1. LoRA is applied to the query, key, value, output, gate, up, and down projections with rank 64 and LoRA alpha 128.

ACSD and OPSD use distillation temperature $T_{\mathrm{d}} = 1.1$ and pointwise clipping threshold $\tau = 0.06$. ACSD uses the windows in Table 3; OPSD uses 4,096 tokens. DeepSeek and Qwen student rollouts use temperature 1.1, top-p 0.95, and top-k 20. OLMo uses temperature 0.6, top-p 0.95, and no top-k truncation.

SFT trains on reference reasoning trajectories with a 16,000-token window. GRPO samples eight responses per problem with a 16,384-token window and uses the answer verifier to assign binary rewards; its KL coefficient is zero. RLCSD uses its released implementation for 450 updates on the same training split. It samples eight responses per problem, uses learning rate 10^{-6} with 50 warmup updates and a 16,384-token window, and saves checkpoints every 50 updates. Four negative hints are selected from the eight responses to the same problem.

Teacher Computation and Student Updates. For each update, the student first generates a response. The student and frozen teacher then compute next-token distributions on the same student prefixes. ACSD applies the direction starting at the final prompt state, which predicts the first response token, and continues at all subsequent states that predict response tokens. The teacher parameters, direction, layer, and strength remain fixed. We treat the sampled token sequence and teacher distributions as constants and update the student using the clipped objective in Section 3.4. Subsequent rollouts use the updated student, which is also used for benchmark evaluation. Algorithm 1 summarizes the procedure.

Mathematical Evaluation. We sample 12 responses per problem on AIME 2024, AIME 2025, HMMT February 2025, and AIME 2026, 30 problems each (Hub identifiers `HuggingFaceH4/aime_2024`, `MathArena/aime_2025`, `MathArena/hmmt_feb_2025`, and `MathArena/aime_2026`). Table 4 lists the decoding settings for each model family. Methods using the same model share these settings, the answer parser, and the verifier. Min-p and the presence penalty are zero. DeepSeek-family generation uses a 40,960-token context and begins reasoning with `<think>`; Qwen3 uses thinking mode.

Algorithm 1 Activation-Conditioned Self-Distillation

Require: Base model π_0, problems $\mathcal{D}$, verifier V, budget B, candidate layers $\mathcal{L}$, strengths $\mathcal{A}$, distillation temperature T_{d}, clipping threshold τ

Direction discovery

Generate closed-book rollouts $r_i \sim \pi_0(\cdot \mid x_i)$
Label $u_i \leftarrow \mathbf{1}\{V(x_i, r_i) = 1 \text{ and } |r_i| \leq B\}$
Collect activations within B, normalize each to unit norm, and pool them by u_i
Compute $\boldsymbol{d}^{(\ell)}$ for each $\ell \in \mathcal{L}$ using Equation (3)
Choose (ℓ^*, α) by recovery on the calibration pool

On-policy self-distillation

Initialize $\pi_\theta \leftarrow \pi_0$ and freeze a copy of π_0 as the teacher $\widetilde{\pi}_0$
for each update **do**
    Sample $x \sim \mathcal{D}$
    Sample unsteered $y \sim \pi_\theta(\cdot \mid x)$
    Compute p_t^S and p_t^T on $(x, y_{<t})$ using Equations (4) and (5)
    Compute $\ell_{t,v}$ using Equation (6)
    Update θ with the clipped objective in Equation (7)
Discard the frozen teacher and $\boldsymbol{d}$

Table 4: Decoding settings for mathematical reasoning benchmarks.

Model family	Max tokens	Temperature	Top-p	Top-k
DeepSeek-R1	38,912	0.6	0.95	–
OLMo-3-Think	38,912	0.6	0.95	–
Qwen3	38,912	1.0	0.95	20

Code Generation. Code experiments use 5,000 Codeforces problems from the Open-R1 release (`open-r1/codeforces`, configuration `verifiable-prompts`), restricted to problems with a Python prompt. ACSD trains on the problem statements alone; SFT and OPSD use the Python reference implementations of the same problems from `open-r1/codeforces-cots`. The direction is extracted from a separate pool of trajectories: four closed-book trajectories for each of 512 Codeforces problems, generated with a 32,768-token limit and labeled by the official tests, with $B = 16{,}384$, layer 18, and $\alpha = 0.10$. ACSD and OPSD use the optimizer, LoRA settings, sampler, and clipping of the mathematical experiments with a 4,096-token window and 200 updates, saving an adapter every 25 updates; GRPO and RLCSD train for 450 updates with a 16,384-token window, RLCSD with full-parameter updates. Evaluation uses the 175 problems of LiveCodeBench v6 (`livecodebench/code_generation_lite`) with 12 samples per problem, temperature 0.6, top-p 0.95, no top-k truncation, and up to 32,768 generated tokens, scored with the official checker.

A.2 Recovery Evaluation and Configuration Selection

Recovery evaluation uses the calibration pool: 256 problems held out from direction extraction. We shuffle them with seed 17, select one eligible contrast-group trajectory per problem, and assign prefix fractions of 25%, 50%, and 75% in round-robin order. The initial DeepSeek-R1-0528-Qwen3-8B scan uses the full contrast group. Later model scans use contrast-group trajectories that terminate naturally and contain a boxed answer.

We continue each fixed prefix with and without the direction, using the same sampling settings and continuation budget across configurations. Each prefix receives two continuations, each capped at 12,288 generated tokens. Recovery is the fraction of continuations whose final answers pass verification, with 512 continuations per layer–strength setting. DeepSeek and Qwen use temperature 1.1, top-p 0.95, and top-k 20. OLMo uses temperature 0.6, top-p 0.95, and no top-k truncation.

The initial scan on DeepSeek-R1-0528-Qwen3-8B includes layers $\{2, 4, 6, 13, 18, 23\}$ and strengths $\{0.0125, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.3, 0.5, 0.75, 1, 1.5, 2\}$. Figure 5 shows higher recovery in intermediate layers. The selected layer 18 and $\alpha = 0.3$ raise recovery from 26.6% to 66.0%. Later models use five candidate layers and $\alpha \in \{0.05, 0.1, 0.2, 0.3\}$. The layer grids are $\{6, 10, 14, 18, 22\}$ for DeepSeek-R1-Distill-Qwen-7B, $\{6, 11, 16, 20, 25\}$ for OLMo-3-7B-Think,

and $\{6, 13, 18, 23, 29\}$ for both Qwen3 models. Trajectory labels and directions at each layer are fixed before the scan; recovery selects the intervention layer and strength.

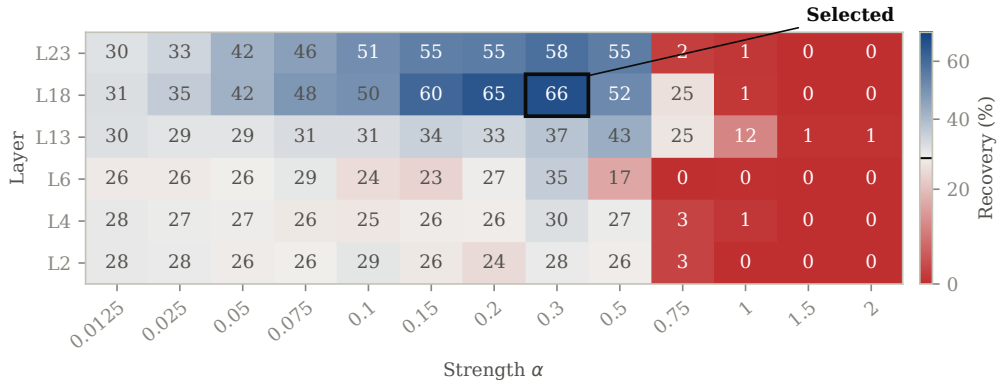

Figure 5: Recovery scan on DeepSeek-R1-0528-Qwen3-8B. Each cell reports the percentage of 512 continuations that pass answer verification. The black outline marks layer 18 and $\alpha = 0.3$, used for distillation. Recovery without steering is 26.6%.

Downstream Sensitivity. We compare six intervention configurations in distillation: at layer 18, $\alpha \in \{0.05, 0.3, 0.75\}$; at $\alpha = 0.3$, layers 4, 13, 18, and 23. All settings share the model, training data, optimizer, and evaluation protocol. The recovery-selected setting reaches 71.9 Avg@12, while the five alternatives range from 68.7 to 69.9.

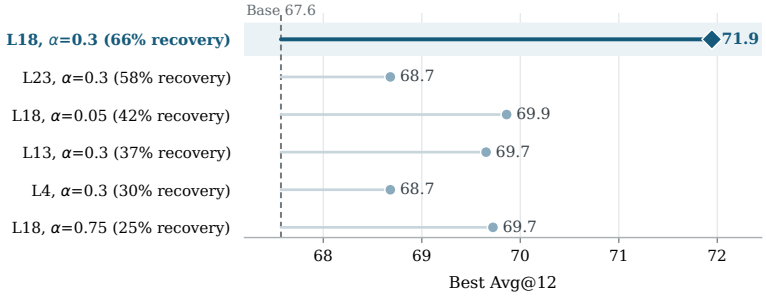

Figure 6: Distillation results for six intervention settings on DeepSeek-R1-0528-Qwen3-8B. The horizontal axis shows the best mean Avg@12 across four mathematical reasoning benchmarks. Labels also give recovery; the outline marks the main setting.

A.3 Complete Direction Controls

Trajectory Contrasts. The correct short–long direction contrasts 914 correct trajectories that finish within the generation budget with 913 correct trajectories that finish beyond it. The length-matched correctness direction contrasts all 221 incorrect trajectories with 221 correct trajectories selected by nearest-neighbor matching on response length. Their mean lengths differ by 78 tokens. All groups use the same activation normalization and difference-in-means construction.

Direction Perturbations. The random direction is an isotropic unit vector. The shuffled direction randomly permutes the coordinates of the ACSD direction, and the reversed direction is $-\boldsymbol{d}$. These controls share ACSD's intervention layer, strength, training data, and optimizer. Figure 3 shows the trajectory and perturbation comparisons.

A.4 Token-Level Update Analysis

Fixed-Trajectory Evaluation. We use 512 fixed student trajectories from DeepSeek-R1-0528-Qwen3-8B, one per problem, split into 256 discovery and 256 held-out trajectories. The base model, ACSD teacher, and OPSD teacher process the same problem and student prefix at every position.

Marker groups are determined on the discovery set, and measurements are reported on the held-out set. The role analysis uses full held-out trajectories to retain less frequent roles.

Update Magnitude. At prefix s, let $\boldsymbol{u}_s$ denote the negative gradient of the clipped distillation loss with respect to student logits, with $u_s(v)$ denoting its coordinate for vocabulary entry v. We group prefixes by absolute response position into five bins: 0–2k, 2–4k, 4–8k, 8–16k, and beyond 16k tokens. For bin b containing N_b prefixes, the update magnitude is

$$R_b = \sqrt{\frac{1}{N_b} \sum_{s \in b} \|\boldsymbol{u}_s\|_2^2}. \tag{8}$$

Both teachers compute updates on the same observed text. OPSD's magnitude relative to ACSD is 97% in the first 2k tokens, 85% in the next 2k, and 66% beyond 16k. Over the same range, the cosine similarity between their update vectors falls from 0.298 to 0.108.

Role Annotation. We adapt the reasoning-behavior taxonomy of Venhoff et al. (2025), adding a `final-answer` category; spans are segmented and annotated with the protocol described below. We split trajectories at sentence and paragraph boundaries, with at most 900 characters per span. The annotator, `deepseek-chat`, uses temperature 0 and assigns one role per span from the response text and the definitions in Table 5. Malformed or missing outputs are marked unlabeled and excluded from role aggregates.

Table 5: Reasoning roles used in the token-level audit.

Figure label	Annotator label	Definition
Setup	`initializing`	Restating the task, setting up variables, or planning.
Deduction	`deduction`	Deriving a conclusion from the current assumptions.
Known facts	`adding-knowledge`	Recalling a theorem, formula, identity, or external fact.
Examples	`example-testing`	Checking a case, example, or sanity check.
Uncertainty	`uncertainty-estimation`	Expressing uncertainty, hedging, or assessing confidence.
Backtracking	`backtracking`	Abandoning, revising, or correcting an approach.
Final answer	`final-answer`	Presenting, boxing, or restating the final answer.

The annotation instruction is reproduced below.

```
You are annotating reasoning traces for a mechanistic analysis.
Label each span with exactly one label. Do not merge or drop spans.

Allowed labels:
- initializing: restating the task, setting up variables, or planning
- deduction: deriving a conclusion from current assumptions
- adding-knowledge: recalling a theorem, formula, identity, or fact
- example-testing: checking a case, example, or sanity check
- uncertainty-estimation: expressing uncertainty, hedging, or confidence
- backtracking: abandoning, revising, or correcting an approach
- final-answer: presenting, boxing, or restating the final answer

Return strict JSON mapping every span id to one allowed label.
```

Marker Groups and Aggregation. We use the seven groups of single-token markers in Table 6. Leading-space variants are retained when the tokenizer maps them to distinct single tokens. Marker groups and reasoning role labels are determined separately.

Let $\mathcal{S}_r$ be the set of held-out prefixes with role r and $\mathcal{V}_g$ the vocabulary entries in marker group g. The percentage shown in each heatmap cell is

$$A_{g,r} = 100 \times \frac{\sum_{s \in \mathcal{S}_r} \sum_{v \in \mathcal{V}_g} u_s(v)}{\sum_{s \in \mathcal{S}_r} \|\boldsymbol{u}_s\|_1}. \tag{9}$$

Figure 4 shows uncertainty estimation and final answers; Figure 7 shows all seven roles. Both teachers use the same marker groups, role labels, normalization, and color scale. An expanded marker

Table 6: Marker groups used in the token-level update analysis.

Group	Examples
Hedging	probably, perhaps, maybe, possibly
Contrast	But, but, However, however
Revision	wait, Wait, Actually, actually, reconsider
Verification	check, verify
Progress	so, then, Then, implies
Conclusion	Therefore, Thus, hence, conclude
Answer form	final, Final, answer, Answer, boxed

lexicon and leave-one-marker-out analysis preserve the main sign patterns described in Section 4.4. We also compare random token groups matched by discovery-set occurrence and update scale. The aggregate EOS update is near zero and slightly negative.

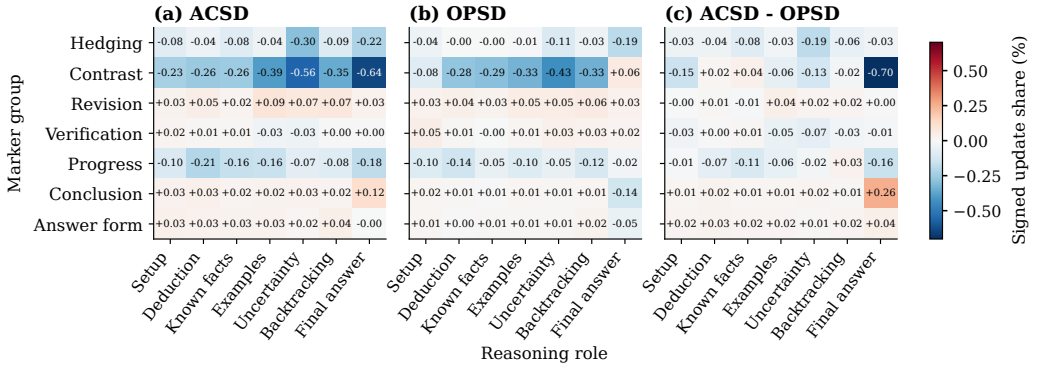

Figure 7: Complete role and marker updates on 256 held-out trajectories. (a) ACSD. (b) OPSD. (c) ACSD minus OPSD normalized updates. Rows are marker groups; columns are reasoning roles.

A.5 Training-Window Experiments

We run a separate 100-update experiment on DeepSeek-R1-0528-Qwen3-8B. ACSD uses windows of 1k, 2k, 4k, 8k, and 16k tokens; OPSD uses 1k, 2k, 4k, and 8k. Within each method, we hold the model, optimizer, training data, and evaluation protocol fixed. ACSD also keeps the direction and intervention settings fixed. Only the training window changes.

Figure 8 shows the full training curves. ACSD improves over the base model at every tested window. OPSD at 4k exceeds the base model early in training and reaches 67.4 at update 100, slightly below the base model's 67.6. The curves for its other tested windows remain below the base model.

A.6 Direction Transfer and Discovery-Set Size

Transfer across Data and Difficulty. The OpenThoughts transfer setting retains the direction extracted from OpenThoughts-Math-30K and replaces the distillation problems with DeepMath Level 7–10, using the split released with RLCSD (`Leyiii/RLCSD`). In the easy-to-hard comparison, we collect 2,048 trajectories each from DeepMath problems with difficulty at most 3 and from Level 7–10 problems. Their budgets are 6,144 and 19,456 tokens, yielding 50.6% and 50.7% within-budget verified trajectories. Both directions are used for distillation on the same hard problems, with the same model, intervention layer, strength, optimizer, and evaluation settings.

Table 2(a) summarizes the transfer results. At layer 18, the directions from easy and hard DeepMath problems have cosine similarities of 0.949 and 0.711 with the OpenThoughts-Math-30K direction, respectively. These comparisons use the direction employed for distillation in each setting and its corresponding generation budget.

Discovery-Set Size. We use nested subsets of 512, 256, 128, 64, 32, and 16 discovery problems for DeepSeek-R1-0528-Qwen3-8B, sampling four trajectories per problem. Table 2(b) gives the budget,

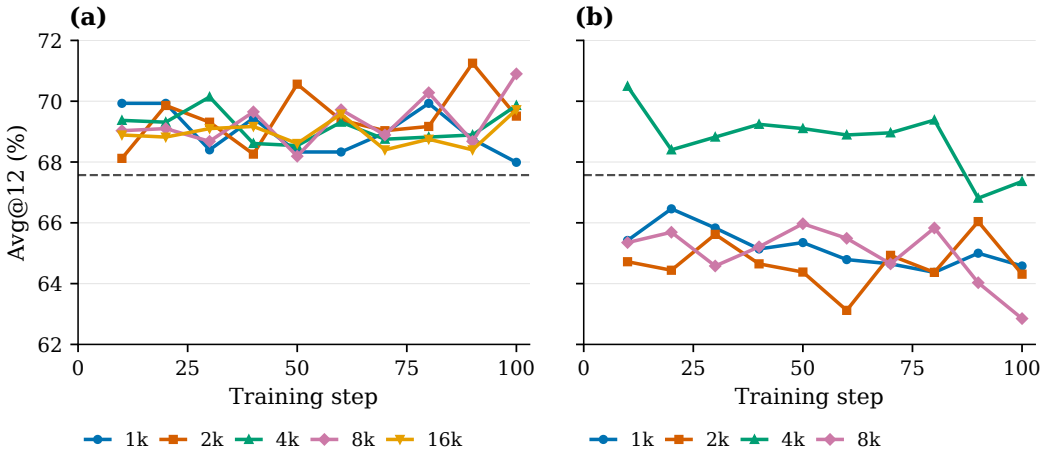

Figure 8: Training curves for different windows on DeepSeek-R1-0528-Qwen3-8B. We report mean Avg@12 across four mathematical reasoning benchmarks over the first 100 updates. (a) ACSD. (b) OPSD. Lines connect evaluations; dashed lines mark the base model.

cosine similarity to the 512-problem direction, and distillation result for each subset. Cosines are computed from the normalized direction vectors at layer 18. As the discovery set shrinks, we keep layer 18 and $\alpha = 0.3$ fixed and use the main mathematical evaluation settings.

A.7 Inference-Time Steering

We apply the direction, layer, and strength selected for teacher construction throughout full autoregressive generation from the base model. The intervention is active whenever the model predicts a response token. Evaluation follows the main mathematical decoding settings. Table 7 reports the results.

Table 7: Direction intervention throughout generation. Values are average Avg@12 (%) across four mathematical reasoning benchmarks; Δ denotes Steered minus Base. Intervention settings come from Table 3.

Model	Layer / α	Base	Steered	Δ
DS-R1-0528-Qwen3-8B	$(18, 0.30)$	67.6	25.6	-42.0
DS-R1-Distill-Qwen-7B	$(14, 0.30)$	40.4	13.8	-26.6
OLMo-3-7B-Think	$(16, 0.10)$	64.1	62.2	-1.9
Qwen3-8B	$(13, 0.05)$	63.3	62.1	-1.2
Qwen3-4B	$(13, 0.05)$	62.3	62.4	$+0.1$

This evaluation changes the autoregressive trajectory from the first generated token. Recovery evaluation starts from a fixed prefix and compares success under the same continuation budget. The ACSD teacher computes distribution targets on prefixes already generated by the student, and final evaluation uses the distilled student. These settings examine full generation, fixed-prefix continuation, and performance after student training, respectively.