TRANSFERABLE GUIDANCE WITH CONSTRAINED ALLOCATION FOR ON-POLICY SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy self-distillation uses training-only privileged information to supervise a model on its own responses. Its effectiveness depends on both the supervision targets and the allocation of training weight across response positions. We propose `TransCAP` to coordinate these choices through a shared task-conditioned residual. The residual contrasts task-plus-privilege predictions with a privilege-only reference, reweighting a task-only anchor and defining initial position priorities. A constrained KL projection then adjusts these priorities using local directional conflict at the current student distribution. The projection bounds measured directional conflict and prevents the allocation from becoming more concentrated than its initial form. On Qwen3.5-9B, `TransCAP` achieves reasoning and code averages of 80.65% and 72.22%, leading the strongest baselines by 6.58 and 4.32 percentage points. It also achieves state-of-the-art performance on Qwen3-4B and Qwen3.5-27B across both task groups.

1 INTRODUCTION

On-policy distillation trains language models on student-generated responses, with a teacher providing next-token supervision at each response prefix (Agarwal et al., 2024; Ye et al., 2026; Jiang & Ferraro, 2026). On-policy self-distillation (OPSD) obtains this supervision from the model itself through privileged information unavailable to the student at inference (Zhao et al., 2026). This provides dense supervision without a separate teacher, but leaves two coupled choices: what targets to construct and where to emphasize them.

Recent work refines supervision targets (Shen et al., 2026; Wu et al., 2026; Xu et al., 2026), emphasizes informative response positions (Liu et al., 2026; Wang et al., 2026; Yang et al., 2026), or adapts the privileged teacher to the evolving student (Feng et al., 2026; Chen et al., 2026; Ding et al., 2026). The target specifies the desired distribution at a position; its weight determines that position's contribution to training. A target correction can receive little emphasis, while a large weight can amplify a correction that moves the student away from its base-model anchor. This raises a central question: **how should supervision targets and position weights be coordinated across a student-generated response?**

The first challenge is *to align target construction with initial supervision priorities*. Purified OPSD filters the target while retaining uniform position weights (Shen et al., 2026), whereas position-weighted OPSD changes the weights while retaining the privileged target (Liu et al., 2026). USD jointly adjusts privileged-information strength and token weights according to student capacity (Yang et al., 2026). We instead use a shared task-conditioned contrast: adding the task to a privileged context changes the relative support for vocabulary candidates. Its signed pattern defines a target correction, and its magnitude supplies an initial position priority. This connects allocation to the signal used in the target, without equating signal strength with correctness or student error.

The second challenge is *to redistribute supervision while controlling directional conflict and concentration*. At the current student distribution, moving toward the target can oppose moving toward the task-only anchor. Direction-adaptive guidance and geometric regularization address related tensions (Zhang et al., 2026; Jukić & Titov, 2026), but position weighting also determines where the remaining supervision goes. Because weights sum to one, reducing emphasis at one position increases it elsewhere. Prioritizing low-conflict positions can therefore concentrate supervision on a

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.

few tokens. A useful allocation rule must account for both measured conflict and the concentration created by redistribution.

We propose `TransCAP`, which couples guidance construction with constrained supervision allocation. Building on privileged-target refinement (Shen et al., 2026; Wu et al., 2026; Xu et al., 2026), we contrast task-plus-privilege predictions with a privilege-only reference. The resulting residual reweights a task-only anchor and defines the initial position weights. A KL projection then makes the smallest change in allocation needed to satisfy a conflict budget and sorted cumulative bounds. The risk budget controls measured conflict, while the cumulative bounds prevent greater concentration than the initial allocation. Together, these constraints give an explicit allocation rule whose properties can be examined separately from downstream performance.

Our main contributions are:

- We coordinate supervision targets and initial position weights through a shared task-conditioned residual, extending target refinement to response-level supervision allocation.
- We formulate allocation as a constrained KL projection, with a risk budget and cumulative bounds that preserve or increase effective sample size.
- `TransCAP` achieves the best reasoning and code averages among evaluated methods across five benchmarks with 4B, 9B, and 27B models. Averaged across these models, it improves mathematical reasoning and code scores over the corresponding base models by 11.18 and 4.83 percentage points, respectively.

2 Related Work

2.1 Privileged Supervision in On-Policy Self-Distillation

On-policy distillation obtains teacher supervision on student-generated trajectories (Agarwal et al., 2024), while OPSD uses privileged context to obtain supervision from the model itself (Zhao et al., 2026). Recent work examines how this additional information should shape the supervision target. Purified OPSD isolates task-conditioned corrections using a privilege-only reference (Shen et al., 2026). DAPD addresses teacher–student information asymmetry through matched-information anchoring (Wu et al., 2026), while β-OPSD interpolates between a reference policy and the privileged teacher (Xu et al., 2026). Other methods adapt the teacher using student trajectories or verified outcomes (Feng et al., 2026; Chen et al., 2026; Ding et al., 2026). `TransCAP` derives both targets and position scores from a frozen base model, linking target construction and supervision allocation through a shared task-conditioned residual.

2.2 Selective and Token-Weighted Distillation

Token-weighted distillation assigns different amounts of supervision to positions within a response. PW-OPSD uses increasing position weights motivated by teacher reliability along a trajectory (Liu et al., 2026). RP-OPSD uses shifts between teacher views to emphasize reasoning pivots in multilingual transfer (Wang et al., 2026). USD jointly adjusts token selection and privileged-information strength under a student learning-capacity budget (Yang et al., 2026). In `TransCAP`, the magnitude of the residual used to construct each target determines its initial position weight. The allocation thus reflects the task-conditioned corrections retained in the targets. A subsequent projection adjusts these weights under conflict and concentration constraints.

2.3 Behavior Preservation and Constrained Allocation

Several self-distillation methods explicitly address the preservation of existing reasoning behavior. AD-OPSD selectively anchors tokens at risk of suppression to a reference prior from the frozen base model (Peng et al., 2026). DASD varies the direction of teacher supervision with student entropy (Zhang et al., 2026), while GeoSD combines Hellinger matching with a Fisher–Rao proximal penalty on predictive drift (Jukić & Titov, 2026). These methods distinguish the content of supervision from its effects on existing behavior. Our formulation uses information projection and majorization to constrain supervision weights across response positions (Csiszár, 1975; Marshall

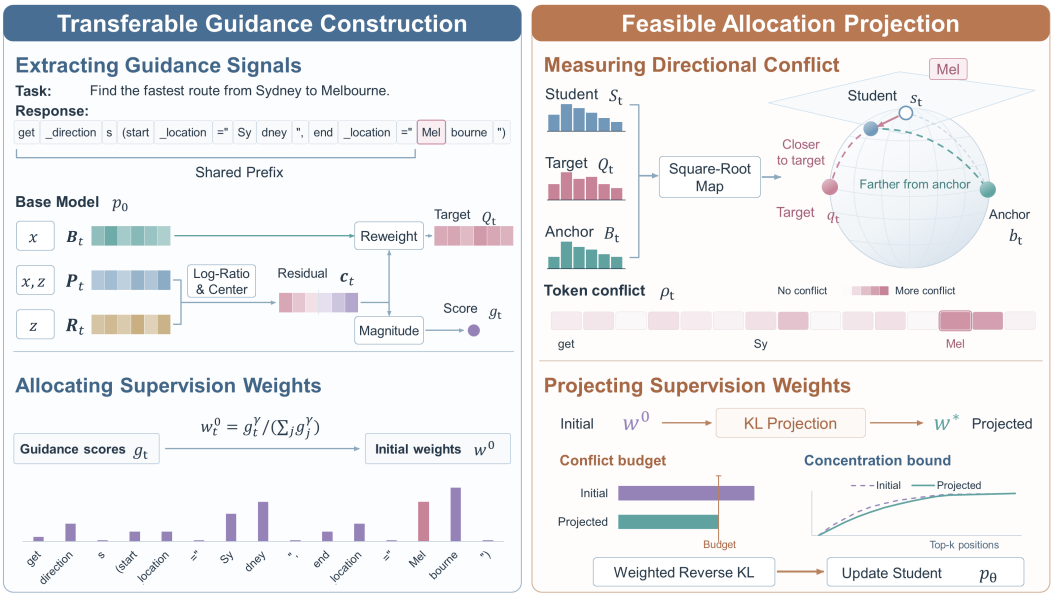

Figure 1: **Overview of `TransCAP`.** A shared task-conditioned residual defines supervision targets and initial token weights. A KL projection adjusts the weights under risk and cumulative concentration constraints before the student receives a weighted distillation update.

et al., 2011). It jointly bounds average measured directional conflict and weight concentration, selecting the feasible allocation closest to the initial weights in KL divergence. The conflict score is a local proxy in distribution space that guides this allocation.

3 METHOD

We introduce `TransCAP`, which jointly constructs supervision targets and allocates supervision weights along student-generated trajectories under privileged supervision. As illustrated in Figure 1, `TransCAP` comprises two coupled modules. **Transferable Guidance Construction** uses a task-conditioned contrast to define supervision targets and initial position priorities. **Feasible Allocation Projection** then adjusts these weights according to directional conflict with the clean anchor, using a risk budget and cumulative constraints to control conflict and concentration after redistribution. Together, the two modules determine what the student learns at each position and how much supervision it receives.

3.1 PROBLEM STATEMENT

In on-policy self-distillation (OPSD), the student $p_{\boldsymbol{\theta}}$ samples a response $\tau = (y_1, \ldots, y_L)$ from task input x, while privileged information z, such as a reference solution, is available only to the teacher during training (Zhao et al., 2026). The response has $L \geq 1$ effective tokens, and $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the prefix at position t. At each sampled prefix $y_{<t}$, let $\boldsymbol{S}_t = (p_{\boldsymbol{\theta}}(v \mid x, y_{<t}))_{v \in \mathcal{V}}$ be the student distribution over vocabulary $\mathcal{V}$. `TransCAP` constructs a teacher-derived target $\boldsymbol{Q}_t$ and position weights $\boldsymbol{w} \in \Delta^{L-1}$, where Δ^{L-1} is the simplex of nonnegative weights summing to one.

All teacher distributions are evaluated along the student's sampled response, so supervision addresses the states encountered by the current policy. The teacher supplies next-token distributions rather than a separately generated target response. We optimize a weighted reverse-KL objective.

3.2 TRANSFERABLE GUIDANCE CONSTRUCTION

We construct a task-conditioned contrast from frozen reference distributions evaluated along the student trajectory. Building on target purification (Shen et al., 2026), we derive targets and position scores from a shared task-conditioned residual.

Extracting Guidance Signals. A frozen base model p_0, which also initializes the student, supplies three distributions at the same student-generated prefix:

$$P_t(v) = p_0(v \mid x, z, y_{<t}), \quad R_t(v) = p_0(v \mid z, y_{<t}), \quad B_t(v) = p_0(v \mid x, y_{<t}). \tag{1}$$

These are the privileged teacher, privilege-only reference, and clean anchor, respectively. The three evaluations share both the model checkpoint and response prefix; only their conditioning information differs. The clean anchor retains the task-only distribution, while the privilege-only reference exposes predictions that can be induced without the task. The teacher-to-reference log-ratio measures the task-conditioned contrast within the privileged context:

$$r_t(v) = \log P_t(v) - \log R_t(v), \qquad c_t(v) = r_t(v) - \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} r_t(u). \tag{2}$$

We use this centered contrast to reweight the task-only anchor:

$$Q_t(v) = \frac{B_t(v) \exp(c_t(v))}{\sum_{u \in \mathcal{V}} B_t(u) \exp(c_t(u))}. \tag{3}$$

This exponential tilt favors candidates supported by the task-plus-privilege context relative to the privilege-only reference. It represents task-conditioned reweighting, rather than the isolated incremental effect of privileged information. Appendix D.1 gives its variational interpretation. Centering leaves the normalized target unchanged and removes vocabulary-wide offsets from the position score. We score each position by the mean absolute residual:

$$g_t = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} |c_t(v)|. \tag{4}$$

The score is the mean absolute centered logit difference, measuring relative-evidence strength rather than a probability-distribution distance. Equal vocabulary weighting retains changes in low-probability candidates instead of suppressing them according to their current probability. Thus g_t supplies a reference-derived priority, not an estimate of correctness or student error; student dependence enters through the conflict score below.

Allocating Supervision Weights. To prioritize positions with stronger residuals, we normalize the scores with a concentration exponent $\gamma \geq 0$:

$$w_t^0(\gamma) = \begin{cases} 1/L, & \gamma = 0 \text{ or } \sum_{j=1}^{L} g_j = 0, \\ \dfrac{g_t^\gamma}{\sum_{j=1}^{L} g_j^\gamma}, & \gamma > 0 \text{ and } \sum_{j=1}^{L} g_j > 0. \end{cases} \tag{5}$$

We use $\gamma = 1$ by default, giving $w_t^0 = g_t / \sum_j g_j$. The setting $\gamma = 0$ recovers uniform supervision, while larger exponents concentrate mass on high-scoring positions; Q1 examines this choice. The allocation is unsmoothed, preserves score ordering for $\gamma > 0$, and is invariant to a common positive rescaling of the scores. The same $\boldsymbol{w}^0$ supplies the KL reference, risk budget, and cumulative bounds in the projection.

3.3 FEASIBLE ALLOCATION PROJECTION

Measuring Directional Conflict. We assess conflict by comparing the direction toward the guidance target with the direction toward the clean anchor at the current student. Following the spherical geometry associated with the Fisher–Rao metric (Jukić & Titov, 2026), we use the square-root embedding:

$$\boldsymbol{s}_t = \sqrt{\boldsymbol{S}_t}, \qquad \boldsymbol{q}_t = \sqrt{\boldsymbol{Q}_t}, \qquad \boldsymbol{b}_t = \sqrt{\boldsymbol{B}_t}. \tag{6}$$

The element-wise square roots have unit norm, placing both directions in the student's tangent space. For unit vectors $\boldsymbol{\xi}, \boldsymbol{\zeta}$ in the positive orthant, the spherical logarithmic map is

$$\mathrm{Log}_{\boldsymbol{\xi}}(\boldsymbol{\zeta}) = \frac{\phi}{\sin\phi}(\boldsymbol{\zeta} - \cos\phi\,\boldsymbol{\xi}), \qquad \phi = \arccos(\boldsymbol{\xi}^\top \boldsymbol{\zeta}). \tag{7}$$

The map is zero at $\phi = 0$ by continuity. Applying it at the current student gives the directions toward the target and clean anchor:

$$\boldsymbol{u}_t = \mathrm{Log}_{\boldsymbol{s}_t}(\boldsymbol{q}_t), \qquad \boldsymbol{v}_t = \mathrm{Log}_{\boldsymbol{s}_t}(\boldsymbol{b}_t), \tag{8}$$

A negative inner product indicates that movement toward the target locally increases squared distance from the anchor. We normalize its opposing component to measure directional conflict:

$$\rho_t = \frac{2[-\langle \boldsymbol{u}_t, \boldsymbol{v}_t\rangle]_+}{\|\boldsymbol{u}_t\|_2^2 + \|\boldsymbol{v}_t\|_2^2}, \tag{9}$$

where $[a]_+ = \max(a, 0)$ and $\rho_t = 0$ when the denominator vanishes. The score lies in $[0, 1]$ and combines angular opposition with the relative lengths of the two directions. It measures local distribution-space conflict, rather than interference between reverse-KL parameter gradients. At the anchor, the anchor direction vanishes and $\rho_t = 0$; between anchor and target, the directions can oppose each other. This permits initial movement away from the anchor while measuring subsequent local tension. Appendix D.3 derives this geometry and shows that the score is not a monotonic measure of learning progress.

Controlling Risk and Concentration. Let $\bar{\rho} = L^{-1}\sum_t \rho_t$ be the conflict under uniform supervision. We set the risk budget to

$$b_\kappa = \bar{\rho} + (1-\kappa)[(\boldsymbol{w}^0)^\top \boldsymbol{\rho} - \bar{\rho}]_+, \qquad \kappa \in [0, 1]. \tag{10}$$

The default $\kappa = 1$ caps conflict at the uniform reference; $\kappa = 0$ leaves $\boldsymbol{w}^0$ feasible and unchanged. When the initial conflict exceeds the uniform reference, the budget removes at least a fraction κ of that excess. Risk control can shift mass onto a small set of lower-conflict positions. To prevent increased concentration, we impose majorization:

$$\boldsymbol{w} \prec \boldsymbol{w}^0 \quad \Longleftrightarrow \quad \sum_{j=1}^{k} w_{[j]} \le \sum_{j=1}^{k} w^0_{[j]}, \quad k = 1, \ldots, L-1, \tag{11}$$

where $w_{[j]}$ and $w^0_{[j]}$ are independently sorted in descending order and both vectors belong to Δ^{L-1}. The cumulative bounds limit the total weight on any k positions while allowing individual weights and their ordering to change. They imply $\mathrm{ESS}(\boldsymbol{w}) \ge \mathrm{ESS}(\boldsymbol{w}^0)$ and $H(\boldsymbol{w}) \ge H(\boldsymbol{w}^0)$ (Appendix D.5). Thus the budget controls measured conflict, while majorization prevents increased allocation concentration.

Projection and Update. We use the initial priorities as a reference and choose the feasible allocation with minimum KL change:

$$\begin{aligned} \boldsymbol{w}^* = \underset{\boldsymbol{w}\in\Delta^{L-1}}{\arg\min}\; & D_{\mathrm{KL}}(\boldsymbol{w}\|\boldsymbol{w}^0) \\ \text{subject to} \quad & \boldsymbol{w}^\top\boldsymbol{\rho} \le b_\kappa, \qquad \boldsymbol{w} \prec \boldsymbol{w}^0. \end{aligned} \tag{12}$$

Here $D_{\mathrm{KL}}(\boldsymbol{w}\|\boldsymbol{w}^0) = \sum_t w_t \log(w_t/w^0_t)$ measures the change in allocation over response positions. If $\boldsymbol{w}^0$ already satisfies the budget, the projection retains it; otherwise the budget and cumulative bounds jointly determine how its mass can be redistributed. Appendix D.2 describes the solver, and Appendix D.4 establishes existence and uniqueness for positive initial weights. The student minimizes

$$\mathcal{L}_{\mathrm{final}} = \sum_{t=1}^{L} w^*_t D_{\mathrm{KL}}(\boldsymbol{S}_t\|\boldsymbol{Q}_t). \tag{13}$$

We average this loss over responses and refresh the on-policy samples and allocations at each iteration. During differentiation, sampled prefixes, targets, and weights are fixed, so gradients flow only through the student distributions.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on AIME 2024, AIME 2025, and HMMT 2025 for mathematical reasoning (Hugging Face H4, 2025; Zhang & Math-AI Team, 2025; HMMT, 2025), LiveCodeBench v6 for code generation (Jain et al., 2025), and BFCL v4 for function calling (Patil et al., 2025). Mathematical performance is measured by Avg@12. Code generation and function calling follow the official evaluation protocols of LCB v6 and BFCL v4, respectively. All scores are percentages; task-group averages weight the three mathematics benchmarks or the two code benchmarks equally.

Baselines. We compare with self-distillation fine-tuning (SDFT) (Shenfeld et al., 2026), DOPD (Yu et al., 2026), OPSD (Zhao et al., 2026), DAPD (Wu et al., 2026), β-OPSD (Xu et al., 2026), and PURIFIED OPSD (Shen et al., 2026). We use published results for matching training datasets and backbones; otherwise, we reproduce methods using their official implementations. Our reproductions use matched training data and evaluation settings within each backbone, with matched rollout budgets for on-policy methods.

Implementation and Evaluation. We use Qwen3-4B, Qwen3.5-9B, and Qwen3.5-27B. Each `TransCAP` student is initialized from the officially released Qwen checkpoints on Hugging Face (e.g., Qwen/Qwen3.5-27B), with a frozen copy providing the privileged teacher, privilege-only reference, and clean anchor. Only student LoRA parameters are updated. We use OpenThoughts Math for mathematical training and Skywork-Code for code-generation training. BFCL v4 evaluates transfer from the code-trained checkpoints. Training runs for 200 steps, and evaluation uses the step-200 checkpoint. Benchmarks, baselines, implementation, and computational costs are detailed in Appendices A, B, E, and G.

4.2 MAIN RESULTS

Table 1 compares the three backbones on mathematical reasoning and code.

Overall Performance. `TransCAP` achieves the highest reasoning and code averages among the evaluated methods on all three backbones. On Qwen3-4B, it reaches average scores of 66.48 and 44.24, outperforming the strongest competing baseline by 0.92 points on reasoning and 0.33 points on code. The gains are larger on Qwen3.5-9B, where `TransCAP` obtains 80.65 and 72.22, exceeding the strongest baselines by 6.58 and 4.32 points, respectively. On Qwen3.5-27B, the corresponding averages are 89.91 and 80.51, exceeding the strongest competing baselines by 2.32 and 2.49 points. These results favor joint target construction and supervision allocation over target purification or position weighting alone.

Effect across Model Scales. The results also show that on-policy or privileged supervision does not automatically improve a stronger base model. For example, DOPD underperforms the raw Qwen3.5-9B model on reasoning, while standard OPSD reduces code performance on Qwen3.5-9B. Similar regressions appear for several baselines on Qwen3.5-27B. In contrast, `TransCAP` remains effective across all three scales. Relative to the raw model, it improves the Qwen3-4B averages by 8.61 points on reasoning and 2.20 points on code. On Qwen3.5-9B, the gains increase to 18.34 and 6.37 points. This suggests that the benefit arises from controlling what privileged guidance is transferred and how it is distributed, rather than from model capacity alone.

Reasoning and Code Transfer. Reasoning performance improves over the base model on all three mathematics benchmarks. On Qwen3.5-9B, `TransCAP` improves over Purified OPSD by 4.17 points on AIME 2025 and 7.78 points on both AIME 2024 and HMMT 2025. The method also transfers effectively to code. On Qwen3-4B, it improves BFCL v4 by 2.58 points over the strongest baseline, while trailing β-OPSD by 1.91 points on LCB v6. On Qwen3.5-9B, it leads both benchmarks by 4.83 and 2.43 points, respectively. The gains across exact-answer reasoning, executable code generation, and function calling indicate that `TransCAP` is effective across substantially different response and supervision structures.

Table 1: Main results (%). Best scores within each backbone group are bold.

Method	Reasoning (Avg@12)				Code		
	AIME 2025	AIME 2024	HMMT 2025	Avg.	LCB v6	BFCL v4	Avg.
Qwen3-4B							
Raw Base	60.28	70.83	42.50	57.87	48.40	35.68	42.04
SDFT (Shenfeld et al., 2026)	62.78	75.00	41.94	59.91	47.62	35.51	41.57
DOPD (Yu et al., 2026)	65.00	73.61	38.89	59.17	47.42	36.07	41.75
OPSD (Zhao et al., 2026)	67.78	76.67	43.33	62.59	47.89	35.70	41.80
DAPD (Wu et al., 2026)	72.22	**78.06**	46.39	65.56	49.06	36.24	42.65
β-OPSD (Xu et al., 2026)	70.83	71.67	44.17	62.22	**51.11**	36.70	43.91
Purified OPSD (Shen et al., 2026)	67.50	76.67	45.00	63.06	50.11	36.28	43.20
TransCAP	**74.17**	77.22	**48.06**	**66.48**	49.20	**39.28**	**44.24**
Qwen3.5-9B							
Raw Base	63.06	73.61	50.28	62.31	65.60	66.10	65.85
SDFT (Shenfeld et al., 2026)	70.28	76.39	56.11	67.59	67.79	67.04	67.42
DOPD (Yu et al., 2026)	59.17	73.06	51.39	61.20	64.27	65.97	65.12
OPSD (Zhao et al., 2026)	62.50	71.67	52.78	62.31	63.32	66.20	64.76
DAPD (Wu et al., 2026)	75.28	75.83	61.39	70.83	67.11	68.67	67.89
β-OPSD (Xu et al., 2026)	73.89	78.89	59.72	70.83	68.50	67.30	67.90
Purified OPSD (Shen et al., 2026)	76.94	79.44	65.83	74.07	67.92	63.31	65.62
TransCAP	**81.11**	**87.22**	**73.61**	**80.65**	**73.33**	**71.10**	**72.22**
Qwen3.5-27B							
Raw Base	74.17	83.89	91.94	83.33	80.70	68.50	74.60
SDFT (Shenfeld et al., 2026)	71.94	83.61	92.22	82.59	79.06	68.37	73.72
DOPD (Yu et al., 2026)	73.89	82.78	92.22	82.96	77.90	68.60	73.25
OPSD (Zhao et al., 2026)	78.33	85.56	92.50	85.46	83.39	69.47	76.43
DAPD (Wu et al., 2026)	81.39	85.28	92.78	86.48	82.56	71.16	76.86
β-OPSD (Xu et al., 2026)	80.56	86.94	92.78	86.76	84.27	69.74	77.01
Purified OPSD (Shen et al., 2026)	82.22	87.22	93.33	87.59	85.24	70.79	78.02
TransCAP	**85.56**	**90.00**	**94.17**	**89.91**	**87.34**	**73.68**	**80.51**

4.3 Ablation and Mechanistic Analysis

We examine four questions: how strongly to concentrate supervision, how tightly to control conflict, whether allocation benefits from token-level alignment, and what cumulative constraints add beyond risk control. The training ablations (Q1–Q4) use Qwen3.5-9B with matched settings and evaluate step-200 checkpoints on AIME 2025 (Avg@12) and LCB v6; all scores are percentages. Q4 additionally compares allocations on fixed model-generated responses to isolate constraint behavior at identical prefixes. Shared configurations are reused across tables; the protocol is detailed in Appendix E.4.

Q1: How concentrated should supervision be? We vary the allocation exponent $\gamma \in \{0, 0.5, 1, 1.5, 2\}$ in Eq. (5), retaining full projection at $\kappa = 1$. Increasing γ emphasizes larger residual scores in the initial weights; $\gamma = 0$ gives uniform allocation. Because these weights also define the projection reference and cumulative bounds, the sweep tests concentration within the full allocation framework.

Table 2 shows gains over uniform allocation at every tested positive exponent. The default $\gamma = 1$ improves AIME 2025 from 74.17% to 81.11% and LCB v6 from 68.12% to 73.33%, gains of 6.94 and 5.21 points. The best tested concentration differs by task: mathematics peaks at 83.33% for $\gamma = 1.5$, whereas code peaks at $\gamma = 1$ and declines to 72.12% at $\gamma = 1.5$. At $\gamma = 2$, scores fall to 79.17% and 71.12%, respectively. Thus, emphasizing strong guidance helps, but concentrating supervision more aggressively eventually reduces performance. The default $\gamma = 1$ balances this trade-off across tasks, while the mathematics peak shows that the preferred concentration can be task dependent.

γ	AIME 2025 (Avg@12)	LCB v6
0	74.17	68.12
0.5	76.11	70.12
1 (default)	81.11	**73.33**
1.5	**83.33**	72.12
2	79.17	71.12

Table 2: Allocation strength ($\kappa = 1$).

κ	AIME 2025 (Avg@12)	LCB v6
0	77.50	61.20
0.25	78.06	63.40
0.5	79.17	67.10
0.75	**81.11**	71.20
1 (default)	**81.11**	**73.33**

Table 3: Risk-control strength ($\gamma = 1$).

Q2: How much risk control is needed? We fix $\gamma = 1$ and vary $\kappa \in \{0, 0.25, 0.5, 0.75, 1\}$ in Eq. (10), retaining the target construction and full cumulative constraints.

Increasing κ from 0 to 1 raises AIME 2025 from 77.50% to 81.11% and LCB v6 from 61.20% to 73.33% (Table 3). Mathematics reaches its highest observed score at $\kappa = 0.75$ and plateaus, whereas code improves at every tested increase, gaining another 2.13 points from 0.75 to 1. The code result makes the role of protection particularly clear: without risk correction, the score falls 4.40 points below Raw Base; with default control, it exceeds Raw Base by 7.73 points. These observations support correcting conflict rather than relying on guidance strength alone. The default $\kappa = 1$ achieves the highest tested code score and ties for the highest mathematics score. This sweep activates risk control within the full projection framework; Q4 tests the additional performance and allocation effects of cumulative constraints.

Q3: Does allocation benefit from token-level alignment? We compare uniform (U), guidance-based (G), and within-bin permuted (ΠG) initial allocations under the same Full projection with $\kappa = 1$. Here G denotes the initial allocation rule ($\gamma = 1$); G + Full is the complete `TransCAP` method. For ΠG, we divide each response into eight relative-position bins and randomly permute the initial guidance weights within each bin before projection. Tokens and targets remain unchanged, as do the initial weight distribution and coarse positional mass. Thus, Q3 varies initial allocation while holding the projection rules fixed; Raw Base and G (no projection) provide additional references. Appendix E.4 specifies the preserved quantities and projection behavior.

Configuration	AIME 2025 (Avg@12)	LCB v6
Raw Base	63.06	65.60
G (no projection)	77.50	61.20
U + Full	74.17	68.12
ΠG + Full	78.61	69.74
G + Full (`TransCAP`)	**81.11**	**73.33**

Table 4: Allocation ablation with Full projection.

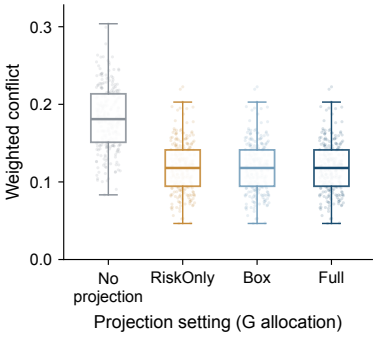

Figure 2: Fixed-prefix conflict with G allocation.

`TransCAP` improves over U + Full by 6.94 points on AIME 2025 and 5.21 points on LCB v6 (Table 4). ΠG + Full retains gains of 4.44 and 1.62 points over U + Full but trails `TransCAP` by 2.50 and 3.59 points, respectively. Both tasks favor the original alignment over within-bin permutation, while permutation retains useful allocation structure relative to uniform weighting. The control preserves both weight shape and coarse positional mass, so it does not separate their individual contributions. The G (no projection) reference scores 61.20% on code, compared with 73.33% for G + Full, showing that initial alignment alone is insufficient. Together, these comparisons support aligning initial supervision with guidance and controlling its subsequent allocation.

Q4: What do cumulative constraints add beyond risk control? We fix the initial allocation to G ($\gamma = 1$) and vary only the projection setting. G (no projection) uses the initial weights directly; G + RiskOnly adds the risk budget, and G + Box also imposes individual bounds. G + Full (`TransCAP`) combines the risk budget with all cumulative constraints. Constrained variants use $\kappa = 1$; all variants share target construction and training settings but follow their own on-policy trajectories.

Configuration	AIME 2025 (Avg@12)	LCB v6
Raw Base	63.06	65.60
G (no projection)	77.50	61.20
G + RiskOnly	78.61	67.20
G + Box	80.28	71.30
G + Full (`TransCAP`)	**81.11**	**73.33**

Table 5: Projection ablation with G allocation.

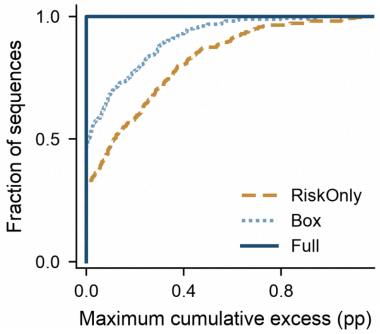

Figure 3: Cumulative excess with G allocation.

G + RiskOnly improves over G (no projection) by 1.11 points on AIME 2025 and 6.00 points on LCB v6 (Table 5). G + Box adds gains of 1.67 and 4.10 points, respectively. G + Full performs best on both tasks, exceeding G + Box by 0.83 and 2.03 points. These comparisons support controlling cumulative concentration beyond individual weight bounds, with a larger additional gain on code in this experiment.

We separately examine allocation behavior on 128 mathematical and 128 coding responses generated by a 9B model. At identical prefixes, the four methods share targets, conflict scores, nominal weights, and budgets. All cases are retained, and both diagnostic figures pool the 256 responses (Appendix E.4). Both figures hold G allocation fixed and label variants by their projection setting. The three projections reduce weighted conflict to similar levels (Figure 2). To distinguish their concentration behavior, we measure maximum cumulative excess:

$$E(\boldsymbol{w}) = \max_{1 \le k < L} \left[\sum_{i=1}^{k} w_{[i]} - \sum_{i=1}^{k} w^{0}_{[i]} \right]_{+}, \tag{14}$$

where the two weight vectors are independently sorted in descending order. Figure 3 reports $100E(\boldsymbol{w})$ in percentage points. RiskOnly retains positive excess, and Box reduces it without eliminating it; Full removes the remaining excess. The fixed-prefix comparison verifies the distinct allocation effects of the constraints, while the training ablation measures their task-level benefits. Appendix F further examines risk, concentration, and projection behavior during training.

5 Conclusion

We introduced `TransCAP`, which combines Transferable Guidance Construction with Feasible Allocation Projection. The first component uses a task-conditioned residual to jointly construct supervision targets and position weights, aligning what the student learns with where learning is emphasized. The second component rebalances this allocation according to directional conflict with the clean anchor while limiting supervision concentration. Across five reasoning and code benchmarks and three model scales, `TransCAP` achieves the highest reasoning and code averages among the evaluated methods on each backbone. Since all auxiliary components are used only during training, the resulting student retains the original inference architecture. Overall, our results show that effective privileged self-distillation depends not only on constructing transferable targets, but also on coordinating their allocation across student-generated trajectories.

AI USE STATEMENT

We used generative AI tools for keyword suggestions, literature searches and assistance with reproducing prior work, error analysis during experiments. The authors are responsible for verifying AI-assisted text, code suggestions, and analyses against the implementation, experimental records, and relevant literature. The authors take responsibility for research decisions, reported experimental results, and the final manuscript.

ETHICS STATEMENT

This work studies on-policy self-distillation for mathematical reasoning, code generation, and function calling. We identify no specific ethical concerns arising from the experiments reported here. The allocation constraints control a measured training-time conflict score and supervision concentration; they do not constitute a safety guarantee for generated programs or tool calls. Applying the trained models in deployed systems would require validation appropriate to the intended use.

REPRODUCIBILITY STATEMENT

Section 3 defines the supervision targets, allocation constraints, and training objective, and Section 4.1 describes the benchmarks and comparison setup. Appendix C gives the overall training algorithm. Appendix D provides the projection solver, numerical feasibility and optimality checks, and theoretical derivations. Appendix E documents the training data, model initialization, LoRA configuration, optimization and sampling settings, compute resources, and loss computation. Tables 7 and 8 summarize the shared and task-specific training settings. Appendix E.4 specifies the mathematical evaluation protocol and the settings held fixed across ablations. Main experiments and ablations use the step-200 checkpoint; mathematical Avg@12 averages correctness over 12 sampled responses per problem.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Yutong Chen, Guangfu Guo, Zhichao Xu, and Kunpeng Liu. Dualopsd: Adaptive privileged teachers for on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.26019.

Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 3(1):146–158, 1975. doi: 10.1214/aop/1176996454. URL https://doi.org/10.1214/aop/1176996454.

Zewen Ding, Zezhong Wu, Zhou Tao, Shida Wang, Shizhuo Hou, YongXiang Hua, Haoyu Cao, and Linli Xu. Vista: Verifier-informed student-to-teacher adaptation for on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.28306.

Yangyang Feng, Zhuoyan Feng, and Junlan Chen. Past: Privileged adaptation from complete student trajectories for on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.08726.

HMMT. Harvard–MIT Mathematics Tournament: February 2025. Official competition archive, 2025. URL https://www.hmmt.org/www/archive/282. Individual tests and solutions. Accessed 2026-09-11.

Hugging Face H4. AIME 2024. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024. Dataset release of the 2024 competition problems. Accessed 2026-09-11.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/pdf?id=chfJJYC3iL.

Yuxuan Jiang and Francis Ferraro. Bridging reasoning trajectories in on-policy distillation via near-future guidance, 2026. URL https://arxiv.org/abs/2606.00305.

Josip Jukić and Ivan Titov. Geometric self-distillation for reasoning generalization, 2026. URL https://arxiv.org/abs/2607.06855.

Xiaogeng Liu, Xinyan Wang, Yingzi Ma, Yechao Zhang, and Chaowei Xiao. When are teacher tokens reliable? position-weighted on-policy self-distillation for reasoning, 2026. URL https://arxiv.org/abs/2605.21606.

LiveCodeBench. LiveCodeBench: Dataset versions. GitHub repository, 2025. URL https://github.com/LiveCodeBench/LiveCodeBench#dataset-versions. Release release_v6: 1,055 problems from May 2023 through April 2025. Accessed 2026-09-11.

Huanzhi Mao, Amitoj Singh, Pracheeti Shikarkhane, Kai Wen, Shishir G. Patil, and Joseph E. Gonzalez. BFCL V4 agentic, part 2: Evaluating tool calling for memory. Gorilla project blog, July 2025a. URL https://gorilla.cs.berkeley.edu/blogs/16_bfcl_v4_memory.html. Released July 17, 2025. Accessed 2026-09-11.

Huanzhi Mao, Raymond Tsao, Jingzhuo Zhou, Shishir G. Patil, and Joseph E. Gonzalez. BFCL V4 agentic, part 1: Web search. Gorilla project blog, July 2025b. URL https://gorilla.cs.berkeley.edu/blogs/15_bfcl_v4_web_search.html. Released July 17, 2025; updated July 20, 2025. Accessed 2026-09-11.

Albert W. Marshall, Ingram Olkin, and Barry C. Arnold. *Inequalities: Theory of Majorization and Its Applications*. Springer, 2 edition, 2011. doi: 10.1007/978-0-387-68276-1. URL https://doi.org/10.1007/978-0-387-68276-1.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Keqin Peng, Chen Li, Yuanxin Ouyang, Yancheng Yuan, and Liang Ding. Diagnosing and mitigating thinking collapse in on-policy self-distillation, 2026. URL https://arxiv.org/abs/2607.10805.

Zhanming Shen, Jintao Tong, Shaotian Yan, Chen Shen, Hao Chen, Wentao Ye, Xiaomeng Hu, Rui Miao, Haobo Wang, Junbo Zhao, Gang Chen, and Jieping Ye. Purified opsd: On-policy self-distillation without losing how to think, 2026. URL https://arxiv.org/abs/2607.02234.

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

Xinye Wang, Junxiao Liu, and Shujian Huang. Rp-opsd: Reasoning-pivot-guided on-policy self-distillation for multilingual reasoning transfer, 2026. URL https://arxiv.org/abs/2608.06347.

Jianyu Wu, Yizhou Wang, Encheng Su, Chen Tang, and Shixiang Tang. Dapd: Dual-anchored policy distillation, 2026. URL https://arxiv.org/abs/2608.01735.

Jiawei Xu, Minghui Liu, Juzheng Zhang, Tom Goldstein, and Furong Huang. β-opsd: Deriving with policy optimization, training with self-distillation, 2026. URL https://arxiv.org/abs/2607.28582.

Yongkang Yang, Zhezheng Hao, Hong Zhang, Yi Liu, Xiankun Lin, Wence Ji, Fanjunduo Wei, Jiarui Yu, Qiang Lin, Xiaoyun Liang, and Hande Dong. Matching supervision to the student's learning capacity: A unified framework for on-policy self-distillation, 2026. URL https://arxiv.org/abs/2608.08176.

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

Xinlei Yu, Gen Li, Qingyi Si, Guibin Zhang, Yuqi Xu, Congcong Wang, Shuai Dong, Kaiwen Tuo, Xiangyu Zeng, Kaituo Feng, Qunzhong Wang, Yang Shi, Xiaobin Hu, Xiangyu Yue, Jiaqi Wang, and Shuicheng Yan. DOPD: Dual on-policy distillation, 2026. URL https://arxiv.org/abs/2606.30626.

Hongbin Zhang, Chaozheng Wang, Kehai Chen, Youcheng Pan, Yang Xiang, Jinpeng Wang, and Min Zhang. Tailoring teaching to aptitude: Direction-adaptive self-distillation for LLM reasoning, 2026. URL https://arxiv.org/abs/2605.22263.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/math-ai/aime25. Accessed 2026-09-11.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

A BENCHMARK DETAILS

Our evaluation covers mathematical reasoning, executable code generation, and function calling. Table 6 summarizes the benchmark releases and reporting conventions. The main comparison uses all five benchmarks, while the training ablations use AIME 2025 and LCB v6 to examine mathematical and coding performance under matched training settings. The fixed-prefix diagnostic in Q4 measures allocation behavior on real model-generated responses and is reported separately from benchmark scores.

Table 6: Benchmark releases and evaluation metrics. Release sizes describe the cited datasets; BFCL v4 counts scoring entries across heterogeneous categories.

Benchmark	Task	Release size	Reported metric
AIME 2024	Mathematical reasoning	30	Avg@12
AIME 2025	Mathematical reasoning	30	Avg@12
HMMT February 2025	Mathematical reasoning	30	Avg@12
LiveCodeBench v6	Code generation	1,055	Pass@1
BFCL v4	Function calling	5,088	Official aggregate

AIME 2024 and AIME 2025. These benchmarks contain competition mathematics problems with numerical answers. AIME 2024 uses the 30-problem release combining AIME I and II from Hugging Face H4 (Hugging Face H4, 2025); AIME 2025 uses the 30-problem `math-ai/aime25` release (Zhang & Math-AI Team, 2025). We sample 12 responses per problem and report Avg@12, averaging binary answer correctness over responses and then over problems. For N problems and correctness indicator a_{ij} for response j to problem i, the percentage score is

$$\text{Avg@12} = \frac{100}{12N} \sum_{i=1}^{N} \sum_{j=1}^{12} a_{ij}. \tag{15}$$

Thus each 30-problem mathematical benchmark contributes 360 sampled responses, and Avg@12 measures average response accuracy rather than success among 12 attempts or majority-vote accuracy.

HMMT 2025. We evaluate the February 2025 individual competition problems, covering algebra and number theory, combinatorics, and geometry (HMMT, 2025). The 30-problem benchmark complements AIME with a different competition source and uses the same Avg@12 metric and mathematical decoding settings. Sampling parameters, response limits, and checkpoint selection are given in Appendix E.4.

LiveCodeBench v6. LiveCodeBench evaluates generated programs through execution against test cases (Jain et al., 2025). Its `release_v6` contains 1,055 problems released between May 2023 and April 2025 (LiveCodeBench, 2025). We report code-generation pass@1 using the official evaluation protocol, assessing functional correctness rather than textual similarity to a reference program.

BFCL v4. The Berkeley Function Calling Leaderboard evaluates the selection and use of functions across several interaction settings (Patil et al., 2025). Version 4 includes agentic web-search and memory tasks alongside single-turn, live, multi-turn, and irrelevance categories (Mao et al., 2025b;a). We follow its category-specific verification and official score aggregation; the overall score is not a pooled accuracy over interchangeable test cases.

Task-group reporting. All benchmark scores are percentages. The reasoning average is the arithmetic mean of AIME 2024, AIME 2025, and HMMT 2025; the code-group average is the arithmetic mean of LCB v6 and BFCL v4. Each benchmark therefore receives equal weight within its group regardless of release size.

B BASELINE METHODS

We compare TransCAP with the unadapted backbone and six distillation methods in Table 1. Published results are used for matching training datasets and backbones; otherwise, methods are reproduced using their official implementations. Within each backbone, our reproductions use matched training data and evaluation settings, with matched rollout budgets for on-policy methods. The methods differ in the source, construction, or allocation of the supervision signal, as summarized below.

Raw Base. The original Qwen checkpoint is evaluated before task-specific distillation. It provides the reference for measuring improvements or regressions after adaptation and is evaluated on the same benchmarks as the trained variants.

SDFT. Self-Distillation Fine-Tuning (Shenfeld et al., 2026) turns demonstrations into on-policy supervision through in-context learning. A demonstration-conditioned version of the model serves as its own teacher, supplying distributional feedback on student-generated responses. This baseline tests learning from demonstrations through self-distillation rather than direct imitation of demonstration tokens.

DOPD. Dual On-policy Distillation (Yu et al., 2026) dynamically routes token-level supervision between privileged teacher and privileged student policies. Routing depends on their advantage gap and relative probabilities, allowing the supervision source, strength, and objective to vary across tokens. It provides a comparison with adaptive supervision designed to address the information gap introduced by privileged context.

OPSD. On-Policy Self-Distillation (Zhao et al., 2026) uses a model as both teacher and student under different conditioning contexts. The teacher receives privileged information such as a reference solution, whereas the student receives only the task input. Training matches their next-token distributions along the student's own sampled responses.

DAPD. Dual-Anchored Policy Distillation (Wu et al., 2026) addresses teacher–student information asymmetry through two levels of anchoring. Dual-path anchoring introduces a self-conditioned bridge to compare policies under matched information, while dual-source anchoring applies reference-to-rollout and rollout-to-reference supervision. The comparison examines an alternative way to retain correctness guidance while reducing reliance on privileged information.

β-OPSD. β-OPSD (Xu et al., 2026) exposes the strength of reference-policy regularization as a tunable parameter. It derives a distillation target through geometric interpolation between a reference policy and the privileged teacher, implemented by mixing token-level logits. This baseline controls the balance between reference behavior and privileged guidance through the target distribution.

Purified OPSD. Purified OPSD (Shen et al., 2026) uses a reference-only teacher to separate reference-induced predictions from question-conditioned corrections. The residual is converted into a normalized target through a pointwise-mutual-information formulation. It is the closest target-construction baseline: TransCAP additionally uses the residual to allocate position weights and refines that allocation under risk and cumulative constraints.

C OVERALL ALGORITHM

Algorithm 1 summarizes the complete training procedure. The training set $\mathcal{D}$ contains pairs of task inputs and privileged information. At each iteration, $\mathcal{M}$ denotes a sampled minibatch and $\mathcal{L}_{\text{batch}}$ accumulates its response-level losses. All response-level quantities are computed separately for each sampled response.

The initial allocation in Eq. (5) defines the KL reference, budget, and majorization bounds. Appendix D.2 gives the constraint-generation procedure for the projection step. The minibatch size $|\mathcal{M}|$ normalizes the accumulated loss so that each response contributes one response-level objective. Only the student distributions contribute gradients to the parameter update. Refreshing the

Algorithm 1 Overall OPSD Training

Require: Training set $\mathcal{D}$, base model p_0
Require: Risk-control strength $\kappa \in [0, 1]$ (default: 1) and allocation exponent $\gamma \geq 0$
Ensure: Trained student policy $p_{\boldsymbol{\theta}}$

```
Initialize p_θ from p_0 and keep p_0 frozen
while training is not complete do
    Sample a minibatch M from D and set L_batch ← 0
    for all (x, z) ∈ M do
        Sample τ = (y_1, ..., y_L) ~ p_θ(· | x)
        for t = 1, ..., L do
            Evaluate S_t = (p_θ(v | x, y_<t))_{v∈V}
            Evaluate P_t, R_t, and B_t using Eq. (1)
            Compute c_t using Eq. (2)
            Set g_t ← |V|^{-1} Σ_{v∈V} |c_t(v)|
            Construct Q_t from B_t and c_t using Eq. (3)
            Compute u_t and v_t using Eq. (8) and Appendix D.3
            Compute ρ_t using Eq. (9), with ρ_t = 0 if its denominator is zero
        end for
        Compute the initial allocation w^0(γ) using Eq. (5)
        Compute b_κ using Eq. (10)
        if Σ_{t=1}^{L} w_t^0 ρ_t ≤ b_κ then
            Set w* ← w^0
        else
            Solve Eq. (12) for w*
        end if
        Hold τ, {Q_t}_{t=1}^{L}, and w* fixed during differentiation
        L_batch ← L_batch + Σ_{t=1}^{L} w_t* D_KL(S_t ‖ Q_t)
    end for
    Update θ using ∇_θ (L_batch / |M|)
end while
return p_θ
```

responses and allocations at each iteration adapts supervision to the student's current behavior while controlling measured conflict and weight concentration.

The projection guarantee under positive initial weights is given in Appendix D.4.

D Theory and Projection Details

D.1 Interpretation of the shared residual

Centered logit contrast. Let $a_t^P(v)$ and $a_t^R(v)$ denote logits whose softmax distributions are P_t and R_t. Writing $\Delta a_t(v) = a_t^P(v) - a_t^R(v)$ gives

$$c_t(v) = \Delta a_t(v) - \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \Delta a_t(u). \tag{16}$$

Indeed, the log-ratio contains $\Delta a_t(v)$ plus a vocabulary-independent difference of log-normalizers, which centering removes. Consequently, g_t is invariant to a common additive shift of either logit vector. It measures the mean absolute variation in the conditional logit contrast, without cancellation between positive and negative components.

Uniform vocabulary weighting counts relative corrections without weighting them by P_t, R_t, B_t, or S_t. Low-probability candidates can therefore contribute substantially, even when their changes move little probability mass. This defines a residual-strength priority rather than a KL, total-variation, or Hellinger distance, and does not imply a ranking by learning benefit. For comparison, writing $Z_t = \sum_v B_t(v) e^{c_t(v)}$ yields $D_{\mathrm{KL}}(Q_t \| B_t) = \mathbb{E}_{Q_t}[c_t] - \log Z_t$. That quantity measures the resulting target displacement, whereas g_t summarizes the contrast used to construct it.

Anchor-relative target construction. For positive B_t, the target has the variational characterization

$$Q_t = \arg\max_{\pi \in \Delta^{|\mathcal{V}|-1}} \{\mathbb{E}_\pi[c_t] - D_{\mathrm{KL}}(\pi \| B_t)\}. \tag{17}$$

To verify this, substitute $\log Q_t = \log B_t + c_t - \log Z_t$: $\mathbb{E}_\pi[c_t] - D_{\mathrm{KL}}(\pi\|B_t) = \log Z_t - D_{\mathrm{KL}}(\pi\|Q_t)$. The unique maximizer is Q_t, and replacing c_t by r_t leaves it unchanged. This construction combines a task-only anchor with the relative support supplied by the task-conditioned contrast.

The contrast compares inclusion of x within the privileged context; it does not isolate the incremental contribution of z. In particular, $P_t = B_t$ gives $Q_t(v) \propto B_t(v)^2/R_t(v)$, which can still reweight the anchor. The no-correction condition is instead $Q_t = B_t$ if and only if $P_t = R_t$, under positive reference distributions. Thus the construction can express both privilege-assisted corrections and task-conditioned contrastive reweighting; their separate performance contributions are not identified by this identity.

D.2 Projection solver

The solver derivation below assumes strictly positive initial weights $\boldsymbol{w}^0$ from Eq. (5). The same initial allocation defines the KL reference, budget b_κ, and cumulative bounds. With $0\log(0/w_t^0) = 0$, the KL objective is finite throughout the simplex. Appendix D.4 states the positive-score condition and the resulting projection guarantee.

Let $T_k(\boldsymbol{w}) = \sum_{j=1}^k w_{[j]}$ and $m_k = T_k(\boldsymbol{w}^0)$ for $1 \le k < L$. The restricted problem retains a collection $\mathcal{C}$ of subset bounds $\sum_{t\in A} w_t \le m_{|A|}$. With nonnegative multipliers λ for the budget and μ_A for these bounds, define the penalty at position t:

$$h_t = \lambda\rho_t + \sum_{A\in\mathcal{C}} \mu_A \mathbf{1}\{t \in A\}. \tag{18}$$

The restricted dual minimizes the following convex objective over $\lambda, \boldsymbol{\mu} \ge 0$:

$$F(\lambda, \boldsymbol{\mu}) = \log\left(\sum_{t=1}^L w_t^0 e^{-h_t}\right) + \lambda b_\kappa + \sum_{A\in\mathcal{C}} \mu_A m_{|A|}. \tag{19}$$

The resulting weights are recovered by normalization:

$$w_t(\lambda, \boldsymbol{\mu}) = \frac{w_t^0 e^{-h_t}}{\sum_{j=1}^L w_j^0 e^{-h_j}}. \tag{20}$$

Algorithm 1 incorporates this solve into training and adds violated subset bounds as needed. Each pass adds a violated subset constraint when needed. A restricted optimum that satisfies all cumulative bounds is also optimal for the full problem.

Numerical checks. Evaluate the logarithmic sum in F with log-sum-exp and check all constraint residuals. For a feasible candidate, $D_{\mathrm{KL}}(\boldsymbol{w}\|\boldsymbol{w}^0) + F(\lambda, \boldsymbol{\mu})$ is a primal–dual upper bound on suboptimality. Separation costs $O(L \log L)$ per pass; total cost also depends on the restricted solves and number of passes.

D.3 Local meaning of the interference score

The square-root embedding in Eq. (6) places each distribution on the positive unit sphere. The spherical logarithmic map is defined in Eq. (7).

At one response position, write $d(\boldsymbol{s}, \boldsymbol{b}) = \arccos(\boldsymbol{s}^\top \boldsymbol{b})$. For a spherical curve $\xi(a)$ with $\xi(0) = \boldsymbol{s}$ and tangent $\xi'(0) = \boldsymbol{u}$, differentiation gives:

$$\frac{d}{da}\frac{1}{2} d(\xi(a), \boldsymbol{b})^2\Big|_{a=0} = -\frac{\phi_b}{\sin\phi_b}\langle \boldsymbol{b}, \boldsymbol{u}\rangle = -\langle \boldsymbol{v}, \boldsymbol{u}\rangle, \qquad \phi_b = d(\boldsymbol{s}, \boldsymbol{b}). \tag{21}$$

The expression extends continuously to $\boldsymbol{s} = \boldsymbol{b}$. Thus a negative inner product increases squared anchor distance to first order along the target direction. Cauchy–Schwarz and $2ab \le a^2 + b^2$ give $0 \le \rho \le 1$.

Behavior between anchor and target. Fix distinct $\boldsymbol{b}, \boldsymbol{q}$ and let $\boldsymbol{s}$ lie a fraction $\alpha \in [0,1]$ along their shortest spherical geodesic. Let $\ell = d(\boldsymbol{b}, \boldsymbol{q}) > 0$ and let $\boldsymbol{e}$ be the unit tangent pointing toward $\boldsymbol{q}$. For $0 < \alpha < 1$, $\boldsymbol{u} = (1-\alpha)\ell\boldsymbol{e}$ and $\boldsymbol{v} = -\alpha\ell\boldsymbol{e}$, giving

$$\rho(\alpha) = \frac{2\alpha(1-\alpha)}{\alpha^2 + (1-\alpha)^2}. \tag{22}$$

The score is zero at either endpoint and reaches one at the midpoint, where the opposing directions have equal length. The total distance ℓ cancels: a larger target–anchor separation does not alone imply a larger normalized conflict. Along this fixed-endpoint path, the score rises and falls rather than monotonically tracking progress toward the target. Its dependence on the current student represents local tension between target-directed movement and anchor-directed movement.

Relation to the training update. At $\boldsymbol{s} = \boldsymbol{b}$, the anchor direction is zero, so the first-order score does not prevent an initial departure from the anchor. The spherical target direction need not equal the direction induced by the reverse-KL loss in parameter space. Shared model parameters also couple changes across response positions. Accordingly, the budget bounds the specified distribution-space score; it does not guarantee parameter-gradient compatibility or preservation of benchmark performance. The reference-derived g_t and student-dependent ρ_t have distinct roles, and neither the formulas nor this geodesic calculation require their correlation to have a particular sign.

D.4 Projection under positive guidance scores

For strictly positive reference distributions, Eq. (4) gives $g_t = 0$ if and only if $\boldsymbol{P}_t = \boldsymbol{R}_t$ over the entire vocabulary: zero mean absolute residual requires a constant log-ratio, and probability normalization makes that constant zero. Thus similar distributions or identical most-probable tokens do not imply a zero score. We analyze the nondegenerate setting $\boldsymbol{P}_t \neq \boldsymbol{R}_t$ at each effective response position. Consequently, $g_t > 0$ and the unsmoothed allocation in Eq. (5) is strictly positive; with the default $\gamma = 1$, $w_t^0 = g_t / \sum_j g_j$.

Proposition D.1 (Existence and uniqueness of the projection)**.** *If $w_t^0 > 0$ for all t and $\kappa \in [0,1]$, Eq.* (12) *has a unique finite minimizer.*

Proof. The uniform vector $\boldsymbol{u} = (1/L, \ldots, 1/L)$ is majorized by every simplex vector and satisfies $\boldsymbol{u}^\top \boldsymbol{\rho} \leq b_\kappa$ by Eq. (10), so the feasible set is nonempty. Each sum of the largest k weights is the maximum of finitely many linear subset sums; hence the feasible set is closed and convex, and it is compact as a subset of the simplex. For positive $\boldsymbol{w}^0$, the KL objective is finite, continuous, and strictly convex on the simplex, using $0 \log(0/w_t^0) = 0$. Existence follows from compactness and uniqueness from strict convexity. □

D.5 Concentration guarantees

Proposition D.2 (Concentration consequences of majorization)**.** *For $\boldsymbol{w}, \boldsymbol{w}^0 \in \Delta^{L-1}$ with $\boldsymbol{w} \prec \boldsymbol{w}^0$, define* $\mathrm{ESS}(\boldsymbol{w}) = 1/\sum_t w_t^2$ *and* $H(\boldsymbol{w}) = -\sum_t w_t \log w_t$*, using* $0 \log 0 = 0$*. Then*

$$\mathrm{ESS}(\boldsymbol{w}) \geq \mathrm{ESS}(\boldsymbol{w}^0), \qquad H(\boldsymbol{w}) \geq H(\boldsymbol{w}^0). \tag{23}$$

Moreover, every size-k subset A obeys $\sum_{t \in A} w_t \leq \sum_{j=1}^{k} w_{[j]}^0$.

Proof. Majorization expresses $\boldsymbol{w}$ as a convex combination of permutations of $\boldsymbol{w}^0$. For any convex scalar function f, Jensen's inequality therefore gives $\sum_t f(w_t) \leq \sum_t f(w_t^0)$. Taking $f(x) = x^2$ and $f(x) = x \log x$ proves the ESS and entropy inequalities, respectively. Every size-k subset sum is bounded by the sum of the k largest entries of $\boldsymbol{w}$, which is bounded by majorization. □

These properties prevent increased concentration relative to the initial allocation while permitting its mass to move between positions. They do not require the final allocation to be uniform or to retain its initial ordering. The guarantees concern allocation geometry; benchmark gains from imposing these bounds are a separate empirical question.

E Implementation Details

E.1 Data and Model Initialization

We train Qwen3-4B, Qwen3.5-9B, and Qwen3.5-27B. For each backbone, the student is initialized from the officially released Qwen checkpoints on Hugging Face (e.g., Qwen/Qwen3.5-27B), and a frozen copy supplies the privileged teacher, privilege-only reference, and clean anchor. Only the student LoRA parameters are updated. Both main experiments and ablations use the step-200 checkpoint.

Mathematical reasoning. We use 29,434 OpenThoughts Math examples. The task input x contains the problem statement, and the privileged information z contains the reference answer, formatted as a final-answer block. The student receives only the problem. The frozen model is evaluated under task-plus-answer, answer-only, and task-only contexts to obtain $\boldsymbol{P}_t$, $\boldsymbol{R}_t$, and $\boldsymbol{B}_t$. All evaluations use the same student-generated response prefixes, so the resulting distributions are aligned at each response position.

Student prompts are limited to 2,048 tokens, and overlength prompts are filtered. Training responses are limited to 1,024 tokens, while the privileged prompt-length limit is 18,976 tokens. Tokens generated before the response limit remain eligible for distillation; reaching this limit does not by itself cause the trajectory to be discarded.

Code generation. We use the processed Skywork-Code training set, with the programming task as x and training-only solution or verifier evidence as z. The privileged context presents the evidence in a separate block and instructs the teacher to infer task requirements and edge cases. The student generates from the task-only prompt. The privilege-only reference retains the evidence wrapper without the task, and the clean anchor receives the task without the evidence. As in mathematical training, supervision is evaluated at the student's sampled prefixes rather than independently generated teacher responses.

Code training allows responses of up to 2,048 tokens within an 8,192-token model context. Verifier evidence serves as additional conditioning information; it is not used as an execution reward for policy-gradient training. BFCL v4 evaluates transfer to function calling using the same step-200 checkpoints trained on Skywork-Code, without additional function-calling training.

E.2 Optimization and Sampling

We use AdamW with moment coefficients $(0.9, 0.999)$, zero weight decay, and a linear learning-rate schedule without warmup. LoRA targets linear modules, excludes visual modules, and leaves bias parameters frozen. Table 7 summarizes the shared optimization settings.

Training framework and compute resources. We extend and maintain model-specific LoRA training support within the `verl` framework[1], addressing gradient synchronization issues in distributed training for the evaluated backbones. These changes enable LoRA optimization in our training setup. Updating adapters while keeping the backbone frozen reduces trainable-parameter and optimizer-state memory requirements relative to full-parameter fine-tuning, making the experiments more resource-efficient. The main experimental configurations are trained on two nodes, each equipped with eight NVIDIA H200 GPUs, for a total of 16 GPUs.

Task-specific learning rates and rollout settings are listed in Table 8. Each iteration samples fresh student responses while keeping the reference frozen; no RL policy-gradient objective or external reward model is used.

Thinking modes. Mathematical training follows OPSD (Zhao et al., 2026): student thinking is disabled and privileged-teacher thinking is enabled. For code, both are disabled to maintain a shared program-generation format, while verifier evidence supplies the teacher's additional guidance. This is a design choice rather than a separately validated advantage. Thinking mode controls the prompt

[1] `https://github.com/verl-project/verl`

Table 7: Shared training settings for mathematical reasoning and code generation.

Setting	Value
Optimizer / moment coefficients	AdamW / $(0.9, 0.999)$
Weight decay / warmup	0 / 0 steps
Learning-rate schedule	Linear
LoRA rank / scaling parameter	64 / 128
Prompts per update / responses per prompt	32 / 1
Per-GPU microbatch / passes per minibatch	1 / 1
Training steps / evaluation checkpoint	200 / step 200

Table 8: Task-specific optimization and training-rollout settings.

Setting	Mathematics	Code
Learning rate	5×10^{-6}	10^{-6}
Sampling temperature	1.1	1.6
Top-p / top-k	0.95 / 20	0.8 / 20
Maximum response tokens	1,024	2,048
Student thinking	Disabled	Disabled
Privileged-teacher thinking	Enabled	Disabled

format; the teacher provides next-token distributions without generating a separate reasoning trajectory. These settings are held fixed within each task across the full method and its ablations.

E.3 Target Construction and Loss Computation

Targets and position scores follow Eqs. (3) and (4), respectively, using the same centered residual $\boldsymbol{c}_t$. The target is normalized in log space using log-sum-exp applied to $\log B_t(v) + c_t(v)$. Initial position weights follow Eq. (5) without an additive uniform mixture.

Reverse KL is evaluated over the full vocabulary in chunks of 32 response positions. Prompt and padding tokens are excluded. The sampled responses, target distributions, and allocation weights are held fixed during differentiation, so gradients flow only through the student probabilities. With a token-mean loss reducer, each token receives multiplier Lw_t^* for a response of length L:

$$\frac{1}{L}\sum_{t=1}^{L}(Lw_t^*)D_{\mathrm{KL}}(\boldsymbol{S}_t\|\boldsymbol{Q}_t) = \sum_{t=1}^{L} w_t^* D_{\mathrm{KL}}(\boldsymbol{S}_t\|\boldsymbol{Q}_t). \tag{24}$$

The minibatch loss averages these response-level objectives. This applies the allocation once and prevents response length alone from increasing a sample's total weight.

E.4 Evaluation and Ablation Protocol

Mathematical reasoning. Mathematical evaluation reloads the step-200 checkpoint and samples 12 responses per problem at temperature 1.0 and top-$p = 0.95$, with top-k disabled. The generation limit is 38,912 tokens within a 40,960-token model context, independently of the shorter training-response limit. Student thinking remains disabled. Avg@12 averages binary correctness over the 12 responses to each problem and then across problems; it is not pass@12 or majority-vote accuracy.

Code generation and function calling. We follow the official evaluation protocols for LCB v6 and BFCL v4 (Jain et al., 2025; Patil et al., 2025; Mao et al., 2025b;a). LCB v6 uses execution-based testing to compute pass@1, while BFCL v4 applies category-specific checks and its official score aggregation. Both scores are reported as percentages, and their arithmetic mean gives the code-group average.

Ablation protocol. The training ablations in Q1–Q4 retain the training and evaluation configuration of the corresponding full-method experiment within each task and backbone. Only the parameter or module being examined is changed. Training data, model initialization, LoRA configuration, optimizer, batch sizes, thinking modes, sampling settings, and generation limits remain fixed. Each

training variant is trained for 200 steps and evaluated at step 200, giving the full method and training ablations the same training budget and checkpoint step. Shared endpoint configurations are reused across tables. Q4 fixes $\gamma = 1$ and uses $\kappa = 1$ for the constrained variants; each variant follows its own on-policy trajectory.

Allocation-alignment control. Q3 compares U + Full, G + Full (`TransCAP`), and ΠG + Full, fixing the projection rules and $\kappa = 1$ while varying initial allocation. For ΠG, each response is divided into eight relative-position bins, and the initial G weights are randomly permuted within each bin while tokens and targets remain unchanged. Permutation preserves the initial weight multiset, total mass, entropy, effective sample size, and each bin's total mass. Each configuration is projected relative to its own initial allocation. U remains uniform because majorization by a uniform vector admits only uniform weights, which already satisfy the default risk budget. G and ΠG have the same initial sorted cumulative bounds, but differ in the alignment between weights and conflict scores; their projected allocations need not retain identical weight multisets. The comparison therefore tests initial token-level alignment within the full constrained framework, while preserving both weight shape and coarse positional structure.

Fixed-prefix constraint diagnostic. The separate fixed-prefix diagnostic in Q4 uses 128 mathematical and 128 coding responses generated by a 9B model. At each identical response prefix, student and frozen reference forward passes provide the target, guidance score, conflict score, and nominal weight using the training formulas, response masks, and truncation settings. Targets, conflict vectors, nominal weights, and budgets are shared across G (no projection), G + RiskOnly, G + Box, and G + Full; the projections share the KL objective. Responses are selected before projection, retaining correct, incorrect, truncated, and already-feasible cases. This diagnostic performs forward calculations and allocation solves without retraining the four configurations. Both figures pool all 256 responses; box centers and edges show medians and quartiles. The cumulative distributions report $100E(\boldsymbol{w})$ from Eq. (14) in percentage points. These measurements describe allocation and constraint behavior, rather than benchmark accuracy.

F Constraint Behavior During Training

We complement the fixed-prefix analysis with 200-step diagnostics for Qwen3.5-9B. RiskOnly and Full abbreviate G + RiskOnly and G + Full (`TransCAP`) throughout these diagnostics. Each follows its own student-generated trajectories; each point summarizes a training step across microbatches and devices.

F.1 Risk Control

Figure 4 tracks weighted conflict before and after Full projection, together with the corresponding risk budget. The initial allocation remains above the budget throughout training, whereas the projected mean conflict stays at or below the budget mean. The mean risk-activation fractions across steps are 99.80% for RiskOnly and 99.73% for Full, indicating that risk correction is repeatedly used. Both conflict and budget increase during training because they depend on the evolving distributions and sampled responses. The relevant criterion is therefore compliance with the current budget, rather than a monotonic decrease in absolute conflict.

F.2 Allocation Adjustment

For each response, we measure redistributed mass by $\frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}^0\|_1$ and report it in percentage points. The medians of the per-step mean redistribution are 7.18 points for RiskOnly and 7.25 points for Full (Figure 5). Their similar overall adjustment magnitudes do not imply identical allocations: the same amount of mass can be moved to different positions and satisfy different cumulative bounds. This distinguishes the amount of redistribution from its concentration, complementing the fixed-prefix comparison in Figure 3. The training traces show how much the initial allocation is modified over the course of optimization.

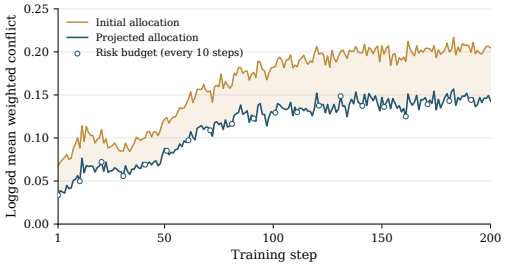

Figure 4: Risk control over 200 training steps. Lines show the initial and projected mean conflict; open markers show the budget every ten steps.

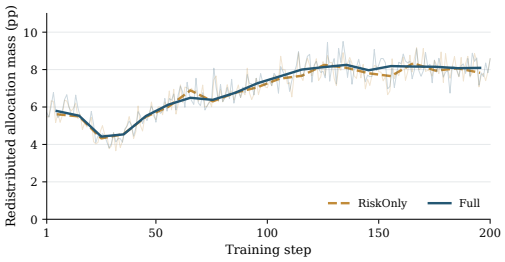

Figure 5: Redistributed allocation mass. Faint lines retain all per-step records; thick lines show medians in non-overlapping ten-step windows.

F.3 Concentration and Projection Divergence

Change in effective sample size. For normalized allocation weights, $\mathrm{ESS}(\boldsymbol{w}) = 1/\sum_t w_t^2$. We measure the change in ESS/L before and after projection on each response and summarize the per-step means (Figure 6). A positive change indicates a less concentrated allocation. The median per-step increase is 1.73 percentage points for Full and 0.49 for RiskOnly. Together with the similar redistributed mass reported above, these observations distinguish how much weight is moved from how broadly it is distributed. The comparison describes the two on-policy training trajectories; the fixed-prefix experiment in the main text isolates the constraint mechanisms on shared inputs.

Allocation divergence and risk reduction. Figure 7 relates the mean projection divergence, $\mathrm{KL}(\boldsymbol{w}\|\boldsymbol{w}^0)$, to the reduction in mean weighted conflict within each training step. The cross-step median KL values are 0.0206 for Full and 0.0198 for RiskOnly, with median conflict reductions of 5.13 and 5.07 percentage points, respectively. The overlapping distributions indicate comparable overall adjustment and risk-reduction scales in these runs. Thus, the additional cumulative constraints primarily distinguish allocation concentration in these diagnostics, rather than producing a clearly separated level of risk reduction. KL here measures the change in allocation, not computational overhead.

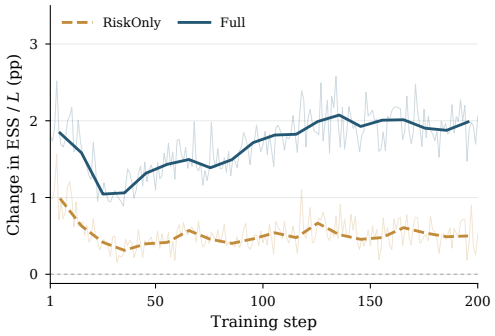

Figure 6: Change in normalized ESS after projection. Faint traces show all 200 step means; thick traces show medians in non-overlapping ten-step windows.

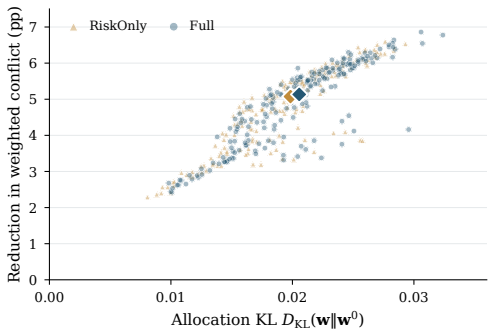

Figure 7: Allocation divergence and conflict reduction. Each point represents one training step; diamonds mark coordinate-wise medians over 200 steps.

F.4 Periodic Evaluation of Full

Figure 8 evaluates Full on AIME 2025 every 25 steps, with 12 responses per problem across 30 problems. Avg@12 is the fraction correct among these 360 responses. Step 0 uses Raw Base

(63.06%, 227/360). Performance rises to 77.22% at step 50 and reaches 81.11% (292/360) at step 150, remaining unchanged at steps 175 and 200. The curve shows rapid early improvement followed by smaller gains and a plateau over the final three evaluations.

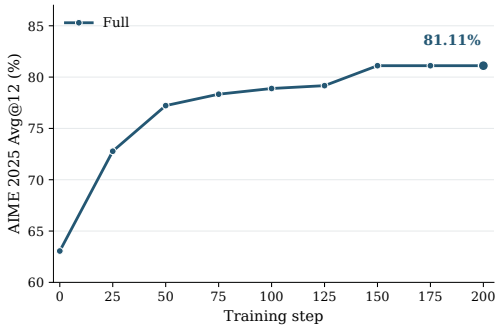

Figure 8: Periodic AIME 2025 evaluation of Full (Avg@12). Step 0 is the Raw Base reference; later markers show evaluation checkpoints.

G Computational Cost

We report constraint-module latency and training-stage timings for the runs in Appendix F, distinguishing projection costs from generation and optimization.

G.1 Conflict Computation and Projection

The timer includes conflict computation, projection, and associated diagnostics, with CUDA synchronization around the call. Figure 9 summarizes all 200 steps: median latency is 47.2 ms for RiskOnly and 148.8 ms for Full, reflecting the additional computation for cumulative constraints. Values are averaged across microbatches and devices; they measure module latency, not added time per training step or end-to-end overhead.

G.2 Training-Stage Timings

Figure 10 summarizes 191 Full steps, excluding the first step and checkpoint-saving steps at multiples of 25. Median times are 32.5 s for rollout generation, 12.8 s for actor log-probability computation, 45.9 s for the actor update, and 13.6 s for weight synchronization. Actor updates and rollouts dominate these measured stages. Stage medians are reported separately and are not summed to estimate total-step latency.

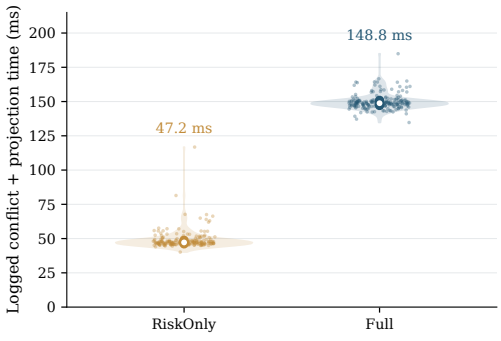

Figure 9: Conflict and projection time. White markers and bars show medians and interquartile ranges.

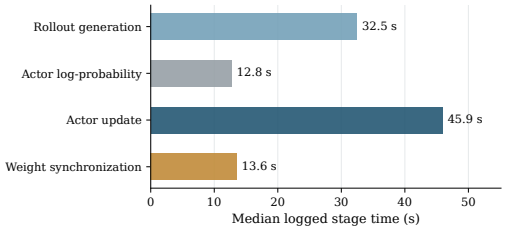

Figure 10: Median training-stage timings for Full over 191 steps.