When Selective On-Policy Dstillation Rules Conflict: From Supervision Collapse to Minimal Repair

Anonymous authors
Paper under double-blind review

Abstract

On-policy distillation (OPD) uses teacher feedback to train a student on responses generated by its current policy. Existing methods select which tokens contribute to the training loss, using criteria such as uncertainty or teacher-student disagreement. However, these methods can select different positions, leaving too little shared feedback when every retained token must satisfy two rules. We find that strict combination of rules can eliminate all teacher feedback, yielding a zero joint gradient even when each rule alone produces a nonzero gradient on the same response, i.e., phenomenon supervision. To this end, we introduce SupportGuard. It preserves the supervision retained by strict combination while recovering training signal when that supervision is insufficient. SupportGuard measures the retained supervision using the mean loss weight over the full response and compares it with a fixed target effective update ratio. We conduct experiments on two groups of tasks, mathematical reasoning and code generation, based on Qwen and Gemma. The experimental results showing consistent improvements of SupportGuard over strict combination. SupportGuard achieves the highest performacne on seven Qwen benchmarks and all four Granite benchmarks up to 1.31 and 2.04 pp.

1 Introduction

Knowledge distillation (KD) transfers knowledge from a teacher model to a student by training the student to match teacher predictions (Hinton et al., 2015). Conventional KD uses reference or teacher-generated responses for training. These responses may not cover the mistakes the student makes when generating its own responses. On-policy distillation (OPD) addresses this gap by training on student-generated responses. The teacher provides token-level feedback on these responses, allowing the student to learn from its own mistakes (Lin et al., 2020a; Agarwal et al., 2024a). However, teacher feedback is not equally useful at every position. Some student predictions already match the teacher closely, while others still differ from the teacher and may provide more useful learning signals. Vanilla OPD gives both types of positions the same loss weight.

This motivates selective OPD methods that reweight token losses to focus training on more useful teacher feedback. Recently, some selective OPD use uncertainty in student or teacher predictions to guide supervision (Kim & Baek, 2026a; Jin et al., 2026a). Other methods assess the reliability of teacher feedback and whether the student can learn from it (Huang et al., 2026c; Wang et al., 2026). The third group focuses on early response positions or relevance to the task input (Zhang et al., 2026; Li et al., 2026a). Although these approaches have improved distillation performance, a single criterion may not fully capture the value of teacher feedback for the student. For example, the teacher may be equally confident at two positions. The student may already match the teacher distribution at one position but remain uncertain at the other. This motivates combining selection rules that use different information. Using teacher confidence and student uncertainty together can help select more useful feedback.

A naive approach is strict combination, which retains a loss only when both rules select the same response position. We find that strict combination can eliminate the entire distillation signal even when each rule alone produces a nonzero gradient on the same response. Each rule retains feedback at its selected positions, but the two selections may share few or no positions. When the overlap

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.

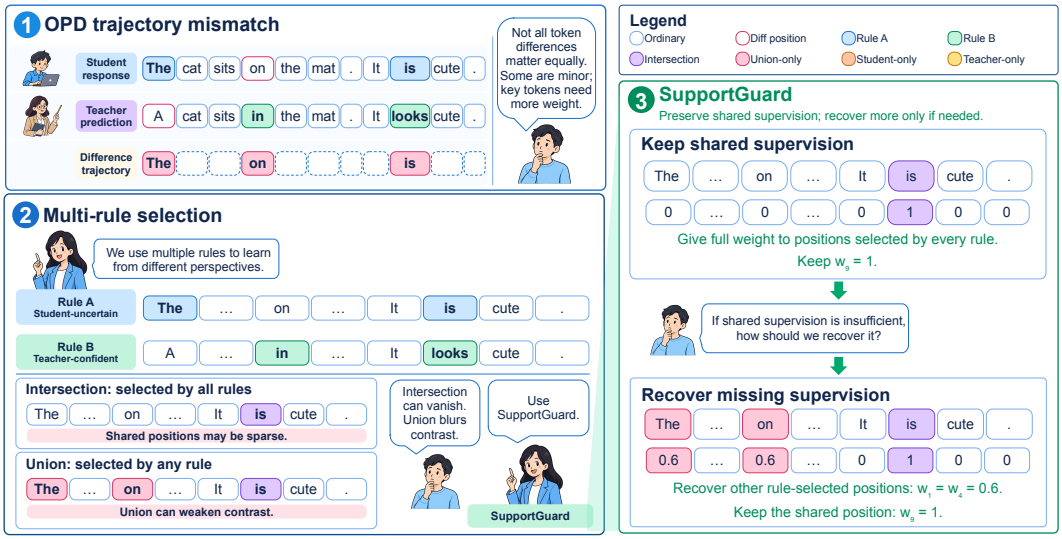


Figure 1: The motivation of SupportGuard. Different token-level selection rules can retain different supervision signals. Strict combination may leave too little shared supervision, whereas the union may weaken token-level contrast. SupportGuard preserves shared supervision and recovers additional supervision only when needed.

is empty, strict combination excludes every token loss and the joint distillation gradient becomes zero. We name this issue supervision collapse. Under our training protocol, steps with zero total loss weight skip the optimizer update and leave the student parameters unchanged. We observe cases where this failure persists throughout training, resulting in the final model identical to its initialization. To this end, SupportGuard preserves jointly selected losses and restores feedback from positions selected by only one rule when needed. Shared losses retain their full weight, while additional losses receive a common weight determined by how much feedback needs to be restored.

To address supervision collapse issue, we introduce SupportGuard. SupportGuard preserves the supervision retained by strict combination. It then measures the retained supervision using the mean loss weight over the full response and compares it with a fixed target effective update ratio. If the retained supervision is sufficient, the current loss weights remain unchanged; otherwise, SupportGuard assigns a shared weight to the remaining rule-selected positions to restore the missing supervision needed to reach the target.

Our contributions are threefold. **1)** We identify the issue of supervision collapse in selective OPD. The supervision collapse a consequential failure mode: strict combination often retains little to no shared supervision, and can eliminate the joint distillation signal entirely. **2)** We introduce SupportGuard. which preserves jointly selected losses at full weight and assigns a common weight to losses selected by only one rule. It adds the weight needed to reach a fixed mean loss weight target when feasible, while leaving combinations that already meet the target unchanged. **3)** We conduct experiments across mathematical reasoning and code generation to show the effciveness of our SupportGuard.

2 Supervision Collapse under Strict Combination

Selective OPD applies supervision only to selected response tokens. When multiple selective criteria are combined, a natural choice is to keep only the tokens selected by all of them, since those tokens are the ones that different criteria all regard as important. We call this strategy *strict combination*. However, in practice, different criteria may share very few tokens, or almost none at all. We call this phenomenon *supervision collapse*. An illustrative real case is given in Figure 2.

Consider a real example from a circular-seating problem in MATH benchmark. It combines an EOPD-derived criterion and a SelecTKD-derived criterion. Their union contains 53 tokens, but strict combination keeps only one. This is a typical case in which two active criteria select almost no tokens

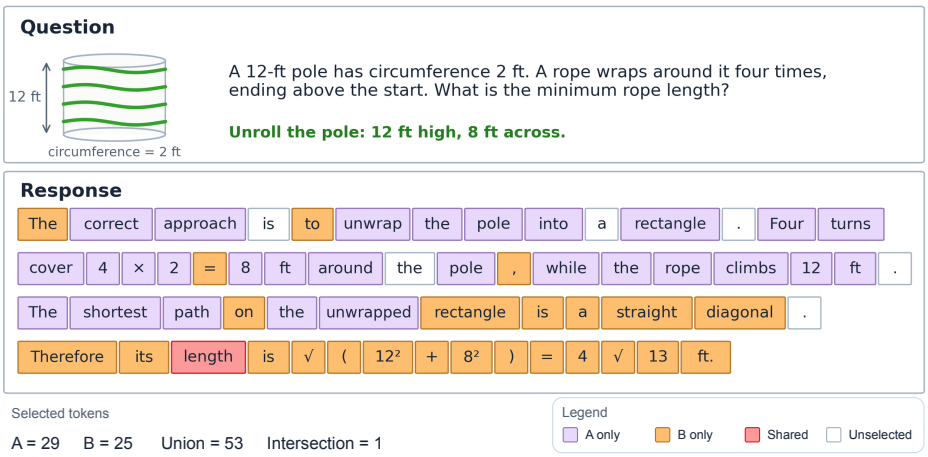


Figure 2: A real case of supervision collapse.

in common. Each criterion alone retains a meaningful set of tokens, yet their overlap is nearly empty. Therefore, this issue is not that one criterion fails to select anything, but that the two criteria focus on different parts of the identical response.

Next, we reoprt the statistics on Qwen-MATH benchmark. The same pattern appears repeatedly in the full Qwen-MATH analysis. There are 280 responses on which both criteria retain at least one token. Among them, 133 retain no shared token after strict combination, and another 109 retain only one shared token. In total, 242 of the 280 responses retain at most one shared token. In other words, strict combination often keeps little to no shared supervision.

Furthermore, we examine whether the 133 empty intersections occur simply because the two criteria select few tokens. For each response, we keep the number of tokens selected by each criterion unchanged and randomly change where those tokens are selected. Under this randomized control, only about 3.1 responses are expected to have no overlap, compared with 133 in the actual selections. Selecting few tokens can make an empty intersection possible, but it does not explain why the two criteria fail to overlap so frequently.

This phenomenon matters because strict combination defines the joint supervision signal through the shared selected tokens. When no shared token remains, the response contributes no joint distillation signal. More broadly, even when strict combination does not become completely empty, the retained supervision is often extremely weak. Therefore, in our analysis strict combination frequently keeps either no common supervision or only a very small amount. These observations motivate the repair strategy in the next section.

3 METHOD

3.1 PRELIMINARIES

On-Policy Distillation (OPD). Conventional off-policy distillation does not directly provide supervision on student-generated prefixes. OPD addresses this issue by first letting the student generate a trajectory and then asking the teacher to provide supervision on the same student-generated prefixes. Given an input x, the student policy π_θ generates a sequence of N tokens as follows,

$$\mathbf{y} = (y_1, \ldots, y_N) \sim \pi_\theta(\cdot \mid x), \tag{1}$$

where N is the number of tokens in student response. The prefix used to predict token t is given as:

$$h_t = (x, y_{<t}). \tag{2}$$

The student policy π_θ and the teacher policy π_ϕ both produce next-token distributions conditioned on the same h_t. The OPD loss at position t is defined as follows,

$$\ell_t^{\mathrm{OPD}} = D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h_t) \| \pi_\phi(\cdot \mid h_t)\right). \tag{3}$$

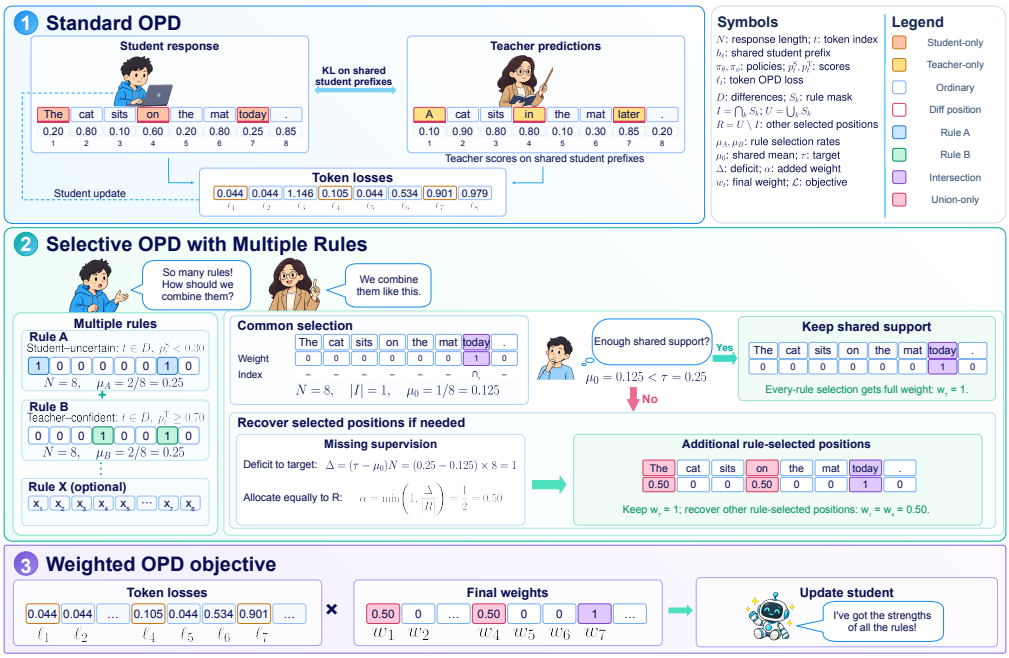


Figure 3: The overview of our proposed SupportGuard.

Therefore, the teacher can provide supervision directly on prefixes generated by the student, rather than only on teacher-generated trajectories.

Selective OPD. Standard OPD applies supervision in the same way across token positions, although teacher feedback is not equally useful at every token position. Let D denote the set of token positions where the student and teacher predictions differ, and let $|D|$ denote the number of such positions. For a token-level rule r, we use a binary indicator r_t to denote whether token t is selected for the update:

$$r_t \in \{0, 1\}, \tag{4}$$

where $r_t = 1$ indicates that token t is selected. A basic Selective OPD objective can be written as:

$$\mathcal{L}_{\text{SOPD}} = \frac{1}{N} \sum_{t=1}^{N} r_t \, \ell_t^{\text{OPD}}. \tag{5}$$

Different token-level criteria can evaluate these prediction differences from different perspectives. For example, student uncertainty and teacher confidence provide signals from the student and teacher sides, respectively. However, a single rule can use only one criterion. We therefore study multi-rule selective supervision and introduce SupportGuard to combine the selection signals of different rules.

3.2 Multi-Rule Selection and Supervision Collapse

We consider K token-level rules, each selecting tokens according to a different criterion. Let D denote the set of token positions where the student and teacher predictions differ. Each rule selects tokens from D. For rule k, we define its binary selection indicator and selected-position set as:

$$r_t^{(k)} \in \{0, 1\}, \qquad \mathcal{S}_k = \{t \in D : r_t^{(k)} = 1\}, \tag{6}$$

where $r_t^{(k)} = 1$ means that token t is selected by rule k, and $\mathcal{S}_k$ contains the token positions selected by rule k from D.

For multiple rules, we define the positions selected by all rules and by at least one rule as follow,

$$I = \bigcap_{k=1}^{K} \mathcal{S}_k, \qquad U = \bigcup_{k=1}^{K} \mathcal{S}_k. \tag{7}$$

A direct choice is to keep only the positions in I. Because different criteria may select different positions, a small overlap can result in fewer tokens contributing to the update. In the extreme case where $I = \varnothing$, the joint gradient becomes zero. Another choice is to keep all positions in U, which retains more rule-specific selections than the intersection. However, when U contains many tokens, many positions contribute to the update at the same time. This weakens the token-level contrast that Selective OPD is intended to preserve.

Therefore, we propose SupportGuard to preserve the supervision retained by strict combination and restore missing supervision when it is insufficient.

3.3 SUPPORTGUARD

We introduce SupportGuard. Basically, our SupportGuard preserve genuinely shared supervision when it exists, but it should also restore a small amount of one-sided supervision when the overlap becomes too weak. The overview is given in Figure 3.

For a student response of N tokens, let $w_t \in [0, 1]$ denote the loss weight at position t. SupportGuard keeps the full loss contribution of positions selected by all rules by assigning $w_t = 1$ for $t \in I$. The resulting mean loss weight over the full student response is given as:

$$\mu_0 = \frac{|I|}{N}. \tag{8}$$

Here, $|I|$ is the number of positions selected by all rules, and N is the number of tokens in the response. Since every token in I has full weight 1, μ_0 represents the fraction of response tokens that currently contribute to the update with full weight.

Furthermore, we use a target effective update ratio $\tau \in (0, 1]$. It is fixed before training. The ratio specifies the target total loss weight as the equivalent fraction of response tokens receiving full weight. If $\mu_0 \geq \tau$, the positions selected by all rules already provide enough effective supervision. SupportGuard keeps the current loss weights unchanged. If $\mu_0 < \tau$, the current supervision is insufficient. SupportGuard restores supervision using positions selected by at least one rule but not by all rules.

We denote the positions selected by at least one rule but not by all rules as:

$$R = U \setminus I, \tag{9}$$

where $\setminus$ denotes the minus of two sets. Let Δ denote the missing full-weight-equivalent supervision and α the shared weight assigned to each remaining rule-selected position. They are given as follows:

$$\Delta = (\tau - \mu_0)N, \qquad \alpha = \min\left(1, \frac{\Delta}{|R|}\right). \tag{10}$$

where $|R|$ is the number of such positions. Let ℓ_t^{OPD} denote the token-level OPD loss at position t, computed from the student and teacher distributions conditioned on the same student-generated prefix. Thus, applying the token weights to this loss gives the final training objective, i.e.,

$$\mathcal{L} = \frac{1}{N} \sum_{t=1}^{N} w_t \, \ell_t^{\mathrm{OPD}}. \tag{11}$$

Thus, SupportGuard combines multiple rule selections into unified token-level loss weights. This preserves the losses at positions selected by all rules while restores the supervision needed to reach the target effective update ratio.

4 EXPERIMENTAL SETUP

Models and training data. We conduct mathematical reasoning and code generation experiments with the pretrained Qwen2.5-3B and Qwen2.5-7B models (Qwen et al., 2024) as the student and teacher, respectively. In addition, we evaluate mathematical reasoning using the pretrained Gemma 3 4B and 14B models (Gemma Team, 2025). The mathematical training dataset is constructed from MATH (Hendrycks et al., 2021), and the code training dataset from MBPP (Austin et al., 2021).

Table 1: Performance of mathematical reasoning and code generation on Qwen. The MATH500 column follows the existing W21 MATH development set of 500 problems. Bracketed markers retain the source protocol or seed scope.

Method	Math				Code		
	MATH500	GSM8K	Minerva Math	Olympiad Bench	HE+	MBPP+	LiveCode Bench
Base model / initial student	43.20	77.92	23.51	26.80	37.07	50.64	8.04
Vanilla OPD	46.30	80.11	25.19	30.10	43.85	58.65	13.00
REOPOLD (StudentEntropy component)	46.35	81.55	26.27	31.30	44.89	58.99	13.67
EOPD (TeacherEntropy component)	46.60	81.97	26.36	32.30	44.97	59.02	13.67
SelecTKD (Confidence component)	46.85	81.15	26.66	31.10	44.67	60.41	14.49
TIP (TIP-style allocation control)	46.20	81.82	27.03	32.20	45.16	60.11	15.21
TA-OPD (IW component)	46.50	81.63	26.91	31.90	45.27	60.88	15.03
FiRe-OPD	47.00	81.38	26.75	32.50	45.36	60.62	15.86
SupportGuard							
REOPOLD + SelecTKD (StudentEntropy $\times$ Confidence)	47.30	82.82	27.46	33.50	46.48	60.79	16.68
EOPD + SelecTKD (TeacherEntropy $\times$ Confidence)	47.10	82.37	26.98	32.90	46.09	61.20	16.05
SelecTKD + TA-OPD (Confidence $\times$ IW)	47.55	83.11	27.24	33.70	46.67	61.31	16.54

Baselines. Vanilla OPD uses teacher supervision at every response position and serves as the reference. We also compare three approaches to selective supervision. REOPOLD (Ko et al., 2026a) uses student entropy to select tokens for training. EOPD (Jin et al., 2026b) adds a forward KL loss at positions with high teacher entropy. In addition, SelecTKD (Huang et al., 2026a) gives greater weight to tokens accepted by the teacher.

Evaluation benchmarks and metrics. We evaluate mathematical reasoning on MATH-500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024), and code generation on HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench (Jain et al., 2024). For mathematics, we report $\mathrm{Avg@8}$, averaged over eight responses for each problem. For code, we report $\mathrm{Avg@1}$, based on one generated program for each task. The results are averaged over three training seeds.

Implementation details. Each comparison uses the identical student initialization, training data and order, optimizer configuration, and training budget. Strict combination, full union, and SupportGuard optimize the same token-level reverse-KL objective and differ only in their token weights. Each configuration is trained for 250 steps and repeated over three random seeds. During training, student responses are generated online using greedy decoding with a maximum response length of 2048 tokens. The weighted loss is normalized by the number of valid response positions, and steps with zero total loss weight skip the backward pass and optimizer update. We determine the SupportGuard threshold τ in separate calibration runs based on update behavior. All main training runs use eight NVIDIA H20 GPUs.

5 Experimental Results and Analysis

5.1 Overall Performance

We report the Qwen results in Table 1. We first observe that Vanilla OPD itself provides a strong improvement over the initial student. On Qwen, it improves all seven benchmarks, with gains of 1.68-3.30 percentage points on mathematical reasoning and 4.96–8.01 points on code generation. The improvement is even larger on Granite, where Vanilla OPD raises MATH-500, GSM8K, Minerva Math, and OlympiadBench by 9.45, 14.90, 8.88, and 7.70 points, respectively. These results show that the teacher already provides useful supervision in both model families. SupportGuard is therefore evaluated on top of an effective OPD setting rather than one in which distillation itself fails.

On Qwen, SupportGuard provides consistent additional gains when multiple selection signals are combined. The best SupportGuard configuration improves over Vanilla OPD by 1.25 points on MATH-500, 3.00 on GSM8K, 2.27 on Minerva Math, and 3.60 on OlympiadBench, as well as by 2.82, 2.66, and 3.68 points on HumanEval+, MBPP+, and LiveCodeBench. More importantly, these gains remain when compared with the strongest selective baselines. The highest SupportGuard score exceeds the best non-SupportGuard point estimate on every benchmark, with margins ranging from 0.43 to 1.31 points. This pattern is not driven by a single pair of selectors: each of the three SupportGuard combinations outperforms the stronger of its two constituent methods on all seven Qwen benchmarks.

Table 2: Cross-family mathematical reasoning on Granite.

Method	MATH-500	GSM8K	Minerva Math	Olympiad Bench
Base model	25.20	44.42	10.83	4.50
Vanilla OPD	34.65	59.32	19.71	12.20
REOPOLD (StudentEntropy component)	35.40	61.84	19.62	12.40
EOPD (TeacherEntropy component)	36.20	61.06	20.12	12.70
SelecTKD (Confidence component)	36.35	61.11	19.95	13.20
TIP (Xu et al., 2026)	36.90	62.69	20.33	13.90
FiRe-OPD (Li et al., 2026)	36.80	61.21	21.12	13.70
SupportGuard				
REOPOLD + SelecTKD	38.40	63.21	22.50	14.60
EOPD + SelecTKD	37.05	62.84	22.12	14.50
SelecTKD + TA-OPD	37.50	63.42	23.16	15.30

Furthermore, Table 2 reports cross-family results on Granite. Table 2 shows that the same behavior extends to Granite. The strongest SupportGuard results reach 38.40 on MATH-500, 63.42 on GSM8K, 23.16 on Minerva Math, and 15.30 on OlympiadBench. These scores improve over Vanilla OPD by 3.75, 4.10, 3.45, and 3.10 points, respectively. They also exceed the strongest non-SupportGuard baseline on all four benchmarks, by 0.73-2.04 points. For the two selector pairs whose individual components are both reported, SupportGuard improves over the stronger constituent in all eight pair–benchmark comparisons. The composition benefit therefore reproduces in a second model family with a substantially different initial level of mathematical performance. The strongest selector pair nevertheless varies across tasks.

5.2 Ablation Study

We ablate six variants in SupportGuard to examine how each part of the repair rule affects accuracy. The six variants are briefly described as follows. **w/o Repair** retains only losses at jointly selected tokens. **w/o Nonempty Repair** repairs empty intersections but leaves nonempty, below-target intersections unchanged. **w/o Shared Priority** redistributes the same total weight uniformly over the union of the two selections. **w/o Distributed Supplement** preserves the shared-token weights and total added weight but assigns the supplement to a random subset of singly selected tokens. **w/o Minimal Repair** assigns weight one to the full union whenever repair is triggered. **w/o Strict Retention** replaces an already-sufficient strict selection with the full union.

Table 3: SupportGuard ablation results.

Variant	Accuracy (%)	Δ (pp)
SupportGuard	**54.00**	–
w/o Repair	51.60	−2.40
w/o Nonempty Repair	52.45	−1.55
w/o Shared Priority	52.00	−2.00
w/o Distributed Supplement	53.75	−0.25
w/o Minimal Repair	52.45	−1.55
w/o Strict Retention	52.60	−1.40

Table 3 report the ablation results. It uses the locked SupportGuard result of 54.00% as the baseline. **First,** we note that removing repair lowers accuracy from 54.00% to 51.60% (-2.40 pp). Repairing only empty intersections reaches 52.45% (-1.55 pp). These differences suggest that the benefit of SupportGuard is not limited to completely empty intersections: a nonempty strict combination can still retain too little supervision.

Second, we note that removing shared priority lowers accuracy to 52.00% (-2.00 pp), even though the same total weight is redistributed over the selected union. In contrast, assigning the same added

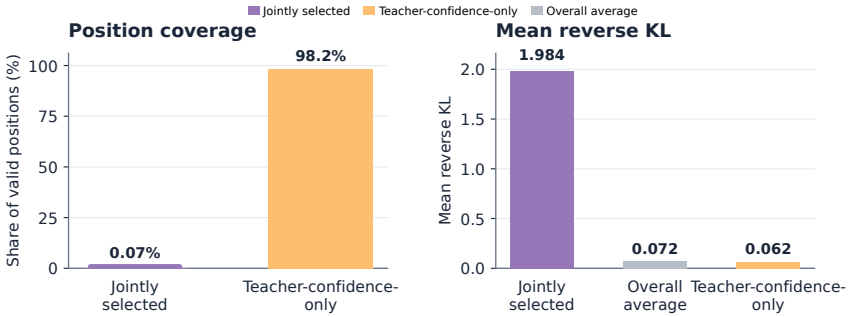


Figure 4: Jointly selected positions are rare but exhibit much larger teacher–student mismatch. They account for only 0.07% of valid positions, while their mean reverse KL is 1.984, which is 27.7× the overall average and 32× that of teacher-confidence-only positions.

weight to a sparse random subset reaches 53.75% (-0.25 pp). The results are therefore more consistent with preserving full weight on jointly selected tokens than with requiring the supplement to cover every singly selected token.

Third, we note that using the full selected union whenever repair is triggered reaches 52.45% (-1.55 pp), favoring a rule that stops once the target is reached. Removing strict retention reaches 52.60% (-1.40 pp), but this variant changes only about 0.7%–1.0% of the observed states, so the endpoint difference should be interpreted cautiously.

Fourth, across the tested variants, the largest degradations come from removing repair and shared priority, whereas sparse-random allocation remains close to SupportGuard. The descriptive pattern therefore favors repairing nonempty shortfalls, preserving full weight on jointly selected tokens, and stopping after the target is reached, but does not show that the added weight must be distributed across every singly selected token.

5.3 Furthermore Analysis

We analyze our SupportGuard by answering four issues. For the position-level analysis, we use one response generated by the initial student for each of the 500 development problems.

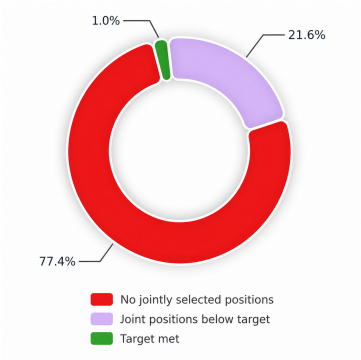


Figure 5: Distribution of supervision availability.

What supervision is recovered? Among the 500 responses, 387 (77.4%) contain no position selected by both rules, 108 retain some jointly selected positions but remain below the target, and only five meet it. In the 387 responses, every selected position is selected by only one rule, so strict combination gives them no distillation loss. Positions selected only by the teacher-confidence rule have relatively small per-position mismatch, but they cover 98.2% of valid positions and hold 84.8% of the initial total KL. SupportGuard brings this supervision back into training with small weights. As a result, SupportGuard executes all 750 scheduled updates, whereas strict combination executes only 147.

Why keep jointly selected positions at full weight? As shown in Figure 4, jointly selected positions account for only 0.07% of valid positions, yet their mean reverse KL, 1.984, is 27.7 times the overall average and 32 times that of positions selected only by the teacher-confidence rule. Positions selected by both rules are therefore rare, but their per-position teacher–student mismatch is much larger, providing a direct reason to retain their full weight.

Keeping them at full weight also preserves the direction of the update. On four diagnostic examples with at least one jointly selected position, the gradient of SupportGuard has a cosine above 0.998 with that of strict combination but only 0.146-0.559 with that of Uniform. Uniform uses the same total weight but distributes it evenly over the union. Thus, even with the same amount of supervision, different weight allocations produce substantially different update directions.

To isolate the effect of where the full weight is placed, we use two examples with exactly one jointly selected position and keep all weight values and weighted positions unchanged. We exchange only the weight of 1 at the jointly selected position with the smaller weight of a randomly chosen position selected only by the teacher-confidence rule. After the exchange, the gradient becomes almost identical to that of Uniform, with cosine similarity above 0.9998. Therefore, changing only which position receives full weight is sufficient to move the update away from the direction induced by jointly selected positions and toward Uniform. Thus, SupportGuard can add supervision while preserving the influence of positions selected by both rules.

How much supervision to add? SupportGuard stops once the target is reached, adding less than 1% of the weight that Full OR adds over strict combination. Adding more supervision does move the student closer to the teacher: at positions selected by either rule, Full OR reduces the KL 1.5-2.7 times as much as SupportGuard. However, it does not improve accuracy: Full OR reaches 53.13% against 54.00% for SupportGuard, and raising the ratio τ from 0.001 to 0.6 yields no consistent gain. Thus, beyond the target, additional supervision mainly makes the student imitate the teacher more closely without improving the performance.

How much is computational cost? Given the two rules' selections, SupportGuard counts jointly selected and union positions, computes one coefficient, and assigns one weight per position. This requires $O(N)$ work for a response of length N and no additional generation or teacher forward pass. Under the same model, input, and optimizer state, weight construction takes 0.353 ms, or 0.052% of a training step, without increasing peak memory. The additional cost relative to strict combination comes from running the updates that strict combination skips, which Full OR also requires.

6 Related Work

Selective OPD. Imitation-based distillation and Generalized Knowledge Distillation (GKD) place supervision on student-generated contexts, reducing the mismatch between training and generation (Lin et al., 2020b; Agarwal et al., 2024b). Selective distillation further shows that teacher supervision need not be applied uniformly across token positions (Wang et al., 2021). Recent OPD methods therefore use different signals to select or modify token-level feedback, including student uncertainty, teacher entropy or confidence, and teacher acceptance of student proposals (Ko et al., 2026b; Jin et al., 2026c; Huang et al., 2026b; Kim & Baek, 2026b). Because these criteria capture different aspects of the student–teacher interaction, they need not retain the same positions.

Combining selection criteria in OPD. Combining multiple distillation signals can occur at different levels. Prior work modifies the distillation objective itself through alternative divergence formulations or mixed teacher-student data (Gu et al., 2024; Ko et al., 2024; 2025; Jung et al., 2025). Other methods combine token-selection signals through soft scoring or multiplicative weighting (Xu et al., 2026; Li et al., 2026b;c). Existing safeguards also constrain excessive filtering, for example by enforcing a minimum retained count or limiting the removed distillation mass (Li et al., 2026b; Jiang et al., 2026). SupportGuard addresses a different failure mode of strict rule combination. It checks the remaining mean loss weight and relaxes only insufficient combinations within the original union.

7 Conclusion

We show that requiring agreement between different selection rules in selctive OPD can eliminate supervision. In this paper, we propose SupportGuard, which keeps jointly selected positions at full weight and adds only the weight needed when supervision is insufficient. Across mathematical reasoning and code generation evaluation, SupportGuard achieves the highest performacne on seven Qwen benchmarks and all four Granite benchmarks up to 1.31 and 2.04 pp. These results suggest that when multiple selection signals are used to improve OPD, their combination should preserve complementary supervision rather than rely solely on agreement.

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 *The Twelfth International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2306.13649.

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*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. SelecTKD: Selective token-weighted knowledge distillation for LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Huang_SelecTKD_Selective_Token-Weighted_Knowledge_Distillation_for_LLMs_CVPR_2026_paper.html.

Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. SelecTKD: Selective Token-Weighted Knowledge Distillation for LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/html/Huang_SelecTKD_Selective_Token-Weighted_Knowledge_Distillation_for_LLMs_CVPR_2026_paper.html.

Jiazhen Huang, Xiao Chen, Xiao Luo, Yong Dai, Senkang Hu, and Yuzhi Zhao. Skill-conditioned gated self-distillation for LLM reasoning, 2026c. URL https://arxiv.org/abs/2605.28791. Accepted by Findings of EMNLP 2026, as indicated in the author-updated arXiv record.

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, 2024. URL https://arxiv.org/abs/2403.07974.

Yinuo Jiang, Yongjie Ye, Zhou Tao, Xiang Zhuang, Qiang Zhang, Huajun Chen, and Tiankai Li. When Teachers Mislead: Spurious-Signal-Aware On-Policy Distillation. arXiv preprint arXiv:2608.03632v1, 2026. URL https://arxiv.org/html/2608.03632v1.

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. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2603.07079.

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:2603.07079v3, 2026b. URL https://arxiv.org/html/2603.07079v3. ICML 2026, per IBM Research (proceedings metadata not verified).

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. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026c. URL https://openreview.net/forum?id=J5i09faOOf.

Seongryong Jung, Suwan Yoon, DongGeon Kim, and Hwanhee Lee. ToDi: Token-wise Distillation via Fine-Grained Divergence Control. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8078–8091, 2025. doi: 10.18653/v1/2025.emnlp-main.409. URL https://aclanthology.org/2025.emnlp-main.409/.

Minsang Kim and Seung Jun Baek. Explain in your own words: Improving reasoning via token-selective dual knowledge distillation. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2603.13260.

Minsang Kim and Seung Jun Baek. Explain in Your Own Words: Improving Reasoning via Token-Selective Dual Knowledge Distillation. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a89331e2ecbbd9d9618bc94717a91155-Abstract-Conference.html.

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.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A Contrastive Approach Boosts the Distillation of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 31044–31062. PMLR, 2025. URL https://proceedings.mlr.press/v267/ko25a.html.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. arXiv preprint arXiv:2603.11137v1, 2026a. URL https://arxiv.org/abs/2603.11137v1.

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, 2026b. URL https://arxiv.org/abs/2603.11137.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Enhan Li, Junhao He, and Hongyang Du. CROP: Task relevance via counterfactuals for selective on-policy distillation, 2026a. URL https://arxiv.org/abs/2608.13387.

Enhan Li, Junhao He, and Hongyang Du. CROP: Task Relevance via Counterfactuals for Selective On-Policy Distillation. arXiv preprint arXiv:2608.13387v4, 2026b. URL https://arxiv.org/html/2608.13387v4.

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:2606.02684v2, 2026c. URL https://arxiv.org/html/2606.02684v2.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6121–6133. Association for Computational Linguistics, 2020a. doi: 10.18653/v1/2020.emnlp-main.494. URL https://aclanthology.org/2020.emnlp-main.494/.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive Knowledge Distillation through Imitation Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6121–6133. Association for Computational Linguistics, 2020b. doi: 10.18653/v1/2020.emnlp-main.494. URL https://aclanthology.org/2020.emnlp-main.494/.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

Qwen, An Yang, Baosong Yang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Fusheng Wang, Jianhao Yan, Fandong Meng, and Jie Zhou. Selective Knowledge Distillation for Neural Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6456–6466. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.504. URL https://aclanthology.org/2021.acl-long.504/.

Yuanyi Wang, Su Lu, Yanggan Gu, Pengkai Wang, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. Not all disagreement is learnable: Token teachability in on-policy distillation, 2026. URL https://arxiv.org/abs/2605.26844.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token Importance in On-Policy Distillation. arXiv preprint arXiv:2604.14084v4, 2026. URL https://arxiv.org/html/2604.14084v4.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D. Lyng, Sanjit Singh Batra, and Robert E. Tillman. Fast and effective on-policy distillation from reasoning prefixes. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25553–25569. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1276. URL https://aclanthology.org/2026.findings-acl.1276/.

A AI USAGE DISCLOSURE

In this work, we used generative AI tools to assist with polishing the manuscript text. We have carefully reviewed and verified all AI-assisted content and data processing. We take full responsibility for the final content of this work, including all text, claims, analyses, experimental results, and artefacts produced with the assistance of generative AI.

B ADDITIONAL METHOD DETAILS AND PROOFS

This appendix gives the formal statements needed to support the claims in the main paper. Unless stated otherwise, all comparisons condition on a fixed sampled response, fixed teacher targets, fixed selector outputs, and a fixed set of valid response positions. These statements concern the weighted distillation objective rather than convergence guarantees for on-policy distillation.

B.1 FIXED-RESPONSE OBJECTIVE AND ZERO FEEDBACK

Let a sampled response contain $N > 0$ valid prediction positions, indexed by $\mathcal{T} = \{1, \dots, N\}$. Prompt and padding positions are excluded. Let $\ell_t(\theta)$ denote the token-level reverse-KL loss at position t, and let $w_t \in [0, 1]$ be its detached loss weight. The response-level objective is

$$\mathcal{L}(w; \theta) = \frac{1}{N} \sum_{t=1}^{N} w_t \ell_t(\theta), \tag{A1}$$

with direct student gradient

$$g(w) = \frac{1}{N} \sum_{t=1}^{N} w_t \nabla_\theta \ell_t(\theta). \tag{A2}$$

Every allocation applied to the same response uses the same denominator N. We do not renormalize by the number of selected positions or by $\sum_t w_t$. Thus, for a fixed response and student state, $g(cw) = c\, g(w)$ for every $c \geq 0$.

Let $A, B \subseteq \mathcal{T}$ denote the positions selected by two binary rules. Define

$$I = A \cap B, \qquad U = A \cup B.$$

Strict combination uses $w^{\cap} = \mathbf{1}_I$ and Full OR uses $w^{\cup} = \mathbf{1}_U$.

Proposition A.1 (zero direct feedback). If $I = \varnothing$, then

$$\mathcal{L}(w^{\cap}; \theta) = 0, \qquad g(w^{\cap}) = 0. \tag{A3}$$

Proof. Every strict-combination weight is zero, so every term in both sums vanishes. □

We reserve **supervision collapse** for the narrower composition-specific event

$$A \neq \varnothing, \qquad B \neq \varnothing, \qquad A \cap B = \varnothing. \tag{A4}$$

An empty individual selection can also produce zero strict feedback, but it has a different cause and is reported separately. Nonempty individual selections do not by themselves prove nonzero individual gradients; the stronger signature

$$\|g(\mathbf{1}_A)\|_2 > 0, \qquad \|g(\mathbf{1}_B)\|_2 > 0, \qquad \|g(w^{\cap})\|_2 = 0 \tag{A5}$$

requires a same-state gradient measurement.

A zero direct gradient, a skipped optimizer call, and unchanged parameters are distinct observations. In the controlled trainer, zero total loss weight skips backward computation and the optimizer call. Under that recorded behavior, and absent another parameter-changing operation, the parameters remain unchanged at that step.

B.2 Repair path and path-wise minimality

SupportGuard moves from strict combination toward Full OR along

$$w(\lambda) = w^{\cap} + \lambda \left(w^{\cup} - w^{\cap}\right), \qquad 0 \le \lambda \le 1. \tag{A6}$$

For binary selectors,

$$w_t(\lambda) = \begin{cases} 1, & t \in I, \\ \lambda, & t \in U \setminus I, \\ 0, & t \notin U. \end{cases} \tag{A7}$$

Thus shared losses remain at weight one, singly selected losses receive a common supplementary coefficient, and positions rejected by both rules remain excluded. Let

$$m_{\cap} = \frac{|I|}{N}, \qquad m_{\cup} = \frac{|U|}{N}.$$

Along the path,

$$m(w(\lambda)) = m_{\cap} + \lambda(m_{\cup} - m_{\cap}). \tag{A8}$$

For target $\tau \in (0, 1]$, SupportGuard uses

$$\lambda^{\star} = \begin{cases} 0, & m_{\cap} \ge \tau, \\ \min\left\{1, \dfrac{\tau - m_{\cap}}{m_{\cup} - m_{\cap}}\right\}, & m_{\cap} < \tau,\ m_{\cup} > m_{\cap}, \\ 0, & m_{\cap} < \tau,\ m_{\cup} = m_{\cap}. \end{cases} \tag{A9}$$

If $m_{\cup} < \tau$, the coefficient saturates at one and the method uses the complete available union while leaving the target unmet.

Proposition A.2 (smallest feasible coefficient on the declared path). If $m_{\cap} < \tau \le m_{\cup}$, then

$$\lambda^{\star} = \frac{\tau - m_{\cap}}{m_{\cup} - m_{\cap}} \tag{A10}$$

is the unique smallest coefficient on the path in Equation (A6) whose mean loss weight reaches τ.

Proof. Equation (A8) is nondecreasing in λ. Solving $m(w(\lambda)) \ge \tau$ gives

$$\lambda \ge \frac{\tau - m_{\cap}}{m_{\cup} - m_{\cap}}.$$

The lower endpoint is feasible and every smaller coefficient fails. □

This is minimality on the specified strict-to-Full-OR path, not a claim of global optimality over all possible token-weighting rules. Reaching the target also does not guarantee a nonzero gradient or improved task accuracy.

B.3 When nonempty repair can occur

Let $a = |A|$, $b = |B|$, and $k = |A \cap B|$. The exact intersection bounds are

$$\max(0, a + b - N) \leq k \leq \min(a, b). \tag{A11}$$

For binary masks, a nonempty strict combination is below the target exactly when

$$1 \leq k \leq \lceil \tau N \rceil - 1. \tag{A12}$$

At $\tau = 0.01$, this branch cannot occur when $N \leq 100$: a single shared position already gives mean weight at least 0.01. This matters when interpreting short-response diagnostics. A 96-token experiment can test repair of an empty intersection, but it cannot test supplementation of a nonempty intersection below the target. At $N = 512$, by contrast, intersections with one through five shared positions can be nonempty yet still fall below the target.

B.4 Equal total weight does not imply equal gradients

Consider a reachable nonempty-insufficient response with

$$0 < k < M = \tau N < u,$$

where $u = |U|$. A matched Uniform control assigns every union position the same weight

$$w_t^{\mathrm{U}} = \frac{M}{u} \mathbf{1}_U(t).$$

SupportGuard and Uniform have the same total loss weight and the same positive-weight support. They differ only in how weight is distributed between shared and singly selected positions. Let

$$\mu_I = \frac{1}{k} \sum_{t \in I} \nabla_\theta \ell_t, \qquad \mu_X = \frac{1}{u - k} \sum_{t \in U \setminus I} \nabla_\theta \ell_t.$$

Then

$$\boxed{g_{\mathrm{SG}} - g_{\mathrm{U}} = \frac{k(u - M)}{Nu} (\mu_I - \mu_X)} \tag{A13}$$

under the common response-length normalization. Therefore matching total loss weight and positive support does not generally match the gradient. This identity motivates the matched-weight control used later in the appendix.

C Experimental and Implementation Details

C.1 Experimental setup

Models and training data. We conduct mathematical reasoning and code-generation experiments with Qwen2.5-3B-Base as the student and Qwen2.5-7B-Base as the teacher. Mathematics and code are trained separately, using MATH and MBPP as the corresponding training datasets. To examine whether the findings extend beyond the Qwen family, we additionally evaluate mathematical reasoning with the pretrained Gemma 3 4B and 12B models as student and teacher, respectively. Historical Granite experiments are retained as mechanism checks and are not pooled with the current public-benchmark results.

Controlled comparison. Each comparison uses the same student initialization, training data and order, optimizer configuration, and training budget. At each step, the current student generates an

on-policy response, and the teacher provides next-token distributions at the corresponding response positions. Strict combination, Full OR, and SupportGuard optimize the same token-level reverse-KL objective and differ only in their token weights. Because generation is on policy, their training trajectories may diverge after the student parameters begin to differ.

Responses are generated greedily with a maximum length of 2048 tokens. The weighted loss is normalized by the number of valid response positions, and steps with zero total loss weight skip the backward pass and optimizer update. We determine the SupportGuard target τ in separate calibration runs based on update behavior rather than downstream task accuracy. All main training runs use eight NVIDIA H20 GPUs. Additional optimization, batching, precision, and evaluation settings are recorded in the released run configurations.

Evaluation. Mathematical evaluation uses MATH-500, GSM8K, Minerva Math, and OlympiadBench, all reported as Avg@8. Code evaluation uses HumanEval+, MBPP+, and LiveCodeBench, all reported at @1. Within each benchmark comparison, every method uses the same prompt template, decoding configuration, generation limit, stopping rule, answer extractor, and evaluator version.

C.2 Selection rules

The controlled selector pair is

$$a_t = \mathbf{1}[H_S(t) > 3 \text{ bits}], \qquad b_t = \mathbf{1}\left[\max_v p_T(v \mid x, y_{<t}) \geq 0.5\right]. \tag{B1}$$

Entropy is computed in bits from the student distribution. Teacher confidence is the largest teacher probability over the full output vocabulary. A shared position satisfies both selection criteria. This motivates priority, but does not imply that the teacher is necessarily correct or that every shared loss has greater task value.

C.3 Token alignment, normalization, and zero-weight handling

Student and teacher predictions use the same prompt and the student-generated prefix. Prompt and padding positions are excluded from the response loss. Generated EOS is included when it belongs to the saved response. In a causal language model, the final prompt logit predicts the first response token, after which the usual one-position shift applies.

Training uses

$$\frac{1}{N}\sum_{t=1}^{N} w_t \,\mathrm{KL}(p_{S,t} \| p_{T,t}). \tag{B2}$$

The sampled response, teacher targets, selector masks, and repair weights are detached. Gradients pass only through the student distribution. A zero-weight token remains part of the autoregressive context for later tokens.

C.4 Evidence groups

The paper combines several evidence types that answer different questions.

Evidence group	What is held fixed	What it supports
Complete training	initialization and declared protocol	endpoint benchmark results and update coverage
Fixed-context forward analysis	responses, teacher targets, token groups	checkpoint alignment on common contexts
Fixed-response gradients	student state, response, teacher targets, masks	local effect of weight allocation
Historical endpoint audits	original run-specific protocol	exact no-update and mechanism checks

Same-state measurements do not replace complete-training comparisons, and historical settings are not pooled with the current benchmark protocol.

D Additional Evidence for Supervision Collapse

D.1 Full zero-feedback breakdown

Across three strict-combination runs, 603 of 750 scheduled responses retain no shared token loss. The causes are distinct.

Seed	A empty	Both nonempty but disjoint	Total zero feedback	Strict updates	Longest zero-feedback run
42	150	51	201	49	19
123	166	45	211	39	16
456	154	37	191	59	13
Total	**470**	**133**	**603**	**147**	-

The 470 A-empty responses inherit sparsity from an individual selector. The other 133 are supervision-collapse cases: both rules retain positions, but none coincide. Among the 280 responses on which both selectors are nonempty, the conditional collapse rate is

$$\frac{133}{280} = 47.5\%.$$

The remaining 147 responses execute an optimizer call. Positive but sparse feedback can therefore train the model; collapse is reserved for complete removal of the direct distillation signal rather than every small intersection.

D.2 Testing whether low overlap is structural

The original structural-overlap analysis asks whether a small intersection is merely the inevitable consequence of sparse selectors, or whether the two rules also choose systematically different positions. We answer the two parts separately: exact count bounds determine what overlap is feasible, and a count-matched random reference evaluates the placement of the observed selections.

For each response define $t = \lceil \tau N \rceil$, $L = \max(0, a + b - N)$, and $H = \min(a, b)$. If $H < t$, the strict intersection cannot reach the target for any placement of the observed selections. If $L \geq t$, every placement reaches the target. Otherwise the outcome depends on position.

Count-based category	All 750 responses	Both selectors nonempty
Necessarily below target	719	249
Guaranteed to meet target	0	0
Position dependent	31	31

The observed states within these categories are:

Count category	Empty intersection	Nonempty below target	Target met
Necessarily below target	597	122	0
Position dependent	6	16	9

Thus the marginal selection counts explain why the strict intersection is usually too small. They do not explain why the two nonempty selections so often fail to share any position.

D.2.1 Derivation of the count-matched overlap reference

Fix one response with N valid positions. Let A be the observed set selected by the first rule, with $|A| = a$. As a reference, draw the second set B uniformly from all subsets of size b. There are $\binom{N}{b}$ possible choices for B. If the overlap $K = |A \cap B|$ equals j, then j positions must be chosen from the a positions in A and the remaining $b - j$ positions from the $N - a$ positions outside A. Therefore

$$\Pr(K = j \mid N, a, b) = \frac{\binom{a}{j}\binom{N-a}{b-j}}{\binom{N}{b}}. \tag{C1}$$

The expected overlap follows from indicator variables. For each position $t \in A$, let $I_t = \mathbf{1}[t \in B]$. A uniformly drawn size-b subset contains any particular position with probability b/N, so

$$K = \sum_{t \in A} I_t \quad \Longrightarrow \quad \mathbb{E}[K \mid N, a, b] = \frac{ab}{N}. \tag{C2}$$

Setting $j = 0$ in Equation (C1) gives

$$\Pr(K = 0 \mid N, a, b) = \frac{\binom{N-a}{b}}{\binom{N}{b}}. \tag{C3}$$

Because the responses have different lengths and selection counts, the reference is applied separately to each response. For a collection $\mathcal{S}$,

$$\mathbb{E}\left[\sum_{s \in \mathcal{S}} K_s\right] = \sum_{s \in \mathcal{S}} \frac{a_s b_s}{N_s}, \qquad \mathbb{E}[Z] = \sum_{s \in \mathcal{S}} \frac{\binom{N_s - a_s}{b_s}}{\binom{N_s}{b_s}}, \tag{C4}$$

where Z is the number of zero-overlap responses under the reference.

Applying the calculation to the same 280 responses on which both selectors are nonempty gives:

Quantity	Observed	Count-matched reference
Empty intersections	**133**	**3.114**
Total shared positions	**229**	**598.651**

The reference preserves response length and both selector counts but not token meaning. It is therefore a conditional positional reference, not a training intervention. Together, the count bounds and the random reference support a narrow conclusion: selector sparsity strongly constrains how much strict supervision can remain, while complete loss of overlap occurs much more often than the observed counts alone would predict under uniform placement.

D.3 Direct update evidence

Three additional measurements connect missing overlap to optimization.

Same-state gradients. On fixed zero-overlap responses, the student state, response, teacher targets, and masks are held constant. Both individual selector objectives produce nonzero measured gradients, while the strict objective produces an exactly zero direct gradient. This locates the failure before a nonzero combined gradient forms.

Controlled overlap intervention. In Granite-MATH, the selection counts, response allocation, data order, and schedule are held fixed while the shared positions are changed. Zero overlap gives zero retained training weight and zero measured parameter movement in all three schedules. At the largest recorded overlap, relative parameter movement is 3.9303×10^{-5}, 3.6646×10^{-5}, and 4.2283×10^{-5} for seeds 42, 123, and 456.

Exact no-update endpoint. In a separate Qwen-Code setting, each of three strict-combination runs completes 250 scheduled steps with zero optimizer calls. All 434 compared saved tensors remain identical to initialization, with maximum absolute change zero. The result shows why task scores alone are insufficient for diagnosing whether a scheduled training run actually updated the model.

E Additional Analysis of Feedback Allocation

E.1 Repair branches visited during training

On the recorded SupportGuard trajectories, the candidate strict states split into three observed branches:

Branch	Count / 750	SupportGuard action
Empty intersection	595	restore feedback from the union
Nonempty but below target	150	keep shared losses at 1 and supplement singly selected losses
Already sufficient	5	keep strict weights unchanged

Every observed union in this panel can reach $\tau = 0.01$, so no target-unreachable state is visited. These branch counts describe when the rule acts on its own on-policy responses; they do not partition the final accuracy difference into branch-specific gains.

E.2 Four token groups on common frozen responses

A common fixed-context analysis uses 500 initial-student responses containing 218,983 valid positions. Each token is assigned to one of four groups according to the two selector decisions and kept in that group across checkpoint comparisons.

Position group	Position share	Initial mean reverse KL	Share of total KL
Shared	0.074%	1.984	2.04%
A-only	0.116%	0.903	1.47%
B-only	98.216%	0.0618	84.80%
Neither	1.594%	0.525	11.70%

Shared positions are extremely rare but have the largest average teacher-student disagreement. B-only positions have much smaller disagreement individually but dominate the total divergence because they cover almost the whole response. This profile motivates retaining shared-position priority without making strict agreement the only admissible source of feedback. It does not imply that the teacher is correct at every shared position or that reverse KL alone determines task usefulness.

E.3 Fixed-context teacher alignment

The same 500 frozen responses and teacher targets are used to compare trained checkpoints. For position group g and method m, define

$$\Delta\mathrm{KL}_{g,m} = \overline{\mathrm{KL}}_{g,\mathrm{initial}} - \overline{\mathrm{KL}}_{g,m}. \tag{D1}$$

Strict combination, SupportGuard, and Full OR are evaluated on exactly the same contexts. This analysis measures how the trained endpoints differ in teacher alignment on shared, A-only, B-only, and neither positions. It is deliberately separate from task evaluation: a larger KL decrease means stronger alignment to the teacher on those contexts, not necessarily higher benchmark accuracy.

E.4 SHARED-POSITION PRIORITY AT THE SAME TOTAL WEIGHT

A matched Uniform control isolates weight placement. On a fixed response with $0 < m_{\cap} < \tau < m_{\cup}$, Uniform assigns

$$w_t^{\mathrm{U}} = \frac{\tau}{m_{\cup}} \mathbf{1}_U(t),$$

so it has the same mean loss weight as SupportGuard and positive weight on the same union positions. The difference is that Uniform also reduces the shared-position weights below one.

On two nonnested fixed-response anchors, Uniform produces 9.07% and 7.99% of the SupportGuard gradient norm. The gradient cosine similarities between SupportGuard and Uniform are 0.146 and 0.387; after one fresh AdamW step, the update cosines are 0.119 and 0.231. Equal total loss weight and equal positive support therefore do not imply equal local optimization.

A second control moves the unit weight away from the shared position while preserving the positive support and the multiset of weight values. On the tested single-shared-position anchors, the exchanged gradient becomes nearly aligned with Uniform and less aligned with SupportGuard and Strict. This narrows the interpretation: at these anchors, where the large weight is placed matters, not only how concentrated the allocation is.

E.5 NUMERICAL SCALE AND COMPUTATIONAL OVERHEAD

The fixed-response diagnostics include same-condition repeats and gradient-component reconstructions. These provide a numerical reference for interpreting small differences. Differences comparable to the repeat variation or reconstruction residual are not treated as resolved allocation effects; the large SupportGuard-vs-Uniform directional differences above are well outside that scale.

Given the two selector masks, SupportGuard requires elementwise intersection and union, two reductions over the valid positions, one scalar coefficient, and elementwise weight assignment. Its incremental complexity is $O(N)$ and it requires no additional rollout or teacher forward pass. In a matched microbenchmark, the weight construction is about 0.353 ms against roughly 679 ms for a full executed training step, approximately 0.052% of that step. This measures the repair calculation itself; restoring feedback can still cause backward and optimizer work that Strict would skip.