WHEN DOES TEACHER FEEDBACK HURT? OUTCOME-ALIGNED REWEIGHTING FOR ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) provides dense teacher feedback on student-generated trajectories, but not every teacher-induced update improves task success. We measure the causal utility of this feedback by applying the OPD update contributed by a single token position and measuring the resulting change in the student's probability of solving the same prompt. These interventions reveal that most token updates have small causal effects on accuracy, while a few produce large improvements or substantial degradation. To estimate utility without performing costly interventions, we derive a first-order score based on the alignment between each token's distillation update and the student's outcome policy gradient. The scores correlate strongly with measured intervention effects across training checkpoints, with Pearson correlations of over 0.9. We then propose ALIGN-OPD (outcome-ALIGNed On-Policy Distillation), which uses these scores to reweight token-level teacher feedback during training. Across instruction following, code, and math domains, ALIGN-OPD improves mean domain accuracy by $+1.6$ points over the RL teacher, $+1.7$ over vanilla OPD, and $+1.4$ over the strongest competing baseline.

1 INTRODUCTION

Knowledge distillation offers a direct way to learn from a stronger model by training a student to match its predictions (Hinton et al., 2015). For autoregressive language models, the prefixes in a fixed training corpus may differ from those the student generates at inference time. On-policy distillation (OPD) addresses this mismatch by querying the teacher on prefixes generated by the current student (Agarwal et al., 2024), providing dense token-level feedback throughout the student's own responses, even when a complete answer may receive only a single correctness signal.

Learning from the teacher, however, does not guarantee that each token update improves the student's ability to solve the task. A teacher can be stronger overall while still providing locally unhelpful or harmful feedback. Recent methods select or reweight teacher feedback using signals such as teacher confidence, teacher–student probability differences, and trajectory correctness (Li et al., 2026; Xu et al., 2026a). These signals do not directly measure what happens when the student learns from a particular piece of feedback. This leaves a concrete question: **does applying the OPD update at a given token position make the student more likely to answer correctly?**

We answer this question by testing the consequences of individual token updates. Starting from a fixed student checkpoint, we apply the OPD update associated with one position and evaluate the original and updated students using fresh responses to the same prompt. We call the resulting change in the student's probability of answering correctly the causal utility of the feedback. These interventions reveal sharp differences across positions: most updates have little measured effect, while a small number substantially improve accuracy or even substantially reduce it. This pattern persists throughout training. Teacher feedback can therefore have large and opposing effects even at different positions within a single response.

Using these differences during training requires predicting utility without separately updating and evaluating the model for every token. We derive a first-order approximation that relates a token's OPD update to the gradient of the student's success probability. Estimated from the student's suc-

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.

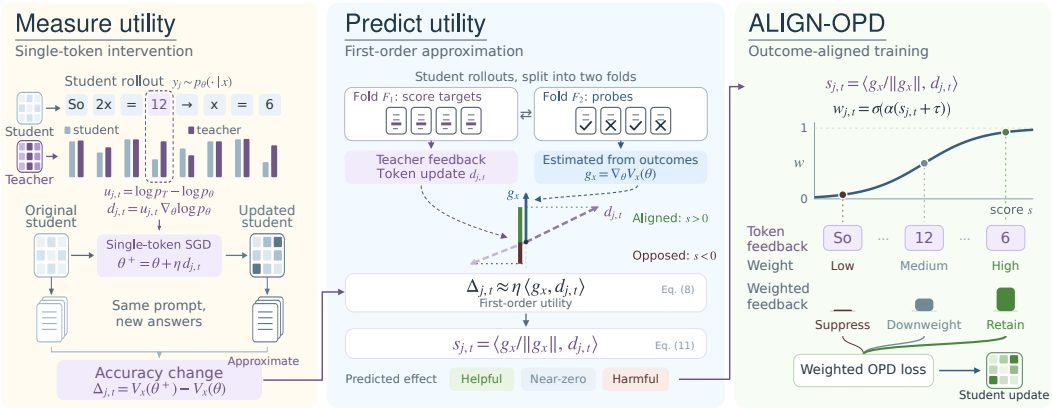


Figure 1: **Overview of our method.** (**Left**) We measure causal utility by applying individual token-level OPD updates to the student and evaluating the change in same-prompt accuracy. (**Mid**) A first-order approximation predicts these effects through alignment between each token update and an outcome policy gradient. (**Right**) ALIGN-OPD uses the resulting scores to reweight teacher feedback during training, downweighting updates predicted to reduce accuracy.

cessful and unsuccessful responses, this outcome gradient indicates how parameter changes affect accuracy. To first order, an update aligned with this gradient improves the probability of success, whereas an opposing update reduces it. This gives a token-level alignment score that correlates strongly with the measured intervention effects throughout training.

Building on this connection, we introduce ALIGN-OPD, which uses the scores to reweight teacher feedback during training. It downweights token updates predicted to conflict with task success while retaining aligned feedback. Outcome information thus guides how much the student learns from each token's teacher signal. On `SmolLM3`, across instruction following (IF), code, and math, ALIGN-OPD improves mean domain accuracy from 23.4 to 31.3 over the MixSFT student, and further improves over vanilla OPD by 1.7 points and the strongest competing baseline by 1.4 points. On `Qwen3-4B-Non-Thinking`, ALIGN-OPD surpasses vanilla OPD by 0.9 points on code generation and 1.3 points on math reasoning.

Overall, our contributions are threefold:

- We measure the causal utility of individual OPD token updates through parameter interventions, revealing that a small number of positions produce large beneficial or harmful effects on accuracy.
- We derive an outcome-gradient alignment score and validate its predictions against measured intervention effects across training checkpoints.
- We introduce ALIGN-OPD, which reweights teacher feedback according to its alignment with an outcome-improving direction and consistently improves over vanilla OPD across instruction following, code, and math.

2 Background and Related Work

We first introduce OPD (On-Policy Distillation) and the notations used throughout the paper, then discuss prior work on teacher feedback adaptation and gradient-based attribution.

2.1 On-policy distillation

Let $x \sim \mathcal{D}$ be a prompt, let p_θ be the student policy, and let q be a fixed teacher. At the beginning of an update, the behavior student is frozen as $p_{\bar{\theta}}$. It samples a response $y = (y_1, \ldots, y_T) \sim p_{\bar{\theta}}(\cdot \mid x)$, inducing states $z_t = (x, y_{<t})$. OPD evaluates the teacher on these student-visited states (Agarwal et al., 2024) rather than on a fixed teacher-generated corpus. This choice reduces

the mismatch between the prefixes used for training and those encountered by the student at inference time.

Specifically, we focus on reverse-KL OPD, widely used when the student cannot cover all modes of a stronger teacher (Gu et al., 2024; Agarwal et al., 2024; Lu & Lab, 2025). With the rollout distribution treated as fixed during an update, its objective is

$$\mathcal{L}_{\mathrm{OPD}}(\theta;\bar{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim p_{\bar{\theta}}}\left[\sum_{t=1}^{T} D_{\mathrm{KL}}(p_\theta(\cdot \mid z_t) \,\|\, q(\cdot \mid z_t))\right]. \tag{1}$$

At $\theta = \bar{\theta}$, sampling a token from the student distribution gives the sampled-token form of gradient,

$$-\nabla_\theta \mathcal{L}_{\mathrm{RKL}}(\theta;\bar{\theta})\big|_{\theta=\bar{\theta}} = \mathbb{E}\left[\sum_{t=1}^{T} u_t \,\nabla_\theta \log p_\theta(y_t \mid z_t)\big|_{\theta=\bar{\theta}}\right], \tag{2}$$

where the residual

$$u_t = \log q(y_t \mid z_t) - \log p_{\bar{\theta}}(y_t \mid z_t). \tag{3}$$

Let $\ell_t(\theta) = \log p_\theta(y_t \mid z_t)$, the contribution from one sampled token can be written as

$$d_t^{\mathrm{OPD}} = u_t \nabla_\theta \ell_t(\bar{\theta}). \tag{4}$$

The vector d_t^{OPD} is the local update direction for that sampled token in the student's parameter space. Standard OPD aggregates these token-level contributions uniformly across the trajectory. Our central question is whether such teacher-induced update is always beneficial for task success and should be executed.

2.2 Selecting and regulating teacher feedback

Recent OPD methods adapt teacher feedback using several types of evidence. *Uncertainty-based methods* identify where supervision may be useful: TIP (Xu et al., 2026b) identifies informative tokens using student uncertainty and teacher–student disagreement. FiRe-OPD (Li et al., 2026) concentrates supervision on uncertain students with confident teachers. EOPD (Jin et al., 2026) adds forward-KL supervision at high-teacher-entropy positions. *Probability-based regulation* controls extreme or unreliable distillation signals. REOPOLD (Ko et al., 2026) clips negative log-probability residuals; TrOPD (Xing et al., 2026) uses teacher–student probability ratios to choose between reverse- and forward-KL supervision.

Another line of work, *Outcome-aware methods*, additionally use correctness to assess feedback. Reward-Gated OPD (Akhondzadeh et al., 2026) filters trajectories by agreement between an outcome advantage and an aggregate teacher–student likelihood gap. SG-OPD (Xu et al., 2026a) uses sign consistency between verifier-derived advantages and token distillation signals to control the strength of supervision. Going beyond trajectory outcomes, SPOT (Qu et al., 2026) probes continuations from selected token branches and uses their success rates to calibrate local teacher targets.

2.3 Gradient-based attribution, token selection and reweighting

Influence functions (Koh & Liang, 2017) estimate how changing training data affects model predictions, while TracIn (Pruthi et al., 2020) traces influence through gradient alignment across training checkpoints. Related ideas guide optimization: meta-reweighting (Ren et al., 2018) uses a clean validation objective to weight training examples, and LESS (Xia et al., 2024) uses gradient-based influence estimates to select instruction-tuning data for target tasks.

Related ideas have recently been applied at the token level in OPD. Unmasking OPD (Armandpour et al., 2026) compares distillation gradients with an ideal next-token logit gradient estimated from branch-conditioned success rates. TIP (Xu et al., 2026b) derives an oracle token weight from the alignment between a token's expected distillation gradient and the population gradient of the distillation loss, but uses student entropy and teacher–student divergence as practical proxies. IDA-OPD (Yang et al., 2026a) derives the first-order influence of a sampled-token logit update on entropy and uses it to regulate distillation. Ours differs in targeting at the change in success probability after applying a token's update to the student's parameters. We score that update against an outcome policy gradient estimated from complete responses to the same prompt.

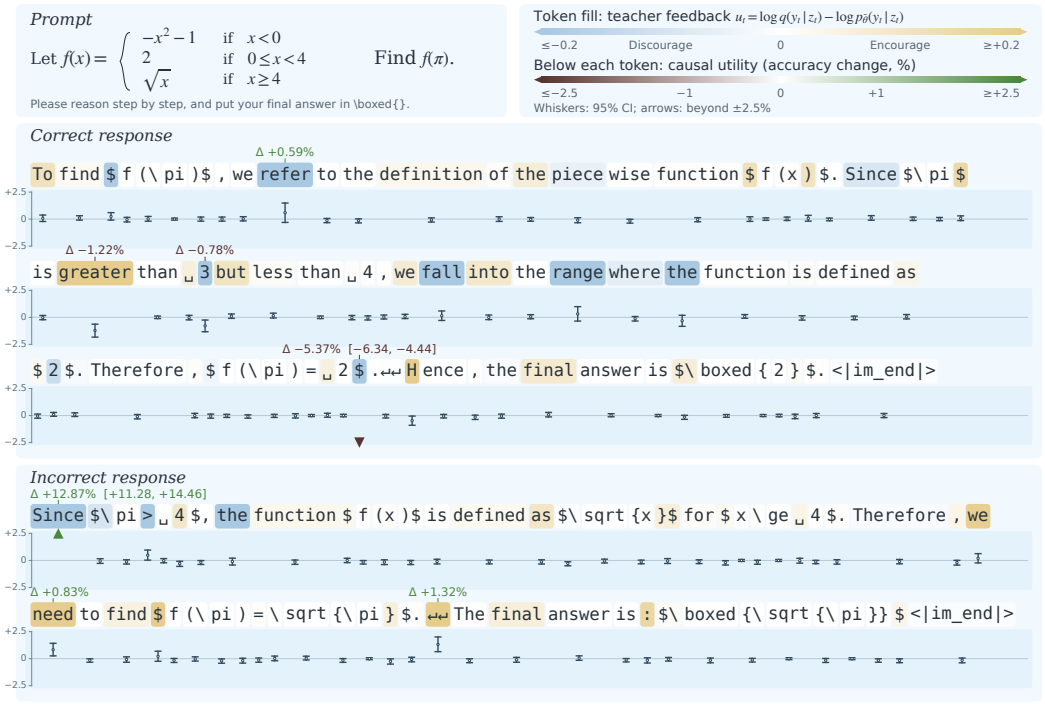


Figure 2: **Large causal effects are concentrated at a few token positions**, regardless of the answer correctness. Most token updates have small effects on accuracy, while a few substantially improve it or cause pronounced degradation. Each effect is measured after a separate single-token OPD update from the same student checkpoint, using repeated paired rollouts on the original prompt. Experiment details are in Appendix B.

3 Causal Utility of Teacher Feedback

OPD provides a teacher feedback signal at every student-generated token, but does following that signal always make the student more likely to solve the problem? We investigate this question by isolating the parameter update contributed by one position and measuring how it changes the student's success probability on the same prompt. We first examine a few complete responses, then test whether the observed pattern recurs across prompts and training checkpoints.

3.1 Measuring the accuracy change from a token update

Consider a response $y \sim p_{\bar{\theta}}(\cdot \mid x)$ generated for prompt x by the student at checkpoint $\bar{\theta}$. At position t, teacher feedback induces the token-level OPD contribution d_t^{OPD} defined in Eq. (4). To isolate the teacher feedback at a single token position t, we update the student using only that position's contribution

$$\theta_t^+ = \bar{\theta} + \eta d_t^{\mathrm{OPD}}, \tag{5}$$

where η is a fixed step size for a single SGD step, and every position is tested from the same original checkpoint. To measure the effect of this update, we use the student's probability of answering x correctly. With $R(x, y) \in \{0, 1\}$ indicating whether a complete response is correct, this probability is

$$V_x(\theta) = \mathbb{E}_{y \sim p_\theta(\cdot|x)}[R(x, y)]. \tag{6}$$

We define the feedback's *causal utility* as the change in this probability after the token update:

$$\Delta_t = V_x(\theta_t^+) - V_x(\bar{\theta}). \tag{7}$$

In particular, although the update uses feedback from a single token, it changes the student's parameters and can affect predictions at other positions. We measure whether the updated student is more likely to answer the prompt correctly, rather than whether teacher feedback encourages next-token choices that lead to more successful continuations from a fixed prefix (Armandpour et al., 2026).

3.2 TOKEN-LEVEL INTERVENTIONS WITHIN COMPLETE RESPONSES

We begin with two responses to the same prompt, one correct and one incorrect, and evaluate every token position. For each intervention, 4,096 paired-seed rollouts from the original and updated students are performed to estimate their success probabilities and the causal utility Δ_t. Each pair uses the same random seed and decoding settings, and both responses are generated from scratch given the original prompt. We use a carefully calibrated learning rate to make parameter updates substantial while still local, avoiding model collapse. Confidence intervals are computed using Tango's score method for matched pairs (Tango, 1998). More implementation details are provided in Appendix B.

As shown in Figure 2, in both responses, most positions have small residuals and little measured effects, while a few updates produce substantial gains or losses. Meanwhile, even some large-residual positions yield near-zero utility, indicating that feedback strength alone does not determine how much it affects accuracy. Furthermore, several positions demonstrate clearly negative utility. This finding supports that a small token subset can retain much of the benefit of full-token distillation (Xu et al., 2026b), while also revealing that some influential updates can even be strongly harmful.

Interestingly, the influential positions may not be semantically decisive. In the incorrect response, discouraging the opening "`Since`" improves accuracy far more than updates at the false comparison "$\pi > 4$". Conversely, in the correct response, discouraging a closing "`$`" causes a significant accuracy drop, and the updated student's responses show failures to produce the required "`\boxed{}`" format. These observations suggest that the most influential updates affect generation behaviors shared across prompts, such as how the student begins a response or formats its answer. Identifying whether these updates help or harm could therefore guide distillation beyond the current example, reinforcing useful behaviors while avoiding changes that undermine task success.

3.3 ACROSS PROMPTS AND TRAINING CHECKPOINTS

We next examine whether these observations hold across prompts and persist beyond the initial student checkpoint. We use the same randomly selected 10 prompts across 6 checkpoints, sampling a new response to each prompt at each checkpoint. For each response, we evaluate 10 token positions with the largest absolute teacher–student residuals $|u_t|$, excluding positions with negligible parameter updates and near-zero utility. Figure 3 summarizes the resulting utility estimates.

Even within this high-divergence subset, most utility estimates remain close to zero. As training progresses, the central distribution becomes narrower, but large positive and negative effects persist, with the largest effects remaining comparable in magnitude. Large teacher–student disagreement alone therefore does not establish whether an update will improve task accuracy.

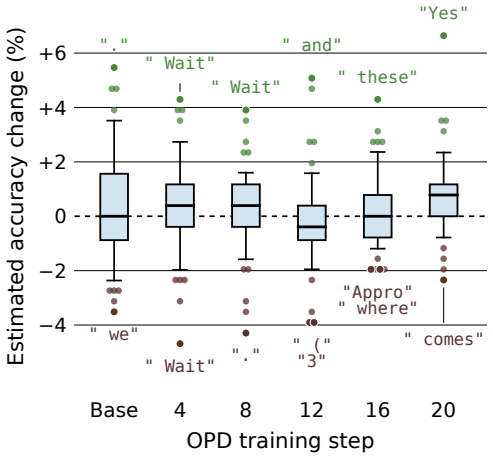


Figure 3: Utility estimates across checkpoints. Boxes show medians and interquartile ranges; whiskers mark the 5th and 95th percentiles. Labels identify extreme tokens.

Takeaway: Together, these interventions show that teacher feedback can induce both beneficial and strongly harmful student updates. This motivates weighting feedback by the causal utility of the resulting update, giving more weight to feedback that improves task accuracy and less to feedback that reduces it. Directly measuring Δ_t for every training token, however, would require a separate model update and rollout evaluation at each position. In Section 4, we derive a first-order approximation to this utility, providing a practical signal for reweighting teacher feedback during training.

4 ALIGN-OPD: LEARNING FROM OUTCOME-ALIGNED FEEDBACK

The intervention study identifies teacher feedback worth suppressing, but direct estimation of Δ_t for every training token would be prohibitively expensive. We therefore seek a first-order approximation to the causal utility defined in Section 3, which relates the predicted accuracy change to the alignment between a token's OPD update and the gradient of the student's success probability. We derive and validate this approximation below, then use the resulting scores to reweight teacher feedback during training.

4.1 ESTIMATING UTILITY THROUGH OUTCOME-GRADIENT ALIGNMENT

Section 3 measures utility as a change in V_x. Its gradient, $g_x = \nabla_\theta V_x(\bar{\theta})$, describes how small changes to the student's parameters affect its probability of solving x. For the token update $\delta\theta_t = \eta d_t^{\mathrm{OPD}}$, a first-order expansion gives

$$\Delta_t = \langle g_x, \delta\theta_t \rangle + \mathcal{O}\big(\|\delta\theta_t\|_2^2\big)\,. \tag{8}$$

The inner product predicts the effect of the OPD update on the student: its first-order contribution to accuracy is positive when aligned with g_x and negative when opposed to it. This approximates the same causal effect measured by the interventions.

We estimate g_x from outcomes of complete responses. The policy-gradient identity expresses it as

$$g_x = \mathbb{E}_{y\sim p_{\bar{\theta}}(\cdot|x)}\left[(R(x,y) - b_x)\nabla_\theta \log p_\theta(y \mid x)\big|_{\theta=\bar{\theta}}\right], \tag{9}$$

where b_x is an action-independent baseline (Williams, 1992). Successful and unsuccessful responses can therefore be used to estimate a direction for improving accuracy, using relative rewards as in GRPO (Shao et al., 2024). For this estimate, we generate M additional responses $\{\widetilde{y}_i\}_{i=1}^M$ to x, called the *probe batch*. They are sampled independently of the response y whose token updates we assess. We compute

$$\widehat{g}_x = \frac{1}{M}\sum_{i=1}^{M}(R_i - \overline{R}_x)\nabla_\theta \log p_\theta(\widetilde{y}_i \mid x)\big|_{\theta=\bar{\theta}}, \tag{10}$$

where R_i is the binary outcome of probe response $\widetilde{y}_i$, and $\overline{R}_x$ is the mean outcome across probes.

The norm of $\widehat{g}_x$ can vary substantially across prompts. Using its raw inner products as scores would carry this scale variation into training, making it less straightforward to map scores to token weights. For convenience, we therefore normalize the probe gradient before comparing it with the token updates. Let $v_x = \widehat{g}_x/\|\widehat{g}_x\|_2$, we define

$$s_t = \langle v_x, d_t^{\mathrm{OPD}} \rangle. \tag{11}$$

This removes the probe gradient's magnitude while retaining the magnitude of each token's OPD contribution. The corresponding first-order estimate of the causal utility can then be expressed as $\widehat{\Delta}_t^{(1)} = \eta\|\widehat{g}_x\|_2 s_t$. The positive factor $\eta\|\widehat{g}_x\|_2$ is shared within a response, so the score preserves the predicted effect's sign and ranking across positions.

We test whether these rankings capture the effects at different training checkpoints using the interventions described in Section 3. At each checkpoint, we compute scores s with direct token gradients and dedicated probe batches of 1,024 responses. For each response, we evaluate 30 positions with the largest residual u_t, excluding those with negligible updates. We then compute the Pearson correlation between scores and measured causal utility across positions in each response. Figure 4 shows high correlations at three training checkpoints. This motivates us to use s_t to predict the relative usefulness of a token feedback before applying the update d_t^{OPD}.

4.2 SCORING AND REWEIGHTING FEEDBACK DURING TRAINING

Naively computing the scores during training introduces two non-negligible costs: *(1) generating a separate probe batch* and *(2) computing a parameter gradient for every token*.

We address the first by reusing the responses already generated for distillation. For each prompt, we divide its N responses into two even folds, F_1 and F_2. To score a response in F_k, we use

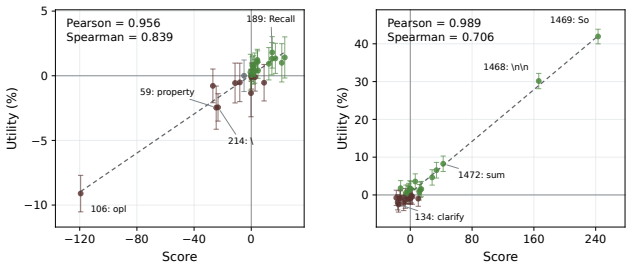


Figure 4: **Outcome-alignment scores track measured causal utility.** Left: the initial student; right: checkpoint 10, evaluated on a different prompt. Each panel shows the 30 positions with largest absolute raw teacher–student residual in one response. Scores use 1,024 independent probe responses; utility is estimated from 4,096 paired rollouts per token update. Error bars show pointwise 95% Tango confidence intervals, and dashed lines are linear fits. Labels identify selected token positions and strings.

Equation (10) on the other fold, F_{3-k}, to obtain $\widehat{g}_{3-k}$. Both folds serve as training targets, each scored using the other's outcomes. This requires computing only two probe gradients per prompt, rather than N separate leave-one-out gradients for every response.

For a response $y_j \in F_k$, its training score estimates

$$\widehat{s}_{j,t} \;\approx\; \left\langle \frac{\widehat{g}_{3-k}}{\|\widehat{g}_{3-k}\|_2}, d_{j,t}^{\mathrm{OPD}} \right\rangle, \qquad k \in \{1,2\}. \tag{12}$$

The remaining cost is the token-level gradients inside $d_{j,t}^{\mathrm{OPD}} = u_{j,t}\nabla_\theta \ell_{j,t}(\bar{\theta})$. We only need their projections onto the probe direction, which is shared by all target tokens in a fold. We use finite differences of token log-probabilities along this direction to estimate these projections together, avoiding separate token-gradient computations. Appendix A provides more details on our implementation of finite differences.

We convert the scores into weights that reduce the contribution of negatively aligned feedback. Let $\overline{R}_{3-k}$ be the mean outcome in the probe fold. For a token in F_k, its weight is

$$w_{j,t} = \begin{cases} \sigma(\alpha(\widehat{s}_{j,t} + \tau)), & 0 < \overline{R}_{3-k} < 1 \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where σ is the logistic sigmoid, $\alpha > 0$ controls its slope, and τ sets its transition. The weight increases with alignment: negative and near-zero feedback is suppressed, while positive feedback retains most of its contribution. The zero-weight case happens when we lack information needed to estimate utility. Specifically, when all probe responses are correct or all are incorrect, their centered rewards vanish, so they provide no outcome direction. We drop the corresponding target fold. All-correct probes also indicate that the student already solves the prompt consistently in the observed attempts, making further training on it less useful, while all-incorrect probes can be seen as an indication of difficult prompts on which the teacher's guidance may be unreliable.

We then optimize the OPD objective with these weights:

$$\mathcal{L}_{\mathrm{ALIGN}}(\theta) = -\frac{1}{Z}\sum_{j,t} \mathrm{sg}(w_{j,t}u_{j,t})\, \ell_{j,t}(\theta). \tag{14}$$

The sum runs over valid response tokens, and Z counts the total number of responses in retained folds. The stop-gradient operator sg treats weights and teacher residuals as constants, so gradients are taken only through the student's log-probabilities $\ell_{j,t}(\theta)$. Each teacher signal is thus weighted by its predicted utility when training the student.

To fall back less frequently, we screen prompts before training. We use the teacher for rollouts given training prompts, and only retain the prompts that are not all-correct or all-incorrect. The choice of the number of responses per prompt also affects the fall-back rate. A larger N trades prompt diversity for less fall-backs . While Lu & Lab (2025); Hou et al. (2026); Fu et al. (2026) have demonstrated the effectiveness of On-Policy Distillation with few prompts, we examined how N affect accuracy and the frequency of dropped folds and found that its effect on final performance is insignificant.

Table 1: **Results across instruction following, code generation, and math on SmolLM3.** Bold and underline mark the best and second-best training variants.

Method	IF (avg@4)			Code (avg@8)			Math (avg@16)			Total
	IFEval	IFBench	Avg.	LCBv5	LCBv6	Avg.	AIME24	AIME25	Avg.	Avg.
Reference models										
Student (MixSFT)	66.5	17.9	42.2	10.8	14.6	12.7	14.6	16.3	15.4	23.4
RL teacher	72.9	21.9	47.4	17.4	17.9	17.6	22.7	25.4	24.1	29.7
Teacher-feedback only										
Vanilla OPD	75.3	23.1	49.2	14.2	18.6	16.4	22.7	23.3	23.0	29.5
+ residual clipping	75.4	22.3	48.9	**16.2**	<u>18.3</u>	**17.3**	22.3	23.3	22.8	29.6
Teacher-feedback + outcome signal										
OPD + GRPO	74.7	23.8	49.2	14.3	16.3	15.3	<u>23.1</u>	25.0	<u>24.1</u>	29.5
SG-OPD	74.4	<u>24.6</u>	<u>49.5</u>	<u>15.4</u>	17.5	16.5	20.8	<u>26.7</u>	23.8	29.9
TIP	<u>75.7</u>	22.8	49.2	14.2	17.9	16.0	21.9	25.8	23.9	29.7
ALIGN-OPD	**76.5**	**25.9**	**51.2**	14.7	**18.8**	<u>16.7</u>	**24.6**	**27.1**	**25.8**	**31.3**

Table 2: **Results across code generation and math for Qwen3.**

Method	Code (Avg@4)				Math (Avg@32)				
	HumanEval+	MBPP+	LCB	Avg.	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.
Student	74.7	64.7	17.9	52.4	21.5	21.9	10.0	8.0	15.4
Teacher	87.2	69.0	26.7	61.0	60.8	54.1	32.4	38.3	46.4
Vanilla OPD	<u>87.0</u>	66.9	25.6	59.9	59.5	55.4	31.5	<u>37.6</u>	46.0
OPD + clipping	85.5	**68.5**	26.0	60.0	**61.2**	**57.3**	<u>32.7</u>	36.5	<u>46.9</u>
SG-OPD	86.8	<u>67.1</u>	<u>26.9</u>	<u>60.3</u>	59.1	55.2	30.3	37.2	45.5
ALIGN-OPD	**87.7**	<u>67.1</u>	**27.7**	**60.8**	<u>59.5</u>	<u>56.6</u>	**34.2**	**38.9**	**47.3**

5 EXPERIMENTS

In this section, we assess the effectiveness of ALIGN-OPD against five baselines across instruction following (IF), math, and code generation. Within each domain, all methods use matched data, model initialization, and optimization settings. We briefly describe the experiment setups below and defer the full details to Appendix C.

Models and training data. We conduct experiments using the `SmolLM3-3B` model (Bakouch et al., 2025) and the `Qwen3` model (Yang et al., 2025). For SmolLM3, we use the `Open-MOPD-SmolLM3-3B-MixSFT` checkpoint as student, and the three RL-trained checkpoints corresponding to `IF`, `Math`, and `Code` domains, released by Gao et al. (2026). For Qwen3, we use `Qwen3-4B-Non-Thinking` as student and the two RL-trained `Math` and `Code` teachers released by Yang et al. (2026b). We use each teacher's corresponding domain-specific RL prompts as training data. Within each domain, the prompts and their appearance ordering are set to the same.

Training and Evaluation. We keep the same training configurations within each pair of domain and model, including learning rate, number of rollouts, maximum update steps. We compare ALIGN-OPD with various methods such as residual clipping (Xing et al., 2026), TIP (Xu et al., 2026b), SG-OPD (Xu et al., 2026a), and the combinition of OPD and GRPO Shao et al. (2024); Xiao et al. (2026). We evaluate instruction following on IFEval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2025) using strict prompt-level accuracy; Math on AIME24 (aim, 2024), AIME25 (Zhang & Math-AI, 2025), and HMMT25 Feb & Nov (Balunovic et al., 2026); and Code on LiveCodeBench v5 and v6 (liv, 2024). We report avg@4 for IF, avg@16 for Math, and avg@8 for Code.

5.1 MAIN RESULTS

The results in Table 1 and 2 show that ALIGN-OPD improves over vanilla OPD across all evaluated domains for both SmolLM3 and Qwen3. The benefits are most pronounced on instruction following and math, while less significant on code. Residual clipping can help, but does not consistently

deliver the same gains, suggesting that deciding which teacher updates to emphasize matters beyond controlling feedback magnitude.

The comparisons with GRPO+OPD, SG-OPD, and TIP further suggest that these gains depend on how outcome information guides learning. ALIGN-OPD uses outcomes to assess the predicted effect of each teacher-induced update, providing a direct basis for reweighting token-level feedback. Its ability to surpass teachers illustrates the broader opportunity: students can learn from a strong teacher while using task outcomes to improve beyond the behavior that teacher provides.

5.2 Ablations

We perform three ablations on IF to test whether ALIGN-OPD's gains arise from the direction of its outcome-alignment signal, rather than from rollout allocation or prompt curation. All results are reported at step 30, matching the IF checkpoint in Table 1.

Reversing outcome alignment reverses the benefit. To test whether ALIGN-OPD benefits from the alignment signal itself rather than from non-uniform token weighting, we reverse the weighting rule from $w_t = \sigma(3(s_t + 1))$ to $w_t^{\text{rev}} = 1 - w_t = \sigma(-3(s_t + 1))$, while keeping the scores and remaining training procedure unchanged. As shown in Table 3, reversing which token updates are emphasized substantially degrades performance, indicating that the direction encoded by the outcome-alignment score matters.

Table 3: **IF ablations at step 30.** Δ is relative to the corresponding reference within each block.

Method	IFEval	IFBench	Avg.	Δ
Alignment direction				
ALIGN-OPD	**76.52**	**25.92**	**51.22**	–
Reverse reweighting	72.46	22.17	47.31	-3.91
Rollout and prompt design				
Vanilla OPD, $N = 64$	75.28	23.08	49.18	–
$N = 32$	73.94	23.92	48.93	-0.25
$N = 1$	74.12	22.67	48.39	-0.79
$N = 32$ + screen/order	73.01	21.92	47.46	-1.47

Effect of the number of rollouts per prompt. Because ALIGN-OPD requires multiple responses per prompt to estimate a cross-fitted outcome direction, we test whether this rollout allocation itself affects vanilla OPD. At a fixed total response budget, varying N produces only small differences and does not improve the baseline as N decreases. Thus, the rollout structure required by ALIGN-OPD does not explain its gain.

Effect of difficulty-aware prompt selection and ordering. We also consider whether explicitly selecting intermediate-difficulty prompts and presenting them in an easy-to-hard order can improve vanilla OPD. This mechanism reduces performance in our setting, so we do not use difficulty screening or curriculum ordering in the main experiments. We report it here as evidence that prompt curation alone does not account for the observed gains; implementation details are given in Appendix E.

6 Conclusion and Limitations

Our single-token interventions show that teacher feedback can induce both beneficial and strongly harmful student updates. We derive a first-order utility score from the alignment between token-level OPD updates and an outcome policy gradient, and use it to reweight teacher feedback in ALIGN-OPD. Experiments with SmolLM and Qwen across instruction following, mathematics, and code show gains over standard OPD and competing methods that also incorporate outcome signals, with benefits varying across benchmarks. By using outcomes to guide which teacher feedback to learn from, ALIGN-OPD can improve beyond imitation and even surpass its teacher.

Building on these results, extending ALIGN-OPD to broader settings could be a future direction. Our utility criterion currently concerns the update's effect on the same prompt. Extending this measurement to other prompts would reveal which feedback supports broader improvements in model behavior and could guide reweighting toward more transferable gains. Our evaluation also focuses on single-teacher settings with matched teacher–student architectures and tokenizers. Large-to-small and cross-family distillation would test how differences in capacity and tokenization affect feedback utility. Multi-teacher distillation offers a further opportunity to use outcome alignment to select or combine feedback from different teachers at each token.

Statement on the Use of Generative AI

In this work, we used generative AI tools to assist with the implementation and execution of experiments, including coding agents used to implement, debug, and run experimental pipelines. We also used generative AI tools to assist with drafting and editing parts of the manuscript, improving readability, formatting and typesetting, and refining the presentation and visual design of figures. All AI-assisted code, experimental outputs, text, and figures were reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including all claims, results, and artifacts produced with the aid of generative AI.

References

Aime 2024: Problems from the american invitational mathematics examination. blog, 2024. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Livecodebench. NeurIPS, 2024. URL https://livecodebench.github.io.

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

Mohammad Sadegh Akhondzadeh, Vijay Lingam, Atula Tejaswi, Chanakya Ekbote, Sujay Sanghavi, and Aleksandar Bojchevski. Reward-gated on-policy distillation. *arXiv preprint arXiv:2607.04037*, 2026.

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

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2026.

Zixuan Fu, Bingxiang He, Yuxin Zuo, Haohuan Huang, Jinqian Zhang, Ruhang Xiao, Cheng Qian, Qinyu Luo, Huan-ang Gao, Yudong Wang, et al. Rethinking on-policy distillation of large language models ii: One training example. *arXiv preprint arXiv:2609.04172*, 2026.

Huan-ang Gao, Haohan Chi, Yong Yan, Shiyuan Feng, Hanlin Wu, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Open-mopd: Diagnosing and fixing capability imbalance in multi-teacher on-policy distillation. *arXiv preprint arXiv:2608.19098*, 2026.

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

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

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, et al. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe. *arXiv preprint arXiv:2605.03677*, 2026.

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

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

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. Pmlr, 2017.

Yuying Li, Leqi Zheng, Yongzi Yu, Wenrui Zhou, Xuchang Zhong, Xing Hu, Jing Jin, Hangjie Yuan, and Tao Feng. Filter, then reweight: Rethinking optimization granularity in on-policy distillation. *arXiv preprint arXiv:2606.02684*, 2026.

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

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in neural information processing systems*, 33: 19920–19930, 2020.

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

Zikun Qu, Min Zhang, Mingze Kong, Zhiwei Shang, Yikun Ban, Shuang Qiu, and Zhongxiang Dai. Spot: Sparse probing and outcome calibration for on-policy distillation. *arXiv preprint arXiv:2608.04419*, 2026.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pp. 4334–4343. PMLR, 2018.

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

Toshiro Tango. Equivalence test and confidence interval for the difference in proportions for the paired-sample design. *Statistics in medicine*, 17(8):891–908, 1998.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*, 2024.

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

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation, 2026. URL https://arxiv.org/abs/2606.01249.

Haoran Xu, Hongyu Wang, Yifei Gao, Jiaze Li, Xiaofeng Zhang, and Xiaosong Yuan. Sg-opd: Sign-gated on-policy distillation via sign-consistency gating and phased teacher sampling, 2026a. URL https://arxiv.org/abs/2606.09304.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Tip: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026b.

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

Run Yang, Runpeng Dai, Jie Sun, Jielei Zhang, Fan Zhou, Hongtu Zhu, Peiyi Li, and Longwen Gao. Influence-directed distillation: Solving the diversity bottleneck in sampled-token on-policy distillation. *arXiv preprint arXiv:2608.29846*, 2026a.

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

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A SCORE COMPUTATION

Let g_F be Equation (10) evaluated on probe fold F. For nonzero g_F, set $v_F = g_F/\|g_F\|_2$ and $\delta = 10^{-5}\|\bar{\theta}\|_2$. Directional finite differences estimate the target-token derivatives as

$$D_{a,t} = \frac{\ell_t(\bar{\theta} + av_F) - \ell_t(\bar{\theta} - av_F)}{2a}, \qquad \widehat{s}_t = u_t \frac{4D_{\delta,t} - D_{2\delta,t}}{3}. \tag{15}$$

Each active fold uses one probe gradient and four forward evaluations shared across all target positions. Perturbations start from the same master-parameter snapshot, which is restored before training. Forwards use BF16; Richardson extrapolation removes the leading central-difference error but not finite-precision error. The source package's implementation record provides the parameter-restoration and numerical-diagnostic details.

B INTERVENTION MEASUREMENT AND SCORE VALIDATION

Single-token interventions. Each intervention starts from the same student checkpoint $\bar{\theta}$ and applies one SGD step using only the OPD contribution from a single token position,

$$\theta_t^+ = \bar{\theta} + \eta d_t^{\mathrm{OPD}},$$

where we use a fixed intervention learning rate of $\eta = 10^{-6}$ for SmolLM3 and $\eta = 5 \times 10^{-6}$ for Qwen3. We use the unclipped teacher–student log-ratio u_t when constructing d_t^{OPD}; no other token contributes to the update. All interventions at a given checkpoint therefore differ only in the selected token position. All intervention updates are applied in FP32.

For the complete-response examples in Figure 2, we intervene on every valid token position in each displayed response. For each intervention, we generate 4,096 paired answers from the original and updated students. Each pair uses the same random seed and decoding settings, and both students generate a fresh complete response from the original prompt.

Let n_{01} denote the number of pairs for which only the updated student answers correctly and n_{10} the number for which only the original student answers correctly. The empirical causal utility is

$$\widehat{\Delta}_t = \frac{n_{01} - n_{10}}{K_{\mathrm{eval}}}, \qquad K_{\mathrm{eval}} = 4096. \tag{16}$$

We compute 95% confidence intervals using Tango's score method for matched proportions (Tango, 1998). Evaluation generations are independent of any probe generations used for score estimation. Baseline generations from the original student may be shared across token interventions on the same prompt.

Across-checkpoint intervention study. For Figure 3, we randomly select 10 prompts once and reuse the same prompts at each of six training checkpoints. At every checkpoint, we sample one new student response for each prompt. Within each response, we select the 10 token positions with the largest absolute teacher–student residual $|u_t|$, excluding positions whose induced parameter update is numerically negligible, and measure $\widehat{\Delta}_t$ using the same intervention procedure.

The resulting study therefore contains the same number of prompts and measured token positions at every checkpoint. Figure 3 summarizes the distribution of these measured utilities without reweighting prompts by response length. Because the study intentionally focuses on high-divergence positions, it characterizes whether large teacher–student disagreement predicts downstream utility rather than the unconditional distribution over all tokens.

Score validation. To validate the alignment score, we compare the predicted token scores with measured causal utilities from independently evaluated interventions. We compute Spearman correlation within each response and then aggregate correlations across prompts. Responses with constant predicted scores or constant measured effects yield undefined rank correlations and are excluded from the correlation average, with their counts reported separately. Prompt-level bootstrap intervals resample prompts together with all associated responses and token measurements.

Probe estimator. The diagnostic score uses probe responses sampled independently of the target response. With M probe responses, the group-mean reward baseline scales the expected policy gradient by $(M-1)/M$. A leave-one-out baseline multiplies this estimator by $M/(M-1)$, changing its magnitude but preserving its normalized nonzero direction. For the diagnostic validation we use dedicated probes and direct token gradients; training instead uses the two-fold cross-fitting procedure described in Section 4.2.

C TRAINING CONFIGURATION

Folds and weights. Each prompt supplies 32 responses. We sort response-token sequences by SHA256 hash, break ties by worker rank and sample index, and alternate entries into two folds of 16. Each fold scores the other. The probe uses summed log-probabilities and the fold-mean reward baseline; residuals and scores receive no clipping or further normalization. The detached weight is

$$w_t = \sigma(3(\widehat{s}_t + 1)), \qquad \sigma(a) = (1 + e^{-a})^{-1}. \tag{17}$$

A homogeneous probe assigns zero weights to the opposite fold. All original valid tokens remain in the reduction denominator. An entirely inactive distributed mini-batch skips the optimizer step, including momentum and weight decay.

Training objective. The local weighted OPD objective is

$$\mathcal{L}_{\text{local}}(\theta) = -\frac{1}{Z} \sum_t a_t \, \text{sg}(w_t u_t) \, \ell_t(\theta). \tag{18}$$

Here a_t marks valid tokens and $Z = \sum_t a_t$ counts all originally valid tokens. We hold weights and residuals fixed during the update. The production actor uses detached advantages $A = wu$ and the ratio $\rho = p_\theta(y_t \mid z_t)/p_{\bar{\theta}}(y_t \mid z_t)$ in the clipped surrogate

$$\begin{aligned} b(\rho, A) &= \min\{\rho A, \text{clip}(\rho, 0.8, 1.2)A\}, \\ \mathcal{L}_{\text{token}} &= -\begin{cases} \max\{b(\rho, A), 3A\}, & A < 0, \\ b(\rho, A), & A \geq 0. \end{cases} \end{aligned} \tag{19}$$

The framework additionally retains token-level rollout importance correction with threshold 5 and its original token-mean reduction. Before this correction, the local gradient at $\theta = \bar{\theta}$ agrees with Equation (18); the subsequent AdamW update is not an SGD intervention. No separate reward-gradient objective is added.

Models and data. The student is `Qwen/Qwen3-4B`; the frozen teacher is `Keven16/Qwen3-4B-Non-Thinking-RL-Math-Step500`. We use the math portion of `Keven16/G-OPD-Training-Data`, with non-thinking chat templates and Math-Verify outcomes binarized at reward greater than zero. Teacher screening retains estimated accuracy in $[1/8, 7/8]$. Using seed 27115, we sample 640 prompts from the 800-prompt pool without replacement, sort by decreasing estimated teacher accuracy, and traverse 20 fixed batches of 32 prompts once.

SmolLM experiments. The experiments in Section 5 use the released `Open-MOPD-SmolLM3-3B-MixSFT` checkpoint as the student and the corresponding domain-specific OpenMOPD RL checkpoints as teachers for instruction following (IF), Math, and Code. Teacher and student share the same architecture, tokenizer, and pre-RL initialization, with the teacher differing through domain-specific RL training.

Each training step samples prompts from the corresponding OpenMOPD training split and generates responses from the current student. We use 16 prompts $\times$ 64 responses for IF, 32 prompts $\times$ 32 responses for Math, and 24 prompts $\times$ 64 responses for Code. Thus, ALIGN-OPD uses probe folds of 32, 16, and 32 responses per prompt, respectively. All methods within a domain use the same prompt order and rollout allocation. Training uses a constant learning rate of 3×10^{-6}, $\lambda = 1$, `opd_top_k=0`, one optimizer step per rollout batch, and data seed 42. Runs are trained for 50 steps on IF, 20 steps on Math, and 5 steps on Code; the main results report steps 30, 20, and 5, respectively. For Qwen experiments, we use a constant learning rate of 1×10^{-5}, and a total training step of 20.

SmolLM main-experiment evaluation. For the SmolLM experiments in Section 5, we evaluate instruction following on IFEval and IFBench using strict prompt-level accuracy with four generations per prompt. Code is evaluated on LiveCodeBench v5 and v6 with eight generations per problem, and Math on AIME24 and AIME25 with sixteen generations per problem. For each domain, we report the mean of its two benchmark scores as the domain-level Avg. used in Table 1; the Total score is the mean of the IF, Code, and Math domain averages.

The main table reports step 30 for IF, step 5 for Code, and step 20 for Math. Evaluation uses the corresponding domain-specific OpenMOPD decoding protocol, with identical settings across training variants within each domain.

Optimization and generation. Each active step uses 32 responses per prompt, one PPO epoch, and one AdamW update. The learning rate is 10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay is 0.01, and there is no warmup. Runs use eight H200 GPUs and save checkpoints at steps 10 and 20. Generation uses temperature one, top-p one, BF16, a 16,384-token response cap, and a 2,048-token prompt cap. Distillation uses teacher probabilities on sampled student tokens, with no vocabulary top-k objective.

D ROLLOUT-ALLOCATION ABLATION DETAILS.

For the IF rollout ablation in Table 3, we fix the total budget at 1,024 responses per step and vary responses per prompt as $N \in \{1, 32, 64\}$, corresponding to 1,024, 32, and 16 prompts per step. All other settings match the main IF vanilla-OPD run, including the teacher, optimizer, learning rate, generation settings, and data seed. The main paper reports step 30.

E DIFFICULTY SCREENING AND PROMPT ORDERING

We additionally considered whether explicitly selecting prompts by estimated difficulty and presenting them in an easy-to-hard order could improve dense distillation. This mechanism is not used in the final main experiments; we report it here because it was evaluated as an alternative training design.

For IF, we sample 5,000 candidate prompts and generate $K = 8$ responses per prompt from the RL-IF teacher using the training generation settings. Each response is scored using the strict `all_constraints_passed` outcome. We retain prompts for which the teacher succeeds between one and seven times out of eight, removing prompts that are always failed or always solved under the screening samples. This retains 2,413 prompts, or 48.3% of the candidate pool.

The retained prompts are then ordered from easier to harder according to the teacher's empirical success rate, with higher-success prompts appearing first, and training is run with shuffling disabled. We train vanilla OPD with $N = 32$ while keeping the remaining IF configuration unchanged. At step 30, the run has therefore consumed $30 \times 32 = 960$ prompts from the ordered pool, corresponding to the easier portion of the retained set.

The screened-and-ordered condition reaches an IF average of 47.46, compared with 48.93 for the corresponding unscreened vanilla-OPD run with $N = 32$, a decrease of 1.47 points. We therefore do not use difficulty screening or easy-to-hard ordering in the main experiments. Since this condition changes both the selected prompt distribution and its ordering, the experiment evaluates the combined mechanism rather than isolating the effect of either component individually.