WHEN DOES FINER TEACHER ROUTING HELP? PER-TOKEN CAPABILITY ROUTING IN ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation from several domain specialists raises a choice a fixed corpus never poses: which teacher supervises each token of the student's own rollout. The standard recipe routes each whole rollout to one teacher by its domain tag, yet rollouts are rarely pure, so one label is wrong on part of the sequence. We introduce CAPABILITY-OPD, which instead assigns a teacher to every *token* by vocabulary identity: offline, each token id goes to the specialist that most improves its next-token distribution over a shared initialization, and the student distills on-policy against that per-token teacher, changing only the granularity of teacher assignment. The benefit is conditional. On fully mixed data, per-token routing beats each single-teacher baseline on its weak axis, by 1.67 points on code and 1.49 on mathematics; the gain falls as rollouts grow purer, reverses to a 1.42-point loss on single-domain data, and vanishes when the student is small, whether a 4B dense model or a 30B mixture-of-experts that activates only 3B parameters per token. Both conditions can be checked in advance: a cheap test on comparisons a practitioner already trains predicts, before the finer router is built, whether per-token routing can help.

1 INTRODUCTION

On-policy distillation trains a student on its own rollouts, with a teacher scoring the tokens the student actually generates (Agarwal et al., 2023; Gu et al., 2023; Song & Zheng, 2026). It removes the exposure bias of learning from a fixed corpus and supplies a dense per-token signal. When one model must hold several capabilities at once, the natural extension is to distill from several domain specialists rather than a single teacher. This forces a choice a fixed corpus never poses: at each of the student's tokens, *which* teacher supplies the target?

The established answer, used in recent multi-teacher systems and production model reports (Ma et al., 2026; NVIDIA et al., 2026), routes one teacher per rollout, selected by the rollout's domain tag. This PER-SAMPLE recipe is simple and lets specialists be developed independently, but it rests on an assumption that is false for any mixed rollout: that one domain label describes the whole sequence. A competitive-programming solution contains natural-language reasoning; a derivation contains symbolic manipulation. Wherever two capabilities interleave within a rollout, a single per-sample label is wrong on part of the sequence, and the obvious refinement is to route *per token*. Figure 1 contrasts the two schemes on one mixed rollout.

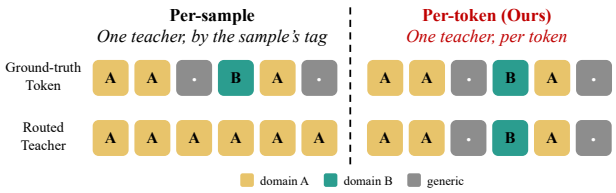

Figure 1: **Per-sample vs. per-token routing** on one mixed rollout. Per-sample routing follows the sample's tag, so the domain-B token is supervised by teacher A; per-token routing follows the token.

Whether that refinement helps has not been tested with granularity as the *only* changed variable: finer supervision schemes tend to also change the routing signal, the teachers, or the objective, so a gain cannot be attributed to granularity itself. We introduce CAPABILITY-OPD (Capability-Routed On-

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.

Policy Distillation), which assigns a teacher to every token by vocabulary identity, computed offline from how much each specialist improves each token id's next-token distribution over the shared student initialization (§2). Holding teachers, data, optimizer, and budget fixed, CAPABILITY-OPD changes only the granularity of teacher assignment, which isolates its effect.

Under that controlled comparison the benefit is *conditional*, and the condition is predictable in advance. On data that mixes domains within a rollout, per-token routing beats each single-teacher baseline on its weak axis, by 1.67 points on code and 1.49 on mathematics, while the two single-teacher baselines themselves pull apart in opposite directions, gaining 2.68 points on code but losing 1.06 on mathematics, so no single per-rollout label can be right on both axes. The gain tracks how much a rollout actually mixes domains: it rises monotonically with the mixed fraction at a rank correlation of 1.00, and reverses to a 1.42-point loss on single-domain data, where the per-sample label is already right. It also requires student capacity: at 4B the two single-teacher baselines still split by 1.84 points, yet per-token routing no longer converts that signal into a gain, and a 30B mixture-of-experts that activates only 3B parameters per token behaves like the small-capacity regime. Averaged over all seven benchmarks it washes out to a non-significant 0.68 points, consistent with a benefit local to one axis.

Read together, these results replace "is finer routing better?" with "*when*": per-token teacher routing is worth its cost when the training data mixes capabilities within a sample and the student is large enough to use the distinction; a necessary condition can be checked before paying for the finer router. Our contributions are:

- **CAPABILITY-OPD, per-token capability routing.** We assign a teacher to every token by an offline vocabulary-identity attribution and distill on-policy against it (§2), changing only the granularity of the standard per-sample multi-teacher recipe. To our knowledge this is the first attribution of a per-token routing gain to granularity alone.
- **When per-token routing helps.** The advantage scales with within-sample domain mixing, reversing on single-domain data, and requires student capacity: detected at 7B, not at 4B or on the 30B-A3B MoE. It sits on one axis, so pooling hides it (§4).
- **A cheap screening test.** A split between the two single-teacher baselines, which a practitioner trains anyway, is a necessary condition for any routing scheme to help, checkable before the finer router is built (§5).

2 CAPABILITY-ROUTED ON-POLICY DISTILLATION

We fix notation and recall on-policy distillation and its standard multi-teacher form (§2.1), then introduce CAPABILITY-OPD, which refines the teacher assignment from one teacher per rollout to one teacher per token (§2.2).

2.1 PRELIMINARIES

On-policy distillation. Let π_0 be a shared student initialization and π_θ the student. Given a teacher π_T, on-policy distillation (Agarwal et al., 2023; Gu et al., 2023; Song & Zheng, 2026) draws a rollout $o = (o_1, \ldots, o_{|o|})$ from the *student* for a prompt $x \sim \mathcal{D}$ and matches the teacher's next-token distribution at every position the student visits:

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, o\sim\pi_\theta(\cdot|x)}\Big[\tfrac{1}{|o|}\textstyle\sum_{t=1}^{|o|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x, o_{<t}) \,\big\|\, \pi_T(\cdot \mid x, o_{<t})\big)\Big]. \tag{1}$$

Sampling o from the student rather than a fixed corpus removes exposure bias: the teacher supervises exactly the states the student reaches at deployment. The consequence that matters for the multi-teacher setting is that the target teacher must be *resolved at every position the student visits*, a choice a fixed offline corpus never surfaces.

Per-sample multi-teacher OPD. To place several capabilities in one model, the standard recipe (Ma et al., 2026; NVIDIA et al., 2026) trains K domain specialists $\{\pi_1, \ldots, \pi_K\}$, each obtained from π_0 by reinforcement learning on a single domain, and distills all of them into the student on its own rollouts. Since a rollout carries one domain tag $d(x) \in \{1, \ldots, K\}$, namely the domain of its

prompt, the entire rollout is routed to that one teacher:

$$\mathcal{L}_{\text{PER-SAMPLE}}(\theta) = \mathbb{E}_{x,\, o\sim\pi_\theta(\cdot|x)}\Big[\tfrac{1}{|o|}\textstyle\sum_{t=1}^{|o|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x, o_{<t}) \,\big\|\, \pi_{d(x)}(\cdot \mid x, o_{<t})\big)\Big]. \tag{2}$$

This keeps OPD's dense signal and its freedom from exposure bias, and lets each specialist be developed independently. Its assumption is that one domain label describes the whole sequence, which is false for any rollout that mixes capabilities: a competitive-programming solution contains natural-language reasoning and a derivation contains symbolic manipulation, so $\pi_{d(x)}$ is the wrong teacher on part of the sequence. Closing that gap is what a finer teacher assignment is for.

2.2 CAPABILITY-OPD: ROUTING TEACHERS PER TOKEN

CAPABILITY-OPD holds every element of Eq. (2) fixed (the same specialists, data, optimizer, rollout sampler, and step budget) and changes only the granularity at which the teacher is resolved, from one teacher per rollout to one teacher per token (Figure 2).

Token–teacher attribution. Teachers are assigned by *vocabulary identity*, computed once and offline before any student is trained. Because all specialists descend from the same π_0 by domain RL, their per-token log-probabilities are commensurable against that common reference; Ma et al. (2026) report this shared initialization as necessary for stable distillation. For a token id $v \in \mathcal{V}$, let $\mathcal{O}(v)$ collect its occurrences as the gold next token in a held-out reference corpus, and define specialist k's *gold-anchored lift* on v as the mean positive part of its log-probability improvement over π_0:

$$\ell_k(v) = \frac{1}{|\mathcal{O}(v)|} \sum_{(x,\, o_{<t})\in\mathcal{O}(v)} \max\big(0,\ \log \pi_k(v \mid x, o_{<t}) - \log \pi_0(v \mid x, o_{<t})\big). \tag{3}$$

Each id is assigned to its highest-lift specialist, subject to an exclusivity cutoff τ on its share of the total lift and a minimum count $n_{\min}$; ids failing either condition fall back to π_0 itself (self-distillation):

$$A(v) = \begin{cases} \arg\max_k\ \ell_k(v), & \text{if } |\mathcal{O}(v)| \geq n_{\min} \text{ and } \max_k \ell_k(v) \big/ \textstyle\sum_j \ell_j(v) \geq \tau, \\ \pi_0, & \text{otherwise.} \end{cases} \tag{4}$$

The result is a static table $A : \mathcal{V} \to \{\pi_1, \ldots, \pi_K, \pi_0\}$, a property of the teacher pair rather than of any training run, recomputed only when the specialists change. It costs one forward pass of π_0 and of each specialist over the reference corpus and adds no online gate, no auxiliary head, and no change to the training loop beyond an indexed lookup. In our main configuration $\tau{=}0$ and $n_{\min}{=}0$, so every id is routed to its higher-lift specialist and the fallback never fires; ids with zero lift under both specialists, including ids absent from the reference corpus, default to the first (mathematics) specialist. No pruned setting significantly improves on $\tau{=}0$ (Appendix B); the value was fixed before the sweep, not tuned.

Per-token objective. At training time the table is applied position by position to the student's own rollout: the token at position t has id o_t, and its target is $\pi_{A(o_t)}$. CAPABILITY-OPD optimizes

$$\mathcal{L}_{\text{CAPABILITY-OPD}}(\theta) = \mathbb{E}_{x,\, o\sim\pi_\theta(\cdot|x)}\Big[\tfrac{1}{|o|}\textstyle\sum_{t=1}^{|o|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x, o_{<t}) \,\big\|\, \pi_{A(o_t)}(\cdot \mid x, o_{<t})\big)\Big], \tag{5}$$

identical to Eq. (2) except that the per-rollout teacher $\pi_{d(x)}$ is replaced by the per-token teacher $\pi_{A(o_t)}$. On a mixed rollout the code region is supervised by the code specialist and the mathematics region by the mathematics specialist, whereas Eq. (2) commits the whole rollout to one of them. Granularity is the only difference between the two objectives, so any measured gap is attributable to it rather than to a changed teacher, data, or signal. The full procedure appears as Algorithm 1 (Appendix L).

Cost relative to per-sample routing. The one-time offline attribution is K forward passes over the reference corpus, recomputed only when the teacher pair changes. At training time the sole difference from per-sample is that resolving a per-token target needs each teacher's log-probability at the positions routed to it, so every rollout is scored by all K frozen teachers rather than by the one tagged teacher; for our two teachers this is one extra teacher forward pass per rollout. The teachers are inference-only and rollout generation and the optimizer step are unchanged, so the overhead is bounded by the teacher-serving share of a step.

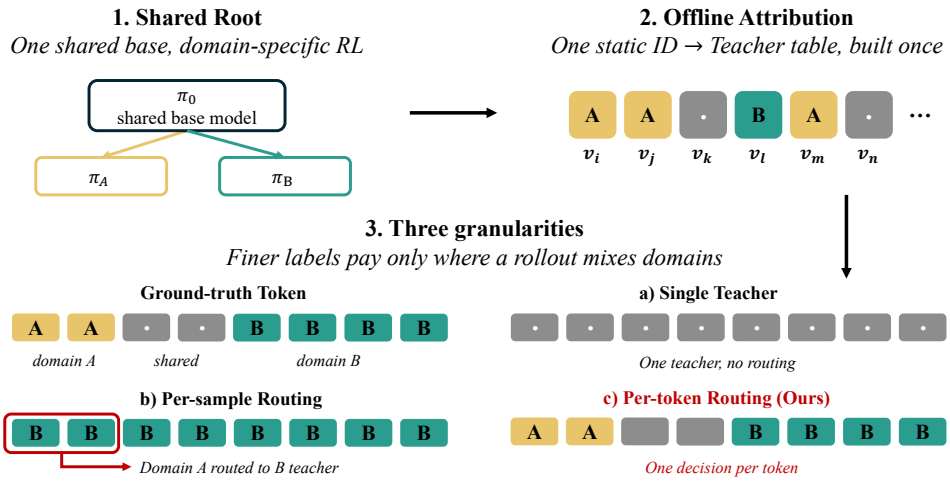

Figure 2: **CAPABILITY-OPD in three steps.** (1) Both specialists are trained from the shared root π_0 by domain RL. (2) Offline, each vocabulary id is assigned to the specialist with the larger gold-anchored lift (Eq. (3)), giving one static id→teacher table. (3) The granularities compared: one teacher for all tokens, one teacher per rollout by its domain tag, and one teacher per token from the table.

Teacher lines. The main results use $\pi_0 \rightarrow$ domain reinforcement learning, the one recipe whose two specialists each lead on their own domain, the teacher-quality condition of §4; on our instruct root, a supervised-finetuning line failed that condition and was not carried to a student. The attribution table is recomputed per line, and App. C reports the alternative lines and their scope.

3 EXPERIMENTS

3.1 SETUP

Models and training. The main comparison distills into a Qwen2.5-7B-Instruct student π_0. Two specialists are obtained from π_0 by GRPO, one on the mathematics set DAPO-Math and one on the competitive-programming set CodeContests, and the token–teacher attribution table of Eq. (4) is computed over a held-out reference corpus. Every method then distills on-policy from these frozen specialists into a fresh copy of π_0 under byte-identical data, optimizer, rollout sampler, and step budget, so methods differ *only* in which teacher supervises each token; the reverse-KL objective and full hyperparameters are in Appendix B. To test the size boundary we repeat the comparison at a smaller dense scale, Qwen3-4B-Instruct, and on a Qwen3-30B-A3B mixture-of-experts with 128 experts, top-8 routing, and roughly 3B active parameters per token.

Mixing math and code in one sample. Per-token routing can matter only when the two capabilities meet inside one rollout, so each training sequence concatenates a mathematics segment and a competitive-programming segment; any single per-sample tag is then correct on one segment and wrong on the other. A family of runs sweeps the mixing fraction f from 0 (each sequence single-domain, so the per-sample tag is never wrong) to 1 (the two domains fully interleaved), holding item count, optimizer, and step budget fixed.

Methods. Beside CAPABILITY-OPD (PER-TOKEN), four methods vary the granularity and correctness of the teacher assignment (Table 1). PER-SAMPLE is the standard per-sample recipe of Eq. (2); MATH-ONLY and CODE-ONLY force one domain label on the whole mixed sequence, so their contrast tests whether the two teachers disagree at all; RANDOM-LABEL randomizes the domain tag at matched per-teacher token share, removing the routing signal while preserving its marginal statistics.

Table 1: The comparison methods. All share π_0, both specialists, and the full training configuration, differing only in which teacher supplies each token's target.

Method	Granularity	Teacher assignment	Role
PER-TOKEN (CAPABILITY-OPD)	per token	identity table (Eq. 4)	our method
PER-SAMPLE	per rollout	sample domain tag	MOPD baseline
MATH-ONLY	per rollout	math label, all tokens	granularity contrast
CODE-ONLY	per rollout	code label, all tokens	granularity contrast
RANDOM-LABEL	per rollout	random tag, matched share	signal removed

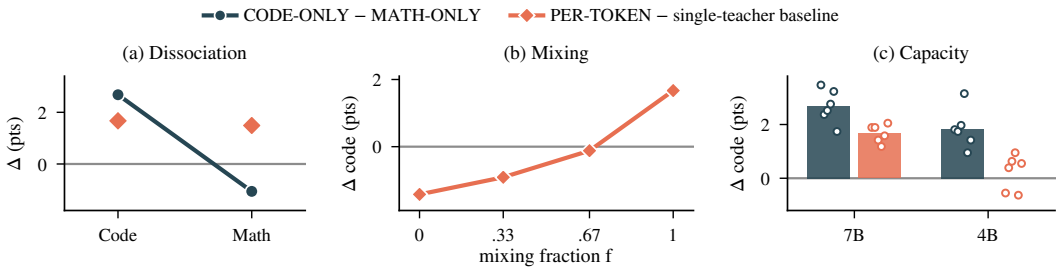

Figure 3: **Paired differences on mixed data** (six-seed pool, points). (a) The two single-teacher baselines pull apart in opposite directions; CAPABILITY-OPD beats the baseline that is wrong on each axis. (b) The per-token code gain over MATH-ONLY grows with the mixed fraction f. (c) Both halves are positive at 7B; at 4B the per-token seeds (circles) straddle zero, so no pooled bar is drawn. Exact values and tests in App. H.

Evaluation panel. We evaluate on seven benchmarks spanning 3828 items, detailed in Appendix G. Three are mathematics: a held-out split of the training distribution, MATH-500 (Lightman et al., 2023), and GSM8K (Cobbe et al., 2021). Three are code: HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LiveCodeBench (Jain et al., 2024). The seventh is instruction-following, IFEval (Zhou et al., 2023), kept as a regression check. BigCodeBench (Zhuo et al., 2024) is held out as a second low-baseline code benchmark; because the four code benchmarks do not agree on method ordering, code summaries are reported against all of them. Mathematics is scored by a format-robust answer matcher, code by execution against each benchmark's test suite, and instruction-following by strict constraint satisfaction.

Evaluation and statistics. Paired method comparisons use McNemar's test on the item-level agreement table; comparison families were fixed in advance and Bonferroni-corrected within each, and the main comparison pools six seeds. A non-significant difference is reported as the 95% confidence-interval upper bound rather than as evidence of equivalence. Because answer extraction is conservative and some benchmarks cap long generations, absolute scores compare only within one family and harness; every reported effect is therefore a paired within-family difference, never a cross-family or cross-size score gap.

4 RESULTS

Main results use the reinforcement-learning teacher line, the only recipe whose specialists each lead on their own domain on our root (App. C). Code and math contrasts each pool three benchmarks in a six-seed McNemar test, corrected for multiplicity, under byte-identical decoding.

4.1 PER-TOKEN ROUTING WINS EACH BASELINE'S WEAK AXIS

The per-sample recipe of Eq. (2), the MOPD recipe of Ma et al. (2026), must commit each rollout to one teacher, so its two instantiations each lag on the axis their teacher does not serve: at 7B, MATH-

Table 2: **Main results on data that mixes math and code within each sample** (accuracy). MATH-ONLY and CODE-ONLY are the per-sample MOPD recipe (Ma et al., 2026) with its math or code teacher; boxed cells are the axis that teacher does not serve. Dense students: six-seed mean, CodeContests as code source (†: AceCode instead, same π_0). MoE student: one run at step 100, GSM8K not evaluated. Avg: mean of the six columns (‡: of the five evaluated). Bold: best trained method per block.

	Mathematics			Code			
Method	MathVal	MATH-500	GSM8K	HumanEval	MBPP	LCB	Avg
Qwen2.5-7B-Instruct							
π_0	21.00	76.20	90.67	78.66	70.67	26.34	60.59
MATH-ONLY	**27.08**	78.00	91.09	**81.30**	**70.44**	25.06	62.16
CODE-ONLY	22.25	77.97	90.21	78.46	69.87	**30.66**	61.57
PER-TOKEN (ours)	26.75	**78.87**	**91.47**	80.59	70.13	28.28	**62.68**
MATH-ONLY†	**26.92**	**78.37**	**91.60**	**80.49**	**70.18**	25.31	62.14
CODE-ONLY†	22.00	76.83	90.85	78.35	69.82	**30.89**	61.46
PER-TOKEN (ours)†	25.67	78.20	91.23	79.78	69.38	28.88	**62.19**
Qwen3-4B-Instruct							
MATH-ONLY	54.75	**87.07**	**92.05**	83.64	68.04	47.44	72.17
CODE-ONLY	51.75	87.03	91.66	**84.65**	**68.27**	**50.30**	72.28
PER-TOKEN (ours)	**56.17**	86.77	91.90	83.84	67.69	47.97	**72.39**
Qwen3-30B-A3B-Instruct (MoE)							
Single mixed teacher	57.00	**90.80**	–	90.24	**77.07**	**58.98**	**74.82**‡
PER-SAMPLE	**58.00**	90.40	–	**91.46**	76.27	56.93	74.61‡
PER-TOKEN (ours)	57.50	87.00	–	90.85	75.47	55.97	73.36‡

ONLY drops below π_0 to 25.06 on LiveCodeBench and CODE-ONLY lags at 22.25 on MathVal. CAPABILITY-OPD routes the same teachers per token and, at 7B with either code source, lifts each baseline's weak axis; with CodeContests it takes the highest six-benchmark mean, 62.68, leading MATH-500 and GSM8K. Instruction-following does not separate from MATH-ONLY (73.20 against 73.97, n.s.).

Neither single teacher can serve both axes at once. The code-only router CODE-ONLY beats the math-only router MATH-ONLY on code by 2.68 points but loses to it on math by 1.06 points, all six seeds agreeing in each direction ($p{=}1.2\times10^{-5}$): each single teacher helps its own axis and hurts the other (Fig. 3a). Without being told the domain, CAPABILITY-OPD beats MATH-ONLY on code by 1.67 points and CODE-ONLY on math by 1.49 points (both $p < 10^{-3}$, six seeds). It does not reach the better teacher on the other cells: on math it does not separate from MATH-ONLY ($+0.43$, n.s.), and on code it recovers 1.67 of the 2.68-point gap but stays 1.01 points below CODE-ONLY ($p_{\text{Bonf}}{=}0.010$). The code gain is carried by LiveCodeBench, where MATH-ONLY falls below π_0 (Table 2), and keeps its sign on held-out BigCodeBench. With AceCode as the code source the pattern reproduces: the baselines split by 2.83 points on code and 1.35 in reverse on math, CAPABILITY-OPD gains 1.72 points on code and 0.95 on math over the baseline that is wrong on each axis, and stays 1.10 points below CODE-ONLY on code (App. F).

The gain is per-axis, not aggregate: pooled over all seven benchmarks CAPABILITY-OPD's lead over MATH-ONLY dilutes to 0.68 points (n.s., $[-0.14, +1.49]$; App. I). Distillation works end to end ($+1.15$ over π_0), while on that pooled panel CAPABILITY-OPD and a random-label control matched on teacher shares do not separate (-0.39 points, n.s.).

Finding. The single-teacher baselines pull apart (2.68 points on code, 1.06 the other way on math), so no per-rollout label is right everywhere; CAPABILITY-OPD beats each on its weak axis (1.67 on code, 1.49 on math) and still trails CODE-ONLY on code by 1.01.

Table 3: **The routing gain has a capacity floor.** Paired McNemar differences, three code benchmarks pooled. At 7B/4B PER-TOKEN is compared with the single-teacher method whose teacher does not serve that axis (MATH-ONLY on code, CODE-ONLY on math), at 30B with PER-SAMPLE. Dash: single-teacher baselines not trained. $^{*}p < 0.05$, $^{**}p < 10^{-3}$, family-corrected. Students: Qwen2.5-7B-Instruct (Qwen et al., 2024), Qwen3-4B-Instruct and Qwen3-30B-A3B-Instruct (Yang et al., 2025).

Student	Teachers differ? (CODE-ONLY−MATH-ONLY, code)	Per-token routing gain (PER-TOKEN vs. per-sample, code)	Verdict
7B (Qwen2.5)	$+2.68^{**}$	$+1.67^{**}$ ($+1.49^{**}$ math)	detected
4B (Qwen3)	$+1.84^{**}$	signs disagree, n.s. ($+0.53^{*}$ math)	not detected
30B-A3B (MoE)	—	-0.87 n.s. $[-2.54, +0.80]$	not detected

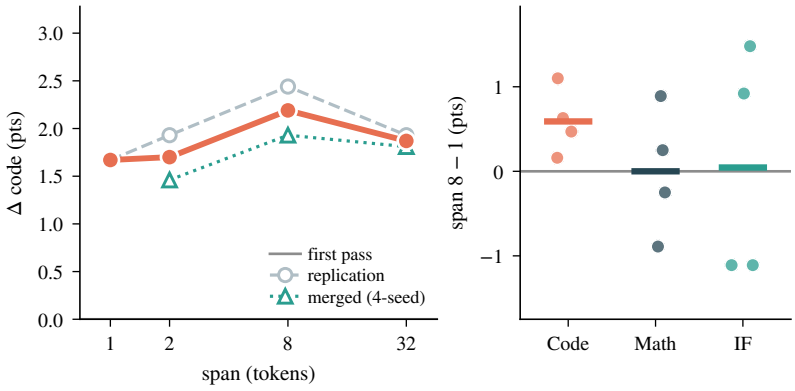

Figure 4: **Coarser routing spans** (f=1, teacher budgets fixed). *Left:* code gain of PER-TOKEN over MATH-ONLY by span length (six seeds at span 1, four at coarser spans). *Right:* span 8 minus span 1 per axis.

4.2 WHEN PER-TOKEN ROUTING HELPS: DOMAIN MIXING AND STUDENT CAPACITY

Domain mixing. If a per-rollout label is wrong only *inside* a mixed sample, the per-token advantage should grow with how much a sample mixes domains and vanish when it does not. Holding item count fixed and varying only the fraction f of each sample that mixes the two domains, the code advantage of CAPABILITY-OPD over MATH-ONLY rises monotonically from a 1.42-point loss at f=0 to a 1.67-point gain at f=1 (Spearman $\rho = +1.00$; Fig. 3b). On single-domain data the sign flips: a few generic tokens in a pure sample are routed to the other teacher while the per-rollout label, already right about the whole sequence, has nothing to lose.

Student capacity. The dissociation has two halves that scale differently (Fig. 3c, Table 3). At 4B (Qwen3-4B-Instruct) the *teacher-quality* half survives: the two baselines still split on code by 1.84 points, all six seeds the same sign ($p_{\text{Bonf}} < 10^{-3}$). The *routing* half does not: CAPABILITY-OPD's lead over MATH-ONLY changes sign across seeds (-0.63 to $+0.95$), none significant, and its code accuracy trails CODE-ONLY (58.44 against 60.05). The same holds for a sparse mixture-of-experts student (Qwen3-30B-A3B-Instruct, about 3B of 30B parameters active per token, the flagship student of Ma et al. (2026)): per-token routing, per-sample routing, and a single mixed teacher do not separate on the pooled code axis (Table 3, App. K), consistent with the $\leq$4B dense regime where the gain is not detected either.

Routing granularity. Per-token and per-sample routing are the two ends of one granularity axis. To probe the points in between, we coarsen the per-token assignment into contiguous spans of 2, 8, and 32 tokens on fully mixed data, holding each teacher's token budget fixed. The code gain over MATH-ONLY stays positive throughout, $+1.67/+1.70/+2.19/+1.87$ points at spans $1/2/8/32$ (Fig. 4). Span 8 beats span 1 by only 0.59 points (p=0.0908), so per token remains the operating

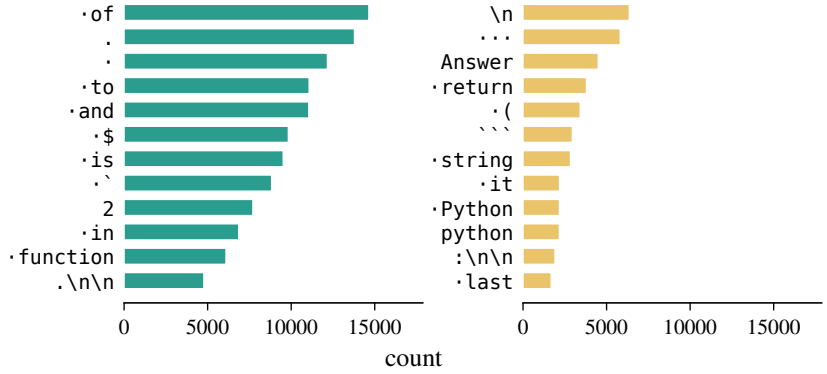

Figure 5: **The most-routed assignments are format and function tokens on both sides.** The twelve most frequent specialist assignments under the inspection gate (τ=0.6, m_c=20), by count in the held-out measurement corpus; a middle dot marks a space inside the token. The panels share one scale, so the code teacher's shorter bars show it receives less routed mass.

point. What breaks the gain is budget, not precision: a majority-vote owner, which lets the denser label win each block and so shifts tokens away from the code teacher, cuts it to $+0.47$ points (n.s.) at span 2 (App. E).

Further checks. A third, instruction-following teacher yields no per-token gain on its axis, as vocabulary-identity routing exposes few such tokens to it; in that configuration per-token routing falls below per-sample routing on math (-1.06) and code (-3.63; App. M).

Finding. Finer routing pays off only where the per-rollout label is wrong and the student can use the difference: the code lead over MATH-ONLY climbs from a 1.42-point loss at f=0 to a 1.67-point gain at f=1, detected at 7B but not at 4B or on the 30B-A3B MoE.

5 WHY THE BENEFIT IS BOUNDED, AND HOW TO SCREEN FOR IT

The gain is small, and its size follows from three measured quantities (full accounting in App. J, Table 11). The token-to-teacher table is reproducible: recomputed on disjoint halves of the data it agrees 29.4% above chance, but it is a property of one teacher pair on one family (changing the pair leaves 3.9%, changing the family 2.6%), so it must be rebuilt per deployment. What the table can buy is bounded: an oracle assignment lifts the gold-token log-probability the router optimizes by only 5.1 to 15.7% relative over the single stronger teacher, and split-half honest estimation halves that to 2.4 to 5.4%, since 32 to 58% of the apparent headroom is arg-max selection bias. Even that residue is not fully realized: a shared fallback over generic tokens puts the gated router 9 to 49% below per-sample routing in the same unit, and the per-token owner switches teachers every 1.8 to 3.0 tokens.

What the router owns. The routed mass is generic, not domain vocabulary. By token position the table splits 61.6%/38.4% between the mathematics and code teachers. On both sides the highest-mass assignments are punctuation, whitespace, and function words, and the domain vocabulary that is routed crosses both ways (Fig. 5; per-id counts in App. B, Fig. 6). This is why coarsening the routing unit barely moves the score (Fig. 4): the mass the router controls is scaffolding both teachers share, and the rare ids that separate them carry little of it.

A necessary-condition test. These bounds have a practical consequence: the condition under which finer routing *can* help is checkable from methods one trains anyway, before committing to the per-token machinery (App. J, Fig. 11).

Proposition 1 (Screen before you route)**.** *Per-token routing can improve over the per-sample baseline only if a single per-rollout label serves some axis wrong, that is, the two single-teacher baselines split across benchmarks. If this check fails, no routing scheme, coarse or fine, has a gap to close.*

The check is cheap and fired for us: the two single-teacher baselines split (2.68 points on code, -1.06 on math; §4.1). It is necessary, not sufficient: at 4B the baselines still split, yet the per-token gain is not detected (§4.2); on the single-domain end of the mixing sweep, where the per-sample label is never wrong, per-token routing loses 1.42 points (§4.2).

6 Related work

Multi-teacher on-policy distillation. Routing a whole rollout to one teacher by its domain tag is the established recipe and our baseline (Ma et al., 2026; NVIDIA et al., 2026; Song & Zheng, 2026). That line documents the failure modes finer routing targets: teachers counteract when mixed (Chen et al., 2026a) and capability recovery is uneven (ang Gao et al., 2026), with consolidating specialists from a shared base as the standing use case (Yu et al., 2026a); its variants change what teachers supply, not how they are assigned (Chen et al., 2026b; Yu et al., 2026b; Yuan et al., 2025). Appendix A positions this recipe against the weight-merge and reinforcement-learning alternatives and Capability-OPD.

Finer-than-sample supervision. Supervision has been refined below the sample at every level: per-prompt routers (Zhang et al., 2025), step-level collaborative decoding (Yun et al., 2026), and token-level assignment by confidence, reweighting, selection, student compatibility, teacher purification, or attribution (Yin et al., 2026; Li et al., 2026b; Huang et al., 2025; Sun et al., 2026; Cui et al., 2026; Jin et al., 2026; Zhu et al., 2026; Zhang et al., 2026). Each refines the routing *signal* together with its granularity, so a win cannot be credited to granularity alone. Selective language modeling ranks tokens by excess loss against a reference model and keeps the top-scoring ones (Lin et al., 2024); our attribution is a difference of the same form, teacher minus shared initialization, aggregated per vocabulary id and used to *assign* a teacher rather than to drop a token, and computed once offline rather than re-ranked at every step.

Concurrent evidence and neighboring settings. Concurrent work bounds the effect from several directions: an oracle router recovers about a third of the teacher gap (ang Gao et al., 2026), a teacher's influence is not confined to its routed tokens (Li et al., 2026c), cross-domain interference is small (Wang et al., 2026), and dense per-token damage may resist reweighting (Jiang et al., 2026; Hou et al., 2026). Distillation scaling laws bound teacher strength by a capacity gap (Busbridge et al., 2025). Our reverse-KL objective (Hinton et al., 2015; Gu et al., 2023; Agarwal et al., 2023) is reported as unstable (Oh et al., 2026; Xing et al., 2026; Jang et al., 2026; Li et al., 2026a), so we treat it as a manipulated axis (App. D). The same granularity question recurs in weight-space merging (Wortsman et al., 2022; Matena & Raffel, 2021; Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2023) and mixture-of-experts gating (Shazeer et al., 2017; Fedus et al., 2021; Jiang et al., 2024), where our screen (Prop. 1) applies unchanged.

7 Conclusion

We asked whether refining multi-teacher on-policy distillation from per-rollout to per-token routing is worth its cost. Isolating granularity with Capability-OPD, the answer is conditional: per-token routing beats each single-teacher baseline on its weak axis, by 1.67 points on code and 1.49 on mathematics, when rollouts mix capabilities, but loses 1.42 points on single-domain data and is not detected below 7B. Finer routing neither uniformly helps nor hurts the recipe it refines; a necessary condition for it to pay can be checked in advance (§5).

AI use statement

We used large language model assistants substantially: to implement the training and evaluation code, orchestrate the experiment matrix, and draft and revise this manuscript. Their use was confined

to execution and exposition under author direction; the research questions, experimental design, pre-registration of hypotheses and decision gates, and interpretation of results are the authors' own. Every reported number is traced to a logged artifact and verified against the logs rather than taken from a model's summary, and we take full responsibility for all content, including any errors.

ETHICS STATEMENT

This work studies how to combine specialist teacher models when distilling into a student, using public mathematics and coding benchmarks and research-released models within their intended terms. It involves no human subjects, no personal data, and no new data collection or release, and we audit train–test overlap in Appendix F. Beyond the usual dual-use considerations of code-generation models we see no direct path to harm. Our central finding is a boundary on when the technique helps, and we report a pre-registered negative result and a null on the seven-benchmark average in full, against publication bias toward positive effects.

REPRODUCIBILITY STATEMENT

All hypotheses, decision gates, and multiplicity families were fixed in dated pre-registrations before any score was read (§3.1); the evaluation panel, judging protocol, routing signal, and teacher lines are specified in §3.1, §2, and Appendices B and C. Every comparison is a paired McNemar test on item-level agreement with the seed count stated inline, and all percentages are recomputed per item from the judged outputs, so any single cell can be regenerated independently.

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, 2023.

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

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

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation scaling laws, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021.

Tianlei Chen, Jiao Ou, Ziyuan Liu, Ruiming Tang, Jian Liang, and Han Li. Counteraction-aware multi-teacher on-policy distillation for general capability recovery with domain preservation, 2026a.

Yunjie Chen, Xiaoxin Chen, and Fang Wang. Regen: Replay-recycling for expert-to-generalist distillation with offline reinforcement learning, 2026b.

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.

Jin Cui, Jiaqi Guo, Ruixuan Yang, Jiayi Lu, Jiepeng Zhou, Jiajun Xu, Jiangcheng Song, Boran Zhao, and Pengju Ren. "the whole is greater than the sum of its parts": A compatibility-aware multi-teacher cot distillation framework, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Wenjin Hou, Shangpin Peng, Weinong Wang, Zheng Ruan, Yue Zhang, Zhenglin Zhou, Mingqi Gao, Yifei Chen, Kaiqi Wang, Hongming Yang, Chengquan Zhang, Zhuotao Tian, Han Hu, Yi Yang, Fei Wu, and Hehe Fan. Uni-opd: Unifying on-policy distillation with a dual-perspective recipe, 2026.

Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. Selectkd: Selective token-weighted knowledge distillation for llms, 2025.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2022.

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.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation, 2026.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026.

Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuo Yang, Chu Yuan Zhang, and Jianhua Tao. Exploring knowledge purification in multi-teacher knowledge distillation for llms, 2026.

Shizhen Li, Zhiyu Shen, Yuyin Lu, Yunhe Pang, Jielin Song, Yanghui Rao, and Fu Lee Wang. Rethinking reverse kl as adaptive entropy distillation, 2026a.

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, 2026b.

Zhaoyi Li, Deyang Kong, Yuan Wei, Evan Yang, Ranran Shen, Mahardika Krisna Ihsani, Ming Yang, Wei Zhang, Chuan Hao, Jian Yang, Ran Tao, Bryan Dai, Shikun Zhang, Wei Ye, Ying Wei, and Defu Lian. Every coin has two sides: On the dual nature of generalization in on-policy distillation of large language models, 2026c.

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.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Not all tokens are what you need for pretraining, 2024.

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.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training, 2026.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging, 2021.

NVIDIA et al. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026.

Minjae Oh, Sangjun Song, Gyubin Choi, Yunho Choi, and Yohan Jo. Kl for a kl: On-policy distillation with control variate baseline, 2026.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report, 2024.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models, 2026.

Yang Sun, Lichao Ma, Houyuan Qin, Yuxin Liu, Hanyang Lu, Yao Zhu, Pinlong Cai, and Guohang Yan. Reopd: Reliability-adaptive reward extrapolation for on-policy distillation, 2026.

Haoqing Wang, Xiang Long, Ziheng Li, Yilong Xu, Tingguang Li, and Yehui Tang. To mix or to merge: Toward multi-domain reinforcement learning for large language models, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation, 2026.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chujie Zheng, Dayiheng Liu, Fei Huang, et al. Qwen3 technical report, 2025.

Qixiang Yin, Huanjin Yao, Yuchen Cai, Jianghao Chen, Ziyi Wang, Min Yang, Fei Su, and Zhicheng Zhao. H-opd: Confidence aware heterogeneous multi-teacher multimodal on-policy distillation, 2026.

Fangxu Yu, Weijia Xu, Michael Xu, Tianyi Zhou, and Zinan Lin. Weak-to-strong on-policy distillation, 2026a.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch, 2023.

Weichen Yu, Xiaomin Li, Yizhou Zhao, Xiaoze Liu, Ruowang Zhang, Haixin Wang, Yinyi Luo, Chen Henry Wu, Gaurav Mittal, Matt Fredrikson, and Yu Hu. Multi-rollout on-policy distillation via peer successes and failures, 2026b.

Xiaoyang Yuan, Yujuan Ding, Yi Bin, Wenqi Shao, Jinyu Cai, Jingkuan Song, Yang Yang, and Heng Tao Shen. More than one teacher: Adaptive multi-guidance policy optimization for diverse exploration, 2025.

Taewon Yun, Jisu Shin, Jeonghwan Choi, Seunghwan Bang, and Hwanjun Song. Distilling long-cot reasoning through collaborative step-wise multi-teacher decoding, 2026.

Hengyuan Zhang, Shiping Yang, Xiao Liang, Chenming Shang, Yuxuan Jiang, Chaofan Tao, Jing Xiong, Hayden Kwok-Hay So, Ruobing Xie, Angel X. Chang, and Ngai Wong. Find your optimal teacher: Personalized data synthesis via router-guided multi-teacher distillation, 2025.

Kangning Zhang, Yixing Li, Shuai Shao, Qingyao Li, Zhengxi Lu, Zhiyuan Yao, Jianghao Lin, Wenxiang Jiao, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. Vad: Attributing visual evidence for target reconstruction in multimodal on-policy distillation, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

Wenhong Zhu, Ruobing Xie, Rui Wang, and Pengfei Liu. Hybrid policy distillation for llms, 2026.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2024.

Table 4: **Capability-integration recipes.** Supervision signal, assignment granularity, on-policy integration, and whether a per-prompt domain label is needed. The integration score is transcribed from Ma et al. (2026), Table 2 (Qwen3-30B-A3B; their RL teacher reads one, the SFT-only student zero) and is not comparable with Table 2. CAPABILITY-OPD changes only the granularity of the per-sample MOPD recipe.

Method	Supervision	Granularity	On-policy	Needs tags	Norm. score
Mix-RL	joint RL	—	✓	✓	0.88
Cascade RL	staged RL	—	✓	✓	0.78
Off-policy finetune	fixed corpus	—	×	✓	0.82
Param-merge (task arith.)	weight space	model	×	×	0.86
Param-merge (average)	weight space	model	×	×	0.33
PER-SAMPLE (MOPD recipe)	teacher logits	rollout	✓	✓	0.94
PER-TOKEN (CAPABILITY-OPD, ours)	teacher logits	token	✓	×	—

APPENDIX ROADMAP

The appendix is organized as follows. Appendix A positions CAPABILITY-OPD among capability-integration recipes; Appendix L states the full CAPABILITY-OPD algorithm; Appendix B gives the routing signal, thresholds, and full training configuration; Appendix C the three teacher lines and their students; Appendix D the design-space axes we treat as manipulated; Appendix E the span sweep on routing granularity; Appendix F the code-source robustness swap and decontamination audit; Appendix G the full evaluation panel; Appendix H the exact paired differences behind the main-text figures; Appendix I the decomposition of the seven-benchmark aggregate; Appendix J the supporting mechanism figures; Appendix K the training dynamics; Appendix M the instruction-following third capability; §5 the screening proposition; §N the difficulty-bounding analysis and the math-source control; and §O the limitations.

A METHOD LANDSCAPE FOR CAPABILITY INTEGRATION

Table 4 places CAPABILITY-OPD among the integration recipes of §6. The rightmost column transcribes the normalized integration score Ma et al. (2026) report in their Table 2 on the Qwen3-30B-A3B student, scaled so their reinforcement-learning teacher reads one and the SFT-only student reads zero. These are their published numbers under their setup and are *not* comparable cell-for-cell with the absolute accuracies of Table 2, which use a different backbone, benchmark panel, and metric; we include them to locate each family, not to rank methods across papers. CAPABILITY-OPD refines the granularity of the strongest entry, the per-sample MOPD recipe, from the whole rollout to the token, and unlike that recipe it consults an offline per-token attribution table rather than a per-rollout domain tag (Appendix B).

B ROUTING SIGNAL AND CONFIGURATION

Attribution procedure. The routing table is built offline, once per teacher pair, before any student is trained. For each teacher π_k and each token position in a held-out measurement corpus, we score the gold next token under the teacher and under the shared initialization π_0, and accumulate, per vocabulary id v, the mean log-probability lift $\Delta_k(v) = \mathbb{E}[\log \pi_k(\text{gold} \mid \text{context}) - \log \pi_0(\text{gold} \mid \text{context}) \mid \text{gold} = v]$. A vocabulary id is assigned to $\arg\max_k \Delta_k(v)$ when the winning lift is positive and the id occurs at least a minimum count in the measurement corpus; ids that clear no teacher, or fall below the count floor, are routed to a shared fallback teacher. The result is a static id→teacher table consulted at training time; no routing decision is made online. The measurement corpus is held out from every evaluation benchmark.

Thresholds. The exclusivity cutoff τ and the minimum-count floor m_c are the only two hyperparameters of the table; the split-half stability of §5 is reported at four count floors to show the table's direction is insensitive to the choice. For the main-result teacher pair both are left open ($\tau = 0$ and $m_c = 0$), so every token id routes to its higher-improvement specialist and none to the shared

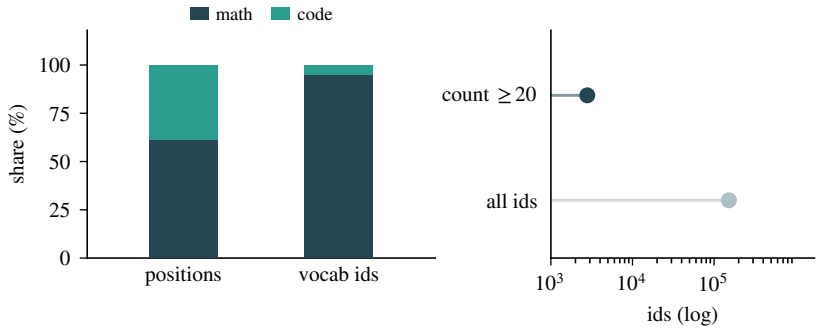

Figure 6: **One routing table, two weightings.** *Left:* the main-result table (τ=0, m_c=0) counted by vocabulary id and weighted by token positions in the measurement corpus. *Right:* of the 152064 vocabulary ids only 2792 occur at least 20 times, so id counts are set by ids that training rarely visits. Coverage figures elsewhere are position-weighted.

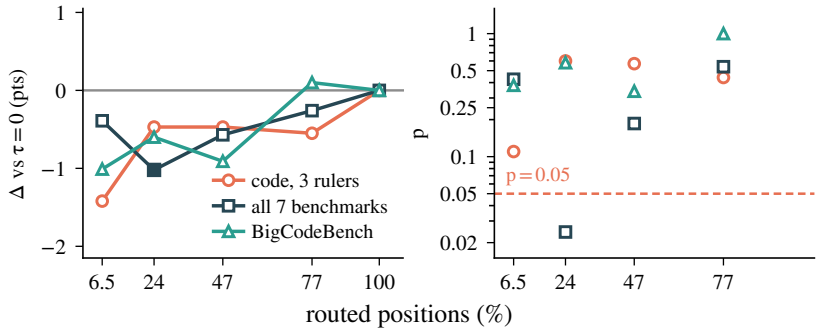

Figure 7: **Pruning the table does not beat full routing.** Each point is one trained student against the τ=0 method on the same items. *Left:* effect against the fraction of positions still routed; hollow markers do not clear their family correction, the filled one is the only cell with raw $p < 0.05$ (-1.02 points at 24.0%, p_{Bonf}=0.098). *Right:* raw p of all twelve cells. τ=0 was fixed before the sweep.

fallback. Weighted by occurrence in the held-out measurement corpus, this sends 61.6% of token positions to the mathematics teacher and 38.4% to the code teacher, with 0% routed to the fallback. Counted by raw vocabulary id the split is instead 95.2%/4.8%; this is dominated by rarely-seen ids that default to the mathematics argmax and does not reflect training-time routing, which the count-floor sweep of §5 progressively prunes (Figure 6).

Threshold sweep. Fixing $m_c = 0$, we swept the exclusivity cutoff over $\tau \in \{0.6, 0.7, 0.8, 0.9\}$, which prunes the specialist-routed fraction of token positions from the 100% of the main $\tau = 0$ method down to 77.3/47.2/24.0/6.5% respectively (the pruned positions revert to π_0 self-distillation rather than to a distinct fallback teacher). Against the $\tau = 0$ method, none of the four pruned settings significantly changes the student's score on the code three-benchmark axis, on held-out BigCodeBench, or on the seven-benchmark aggregate: every comparison is non-significant after Bonferroni correction over the four methods, the only cell with raw $p < 0.05$ being -1.02 points on the aggregate at 24.0% coverage (raw p=0.024, p_{Bonf}=0.098, driven by a single benchmark). The coverage-versus-code trend is a descriptive Spearman $\rho = +0.67$ (p=0.13, n=5), leaning the way full routing would predict but within noise. No pruned setting significantly improves on full routing (Figure 7); $\tau = 0$ was fixed before the sweep, not tuned.

This sweep varies a different quantity than the keep-ratio sweeps reported for token *selection* (Lin et al., 2024): a token dropped by a selection rule loses its loss term, whereas a pruned position here keeps its loss term and exchanges a specialist target for π_0. The two sweeps are not comparable.

Which ids the table owns. The shares above say how much mass each teacher receives, not which tokens it receives. Reading the table out at the inspection gate ($\tau = 0.6$ with $m_c = 20$, a third operating point, distinct from both the $\tau = 0$ method that trains the students and the $m_c = 0$ sweep above) shows that the highest-mass assignments on either side are punctuation, whitespace, function words and formatting scaffolding rather than domain vocabulary (Figure 5). What domain vocabulary does appear crosses in both directions: `·function` and the backtick go to the mathematics teacher, `Answer` and `·math` to the code teacher. Ranking by frequency and by per-token improvement also gives different orders: the most frequent assignment, `·of`, carries one of the smallest gold-token lifts, while the largest lifts sit on ids an order of magnitude rarer.

Evaluation and judging. Mathematics is scored with a format-robust matcher that accepts any answer equivalent to the reference under normalization, rather than a bracket-exact gate; code is executed against each benchmark's own test suite (pass@1 under the benchmark's harness, EvalPlus (Liu et al., 2023) tests where available); instruction-following uses strict per-constraint satisfaction. Paired comparisons use McNemar's test on the item-level 2×2 agreement table. Family definitions, decision gates, and multiplicity corrections were fixed in dated pre-registrations before any score was read.

C TEACHER LINES

The main text reports the reinforcement-learning teacher line: each domain specialist is $\pi_0 \rightarrow$ GRPO (Shao et al., 2024) on that domain (the GRPO configuration, `adv_estimator=grpo` with a KL loss term and no KL-in-reward, is shared across methods). We report here what happened when we tried to vary that recipe, because the outcome constrains the main result's scope.

- **SFT line, instruct root: built, gated out, never carried to a student.** Specialists here are $\pi_0 \rightarrow$ supervised fine-tuning on the domain, with a replay mix to counter catastrophic forgetting of the shared format, the line closest to a realistic "merge separately fine-tuned specialists" deployment. On the instruction-tuned root we built it at two learning rates, and at both the domain-SFT specialist came out significantly *worse* on its own target domain than the root it started from, in mathematics and in code alike. A specialist that has lost ground on the very domain it was specialized for cannot supply the gold-anchored lift the routing table is built from, so the line failed the teacher-quality gate of §5 and we stopped it there rather than spend a student on it.
- **SFT+RL line, instruct root: not run.** Specialists would be $\pi_0 \rightarrow$ domain SFT $\rightarrow$ domain GRPO. The first stage of that chain is the line above, so the gate failure removes its precondition; we did not launch it. Whether an RL stage can repair a specialist that domain SFT has degraded is a question our logs do not settle: our code-side traces put most of the format damage inside the SFT stage, with the subsequent RL stage recovering a fraction of it.

Both alternative lines were carried all the way to trained and evaluated students on the *base*-model root (Appendix D), which is where the recipe's influence can be measured: on that root PER-TOKEN lost to PER-SAMPLE under the SFT line and under the SFT+RL line alike, and the difference between the two lines was not significant, so the teacher recipe is not what decided the granularity outcome there. Within each line the student, corpus, decoding, and routing table are byte-identical across methods; only the teacher checkpoints differ. The granularity result of §4 therefore rests on the one teacher line that clears the gate on the instruct root, and the evidence that the recipe is not itself the active ingredient comes from the base root, where it changes nothing.

Both cells are significant against the by-sample baseline after family correction (n=3828 paired items each). The two lines differ by 0.60 points at the point estimate, inside the ± 2.23 points two-method seed-shift floor, and their intervals overlap across almost their entire range, so the teacher recipe is not what decided the granularity outcome on this root. We rest the "no line difference" reading on the overlapping line-level intervals rather than a jointly-paired difference-in-differences contrast, which our logs do not contain.

Why the instruct-root SFT line never reached a student. On the instruct root the same domain-SFT step fails a teacher-quality gate before any student is trained: each specialist degrades its *own*

Table 5: **Base-root teacher lines.** PER-TOKEN−PER-SAMPLE on the seven-benchmark panel for the two alternative teacher recipes; only the teacher checkpoints differ between methods. Both lines lose, with overlapping intervals across most of their range.

Teacher line	Teacher recipe	Δ ALL7 (pp)	95% CI	Verdict
SFT	$\pi_0 \to$ domain SFT	-4.44	$[-5.70, -3.18]$	worse
SFT+RL	$\pi_0 \to$ domain SFT $\to$ GRPO	-5.04	$[-6.30, -3.79]$	worse

Table 6: **Instruct-root domain-SFT gate.** Each specialist against π_0 on its own domain's benchmarks at two learning rates. Domain SFT overwrites the aligned instruct distribution, so both specialists fall on their own domain and neither can serve as a teacher; the base root shows no such drop (Table 5).

Specialist (own-domain Δ vs π_0, pp)	lr $= 10^{-5}$	lr $= 3 \times 10^{-6}$
math specialist (3 math benchmarks)	-6.0 to -17.2	-6.22 to -16.20
code specialist (4 code benchmarks)	-3.3 to -8.3	-3.43 to -7.93

target domain relative to π_0, at both learning rates we tried, so there is no usable teacher to distill. Table 6 gives the own-domain ranges; every own-domain cell is significant by paired McNemar.

The contrast the main result rests on is the RL teacher line on the instruct root, which clears the same gate: the math specialist lifts its hardest math benchmark by $+12.00$ points ($p < 10^{-3}$) and the code specialist lifts its hardest code benchmark by $+5.21$ points ($p < 10^{-3}$), with no significant cross-domain drop. The KL term in the RL objective prevents the catastrophic forgetting that the cross-entropy SFT pass causes.

A specialist batch-size asymmetry that cancels in the contrast. The two RL specialists share every optimization setting (objective, learning rate, KL coefficient, and epoch count) except three quantities the code reward forces apart: because the code specialist is scored by a sandbox judge that forks one process per rollout, its batch size, rollouts per prompt, and response budget are held at $256/4/2048$ against the math specialist's $512/8/3072$, keeping the fork count within the machine's process budget. This is a fixed property of the teacher pair, not a tuned knob, and it enters every downstream method identically, so it cancels in the PER-TOKEN−PER-SAMPLE contrast that carries the main result: both methods consume the byte-identical pair. It is likewise conservative for the split between the single-teacher baselines: the asymmetry uniformly favors the better-resourced *math* specialist, which works *against* the code-axis crossover we still observe (CODE-ONLY beats MATH-ONLY on code by 2.68 points) rather than creating it. Only the math-axis leg could carry a residual budget advantage atop the domain effect, so we rest that leg on its sign, not its magnitude.

D DESIGN-SPACE AXES TREATED AS MANIPULATED

Several choices that prior work fixes by assumption we instead varied, so that the main comparison holds them at one setting rather than confounds the granularity effect with them:

- **Objective.** The main methods use single-sample reverse-KL on student rollouts; we also ran a top-k forward-KL variant to check that the routing conclusion is not tied to the reverse-KL objective, which is independently reported as unstable.
- **Routing mechanism.** Beyond the hard id→teacher assignment, we ran a soft teacher-mixture variant (convex weighting by per-token lift) and a masking variant (supervise only each teacher's owned tokens), isolating the effect of *how* the routing table is applied from the table itself.
- **Route strength.** A sweep over the fraction of tokens routed away from the fallback separates the contribution of the routed tokens from the shared-fallback backbone.

Table 7: **Base-root design-space ledger.** For each manipulated axis, the PER-TOKEN−PER-SAMPLE difference over the full seven-benchmark panel on the base-model root. "worse" clears family-corrected significance with a negative sign; "tie" does not clear it. No alternative setting is a positive win.

Axis	Alternative setting	Δ ALL7 (pp)	Verdict
Objective	top-k forward-KL	-5.98	worse
Routing mech.	masking (owned tokens only)	-0.52	tie
	soft teacher mixture	$+0.63$	tie
Route strength	heavy dose	-5.22	worse
	medium dose	-3.42	worse
	light dose	$+0.50$	tie
	owned-only (purity)	-5.17	worse
	owned-only + medium dose	-4.26	worse

This sweep was run on the base-root initialization, and on that root no axis produced a robust win for PER-TOKEN over PER-SAMPLE: in aggregate the best case was a statistical tie, with several axes significantly worse, and even the raw code-axis positives that did appear did not survive the held-out BigCodeBench scale. That null outcome is what motivated moving to the instruct-root initialization used in the main results, where the code-axis gain does survive the same held-out scale. We read it as consistent with a weaker base distribution yielding noisier per-token attribution, leaving finer routing no clean signal to exploit, though we do not isolate that as the mechanism.

Across the mechanism and strength alternatives (the pre-registered best-of-eight single-benchmark family), no cell is a positive win: $0/8$ clear family-corrected significance with a positive sign. The ceiling is the soft mixture at $+0.63$ points, whose 95% interval reaches only $+1.68$ points, inside the ± 2.23 points two-method seed-shift floor; masking and the lightest dose sit at the same null, and the objective axis and the heavier doses are outright worse. Neither *how* the table is applied nor *how much* of it is applied buys PER-TOKEN an aggregate advantage over PER-SAMPLE on this root.

The one code-axis positive, and why it does not survive. The design-space cells above are single-benchmark aggregates, so a code-axis breakdown exists only for the two axes we ran with the full code panel: the objective axis and the mixed-domain concatenation construction. On the three code benchmarks the forward-KL objective is a non-significant -0.55 points, and the base-root concatenation gives a nominal $+2.60$ points (95% CI $[+0.98, +4.22]$), but that positive does not survive the held-out fourth code benchmark: BigCodeBench reverses it to -5.34 and -2.92 points across two seeds, so the held-out-scale gate disqualifies it. The same construction on the instruct root keeps its code-axis gain *with* its held-out check intact (BigCodeBench $+0.81$ points, same sign, not reversed; §4), the base-to-instruct survival contrast that motivated the main setup.

E How fine is fine enough? A span sweep on the routing granularity

The main result compares one step of granularity, per token against per sample, which leaves the granularities in between and beyond untested. If the benefit came from per-position precision, it should decay as soon as the routing label is smeared across neighboring positions. We test that directly by coarsening the per-token assignment into contiguous spans of 1, 2, 8, and 32 tokens on fully mixed ($f=1$) data, holding the specialist teachers, student, attribution table, decoding, and step budget byte-identical to the main methods, so that span length is the only variable. Two reassignment modes separate the two effects coarsening has at once. A *proportional* owner preserves each teacher's token budget and changes only *which* positions carry each label; a *majority-vote* owner lets the denser label win each block, which also shifts budget away from the sparser code teacher.

This section carries the full sweep behind Figure 4: both ownership modes, the per-seed increments, and the merge-operator control.

The gain does not decay with coarser spans; it plateaus. Under the proportional owner, the code-axis gain over MATH-ONLY reads $+1.67/+1.70/+2.19/+1.87$ points at spans $1/2/8/32$ (span 1 pooled over six seeds, $N{=}7608$; the coarser spans over four, $N{=}5072$; Figure 4). Every point is positive, and the profile peaks at span 8 rather than at the finest span ($\rho{=}+0.80$ between $\log_2$ span and the gain), the opposite sign to the monotone decay we pre-registered. Because the peak first appeared in a two-seed sweep, we replicated it in a separately pre-registered family on two fresh seeds instead of adding seeds to the original (which had declared a two-seed cap to forestall optional stopping): the independent queue reads $+1.46/+1.93/+1.81$ points at spans $2/8/32$, all three surviving Bonferroni correction over the three methods ($p_{\mathrm{Bonf}}{=}0.030/0.0024/0.0027$), and span 8 is again the argmax. Pooling the two families moved the peak from $+2.44$ points down to $+2.19$ points, a regression toward the mean as N grew.

Why we nonetheless claim per-token, not per-span. The argmax is stable, but being the argmax is not the same as being better than per-token. Tested directly and pairwise on the four seeds that carry both methods, span 8 beats span 1 by $+0.59$ points on the code axis ($N{=}5072$, $p{=}0.0908$; $+0.47/+0.63/+1.10/+0.16$ points, positive in all four seeds). The sign is consistent and the increment is small and only marginally significant, so by our own standard (a claim of superiority requires a paired test that clears correction, and a non-significant contrast may be reported only as a bound) the evidence supports *a plateau that per-token sits on*, not a win for per-span. The paper therefore keeps per-token as the claimed operating point, and reports span 8 as the top of a flat region. Coarsening is free rather than beneficial: the math axis is unmoved by it ($+0.25/-0.25/-0.89/+0.89$ points across seeds, straddling zero), as is instruction-following ($+0.92/-1.11/+1.48/-1.11$ points, straddling zero), and span 8 carries no math cost against MATH-ONLY either ($+0.54/-0.30/-0.25/+0.99$ points).

What a span is, and what the plateau implies. A span of eight tokens is roughly a short run of contiguous context (an identifier and its call, a clause of a proof) rather than a single lexical decision. Read that way, the routing signal appears locally smooth: neighboring positions inside such a run mostly want the same specialist, so committing the whole run to one teacher discards little, while a per-sample tag spanning hundreds of tokens necessarily crosses domain boundaries and misroutes part of the sequence. On this reading the useful granularity is bounded above by the length over which the specialist identity is stable, and per-token routing is simply the finest point inside that bound rather than a knife-edge optimum. This reading is interpretive: the sweep constrains where the gain lives but does not measure the smoothness of the signal directly.

The budget-shifting mode breaks the gain. Under the majority-vote owner the gain dips to $+0.47$ points (n.s.) at span 2 before recovering to $+1.30$ points at span 8, never reaching the proportional profile. Precision and budget therefore come apart: per-position precision is not where the per-token benefit lives, so spending it costs nothing, and the benefit is instead a property of teacher budget and how much a sample mixes the two domains (the mixing sweep of §4.2).

Coarser routing does not buy compute. One might expect a coarser router to be cheaper, and in our setting it is not. The routing table is built offline once per teacher pair and consulted by lookup at training time, so no routing decision is made online and none is avoided by grouping positions. Under discrete per-token routing both specialists still serve every step, since different positions in the same sequence route to different teachers, and the proportional mode holds each teacher's token count fixed by construction. Span length therefore changes neither the number of teacher forward passes nor the number of served teachers. Any efficiency argument for coarse routing would have to come from an online-switching implementation we do not use.

Together these sweeps bound the granularity question from the fine side: below the per-sample scale, resolution finer than necessary buys no additional gain across spans of 1 to 32 tokens, and the cost of getting the granularity wrong is paid in teacher budget rather than in positional precision.

Routing mechanism, swept on the instruct root. We probe a second design choice, *how* teachers are merged at a routed token, on the same instruct root in a single-seed direction check. Replacing the hard id→teacher assignment of the main methods with a *soft* teacher-mixture (each token's supervision a convex combination of the teachers' log-probabilities, weighted by their per-token attribution lift), and holding the routing table, ownership, decoding, and step budget fixed, the granularity

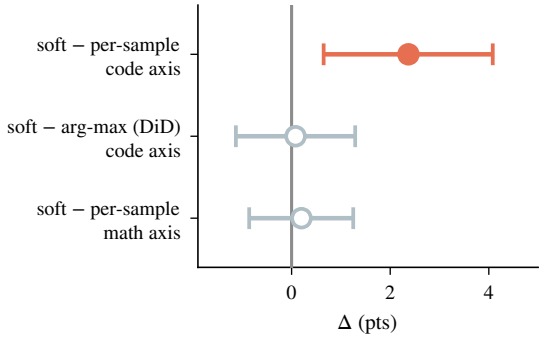

Figure 8: **The advantage survives a soft teacher mixture.** Paired contrasts with 95% CIs; filled markers clear zero. Top: soft routing minus PER-SAMPLE on code; middle: soft minus argmax merge (difference-in-differences); bottom: soft minus PER-SAMPLE on math. Single seed, direction check.

Table 8: Swapping the code training source from CodeContests to AceCode reproduces both halves. Six-seed paired contrasts; CodeContests main-line values in parentheses. $^{*}p_{\text{Bonf}} < 0.05$, $^{**}p_{\text{Bonf}} < 10^{-3}$. LiveCodeBench is the contamination-resistant arbiter (a single benchmark, not commensurable with the three-benchmark pool).

Contrast	Code axis	Math axis
CODE-ONLY − MATH-ONLY *(the two split)*	$+2.83^{**}$ $(+2.68)$	-1.35^{**} (-1.06)
PER-TOKEN − MATH-ONLY	$+1.72^{**}$ $(+1.67)$	-0.40 n.s. $(+0.43$ n.s.$)$
PER-TOKEN − CODE-ONLY	-1.10^{*} (-1.01^{*})	$+0.95^{**}$ $(+1.49)$
PER-TOKEN − MATH-ONLY, LiveCodeBench	$+3.57^{**}$	—
CODE-ONLY − MATH-ONLY, LiveCodeBench	$+5.58^{**}$	—
PER-TOKEN − CODE-ONLY, LiveCodeBench	-2.01^{**}	—

advantage of PER-TOKEN over PER-SAMPLE survives the substitution: it reads $+2.37$ points on the three code benchmarks (95% CI $[+0.65, +4.08]$, $p_{\text{Bonf}} = 0.0179$), clearing the per-sample baseline just as the argmax merge does. The soft and hard-argmax merges did not separate (difference-in-differences $+0.08$ points, 95% CI upper bound $+1.29$ points), and the math axis is unmoved ($+0.20$ points, n.s.), as on the argmax root. The code-axis gain is therefore a property of the routing *table* rather than of the operator that merges teachers at a routed token (Figure 8).

F Robustness to the mixed-data source: swapping the code training source

The code axis of the result rests on a single code distribution (CodeContests, competitive stdin/stdout). To test whether the effect is an artifact of that style, we swap the entire code training source for AceCode (function-completion problems derived from open-source and StackOverflow code, the same family as HumanEval/MBPP), holding the math source, the routing machinery, and the frozen evaluation panel fixed; the code distribution is the only variable. We retrain the code teacher on the new source, recompute the per-token routing table, and train the three methods (PER-TOKEN, CODE-ONLY, MATH-ONLY) at six seeds each, greedy decoding as in the main line.

The six-seed contrasts under the swap are summarized in Table 8, with the CodeContests main-line values alongside; this appendix adds the per-seed range, the three decontamination guards, the LiveCodeBench arbiter, and the pre-registered math-source swap.

Both halves reproduce (Table 8, Figure 9). The single-teacher baselines come apart on code by 2.83 points (six-seed pool $p_{\text{Bonf}} < 10^{-3}$; all six seeds positive, $+2.37$ to $+3.23$) and reverse on math by 1.35 points ($p_{\text{Bonf}} < 10^{-3}$), landing on top of the CodeContests values (2.68 and -1.06). The per-token code gain reproduces and stays significant at 1.72 points ($p_{\text{Bonf}} < 10^{-3}$; cf. 1.67).

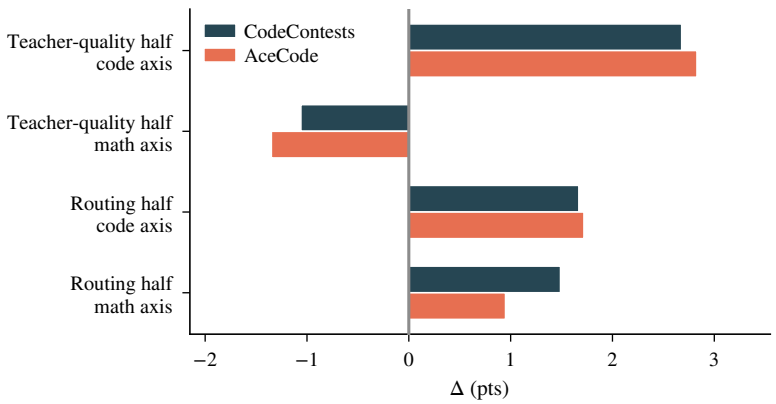

Figure 9: **Swapping the code training source reproduces both halves.** Paired six-seed contrasts under CodeContests (main) and AceCode (swap). Exact cells, corrected p and the LiveCodeBench arbiter are in Table 8.

The per-token math gain reproduces and, with the full six seeds, clears correction at 0.95 points ($p_{\mathrm{Bonf}} = 0.0002$); it attenuates relative to the CodeContests value (1.49), consistent with a milder code teacher leaving the code-only router less room to lose the math axis.

Contamination is the obvious worry, since AceCode shares a family with HumanEval/MBPP. Three pre-registered guards clear it. An 8-gram decontamination audit finds no near-duplicate (0 of 542 evaluation problems at containment ≥ 0.8; mean 0.007). The math legs of both hypotheses are contamination-immune by construction (a code teacher's memorization cannot help a math benchmark), and both reproduce. On LiveCodeBench alone, time-sliced and near-disjoint from StackOverflow completions as our anti-contamination arbiter, the code legs are *larger* than the three-benchmark pool: CODE-ONLY$-$MATH-ONLY $= +5.58$ points and PER-TOKEN$-$MATH-ONLY $= +3.57$ points (both $p < 10^{-3}$). The code-axis gain is therefore not a same-family memorization artifact.

A symmetric math-source swap (BigMath+ORZ) was pre-registered and reached its teacher-differentiation gate as INCONCLUSIVE: the stronger math distribution did not lift the reused teacher above the code teacher on held-out easy math, so no methods were trained there. We report that boundary as a registered negative in §N.

G THE EVALUATION PANEL IN FULL

The seven-benchmark panel of §3.1 is listed in full in Table 9.

H EXACT PAIRED DIFFERENCES BEHIND THE MAIN-TEXT FIGURES

Table 10 gives the exact per-axis paired differences plotted in Figure 3a; the 7B-versus-4B values plotted in Figure 3c are in Table 3 (§4.2).

I THE AGGREGATE NULL, DECOMPOSED

The per-token code gain is a per-axis effect; pooled over all seven benchmarks it dilutes to a null. Figure 10 places that aggregate ($+0.68$ points, n.s., 95% CI $[-0.14, +1.49]$) beside the code component it dilutes ($+1.67$ points) and the two controls that frame it: distillation works end to end (PER-TOKEN over π_0 is $+1.15$ across all seven), while on that same pooled panel PER-TOKEN and a random-label control matched on teacher shares do not separate (-0.39 points, n.s.), so the panel resolves the routing comparison in neither direction. A practitioner optimizing a single averaged score should not expect the gain to move; it appears only on the axis a single label serves badly (§4.1).

Table 9: Evaluation panel: seven scored benchmarks (n=3828 items) for mathematics, code, and an instruction-following regression check. BigCodeBench is a held-out second code benchmark (not in the pooled n); since the code benchmarks disagree on method ordering, code summaries are reported against all of them. Absolute scores are comparable only within a model family (§3.1).

Benchmark	Capability	#Items	Scoring
Held-out math split	Mathematics	200	format-robust answer match
MATH-500	Mathematics	500	format-robust answer match
GSM8K	Mathematics	1319	format-robust answer match
HumanEval	Code	164	execution against test suite
MBPP	Code	375	execution against test suite
LiveCodeBench	Code	729	execution against test suite
IFEval	Instruction-following	541	strict constraint satisfaction
Pooled (main panel)		*3828*	
BigCodeBench *(held-out benchmark)*	Code	1083	execution against test suite

Table 10: Neither single teacher serves both axes. CAPABILITY-OPD beats each single-teacher baseline on the axis that baseline serves badly, but trails CODE-ONLY on code; the two baselines pull apart. Six-seed pool, McNemar. $^{*}p_{\mathrm{Bonf}} < 0.05$, $^{**}p < 10^{-3}$; n.s. not significant after correction.

Comparison	Code axis	Math axis
PER-TOKEN − MATH-ONLY	$+1.67^{**}$	$+0.43$ n.s.
PER-TOKEN − CODE-ONLY	-1.01^{*}	$+1.49^{**}$
CODE-ONLY − MATH-ONLY *(the two split)*	$+2.68^{**}$	-1.06^{**}

J MECHANISM: SUPPORTING FIGURES

The bound decomposition behind §5 is plotted in Figure 12, and the cost and ceiling terms it plots are listed in Table 11; the screening test itself is Figure 11 in §5.

K TRAINING DYNAMICS: ACCURACY VERSUS TRAINING STEP

The headline comparisons read accuracy at a single trained checkpoint. Figure 13 plots the full accuracy-versus-training-step trajectory of the three dense methods (§4.1), showing how the gap opens over the course of distillation rather than only at its endpoint: one representative seed for the per-token router (PER-TOKEN) and the two single-teacher baselines, at training steps 0 (the shared π_0 anchor), 30, and 60. The trajectory is descriptive; cross-method significance is not claimed from it, and the authoritative endpoint verdict is the six-seed, Bonferroni-corrected comparison of §4.1.

The sparse mixture-of-experts student (Qwen3-30B-A3B-Instruct, about 3B of 30B parameters active per token) is read the same way (Figure 14). At the trained endpoint (step 100, n=1268) the three methods do not separate on the pooled code axis: PER-TOKEN does not separate from PER-SAMPLE (point estimate -0.87 pp, 95% CI $[-2.54, +0.80]$, p=0.36), nor from the single mixed teacher (point estimate -2.13 pp, raw p=0.04, not surviving correction). The null is consistent with the $\leq$4B dense regime (§4.2), where the routing gain is already absent.

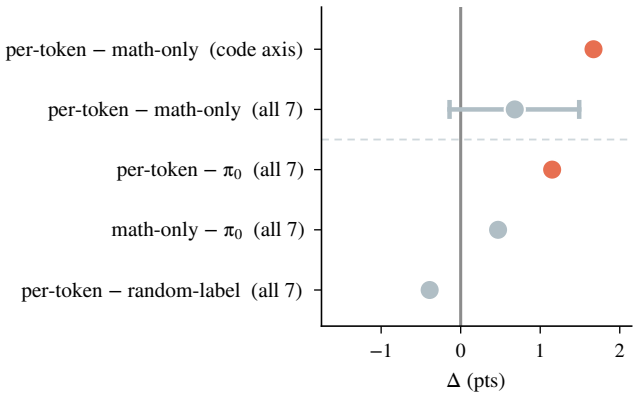

Figure 10: **The aggregate null, decomposed.** Paired differences against zero. Top group: the per-axis code effect ($+1.67$) and the seven-benchmark aggregate it dilutes to ($+0.68$, n.s., with 95% CI). Bottom group: two controls, distillation over π_0 ($+1.15$) and the random-label control (PER-TOKEN minus random $= -0.39$).

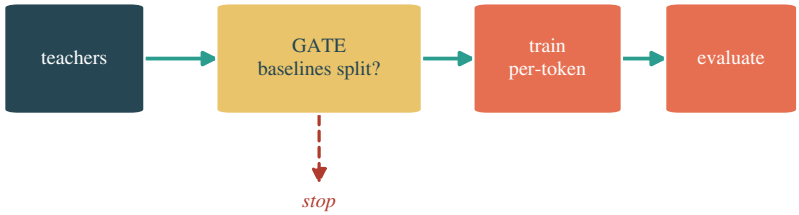

Figure 11: **The screening test** (Proposition 1): one comparison between methods one trains anyway decides, before the finer router is built, whether it can help. It passes on our mixed data and on the AceCode swap (App. F); a math-source swap fails it and was not trained (App. N).

L THE CAPABILITY-OPD ALGORITHM

Algorithm 1 Capability-Routed On-Policy Distillation (CAPABILITY-OPD)

Input: init π_0; specialists $\{\pi_k\}_{k=1}^{K}$; reference corpus $\mathcal{R}$; exclusivity cutoff τ; min count $n_{\min}$
// Stage 1: offline token–teacher attribution (once per teacher pair)
for each token id $v \in \mathcal{V}$ **do**
  compute lifts $\ell_k(v)$ over occurrences $\mathcal{O}(v) \subseteq \mathcal{R}$ ▷ Eq. (3)
  set $A(v)$ by the share/count rule ▷ Eq. (4)
end for
// Stage 2: on-policy distillation with the fixed table
$\theta \leftarrow \pi_0$
for each training step **do**
  sample prompt $x \sim \mathcal{D}$ and rollout $o \sim \pi_\theta(\cdot \mid x)$
  route each position t to teacher $\pi_{A(o_t)}$ and update θ on Eq. (5)
end for
return student π_θ

M A THIRD CAPABILITY: INSTRUCTION-FOLLOWING

To test whether the result is specific to the mathematics/programming pair, we add a third teacher for instruction-following, aligning the setup with the {math, code, IF} triple of production multi-

Table 11: Why the per-axis gain is bounded: the measured cost and ceiling terms behind the null on the seven-benchmark average, grouped by the three mechanisms of §5. Headroom figures are relative to sending every token to the single stronger teacher, in the gold-token log-probability lift the router optimizes; percentages above chance are split-half table-recovery rates.

Quantity	Measured
(a) The routing table is reproducible, but specific to its teacher pair	
Split-half agreement (same pair, same family)	$+29.4\%$ above chance
Same, changing the teacher pair (data fixed)	$+3.9\%$
Same, changing the model family	$+2.6\%$
(b) The routed-objective headroom is small, and half is selection bias	
Oracle headroom, in-sample	$+5.1$ to $+15.7\%$ rel.
Honest headroom, split-half	$+2.4$ to $+5.4\%$ rel.
Selection bias (arg-max inflation)	32 to 58% of apparent
True ceiling	$+2$ to $+6\%$ rel.
Gated router vs. per-sample routing	-9 to -49%
(c) Per-token assignment switches teachers within a sequence	
Owner switch interval	every 1.8 to 3.0 tokens
Adjacent positions that switch	33 to 56%
Switch rate under sequence shuffle	$\times 0.93$ to 1.07

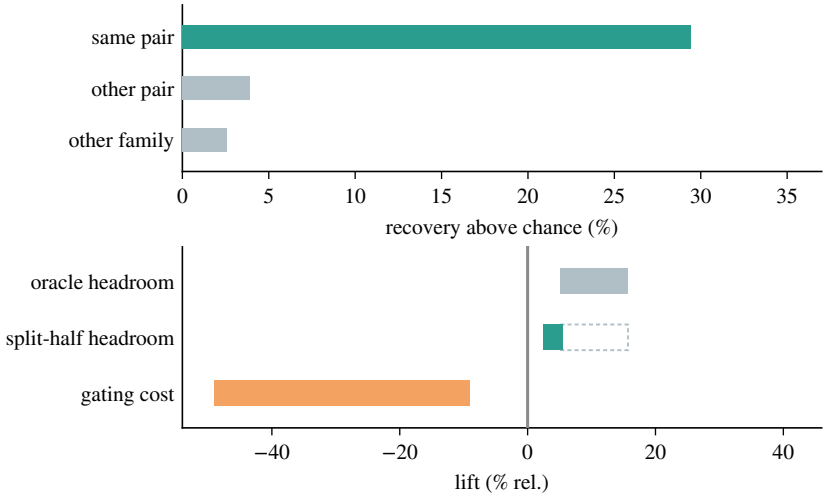

Figure 12: **What bounds the gain.** *Top:* table recovery above chance on disjoint halves: $+29.4\%$ for the same teacher pair but $+3.9\%$ across pairs and $+2.6\%$ across families, so the table is recomputed per deployment. *Bottom:* headroom and gating cost in gold-token log-probability lift; dashed outline: in-sample, solid bar: cross-fitted. Bars are ranges across teacher-pair-by-corpus combinations, not confidence intervals (Table 11).

teacher recipes. Instruction-following is a deliberately non-reasoning axis: its reward is programmatic constraint satisfaction rather than problem-solving correctness. We reuse the frozen math and code teachers, train an instruction-following teacher by reinforcement learning against a verifiable reward, recompute the per-token routing table over the three teachers, and train the per-token and per-sample methods at two seeds with data volume, decoding, and evaluation panel held fixed.

We registered in advance that a significant positive instruction-following result would contradict the paper's thesis. An earlier attempt with a supervised instruction-following teacher produced the largest negative per-token effect we measured, and diagnostics traced it to a structural cause: vocabulary-identity routing exposes only a small share of instruction-following tokens to that teacher,

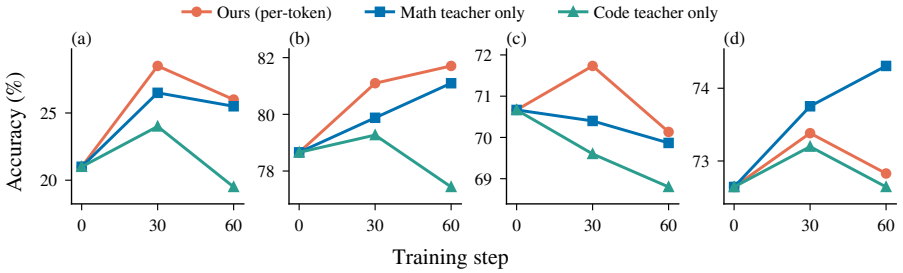

Figure 13: **Dense 7B student (Qwen2.5-7B-Instruct): accuracy against training step** (single seed, descriptive). PER-TOKEN and the two single-teacher baselines from a shared π_0 at step 0. (a)–(d): MathVal, HumanEval, MBPP, IFEval. Endpoint significance uses six seeds (§4.1).

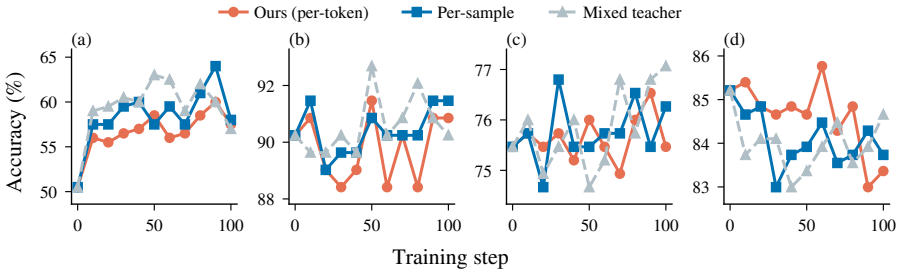

Figure 14: **Sparse-MoE student (Qwen3-30B-A3B-Instruct): accuracy against training step** (descriptive). PER-TOKEN, PER-SAMPLE, and a single mixed teacher from a shared π_0. (a)–(d): MathVal, HumanEval, MBPP, IFEval. The methods do not separate on the pooled code axis (§C).

because generic tokens are claimed by the math specialist. That cause is a property of the routing signal, not of teacher strength.

On instruction-following, per-token routing does not beat the per-sample router: the two seeds land at -0.92 and $+0.00$ points, neither significant and of opposite sign. In this three-teacher configuration PER-TOKEN under-performs PER-SAMPLE on both math (-1.06 points, $p_{\text{Bonf}}{=}0.0313$) and code (-3.63 points, $p_{\text{Bonf}} < 10^{-4}$), the cost of a low-exposure third teacher that perturbs the routing table without adding signal of its own to the axes that already carry it. Moving from the supervised to the reinforcement-learned teacher pulls the result up from a 14.23-point deficit, with a 95% interval of $[-18.20, -10.26]$, to zero. The reinforcement-learned teacher beats the shared start on IFEval by 6.28 points, about a quarter of the 25.14 points the supervised teacher held over its own start, because the instruct root already follows instructions well.

N A REGISTERED NEGATIVE, AND THE MATH GAIN ON AIME

This appendix collects two analyses that sharpen the boundary of the per-token benefit: a pre-registered negative that isolates the teacher-quality precondition, and the math half of the gain on competition-hard AIME, where it is not detected.

A registered negative that sharpens the boundary. We pre-registered a symmetric robustness test that swapped the *math* training distribution for a stronger, broader math-RL source, mirroring the code-source swap that reproduced the result. It did not clear its teacher-differentiation gate: on held-out easy math the swapped teacher did not surpass the reused code teacher, so the teacher-quality half of the result was absent, and under the registered decision rule we trained no student methods there. The asymmetry, the code-source swap reproducing while the math-source swap is inconclusive at the gate, locates the precondition: the per-token math gain rests on a teacher-quality

difference that a stronger data source failed to produce, the failure Proposition 1 is meant to catch before any student is trained.

The per-token math gain on competition math. On the standard math benchmarks (MathVal, MATH-500, GSM8K) the per-token advantage over the code-only router is 1.49 points, and 0.95 points under the AceCode code-source swap, both significant ($p_{\text{Bonf}} \approx 0.0002$ for the latter, six seeds; Figure 15). On competition-hard AIME it is not detected: 0.28 points, with a 95% confidence-interval upper bound of 3.29 points; that set is time-sliced and effectively uncontaminated, scored under sampled decoding, and treated as a family-internal contrast never pooled with the greedy benchmarks. Every distilled method still scores several times the π_0 floor there, so distillation transfers skill on hard math; what we do not detect there is any advantage of routing *granularity*. With the registered math-source negative, this is consistent with a per-token benefit that needs a teacher-quality gap on the axis in question.

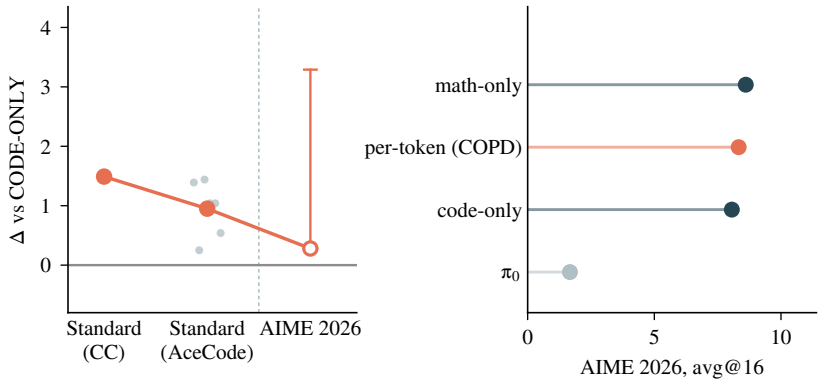

Figure 15: **The per-token math gain over CODE-ONLY: significant on the standard benchmarks, not detected on AIME.** *Left:* CAPABILITY-OPD minus CODE-ONLY on the three standard math benchmarks (CodeContests and AceCode as code source) and on AIME; filled markers clear the family correction, grey dots are the six AceCode seeds. AIME (avg@16 under sampling) is a separate pre-registered family, so only its 95% CI upper bound is plotted. *Right:* on that tier every distilled method is far above the undistilled π_0, while the three routings sit in one narrow band.

O LIMITATIONS

Claims are within-family paired differences. Because our answer extraction is conservative and some benchmarks cap long generations, absolute accuracies are comparable only within one model family under byte-identical decoding. Every cross-method claim is a paired difference between methods that share a student, corpus, and decoding configuration; we never compare absolute scores across families or sizes.

The gain is conditional and the routing table does not transfer. The per-token advantage does not survive pooling across benchmarks (App. I) and reverses on homogeneous data (§4.2), so a practitioner optimizing a single averaged score should not expect it to move. The token-to-teacher table is a property of the specific teacher pair and model family, recomputed per deployment; our offline gold-anchored lift is one routing signal, and online or learned signals may raise the ceiling we measured.

Boundaries of the tested regime. The result rests on a measurable teacher-quality difference; where a stronger data source fails to produce one, the mechanism does not fire, as the registered math-source negative shows (App. N). We study two reasoning capabilities and a third instruction-following teacher (App. M) at two model sizes on a seven-benchmark panel of short-form tasks. The capacity threshold we report, $\geq$ 7B for the per-token half, is an empirical boundary on the sizes measured, not a derived constant.