CROSS-TOKENIZER ON-POLICY DISTILLATION VIA BYTE-PREFIX MARGINALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-weight language models from different families exhibit complementary capabilities, motivating their consolidation into a compact student through on-policy distillation (OPD). However, full-vocabulary OPD typically assumes a shared tokenizer, while existing cross-tokenizer methods may discard teacher probability mass or assign it to student tokens with unrelated content. We introduce Byte-Prefix Marginalization (BPM), which re-expresses the teacher's next-token distribution over the student vocabulary in a shared byte space. Specifically, BPM assigns each teacher token's probability to the longest student token whose byte representation is a prefix of the teacher token's bytes, aggregates mass mapped to the same student token, and places otherwise unmatched mass in an explicit residual category. This produces a vocabulary-complete, byte-aligned, and mass-preserving target for dense OPD. The target exactly recovers the teacher-induced byte-prefix marginal when the relevant prefix does not span multiple teacher tokens (a condition satisfied at more than 99% of training positions) and uses a mass-preserving, chain-factorized lower bound otherwise. Across Qwen3-32B, GLM-Z1-9B-0414, and MiniMax-M2.7 as teachers, BPM consistently outperforms current cross-tokenizer methods on six mathematics and programming benchmarks, improving six-benchmark avg@8 by 3.7–6.6 points over the strongest baselines.

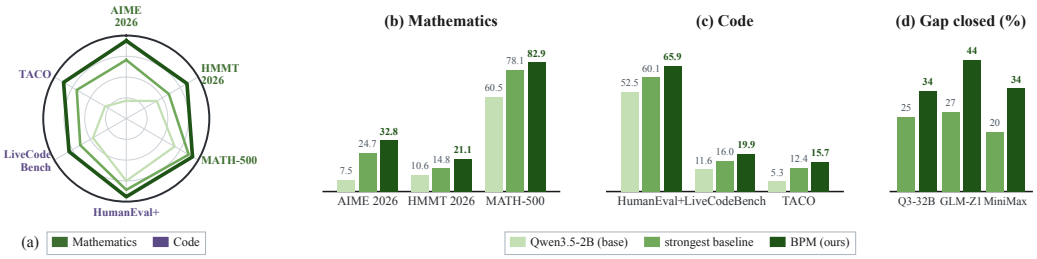


Figure 1: Performance averaged over the three teacher–student pairs (Table 1). **(a)** avg@8 on all six benchmarks, with each axis scaled independently; **(b)** mean mathematics avg@8; **(c)** mean code avg@8; **(d)** fraction of each pair's teacher–student avg@8 gap closed by BPM and its strongest baseline. BPM leads all budget-matched on-policy baselines and the separately trained SeqKD reference.

1 INTRODUCTION

Open-weight language models (DeepSeek-AI, 2026; Qwen Team, 2025; Kimi Team, 2026; GLM-5 Team, 2026; MiniMax, 2026) have advanced rapidly, with different model families exhibiting complementary strengths in mathematical reasoning, complex problem solving, agentic coding, and tool use. In practical deployments, it is often desirable to consolidate these diverse capabilities into a single compact model. Beyond capability integration, such consolidation can improve inference efficiency by reducing computational and memory overhead and avoiding the need to deploy or route requests among multiple specialized models. This motivates transferring knowledge from heterogeneous teacher models into an efficient student model (Jiang et al., 2023).

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.

Knowledge distillation provides a natural mechanism for this transfer by training the student to imitate the teacher's predictive behavior (Hinton et al., 2015; Kim & Rush, 2016; Xu et al., 2024a; Guha et al., 2025). On-policy distillation (OPD) performs this transfer along student-generated trajectories, querying the teacher at the states the student actually visits (Agarwal et al., 2024; Gu et al., 2024; Arora et al., 2022; Wang et al., 2026). Its supervision ranges from sampled-token and top-k approximations (Lu & Thinking Machines Lab, 2025; Gu et al., 2024; Li et al., 2025; Anshumann et al., 2025) to full-vocabulary matching of the teacher's complete next-token distribution (Agarwal et al., 2024; Qwen Team, 2025).

However, dense token-level supervision typically assumes that the teacher and student use the same tokenizer (Sennrich et al., 2016). This assumption restricts teacher selection to models compatible with the student's tokenization scheme, potentially excluding a stronger teacher for the target task (Rust et al., 2021). Extending full-vocabulary OPD to heterogeneous tokenizers therefore requires the transformed target to satisfy three properties. **(D1) Vocabulary Completeness:** the target is well-defined for every token in the student vocabulary. **(D2) Byte-level Alignment:** the teacher's probability mass is mapped to student tokens according to the correspondence between their byte-level representations. **(D3) Mass Preservation:** the transformation preserves the teacher's total probability mass, ensuring that the resulting target distribution, including any explicitly represented residual mass, sums to one.

Existing cross-tokenizer distillation methods (Wan et al., 2024; Boizard et al., 2024; Zhang et al., 2024; Cui et al., 2025; Minixhofer et al., 2025b; Patiño et al., 2025; Sun et al., 2026) make teacher and student distributions comparable by relaxing at least one of these requirements (Section 2). Candidate-set methods such as SimCT (Sun et al., 2026) leave unmatched teacher mass untransferred, violating **(D1) Vocabulary Completeness**; rank matching in ULD (Boizard et al., 2024) and GOLD (Patiño et al., 2025) may assign teacher mass to student tokens with unrelated byte content, violating **(D2) Byte-level Alignment**. SeqKD (Kim & Rush, 2016) instead avoids distribution matching and provides no full-vocabulary next-token target. These baselines therefore either approximate token-level transfer or avoid token-level distribution matching entirely. At the byte level, however, the transformation admits an exact solution.

We therefore formulate teacher supervision in a shared byte space (Clark et al., 2022; Xue et al., 2022; Yu et al., 2023; Pagnoni et al., 2024): although the teacher and student may tokenize a response differently, they agree on its underlying byte sequence (Figure 2). Byte-Prefix Marginalization (BPM) transforms the teacher's next-token distribution into a distribution over the student vocabulary. Specifically, the probability mass of each teacher token is assigned to the longest student token whose byte representation is a prefix of the teacher token's byte representation. The mass assigned to the same student token is then aggregated across teacher tokens, while any mass for which no such student-token prefix exists is allocated to an explicit residual category. This residual is essential for mass preservation: without it, unmatched teacher mass would be discarded and the transformed target could sum to less than one. The resulting target therefore satisfies D1/D2/D3 by construction. At more than 99% of training positions, this target exactly recovers the teacher's byte-prefix marginal. The student is then trained on its own generated trajectories by minimizing a generalized Jensen–Shannon divergence with respect to the transformed target.

Our experiments further identify a failure mode of byte-exact distillation. At positions where the target consists solely of whitespace in code-generation data, byte-level alignment can transfer tokenizer-specific segmentation patterns rather than the teacher's predictive behavior. For one teacher–student pair, this spurious supervision reduces the code pass rate from 49% to 9% (Section 4.3). We address this issue with a precomputed mask that excludes all-whitespace target positions from the distillation objective, thereby preventing the observed collapse.

In summary, our contributions are twofold:

- **Exact full-vocabulary supervision across tokenizers.** We introduce BPM, which re-expresses a teacher's next-token distribution over the student vocabulary while satisfying D1-D3 by construction. We prove that the resulting target is exact whenever the byte prefixes do not span multiple teacher tokens, a condition that holds at more than 99% of the training positions in our experiments (Proposition 1). At the remaining spanning positions, BPM uses a mass-preserving, chain-factorized lower bound.
- **A systematic evaluation across heterogeneous tokenizers.** We independently distill three teachers, including Qwen3-32B, GLM-Z1-9B-0414, and MiniMax-M2.7, into the same

thinking-mode student architecture, covering teacher–student pairs with varying tokenizer distances. All on-policy methods share a training budget; SeqKD follows its standard offline recipe with no smaller budget. The best BPM variant outperforms the strongest baseline for every teacher-student pair, closing 33.5%, 43.9%, and 34.4% of the corresponding teacher-student performance gaps.

2 Related Work

2.1 Distillation for Language Models

Language-model distillation began with matching the teacher's output distribution position by position (Hinton et al., 2015). SeqKD carried this recipe to sequence models (Kim & Rush, 2016). Other lines transfer intermediate features (Romero et al., 2015), reasoning rationales (Hsieh et al., 2023; Magister et al., 2023), or a teacher-refined training corpus (Gu et al., 2025; Xiao et al., 2025; 2026). BPM builds on two refinements of the per-position recipe. MiniLLM replaces the forward KL with a reverse KL (Gu et al., 2024). The student then concentrates on the teacher modes it can represent. Generalized on-policy distillation trains on the student's own generations (Agarwal et al., 2024; Lin et al., 2020; Li et al., 2026). Its divergence interpolates between the two KLs, and later objectives span f-divergences and skew or contrastive variants (Wen et al., 2023; Ko et al., 2024; 2025). On-policy distillation has reached production. KAT-Coder-V2 consolidates five specialized coding experts into one deployable model (KwaiKAT Team, 2026). Its successor extends the same pipeline to repository-scale agentic tasks (Huang et al., 2026). The token-level objectives in this line still compare teacher and student predictions within a shared vocabulary.

2.2 Cross-Tokenizer Distillation

Cross-tokenizer methods recover comparability through alignment, transport, projection, or span-level objectives. FuseLLM hard-aligns the two token sequences by minimum edit distance (Wan et al., 2024). The per-position comparison becomes one approximate token-to-token map. The Universal Logit Distillation loss sorts each distribution by probability and matches it rank by rank as an optimal-transport surrogate. The match discards which token each ranked probability belongs to (Boizard et al., 2024). Multi-level optimal transport extends this rank-based transport across token and sequence levels (Cui et al., 2025). The GOLD implementation adds a generalized Jensen–Shannon term to the sorted loss (Patiño et al., 2025). The term covers only the vocabulary the two models share. Dual-space distillation learns a projection into a common hidden space. The output-side correspondence is then only as reliable as the learned projection (Zhang et al., 2024). SimCT compares the two models on short continuations both tokenizers can express, and supervises only a restricted set of positions. The rest of the student vocabulary receives no signal (Sun et al., 2026). Approximate likelihood matching is the closest prior objective and also supervises across tokenizers on aligned spans (Minixhofer et al., 2025b). Its target for each span is one scalar likelihood, and a scalar cannot place mass on a specific student token. Zero-shot tokenizer transfer instead re-equips a model with a new vocabulary (Minixhofer et al., 2025a).

Inference-time vocabulary projection. GaC, EVA, and UNITE project ensemble distributions onto a shared vocabulary (Yu et al., 2024; Xu et al., 2024b; Yao et al., 2025). These projections operate only at inference time, whereas BPM constructs an explicit training distribution over the full student vocabulary.

3 Method

3.1 Cross-Tokenizer Setup

On-policy distillation queries the teacher along responses generated by the student. Let $\mathcal{V}^{\mathrm{t}}$ and $\mathcal{V}^{\mathrm{s}}$ denote the teacher and student vocabularies. At teacher position j, the teacher predicts $q_j \in \Delta(\mathcal{V}^{\mathrm{t}})$; at student position i, the student predicts $p_i \in \Delta(\mathcal{V}^{\mathrm{s}})$. When both models share a tokenizer, their positions and vocabulary coordinates coincide, giving the standard per-position objective:

$$\mathcal{L}_{\mathrm{OPD}} \;=\; \textstyle\sum_i D\big(q_i \,\|\, p_i\big). \tag{1}$$

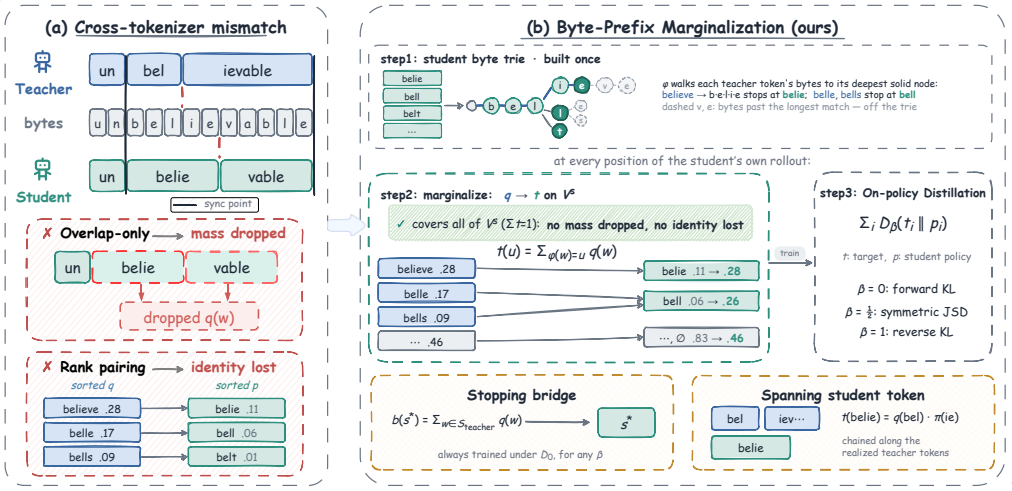


Figure 2: BPM overview. **(a)** Both models cut the same bytes into different tokens (real segmentations of our most dissimilar pair); boundaries both models share are *sync points* (black lines). **(b)** The map ϕ routes each teacher token w to the longest student token whose bytes prefix $\mathrm{bytes}(w)$. Probabilities routed to the same student token are summed into t (here $.17{+}.09$ for *bell*); unmatched mass goes to $\varnothing$, and p is trained toward t under D_β.

Two conditions make this comparison meaningful: q_i and p_i live on the same vocabulary, and index i denotes the same text prefix for both models.

Different tokenizers break both conditions. Their distributions occupy different vocabularies (*vocabulary mismatch*), while equal position indices may refer to different text prefixes because token boundaries occur at different byte offsets (*segmentation mismatch*). We therefore view each response as one byte string $\mathbf{x}$ with teacher segmentation $w_1^{\mathrm{t}}, \ldots, w_M^{\mathrm{t}}$ and student segmentation $w_1^{\mathrm{s}}, \ldots, w_N^{\mathrm{s}}$. Both decode to $\mathbf{x}$, but their boundaries and token counts differ. A boundary shared by both segmentations is a *sync point*, and the bytes between consecutive sync points form a *chunk*. At a sync point, student position i aligns with the teacher position $j(i)$ beginning at the same byte offset. Throughout, assumption (A2) models student segmentation by greedy longest-prefix matching: at each offset, the student emits the longest token in $\mathcal{V}^{\mathrm{s}}$ whose bytes prefix the remaining stream. Under this rule, a teacher prediction at an aligned position induces a distribution over the student's next token. The remaining task is to compute that distribution explicitly.

3.2 Byte-Prefix Marginalization

Every byte-level BPE token deterministically decodes to a byte string. We write $\mathrm{bytes}(w)$ for the bytes of token w, $\mathbf{c} \preceq \mathbf{c}'$ when $\mathbf{c}$ is a prefix of $\mathbf{c}'$, and $\oplus$ for byte concatenation. Although the two tokenizers produce different token sequences, they decode each response to the same byte stream. BPM uses this shared representation to marginalize teacher probability over the student token that greedy segmentation would emit first.

Definition. Fix an aligned student position i and its corresponding teacher position $j(i)$. At this position, $q_{j(i)}(w)$ assigns probability to every possible teacher token w. Each candidate w is mapped to the first student token obtained by greedily segmenting $\mathrm{bytes}(w)$. Candidates that do not contain enough bytes to determine a complete student token are assigned to an explicit residual category. The resulting target is defined as:

$$t_i(u) \;:=\; \Pr_{w \sim q_{j(i)}} \big[\text{first student token of } \mathrm{bytes}(w) = u\big]. \tag{2}$$

Thus, $t_i(u)$ collects the probability of every teacher-token outcome that makes u appear first under the student's greedy segmentation. This *byte-prefix marginal* re-expresses the teacher's next-token distribution on the student vocabulary without matching token ids or probability ranks.

Closed form. The marginal has a one-step closed form in the *refinement regime*, where each student token lies within a single teacher token and the student therefore segments at least as finely

as the teacher. Whenever a candidate teacher token w contains enough bytes to determine a complete student token, greedy matching selects the longest $u \in \mathcal{V}^{\mathrm{s}}$ whose bytes prefix $\mathrm{bytes}(w)$. This defines:

$$\phi(w) \;=\; \underset{u \in \mathcal{V}^{\mathrm{s}},\ \mathrm{bytes}(u) \preceq \mathrm{bytes}(w)}{\arg\max} |\mathrm{bytes}(u)|, \tag{3}$$

with $\phi(w) = \perp$ when no student token qualifies. The map ϕ can be compiled once for a tokenizer pair because it depends only on the two vocabularies. Grouping teacher outcomes by their mapped student token reduces Equation (2) to:

$$t_i(u) = \sum_{w:\ \phi(w)=u} q_{j(i)}(w), \qquad t_i(\varnothing) = \sum_{w:\ \phi(w)=\perp} q_{j(i)}(w). \tag{4}$$

The first sum pools all teacher tokens routed to u, while $\varnothing$ receives outcomes for which no complete student-token prefix exists. Every teacher token is assigned either to exactly one u or to $\varnothing$, so the transformed mass still sums to one. Stop-token mass is handled separately in Section 3.4.

Positions inside a teacher token. Refinement can place a later student position inside the same teacher token. Let $\mathbf{r}_{\mathrm{pre}}$ be the bytes already emitted since the preceding teacher boundary, and let $\pi(\mathbf{c})$ be the teacher mass on tokens beginning with $\mathbf{c}$. For each compatible token w, let $\phi_{\mathbf{r}_{\mathrm{pre}}}(w)$ be its longest student-token continuation, or $\perp$ if none exists. The conditional target is:

$$t_i(u \mid \mathbf{r}_{\mathrm{pre}}) = \frac{\sum_{w:\ \mathbf{r}_{\mathrm{pre}} \preceq \mathrm{bytes}(w),\ \phi_{\mathbf{r}_{\mathrm{pre}}}(w)=u} q(w)}{\pi(\mathbf{r}_{\mathrm{pre}})}, \qquad t_i(\varnothing \mid \mathbf{r}_{\mathrm{pre}}) = 1 - \sum_{u \in \mathcal{V}^{\mathrm{s}}} t_i(u \mid \mathbf{r}_{\mathrm{pre}}). \tag{5}$$

Each compatible teacher token is counted once, so Equation (5) is normalized; Section A gives the equivalent two-cell scatter. Together, the aligned-boundary and within-token cases cover more than 99% of training positions. Section 3.3 handles the remaining spanning cases.

3.3 Spanning-Token Correction

Refinement fails when one student token spans several teacher tokens. For example, the student tokenizer represents `</think>` as one token, whereas the teacher tokenizer splits the same bytes into three. The scatter in Equation (4) examines only teacher tokens at the aligned position, where almost no single token begins with the full bytes of `</think>`. It therefore assigns this student token near-zero mass, incorrectly teaching the student not to close its reasoning span.

The required spanning mass follows from the chain rule along the teacher's realized segmentation. Let student token u begin at teacher position j, fully cover $w^{\mathrm{t}}_j, \ldots, w^{\mathrm{t}}_{j+m-1}$, and end after a residual byte prefix $\mathbf{r}$ of the next teacher token. Write π_k for the prefix mass from Equation (5) evaluated at teacher position k. The chain value is defined as:

$$t_i(u) \;=\; \Big(\prod_{k=j}^{j+m-1} q_k\big(w^{\mathrm{t}}_k\big) \Big) \cdot \pi_{j+m}(\mathbf{r}), \tag{6}$$

which is the probability that the teacher produces $\mathrm{bytes}(u)$ through this realized token path.

Other teacher segmentations may produce the same bytes, so Equation (6) is a lower bound on the full byte marginal. We add this value to the target mass of u and subtract the same amount from the closed-form target in Equation (4), preserving total mass. Positions that neither begin at nor reach a sync point admit neither target and are excluded from the loss. Spanning corrections and excluded positions together account for less than one percent of response positions.

3.4 Training Objective

Training matches the transformed target with the student distribution through the generalized Jensen–Shannon divergence. For $0 < \beta < 1$, it is defined as:

$$D_\beta(t \,\|\, p) = \beta\, \mathrm{KL}(t \,\|\, m_\beta) + (1{-}\beta)\, \mathrm{KL}(p \,\|\, m_\beta), \tag{7}$$

with mixture $m_\beta = \beta t + (1{-}\beta)p$. We set $D_0(t\|p) = \mathrm{KL}(t\|p)$ (FKL) and use the skew reverse-KL endpoint $D_1(t\|p) = \mathrm{KL}\big(p \,\|\, (1{-}\lambda)t + \lambda p\big)$ with $\lambda = 0.1$; Section A gives their normalized-limit interpretation. The default is $\beta = \frac{1}{2}$.

Each divergence uses the student's projection $\widetilde{p}_i$ onto the target's explicit token cells plus one residual complement, as defined in Section A.

When the realized student token contains only spaces or tabs, BPM withholds the target. This mask is precomputed from the student tokenizer and applied independently of the divergence. It covers about 3.5% of aligned rows on the code half and is rare elsewhere, so in practice it acts almost entirely on code (Section 4.3).

Stop tokens require separate treatment because they carry no content bytes and therefore fall outside byte alignment. Moreover, the student's realized stop token need not equal the tokenizer's declared `eos`. We collect teacher and student stop sets $\mathcal{S}^{\mathrm{t}}, \mathcal{S}^{\mathrm{s}}$ from the declared `eos`, generation config, and chat template, then remove those tokens from the content byte stream. Extending ϕ with $\phi(w) = s^*$ for every $w \in \mathcal{S}^{\mathrm{t}}$ maps all teacher stop tokens to the student's realized stop token s^* for that response. The teacher's stopping probability can then join the target in Equation (4) at every aligned position and train under the same D_β as content.

If stopping has already collapsed, an explicit bridge target supplies a restoring signal. At the position i that emits s^*, BPM defines:

$$b_i(s^*) = \sum_{w \in \mathcal{S}^{\mathrm{t}}} q_{j(i)}(w), \qquad b_i(\varnothing) = 1 - b_i(s^*). \tag{8}$$

The corresponding student projection is $\bar{p}_i = [p_i(s^*), 1 - p_i(s^*)]$. The bridge always trains under D_0, regardless of the run's β, because only this endpoint retains a restoring gradient after stopping collapses (Section B). A healthy pipeline does not depend on the bridge.

Finally, let $\mathcal{R}$ collect the unmasked content positions and $\mathcal{R}'$ the stop positions of every response in the batch. The normalizer Z counts all student response tokens, including masked and excluded positions that appear in neither sum. The complete objective is defined as:

$$\mathcal{L} \;=\; \tfrac{1}{Z} \textstyle\sum_{i \in \mathcal{R}} D_\beta(t_i \,\|\, \widetilde{p}_i) \;+\; \tfrac{1}{Z} \sum_{i \in \mathcal{R}'} D_0(b_i \,\|\, \bar{p}_i). \tag{9}$$

4 Experiments

4.1 Experimental Setup

Teacher–student pairs. To isolate the effect of the objective, every on-policy arm within a teacher–student pair uses the same student, prompt set, schedule, and update budget. Across the three pairs, only the teacher changes. We use Qwen3.5-2B (Qwen Team, 2026) as the fixed student, running this 248k-token-vocabulary model in thinking mode for every rollout and evaluation. Tokenizer distance increases across the teachers. Qwen3-32B (Qwen Team, 2025) shares the student's family and has span Jaccard 0.92 between segmentations. GLM-Z1-9B-0414 (GLM Team, 2024; Z.ai, 2025) comes from a different family yet stays close in segmentation (span Jaccard 0.87), whereas MiniMax-M2.7 (MiniMax, 2026) sits farthest (0.81). We abbreviate the second teacher as GLM-Z1-9B and denote the pairs P1, P2, and P3, respectively, in later figures and tables. Because teacher capability also varies, all comparisons are made within a pair.

Training data. Training uses $20{,}000$ prompts, split evenly between mathematics and code. We draw 10k problems with exact integer answers from DAPO-Math (Yu et al., 2025) and 10k TACO problems (Li et al., 2023), each with a recovered executable test suite.

Implementation details. Except for SeqKD, every arm trains on-policy. Rollouts are sampled from the student at temperature 1.0 under its release decode settings, with a maximum length of $27{,}648$ tokens. After each response is re-rendered in the teacher's chat template, the teacher scores the same generated content. The on-policy arms train for two epochs at batch size 256 with a constant learning rate of $5{\times}10^{-7}$ on H200 SXM nodes. SeqKD follows its offline recipe with no smaller budget (Section F). Unless stated otherwise, BPM uses $\beta = \frac{1}{2}$ and the whitespace-row mask (ablated below).

Baselines. We compare against the three cross-tokenizer baselines from Related Work and a sequence-level reference. The on-policy baselines retain their published losses and coefficients under the shared schedule. Rank-based **ULD** (Boizard et al., 2024) matches sorted probability vectors under a transport loss. Hybrid **GOLD** (Patiño et al., 2025) applies a generalized Jensen–Shannon divergence

Table 1: Main results: avg@8 and pass@8 (%). Tinted rows report teacher performance; the base row is the shared undistilled student. For BPM, we report forward KL (FKL), which performs best among its three divergence variants on all three pairs (Table 2). Within each teacher block, dark shading and boldface mark the best distilled result in each metric column, while light shading and underlining mark the second best. Gray Δ rows report BPM minus the strongest baseline in each column.

Method	Mathematics						Code						Avg.	
	AIME 2026		HMMT 2026		MATH-500		HumanEval+		LiveCodeBench		TACO			
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Qwen3.5-2B (base)	7.5	26.7	10.6	18.2	60.5	84.6	52.5	84.0	11.6	17.6	5.3	11.7	24.7	40.5
Qwen3-32B	71.7	86.7	47.3	63.6	93.7	97.2	82.2	96.9	56.5	73.1	53.2	64.3	67.4	80.3
SimCT	35.8	56.7	18.9	30.3	**83.4**	93.6	56.4	87.7	10.6	24.7	6.7	21.2	35.3	52.4
ULD	7.5	20.0	6.1	15.2	60.7	89.0	55.9	79.8	12.0	25.3	8.2	19.1	25.1	41.4
GOLD	12.5	33.3	6.8	21.2	71.0	91.2	**62.1**	**89.6**	14.1	23.6	11.7	24.7	29.7	47.3
SeqKD	22.5	43.3	17.0	30.3	77.8	94.2	50.8	81.0	**18.8**	29.7	**15.7**	**35.7**	33.8	52.4
BPM	**37.5**	**73.3**	**23.5**	**39.4**	82.1	**96.4**	61.8	88.3	15.7	**33.0**	13.3	35.0	**39.0**	**60.9**
Δ vs. best baseline	+1.7	+16.6	+4.6	+9.1	−1.3	+2.2	−0.3	−1.3	−3.1	+3.3	−2.4	−0.7	+3.7	+8.5
GLM-Z1-9B	63.3	90.0	33.3	48.5	92.3	97.4	89.8	96.9	45.6	63.2	53.5	64.0	63.0	76.7
SimCT	15.8	30.0	12.5	27.3	58.6	89.2	50.2	78.5	12.6	23.6	8.4	21.2	26.3	45.0
ULD	17.5	40.0	12.5	24.2	81.2	94.4	66.0	88.3	15.8	25.8	12.4	27.2	34.2	50.0
GOLD	22.1	53.3	17.4	33.3	81.8	**96.4**	63.0	89.0	9.0	23.6	4.8	15.5	33.0	51.9
SeqKD	23.3	50.0	14.8	27.3	74.4	93.4	60.1	87.1	20.1	31.3	**16.8**	34.3	34.9	53.9
BPM	**35.4**	**63.3**	**21.2**	**36.4**	**84.8**	95.4	**68.8**	**91.4**	**22.3**	**33.5**	16.5	**36.4**	**41.5**	**59.4**
Δ vs. best baseline	+12.1	+10.0	+3.8	+3.1	+3.0	−1.0	+2.8	+2.4	+2.2	+2.2	−0.3	+2.1	+6.6	+5.5
MiniMax-M2.7	75.4	93.3	48.1	69.7	94.1	98.4	90.7	97.5	42.3	64.8	41.5	59.4	65.4	80.5
SimCT	3.3	10.0	6.1	15.2	61.2	82.2	61.2	85.9	11.3	18.1	7.0	14.8	25.0	37.7
ULD	15.0	36.7	10.6	21.2	76.5	91.2	63.9	86.5	17.3	25.3	13.6	26.9	32.8	48.0
GOLD	9.2	30.0	6.1	18.2	69.6	87.6	63.5	89.0	11.5	18.7	5.2	11.3	27.5	42.5
SeqKD	18.3	40.0	11.0	24.2	64.5	90.6	46.4	81.0	12.4	23.1	6.0	14.1	26.4	45.5
BPM	**25.4**	**46.7**	**18.6**	**39.4**	**81.8**	**95.6**	**67.1**	**92.0**	**21.8**	**34.1**	**17.4**	**34.6**	**38.7**	**57.1**
Δ vs. best baseline	+7.1	+6.7	+7.6	+15.2	+5.3	+4.4	+3.2	+3.0	+4.5	+8.8	+3.8	+7.7	+5.9	+9.1

on surface-matched tokens and the sorted loss on the remainder. Overlap-based **SimCT** (Sun et al., 2026) applies reverse KL on the span-aligned shared-surface vocabulary. As the separately trained sequence-level reference, **SeqKD** (Kim & Rush, 2016) fine-tunes the student on teacher-generated responses to the same prompts, without constructing a distributional target. We evaluate BPM with the three instantiations of Equation (7) (FKL, JSD, RKL) while holding the target fixed.

Evaluation protocol. Evaluation spans three mathematics and three code benchmarks. A symbolic verifier scores mathematics by accepting any equivalent expression, whereas code is evaluated by sandboxed execution against each benchmark's official tests and limits. The mathematics suite comprises MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) and the February 2026 AIME and HMMT contests (Balunović et al., 2025). For code, we use HumanEval+ (Liu et al., 2023), the most recent official LiveCodeBench window (Jain et al., 2025), and the official TACO test split (Li et al., 2023) at its EASY/MEDIUM labels. From 8 samples per problem, we report **avg@8** and **pass@8**; each arm is represented by the checkpoint with the highest six-benchmark average. Decoding settings, metric definitions, and the paired-bootstrap protocol appear in Sections F and H.4.

4.2 Distillation Performance

Under the matched budget shared by the on-policy arms, the best BPM variant beats all three distributional baselines on every pair and also exceeds the separately trained SeqKD reference (Tables 1 and 2). Across P1–P3, it closes 33.5%, 43.9%, and 34.4% of the teacher–student gap, compared with 19.9–26.6% for the strongest baselines (Figure 3d). Improvements are largest on the contest benchmarks.

Even the default $\beta=\frac{1}{2}$ arm outperforms the strongest baseline on every pair, and it stays ahead on the most dissimilar pair (33.8 against ULD's 32.8). With the best divergence selected, the six-benchmark margins reach +3.7, +6.6, and +5.9 points (Table 1). All three reported best-arm average intervals exclude zero; Section H.4 specifies their comparators.

Two data ablations on the GLM-Z1-9B pair test whether the divergence ranking depends on the original prompt set (Table 3). Replacing the mathematics half with harder competition-style problems shifts the six-benchmark average by at most two points (−2.1 under FKL, +0.3 under JSD, +0.6 under RKL), while expanding the corpus from 20k to 35.5k prompts (1.78×) shifts it by at most one

Table 2: Divergence ablation for BPM: forward KL (FKL, β=0), JSD (β=$\frac{1}{2}$), and reverse KL (RKL, β=1, skew 0.1). Math and Code average the corresponding three benchmarks in Table 1; All averages all six. Tinted rows report teacher performance, and highlighting follows Table 1. The P3 RKL result is measured at its pre-collapse peak (Section H.3).

	Math (3)		Code (3)		All (6)	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Qwen3-32B	70.9	82.5	64.0	78.1	67.4	80.3
FKL	**47.7**	**69.7**	**30.3**	**52.1**	**39.0**	**60.9**
JSD	46.1	65.9	25.5	49.7	35.8	57.8
RKL	47.0	67.0	26.6	49.6	36.8	58.3
GLM-Z1-9B	63.0	78.6	63.0	74.7	63.0	76.7
FKL	**47.1**	**65.0**	**35.9**	**53.8**	**41.5**	**59.4**
JSD	43.1	60.7	34.2	50.8	38.6	55.8
RKL	42.5	59.4	31.1	49.9	36.8	54.6
MiniMax-M2.7	72.5	87.1	58.2	73.9	65.4	80.5
FKL	41.9	**60.6**	**35.4**	**53.6**	**38.7**	**57.1**
JSD	**43.7**	59.1	23.8	48.8	33.8	54.0
RKL	38.5	59.8	18.6	41.7	28.5	50.7

Table 3: Data ablations on the GLM-Z1-9B pair, reported as six-benchmark avg@8 and pass@8. FKL, JSD, and RKL are the three BPM divergence variants; Avg. averages their columns. Gray Δ rows give the change from the original mix, and highlighting follows Table 1.

	FKL	JSD	RKL	Avg.
Original mix (20*k*)				
avg@8	**41.5**	38.6	36.8	39.0
pass@8	**59.4**	55.8	54.6	56.6
Harder-math mix (20*k*)				
avg@8	39.4	**38.9**	37.4	38.6
Δ	-2.1	$+0.3$	$+0.6$	-0.4
pass@8	57.1	57.0	**55.1**	56.4
Δ	-2.3	$+1.2$	$+0.5$	-0.2
Expanded corpus (35.5*k*)				
avg@8	40.7	**38.9**	**37.6**	**39.1**
Δ	-0.8	$+0.3$	$+0.8$	$+0.1$
pass@8	58.4	**58.3**	53.9	**56.9**
Δ	-1.0	$+2.5$	-0.7	$+0.3$

(-0.8, $+0.3$, $+0.8$). Averaged across the three divergences, the respective changes are only -0.4 and $+0.1$. Both perturbations lower FKL and raise JSD and RKL, narrowing the gaps without changing the FKL $>$ JSD $>$ RKL ordering. At this budget, divergence choice therefore matters more than either data change.

Beyond accuracy, Table 4 scores four distributional-closeness quantities on the GLM-Z1-9B pair. Three of them reward any student the teacher finds predictable, and a student collapsed into repetition is easy to predict. The most collapsed baseline (SimCT) therefore posts the best-looking teacher bits-per-byte (0.155). A collapsed student cannot game the fourth quantity, its own likelihood of teacher-written text. That quantity inverts the order. BPM ranks first among the distilled arms (0.237), while SimCT falls below the base (0.248 against 0.242). Section H.2 extends the comparison to the other two pairs.

Table 4: Distributional closeness on the GLM-Z1-9B pair. Teacher BpB, shared teacher mass, and byte-level top-1 agreement are measured on student rollouts; student BpB is measured on teacher-written text. Lower BpB and higher shared mass or agreement are the nominal closer-to-teacher directions.

Method	Tea. BpB ↓	Shared ↑	Byte agr. ↑	Stu. BpB ↓
Base	0.449	0.930	0.933	0.242
SimCT	0.155	0.952	0.977	0.248
ULD	0.231	0.933	0.963	0.241
GOLD	0.226	0.954	0.969	0.238
BPM	0.250	0.943	0.964	0.237

The marginalized stopping term of Section 3.4 folds the teacher's end-of-turn decision into the target, and the students inherit that decision. On the Qwen3-32B pair, BPM cuts the fraction of AIME 2026 responses that hit the length cap from 97% to 34%, against 49–62% for the baselines. Response lengths otherwise stay unchanged. GLM-Z1-9B never caps, and its BPM student reaches 8%. MiniMax-M2.7 runs 55% of its own HMMT responses to the cap, and its students stay above 60%.

4.3 Whitespace Collapse

Among code examples in the mixed on-policy rollouts, executable correctness rises through the first epoch before falling in the second. For the GLM-Z1-9B pair, the decline begins at step 113 of 156 and settles below 0.06 (Figure 3). Meanwhile, mathematics accuracy on the same rollouts remains stable, and the generated code remains syntactically valid. Closer inspection traces this selective loss of executable correctness to indentation.

The collapse concentrates at all-whitespace rows, where byte-exact supervision no longer conveys a lexical content preference. At a content row, target mass away from the student's realized token

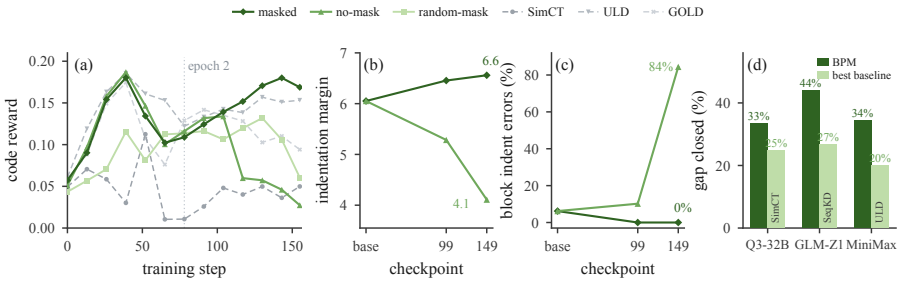


Figure 3: (a–c) Whitespace collapse for GLM-Z1-9B BPM-JSD with/without whitespace-row masking: (a) rollout code reward (9-step average; dotted: epoch 2), (b) mean indentation margin over 1,799 positions, and (c) block indentation errors. (d) Teacher–student avg@8 gap closure for each pair's best BPM arm and strongest baseline (named in bars).

indicates what the teacher would rather write. At an all-whitespace row, by contrast, this mass reflects tokenizer-specific choices among whitespace continuations, conflating token segmentation with indentation depth. During the collapse on the GLM-Z1-9B pair, the target retains 73% of its mass on the realized token on average, but this share falls below half on a quarter of the rows. The next-token entropy at these rows is only 0.53.

Training toward this target erodes the student's indentation margins, and line-level errors accumulate across each program. Block-level indentation errors rise to 84%, while the pass rate falls from 49% to 9% (Figure 3b,c). Masking all-whitespace rows prevents the collapse: although the mask removes only about 3.5% of aligned code rows, the otherwise identical masked arm finishes with an indentation margin of 6.57 and no block-level indentation errors.

Three controls isolate the effect. The masked and unmasked arms differ only in the whitespace-row mask. Masking the same number of random non-whitespace rows per response, matched for loss reduction, does not rescue the arm; executable correctness still declines late and ends at a 0.11 tail mean against 0.17. SimCT, ULD, and GOLD show no comparable collapse under the same mix, and none receives the byte-exact target.

5 Conclusion

We distill on-policy across a tokenizer boundary. BPM marginalizes the teacher's next-token distribution over byte prefixes and re-expresses it on the student's vocabulary. Each teacher token's mass is routed to its longest student-token byte prefix or to the residual, conserving total mass. Our experiments distill Qwen3-32B, GLM-Z1-9B, and MiniMax-M2.7 into one thinking-mode student. The three teachers sit at increasing tokenizer distance from the student. Under the budget shared by the on-policy arms, BPM beats the overlap, rank, and hybrid baselines on every pair and also exceeds the separately trained SeqKD reference. The margins over each pair's strongest baseline run from $+3.7$ to $+6.6$ avg@8 points and are widest on contests held after every teacher's training cutoff. The target also carries the teacher's decision to stop, and the students inherit it. Fidelity backfires where the bytes carry no content. At whitespace-only code positions the target encodes only segmentation noise, and training on that noise collapses executable code. A precomputed whitespace-row mask removes the collapse. Both the routing map and the mask come from the two tokenizers alone, before training. Other cross-tokenizer objectives will face the same choice of where to apply fidelity and where to withhold it.

Limitations. First, BPM discards part of the gradient. The whitespace-row mask withholds the target on about 3.5% of aligned code rows. Positions that reach no sync point leave the loss entirely. Whatever signal these rows carry never reaches the student. Second, data quality bounds what our experiments can show. Whether the remaining teacher–student gap survives higher-quality data and tasks is untested. Such data can sharpen the student's next-token prediction, and sharper prediction amplifies on-policy distillation.

AI USE STATEMENT

We used generative AI tools to draft and polish the appendix. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 4.1 and Table F.1 list the full training, rollout, and evaluation configuration. Section B gives the training algorithm, and Section F the precomputation, the per-step dispatch, and the fidelity checks. Section 4.1 and Section H.2 specify the training-mix construction and its two-stage decontamination. The benchmark harnesses, exclusions, and the bootstrap protocol appear in the appendix. Code will be publicly released.

REFERENCES

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

Anshumann, Mohd Abbas Zaidi, Akhil Kedia, Jinwoo Ahn, Taehwak Kwon, Kangwook Lee, Haejun Lee, and Joohyung Lee. Sparse logit sampling: Accelerating knowledge distillation in llms. *arXiv preprint arXiv:2503.16870*, 2025.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, 2022. doi: 10.18653/v1/2022.findings-acl.58.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. `https://matharena.ai/`, 2025.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: the universal logit distillation loss for llms. *Transactions on Machine Learning Research (TMLR)*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *arXiv preprint arXiv:2103.06874*, 2022.

Xiao Cui, Mo Zhu, Yulei Qin, Liang Xie, Wengang Zhou, and Houqiang Li. Multi-level optimal transport for universal cross-tokenizer knowledge distillation on language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23724–23732, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

GLM Team. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Miniplm: Knowledge distillation for pre-training language models. *arXiv preprint arXiv:2410.17215*, 2025.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

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

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

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

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023.

Bo Huang et al. Kat-coder-v2.5 technical report. *arXiv preprint arXiv:2607.05471*, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1317–1327, 2016.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. Distillm-2: A contrastive approach boosts the distillation of llms. *arXiv preprint arXiv:2503.07067*, 2025.

KwaiKAT Team. Kat-coder-v2 technical report. *arXiv preprint arXiv:2603.27703*, 2026.

Minchong Li, Feng Zhou, and Xiaohui Song. Bild: Bi-directional logits difference loss for large language model distillation. *arXiv preprint arXiv:2406.13555*, 2025.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. *arXiv preprint arXiv:2009.07253*, 2020.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Kevin Lu and Thinking Machines Lab. On-policy distillation. `https://thinkingmachines.ai/blog/on-policy-distillation/`, 2025.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2023.

MiniMax. The MiniMax-M2 series: Mini activations unleashing max real-world intelligence. *arXiv preprint arXiv:2605.26494*, 2026.

Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. *arXiv preprint arXiv:2405.07883*, 2025a.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024.

Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Unlocking on-policy distillation for any model family. Hugging Face research article, `https://huggingface.co/spaces/HuggingFaceH4/on-policy-distillation`, 2025. Introduces General On-Policy Logit Distillation (GOLD); implementation in TRL.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qwen Team. Qwen3.5-2b. `https://huggingface.co/Qwen/Qwen3.5-2B`, 2026.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. *arXiv preprint arXiv:2012.15613*, 2021.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2016.

Jie Sun, Mao Zheng, Mingyang Song, Qiyong Zhong, Yilin Cheng, Bichuan Feng, Pengfei Liu, Junfeng Fang, and Xiang Wang. Simct: Recovering lost supervision for cross-tokenizer on-policy distillation. *arXiv preprint arXiv:2605.07711*, 2026.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-divergence minimization for sequence-level knowledge distillation. *arXiv preprint arXiv:2307.15190*, 2023.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, et al. UI-Genie: A self-improving approach for iteratively boosting MLLM-based mobile GUI agents. *Advances in Neural Information Processing Systems*, 38:166978–167013, 2025. doi: 10.52202/085713-5028.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. UI-Mem: Self-evolving experience memory for online reinforcement learning in mobile GUI agents. *arXiv preprint arXiv:2602.05832*, 2026.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024a.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. Bridging the gap between different vocabularies for LLM ensemble. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 7140–7152, 2024b.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. *arXiv preprint arXiv:2503.02951*, 2025.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*, 2022.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. *arXiv preprint arXiv:2410.03777*, 2025.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *arXiv preprint arXiv:2305.07185*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yao-Ching Yu, Chun-Chih Kuo, Ziqi Ye, Yu-Cheng Chang, and Yueh-Se Li. Breaking the ceiling of the LLM community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Z.ai. GLM-Z1-9B-0414: Model card. Hugging Face model card, `https://huggingface.co/zai-org/GLM-Z1-9B-0414`, 2025.

Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 18164–18181, 2024.

The appendix follows the dependency chain behind the main result. We first restate the notation and target definitions (Section A), turn them into a complete training algorithm (Section B), and then establish the exactness and mass-conservation claims (Section C). The segmentation dispatch is made concrete through worked examples (Section D), while a fixed teacher distribution exposes the differences among target constructions (Section E). Finally, we document implementation cost, prompts, and evaluation details (Sections F and G) before expanding the empirical results (Section H).

A Notation Recap and Restated Definitions

To make the later derivations self-contained, we begin with the shared byte representation. The teacher T has vocabulary $\mathcal{V}^{\mathrm{t}}$, the student S has vocabulary $\mathcal{V}^{\mathrm{s}}$, and every token w from either byte-level BPE tokenizer decodes losslessly to $\mathrm{bytes}(w)$. We write $\mathbf{c} \preceq \mathbf{c}'$ when $\mathbf{c}$ is a byte prefix of $\mathbf{c}'$, and $\oplus$ for byte concatenation. At teacher position j, the teacher predicts $q_j \in \Delta(\mathcal{V}^{\mathrm{t}})$; at student position i, the student predicts $p_i \in \Delta(\mathcal{V}^{\mathrm{s}})$. A byte offset shared by both token boundaries is a *sync point*. Student position i at such an offset aligns with teacher position $j(i)$, and consecutive sync points delimit a *chunk*. The teacher's *prefix mass* at position k is defined as:

$$\pi_k(\mathbf{c}) \;=\; \sum_{w \in \mathcal{V}^{\mathrm{t}}:\, \mathbf{c} \preceq \mathrm{bytes}(w)} q_k(w), \tag{A.1}$$

the probability that its next token starts with $\mathbf{c}$; we drop the subscript when the position is clear.

With this notation fixed, the refinement target can be restated in three steps. First, the longest-prefix map from Equation (3) is defined as:

$$\phi(w) \;=\; \operatorname*{arg\,max}_{u \in \mathcal{V}^{\mathrm{s}},\ \mathrm{bytes}(u) \preceq \mathrm{bytes}(w)} |\mathrm{bytes}(u)|, \tag{A.2}$$

with $\phi(w) = \perp$ when no student token qualifies. Second, the target at an aligned student position i from Equation (2) is defined as:

$$t_i(u) \;:=\; \Pr_{w \sim q_{j(i)}}\big[\text{first student token of } \mathrm{bytes}(w) = u\big], \tag{A.3}$$

Marginalizing the teacher outcomes through ϕ gives the third step, the closed scatter form from Equation (4):

$$t_i(u) = \sum_{w:\, \phi(w) = u} q_{j(i)}(w), \qquad t_i(\varnothing) = \sum_{w:\, \phi(w) = \perp} q_{j(i)}(w). \tag{A.4}$$

The scatter above covers tokens resolved within one teacher step. When a student token u instead starts at teacher position j, covers realized teacher tokens $w_j^{\mathrm{t}}, \ldots, w_{j+m-1}^{\mathrm{t}}$, and extends through a residual byte prefix $\mathbf{r}$ of the next token, it receives the chain value from Equation (6):

$$t_i(u) \;=\; \Big(\prod_{k=j}^{j+m-1} q_k\big(w_k^{\mathrm{t}}\big)\Big) \cdot \pi_{j+m}(\mathbf{r}), \tag{A.5}$$

For a student position beginning *inside* a teacher token after realized bytes $\mathbf{r}_{\mathrm{pre}}$, let $\phi_{\mathbf{r}_{\mathrm{pre}}}(w)$ return the longest student token that prefixes the remaining bytes of compatible teacher token w, or $\perp$ if none exists. The exact conditional scatter is defined as:

$$\begin{aligned} t_i(u \mid \mathbf{r}_{\mathrm{pre}}) &= \frac{\sum_{w:\, \mathbf{r}_{\mathrm{pre}} \preceq \mathrm{bytes}(w),\ \phi_{\mathbf{r}_{\mathrm{pre}}}(w) = u} q(w)}{\pi(\mathbf{r}_{\mathrm{pre}})}, \\ t_i(\varnothing \mid \mathbf{r}_{\mathrm{pre}}) &= \frac{\sum_{w:\, \mathbf{r}_{\mathrm{pre}} \preceq \mathrm{bytes}(w),\ \phi_{\mathbf{r}_{\mathrm{pre}}}(w) = \perp} q(w)}{\pi(\mathbf{r}_{\mathrm{pre}})}. \end{aligned} \tag{A.6}$$

The two cells partition the conditioned teacher support and therefore sum to one without counting nested student-token prefixes more than once. Finally, let $\mathcal{S}^{\mathrm{t}}$ and $\mathcal{S}^{\mathrm{s}}$ be the declared stop sets and s^* the student's realized stop token for the response at hand. The stop bridge and its student-side binary projection from Equation (8) are defined as:

$$\begin{aligned} b_i(s^*) &= \sum_{w \in \mathcal{S}^{\mathrm{t}}} q_{j(i)}(w), \quad b_i(\varnothing) = 1 - b_i(s^*), \\ \bar{p}_i(s^*) &= p_i(s^*), \qquad\qquad \bar{p}_i(\varnothing) = 1 - p_i(s^*). \end{aligned} \tag{A.7}$$

Divergence endpoints and projected support. The generalized Jensen–Shannon expression in Equation (7) is used for $0 < \beta < 1$. Because it vanishes at the endpoints, FKL is defined by the normalized limit $D_0(t\|p) := \lim_{\beta\to 0} D_\beta(t\|p)/\beta = \mathrm{KL}(t\|p)$. The other normalized limit, $\lim_{\beta\to 1} D_\beta(t\|p)/(1-\beta) = \mathrm{KL}(p\|t)$, motivates the numerically stable skew-RKL endpoint:

$$D_1(t\|p) := \mathrm{KL}(p \,\|\, (1-\lambda)t + \lambda p)\,, \qquad \lambda = 0.1. \tag{A.8}$$

For a content row, let $\mathcal{I}_i$ contain the student tokens represented explicitly by its routing map and any spanning correction. The student distribution is projected onto these cells and one complement coordinate:

$$\widetilde{p}_i(u) = p_i(u) \quad (u \in \mathcal{I}_i), \qquad \widetilde{p}_i(\varnothing) = \sum_{v\in\mathcal{V}^{\mathrm{s}}\setminus\mathcal{I}_i} p_i(v) = 1 - \sum_{u\in\mathcal{I}_i} p_i(u). \tag{A.9}$$

Thus t_i and $\widetilde{p}_i$ have identical support and retain all teacher and student mass. The binary stop projection $\bar{p}_i$ is defined separately in Equation (A.7).

B Training Algorithm

Algorithm 1 gives the full training procedure: the per-pair precomputation, the per-response byte alignment, the per-position dispatch among the target constructions of Section A, the whitespace-row mask, and the loss. The dispatch mirrors the case table of Section D (Table D.1).

Stopping-bridge gradient. Stop positions train under $D_0 = \mathrm{KL}(b_i \,\|\, \bar{p}_i)$ rather than the generalized Jensen–Shannon divergence D_β (Equation (7)) because only D_0 keeps a restoring gradient once the student's stop probability has collapsed. Take the gradient of the divergence in the student logit of the realized stop token s^* as $p_i(s^*) \to 0$: under D_β with $\beta \in (0,1)$ it vanishes, and a suppressed stop token receives no signal to recover, whereas under D_0 it approaches $-b_i(s^*)$, bounded away from zero whenever the teacher places stop mass at the aligned position (Equation (A.7)). The ϕ stop-extension (Section 3.4) already folds termination into the content target on a healthy run; the bridge adds the D_0 term to restore stopping in the suppressed regime.

C Exactness of the Byte-Prefix Marginal

Two questions underlie the closed form in Equation (A.4): whether it preserves all teacher mass and whether it equals the byte-prefix marginal. We answer them in that order. After stating the byte-decoding and greedy-segmentation assumptions, we express the target as masses on a byte trie and prove that those masses telescope to one (Lemma 1). The scatter proposition then identifies this construction with the desired marginal (Proposition 1), before the final paragraph discusses the remaining gap between greedy longest-prefix matching and true BPE merge order.

Assumptions.

- (A1) Every token w decodes to a byte string $\mathrm{bytes}(w)$, and decoding a token sequence concatenates the byte strings losslessly (byte-level BPE).
- (A2) At any byte position, the student tokenizer emits the longest token in $\mathcal{V}^{\mathrm{s}}$ whose bytes prefix the remaining byte stream (greedy longest-prefix matching).

(A1) holds for the tokenizers we use by construction. (A2) idealizes BPE merge order; we argue its gap is second-order without proving a bound, and the end of this section reports that in position-by-position cross-checks we could not distinguish the two from exact tokenizer behavior.

Trie node-mass form. Fix an aligned student position i and abbreviate $q = q_{j(i)}$, $\pi = \pi_{j(i)}$ (Equation (A.1)). Arrange $\mathcal{V}^{\mathrm{s}}$ as a byte trie and let $\mathrm{tok}(\mathbf{c}) \in \mathcal{V}^{\mathrm{s}} \cup \{\perp\}$ denote the longest student token whose bytes prefix $\mathbf{c}$'s path. The node-mass target assigns to each node $\mathbf{c}$ the mass $\rho(\mathbf{c})$ that ends exactly there:

$$\rho(\mathbf{c}) \;=\; \pi(\mathbf{c}) - \sum_{b=0}^{255} \pi(\mathbf{c}\oplus b), \tag{C.1}$$

Algorithm 1 BPM: precomputation and one on-policy training step

Require: teacher q over $\mathcal{V}^{\mathrm{t}}$, student p_θ over $\mathcal{V}^{\mathrm{s}}$, prompt batch $\mathcal{B}$, mixing weight β

Precompute once per tokenizer pair:

build the byte trie of $\mathcal{V}^{\mathrm{s}}$; for each $w \in \mathcal{V}^{\mathrm{t}}$, descend $\mathrm{bytes}(w)$ to set $\phi(w)$ per Equation (A.2)

read stop sets $\mathcal{S}^{\mathrm{t}}, \mathcal{S}^{\mathrm{s}}$ from tokenizer, generation configuration, and chat template

One training step:

$\mathcal{L} \leftarrow 0$

for each prompt $x \in \mathcal{B}$ **do**

  sample response $y \sim p_\theta(\cdot \mid x)$ ▷ on-policy rollout

  teacher forward on y re-rendered in the teacher's chat template $\rightarrow q_1, \ldots, q_M$; student forward $\rightarrow p_1, \ldots, p_N$

  strip stop-token bytes from y; extend $\phi(w) \leftarrow s^*$ for all $w \in \mathcal{S}^{\mathrm{t}}$, where $s^* \in \mathcal{S}^{\mathrm{s}}$ ends y

  compute byte offsets of both segmentations; mark sync points and chunks

  for each student position $i = 1, \ldots, N$ **do**

    if i is aligned and its token ends at or inside the aligned teacher token **then**

      $t_i \leftarrow$ scatter of $q_{j(i)}$ through ϕ ▷ Equation (A.4); 1:1 and 1:N head

    else if i starts inside a teacher token and its token ends at or inside that teacher token, realized prefix $\mathbf{r}_{\mathrm{pre}}$ **then**

      $t_i \leftarrow$ compatible teacher mass, normalized by $\pi(\mathbf{r}_{\mathrm{pre}})$, scattered through $\phi_{\mathbf{r}_{\mathrm{pre}}}$ ▷ Equation (A.6); 1:N interior

    else if the token at i spans one or more teacher boundaries **then**

      $t_i \leftarrow$ scatter of $q_{j(i)}$ through ϕ ▷ base row, Equation (A.4)

      $t_i(u) \leftarrow$ chain value of Equation (A.5), conditioned on $\mathbf{r}_{\mathrm{pre}}$ if the row starts mid-token;

      subtract the shallowest candidate's chain value from the head's scatter cell (or $\varnothing$ if unmapped), split across nested candidates by telescoping differences ▷ mass-neutral; N:1 and M:N

    else

      exclude i from the loss ▷ no sync point reachable; $< 1\%$ of rows

    end if

    if i emits s^* **then**

      $b_i(s^*) \leftarrow \sum_{w \in \mathcal{S}^{\mathrm{t}}} q_{j(i)}(w)$ ▷ stop bridge, Equation (A.7); trained under D_0

      $\bar{p}_i \leftarrow [p_i(s^*),\ 1 - p_i(s^*)]$

    end if

    if the whitespace mask is enabled and the realized token at row i is entirely spaces or tabs **then**

      mask row i out of the loss ▷ whitespace mask; Section 4.3

    end if

  end for

  $\mathcal{R} \leftarrow$ unmasked content positions of y; $\mathcal{R}' \leftarrow$ stop positions of y

  project each p_i onto its explicit target cells plus the complement to obtain $\widetilde{p}_i$

  $\mathcal{L} \leftarrow \mathcal{L} + \sum_{i \in \mathcal{R}} D_\beta(t_i \,\|\, \widetilde{p}_i) + \sum_{i \in \mathcal{R}'} D_0(b_i \,\|\, \bar{p}_i)$ ▷ Equation (9)

end for

update θ with the gradient of $\mathcal{L}/Z$, where Z is the batch's total student response-token count (masked and excluded positions included)

and credits it to $\mathrm{tok}(\mathbf{c})$, the deepest student-token ancestor on the path:

$$t_i(u) = \sum_{\mathbf{c}:\, \mathrm{tok}(\mathbf{c}) = u} \rho(\mathbf{c}), \qquad t_i(\varnothing) = \sum_{\mathbf{c}:\, \mathrm{tok}(\mathbf{c}) = \perp} \rho(\mathbf{c}). \tag{C.2}$$

Lemma 1 (Mass conservation)**.** *For the assignment in Equations* (C.1) *and* (C.2)*,* $\sum_{u \in \mathcal{V}^{\mathrm{s}}} t_i(u) + t_i(\varnothing) = 1$.

Proof. Every teacher token w with $q(w) > 0$ contributes $q(w)$ to $\pi(\mathbf{c})$ for exactly the prefixes $\mathbf{c} \preceq \mathrm{bytes}(w)$, which form one root-to-node path in the byte trie. Along this path, the node masses $\rho(\mathbf{c})$ of Equation (C.1) telescope: each interior node's contribution $q(w)$ cancels between $\pi(\mathbf{c})$ and the child term $\pi(\mathbf{c} \oplus b)$ on the path, leaving $q(w)$ exactly once, at the node $\mathbf{c} = \mathrm{bytes}(w)$ (beyond which no prefix of $\mathrm{bytes}(w)$ continues). Hence $\sum_{\mathbf{c}} \rho(\mathbf{c}) = \sum_w q(w) = 1$, and Equation (C.2) partitions this sum by $\mathrm{tok}(\mathbf{c}) \in \mathcal{V}^{\mathrm{s}} \cup \{\perp\}$. □

Proposition 1 (Scatter form of the marginal)**.** *Assume (A1)–(A2). Let student position i be aligned to teacher position $j(i)$, and suppose every student token inside the chunk is contained in a single teacher token (the student refines the teacher there). Then the byte-prefix marginal of Equation* (A.3) *equals the scatter form of Equation* (A.4) *for every $u \in \mathcal{V}^{\mathrm{s}} \cup \{\varnothing\}$, and $\sum_u t_i(u) + t_i(\varnothing) = 1$.*

Table D.1: Target construction for each teacher-to-student segmentation relation. "Realized-path" targets are exact along the realized teacher segmentation and therefore lower-bound the full byte marginal for every spanning token, whether emitted or counterfactual (Section 3.3).

Relation	Example	Target	Guarantee
1:1	`at` → `at`	Equation (A.4)	exact (Proposition 1)
1:N head	`ate` → `at`·`e`	Equation (A.4)	exact (Proposition 1)
1:N int.	the `e` above	Equation (D.1)	exact
N:1	`</think>` spans 3	Equation (A.5)	realized-path
M:N	mid-start span	Equation (A.5)	realized-path

Proof. Draw $w \sim q$. Because the student refines the teacher inside the chunk, the first student token emitted at position i lies within $w^{\mathrm{t}}_{j(i)}$'s bytes, hence within $\mathrm{bytes}(w)$, and is determined by w alone. Under (A2) it is the longest student token prefixing $\mathrm{bytes}(w)$, which is $\phi(w)$ of Equation (A.2) by definition. The event in Equation (A.3) therefore coincides with $\{\phi(w) = u\}$, and taking probabilities under q yields Equation (A.4); the case $\phi(w) = \perp$ collects the teacher tokens no student token prefixes. Conservation follows either directly, since $\{\phi(w) = u\}_u \cup \{\phi(w) = \perp\}$ partitions $\mathcal{V}^{\mathrm{t}}$, or from Lemma 1: the node mass $\rho(\mathrm{bytes}(w))$ is credited to $\mathrm{tok}(\mathrm{bytes}(w)) = \phi(w)$; hence Equation (C.2) and Equation (A.4) agree token by token. □

The (A2) gap. Because BPE merge order occasionally emits a first token shorter than the longest matching prefix, (A2) is an idealization of counterfactual continuations. Two facts delimit its effect, though neither yields a proven bound. First, (A2) never changes *which* token the student emitted (that token is observed), though it can perturb the target probability the marginal assigns to it and to its byte-prefix siblings. Second, the idealization only redistributes teacher mass among student tokens whose bytes prefix one another, and in position-by-position cross-checks we could not distinguish the two from exact tokenizer behavior.

D The Four Segmentation Relations, with a Worked Example

Within a chunk, the teacher-to-student token count takes one of four relations (1:1, 1:N, N:1, M:N). Table D.1 lists the five position cases they induce—1:N contributes head and interior positions—with the target each receives and its guarantee; the paragraphs below give the constructions and a numeric walk-through. Ratios are written teacher : student.[1]

1:1 **and** 1:N **head.** When the student token ends at or inside the aligned teacher token, Proposition 1 applies and the scatter of Equation (A.4) is the exact marginal.

1:N **interior.** A student position inside a teacher token, with realized preceding bytes $\mathbf{r}_{\mathrm{pre}}$ already emitted since the teacher boundary, receives the conditional marginal: restrict the teacher distribution to the tokens whose bytes continue $\mathbf{r}_{\mathrm{pre}}$, renormalize by $\pi(\mathbf{r}_{\mathrm{pre}})$, and scatter the remaining bytes by the student's first token:

$$t_i(u \mid \mathbf{r}_{\mathrm{pre}}) = \frac{\sum_{w:\, \mathbf{r}_{\mathrm{pre}} \preceq \mathrm{bytes}(w),\ \phi_{\mathbf{r}_{\mathrm{pre}}}(w) = u} q(w)}{\pi(\mathbf{r}_{\mathrm{pre}})},$$
$$t_i(\varnothing \mid \mathbf{r}_{\mathrm{pre}}) = \frac{\sum_{w:\, \mathbf{r}_{\mathrm{pre}} \preceq \mathrm{bytes}(w),\ \phi_{\mathbf{r}_{\mathrm{pre}}}(w) = \perp} q(w)}{\pi(\mathbf{r}_{\mathrm{pre}})}. \tag{D.1}$$

capped at the chunk boundary. This is exact: it is Equation (A.3) applied to the conditional teacher distribution, with every compatible outcome assigned to one longest student-token prefix or to $\varnothing$. It restates Equation (A.6) for the 1:N interior case.

N:1 **and** M:N**.** Spanning student tokens receive the realized-path chain of Equation (A.5), applied as a mass-neutral reallocation from the head token's scatter cell to the spanning candidates, and each row thus remains a distribution over its explicit student-token cells and $\varnothing$. Positions that both start

[1] The reference implementation labels the same cases in the opposite order, student : teacher.

Table D.2: Frequency of each segmentation relation among student positions in realized BPM generations. P1/P2/P3 use 240/240/120 responses and 5.4/4.1/3.1M chunks, respectively; examples are the most frequent real instances in the measured corpora. P1 is almost entirely 1:1, P2 is dominated by $1{:}N$ splits of GLM-Z1 tokens, especially digits, and P3 has the largest spanning share. Even on P3, the exact 1:1 and $1{:}N$ cases cover 99.5% of positions.

Relation	Example (teacher → student)	P1	P2	P3
1:1	`the` → `the`	99.65	93.68	94.75
$1{:}N$	`10` → `1·0`	0.27	6.24	4.77
$N{:}1$	`bit·mask` → `bitmask`	0.06	0.06	0.37
$M{:}N$	`.f·act·orial` → `.factor·ial`	0.02	0.02	0.10

Table D.3: Worked byte-prefix marginal. Trie node masses and the ϕ-scatter agree for every student token, and $0.2{+}0.5{+}0.2{+}0.1 = 1$, confirming mass conservation. Each teacher token maps to a distinct student token here; Figure 2b shows the case in which several teacher tokens pool into one student token.

byte prefix $\mathbf{c}$	`a`	`at`	`ate`	`bit`
prefix mass $\pi(\mathbf{c})$	0.9	0.7	0.2	0.1
node mass $\rho(\mathbf{c})$	$0.9{-}0.7$	$0.7{-}0.2$	0.2	0.1
$t(u)$ via trie	0.2	0.5	0.2	0.1
$t(u)$ via ϕ-scatter	0.2	0.5	0.2	0.1

inside a teacher token and span past its end use the same chain conditioned on the observed $\mathbf{r}_{\mathrm{pre}}$; telescoping differences between nested candidates prevent prefix double-counting. Positions that neither start on a sync point nor reach one carry no exact target under the realized factorization and are excluded from the loss. On our training distribution these exclusions and all spanning corrections together account for well under one percent of response positions (measured per step: roughly $3{\times}10^4$ spanning rows against $6.6{\times}10^6$ aligned rows on the most dissimilar tokenizer pair), which is why the closed-form scatter dominates the cost. Table D.2 quantifies all four relations on each pair's realized generations.

Worked example. Take teacher distribution $q = \{\texttt{at}{:}0.5,\ \texttt{ate}{:}0.2,\ \texttt{a}{:}0.2,\ \texttt{bit}{:}0.1\}$ at an aligned position, and a student vocabulary containing `a`, `at`, `ate`, `bit`. Table D.3 computes the target twice: by trie node masses (Equations (C.1) and (C.2)) and by the scatter (Equation (A.4)). Both give the same distribution and the mass telescopes to one, illustrating Lemma 1 and Proposition 1.

For the spanning case, suppose the student emits `</think>` as one token where the teacher reads `</·think·>`, with realized-path probabilities $q_j(\texttt{</}) = 0.7$, $q_{j+1}(\texttt{think}) = 0.9$, and prefix mass $\pi_{j+2}(\texttt{>}) = 0.95$. Equation (A.5) gives $t(\texttt{</think>}) = 0.7 \times 0.9 \times 0.95 \approx 0.60$, whereas the scatter of Equation (A.4) through $\phi(\texttt{</})$ would assign the spanning token essentially zero; the on-policy student would then be trained *against* closing its own reasoning span. This failure is what the chain factorization of Section 3.3 exists to prevent.

E The Three Target Constructions on One Distribution

Figure E.1 holds one teacher distribution fixed so that only the target construction changes. Two teacher tokens, `then` and `they`, are absent from the student vocabulary. SimCT keeps only shared-surface tokens and renormalizes, leaving 44% of the original teacher mass without supervision. ULD retains all rank slots but discards token identity by sorting both distributions. In contrast, BPM routes `then` and `they` to `the`, their longest student-token byte prefix, while assigning genuinely unrepresentable mass to $\varnothing$. The comparison makes the three design requirements concrete: only the final construction preserves byte correspondence and total mass without restricting the student vocabulary.

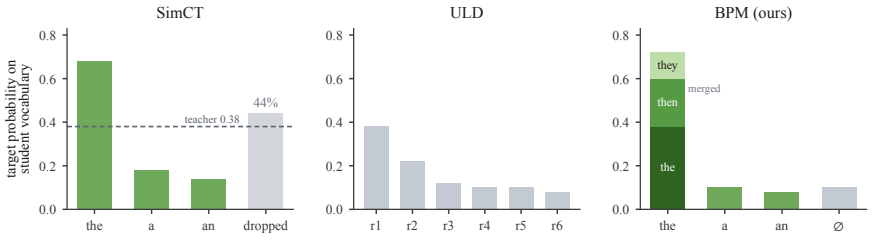


Figure E.1: Student-vocabulary targets constructed from the same teacher distribution over `the` (.38), `then` (.22), `they` (.12), `a` (.10), `an` (.08), and no-prefix mass (.10); `then` and `they` are absent from the student vocabulary, and colors track teacher-token mass. (a) SimCT retains and renormalizes only shared-surface tokens, leaving 44% of teacher mass unused. (b) ULD aligns rank slots, so student-token identity is not preserved. (c) BPM routes the absent tokens to their longest student-token byte prefix `the`, routes unrepresentable mass to $\varnothing$, and preserves total mass.

Table F.1: Training, rollout, and evaluation configuration shared by all on-policy arms. SeqKD is the exception: it fine-tunes offline for three epochs at learning rate 1×10^{-5} on its teacher-generated corpus.

Training	
Student	Qwen3.5-2B (thinking mode)
Epochs	2
Batch size	256
Learning rate	5×10^{-7}, constant
Default divergence	D_β with $\beta = \frac{1}{2}$
Whitespace-row mask	on
On-policy rollouts	
Temperature	1.0
top-p / top-k	0.95 / 20
Length cap	27,648 tokens
Evaluation decoding	
Temperature	0.6
Responses per problem	8 (scored as avg@8 / pass@8)
Hardware per run	one 8×H200 SXM node

F Dispatch and Cost

The implementation separates tokenizer-dependent work from the repeated training path. A one-time precomputation builds the byte maps, trie, and routing table; during training, most positions reduce to one batched gather-scatter, while the rare interior and spanning cases receive targeted corrections. Content positions whose remaining bytes admit exactly one student continuation produce a point-mass target and train under D_0. The following paragraphs describe this dispatch, its fidelity checks and cost, and the evaluation conventions needed to reproduce the main results.

Implementation. We implement BPM and every on-policy baseline arm on the slime training framework[2]. The teacher engine's sleep–wake scheduling follows KDFlow[3]. Table F.1 lists the configuration shared by the on-policy arms. SeqKD instead trains offline for three epochs at learning rate 1×10^{-5} on teacher-generated responses, giving it no smaller training budget.

Hardware. Each training run occupies one $8\times$H200 SXM (141 GB) node with the rollout engine, the teacher, and the trainer colocated. A few arms add data-parallel nodes of the same type for wall-clock efficiency, with the recipe otherwise unchanged.

[2] `https://github.com/THUDM/slime`
[3] `https://github.com/songmzhang/KDFlow`

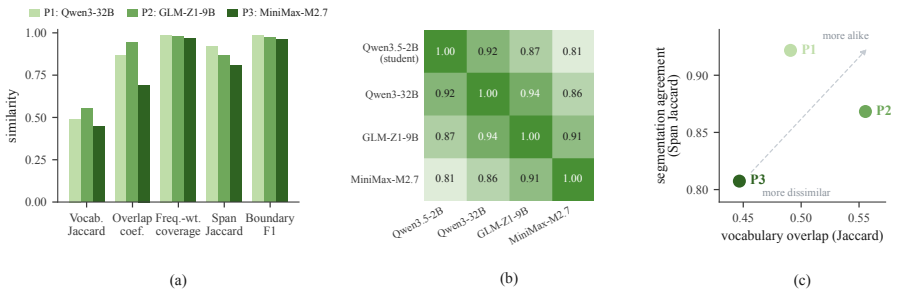


Figure F.1: Tokenizer similarity across the three teacher→student pairs. (a) Vocabulary Jaccard, overlap coefficient, frequency-weighted coverage, span Jaccard, and boundary F_1 on 2,000 training prompts. (b) Pairwise exact-span Jaccard for all four tokenizers on the same documents. (c) P1–P3 in the vocabulary-versus-segmentation similarity plane.

Rollout decoding and stop sets. On-policy rollouts sample under the student's release decode settings (Table F.1). The stop sets $\mathcal{S}^{\mathrm{t}}, \mathcal{S}^{\mathrm{s}}$ are read from each model's declared stop tokens: the tokenizer's and the generation configuration's end-of-sequence entries and the chat template's turn-end token, since the turn-end token of several chat models differs from the tokenizer's `eos` field. Stop tokens are stripped from the byte stream before alignment. Keeping their literal bytes would work against the stopping target: the teacher assigns those characters near-zero probability, and the divergence would then pull the student away from stopping and toward the length limit.

Precomputation. Per tokenizer pair, once: byte maps for both vocabularies, the student byte trie, the map ϕ (one trie descent per teacher token, $O(\sum_w |\mathrm{bytes}(w)|)$ total), and the stop sets $\mathcal{S}^{\mathrm{t}}, \mathcal{S}^{\mathrm{s}}$ from the generation configurations. The content part of ϕ is static; the stop extension $\phi(w) = s^*$ (Section 3.4) is the one response-dependent entry, patched per response in $O(|\mathcal{S}^{\mathrm{t}}|)$.

Per step. Each response position is classified by its byte offsets into the relations of Table D.1 and routed as follows: aligned and $1{:}N$-head positions to one batched gather-scatter of the teacher distributions through ϕ (Equation (A.4)); $1{:}N$-interior positions to the batched conditional marginal (Equation (D.1)); spanning positions to the chain correction (Equations (A.5) and (A.6)), applied as a sparse update on top of the scatter; stop positions to the bridge target (Equation (A.7)). The scatter dominates: it touches each aligned row once, with cost linear in the teacher vocabulary per row, and carries no byte-level traversal; only the rare interior-conditional and spanning rows perform bounded byte-trie descents, each linear in the token length. Classification itself is linear in the response length.

Fidelity. The vectorized scatter is checked against the pure byte-walk reference (Equations (C.1) and (C.2)) position by position: agreement is at floating-point tolerance on $1{:}1$ and $1{:}N$ positions, and the spanning correction reproduces the chain value on realized tokens. The reference test suite also encodes the failure mode of Section D as an assertion: a target built by first-token scatter alone must fail the spanning test, which guards against regressions that would silently reintroduce it.

Tokenizer-similarity metrics. The overlap coefficient of the Design paragraph of Section 4 is $|\mathcal{V}^{\mathrm{t}} \cap \mathcal{V}^{\mathrm{s}}| / \min(|\mathcal{V}^{\mathrm{t}}|, |\mathcal{V}^{\mathrm{s}}|)$, computed byte-exactly on token strings. Span Jaccard is the Jaccard similarity of the two tokenizers' sets of byte-offset segmentation spans on the training corpus. Figure F.1 additionally reports vocabulary Jaccard, boundary F_1, and frequency-weighted coverage on the same 2,000-prompt sample.

Decoding and metrics. Evaluation decoding uses temperature 0.6 with 8 samples per problem. **avg@8** is the mean correctness over the 8 samples (Chen et al., 2021); **pass@8** is the fraction of problems solved by at least one sample. The February 2026 AIME and HMMT contests postdate every teacher's training cutoff. Headline margins carry 95% paired-bootstrap intervals over problems (10,000 resamples), reflecting problem and sample variability in one regenerated evaluation per arm, but not training-run or checkpoint-selection uncertainty (Section H.4).

Benchmark exclusions. For HumanEval+ we exclude one problem, HumanEval/32 (`find_zero`). The exported official test special-cases this problem's oracle and unpacks the returned scalar root as an argument tuple. Every submission then fails with a `TypeError`, including the canonical solution, and the original oracle cannot be reconstructed from the export. We drop the problem and report HumanEval+ on the remaining 163 problems, which run the unmodified official tests.

The full HumanEval+ set is 164 problems, one excluded above. The LiveCodeBench slice is the most recent official window, 2025.01–2025.04 (182 problems), each run against its full official test suite under the sampling protocol above. The TACO-test count (283 problems) and its construction are given in the extended-results section below.

G Task Prompts

Training and every evaluation share one instruction template per task family, filled with the problem instance and then wrapped by the student model's own chat template with thinking enabled; responses are decoded up to 27,648 tokens. Mathematics problems end with an `Answer:` line and code problems with a single fenced program, at training and test time alike. Braces denote fields substituted per instance.

Mathematical reasoning. The verifier reads the last `\boxed{}` if present, else the string on the final `Answer:` line, and compares it to the reference with a symbolic checker. Integer answers (AIME) and the fractions, radicals, and closed forms that HMMT answers take are both handled: any expression equal to the reference counts, regardless of surface form.

Mathematical Reasoning Prompt (train = test)

```
Solve the following math problem step by step. The last line of your
response should be of the form Answer: $Answer (without quotes) where
$Answer is the answer to the problem.

{problem}

Remember to put your answer on its own line after "Answer:".
```

Code generation. The grader extracts the last syntactically valid fenced program from the response and executes it in a sandbox against the benchmark's official tests (stdin/stdout comparison or unit-test suites, per benchmark), at the official resource limits.

Code Generation Prompt (train = test)

```
You are given a competitive programming problem. Think step by step,
then provide a single complete Python program in a ```python code
block. If a starter code / function signature is given, implement it;
otherwise read from standard input and write to standard output.

{problem}
```

H Extended Results

The following analyses unpack the aggregate results from four angles: benchmark-level divergence effects, method-agnostic distributional closeness, training stability and response length, and uncertainty in the headline margins.

Table H.1: Per-benchmark avg@8 and pass@8 (%) for the three BPM divergences on each pair. Each row uses the same selected checkpoint as Table 2, and All reproduces its six-benchmark aggregate. Per-benchmark results for the undistilled student and baselines appear in Table 1.

	AIME 2026		HMMT 2026		MATH-500		HumanEval+		LiveCodeBench		TACO		All (6)	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
P1 (Qwen3-32B teacher)														
FKL	37.5	73.3	23.5	39.4	82.1	96.4	61.8	88.3	15.7	33.0	13.3	35.0	39.0	60.9
JSD	37.1	66.7	19.3	36.4	82.0	94.6	57.4	89.6	10.6	29.1	8.6	30.4	35.8	57.8
RKL	36.2	66.7	22.0	39.4	82.9	95.0	57.1	85.3	12.3	33.5	10.5	30.0	36.8	58.3
P2 (GLM-Z1-9B teacher)														
FKL	35.4	63.3	21.2	36.4	84.8	95.4	68.8	91.4	22.3	33.5	16.5	36.4	41.5	59.4
JSD	26.7	56.7	17.8	30.3	84.7	95.2	69.6	90.2	19.4	32.4	13.6	29.7	38.6	55.8
RKL	28.3	50.0	20.5	33.3	78.8	94.8	65.3	89.0	16.1	30.2	11.8	30.4	36.8	54.6
P3 (MiniMax-M2.7 teacher)														
FKL	25.4	46.7	18.6	39.4	81.8	95.6	67.1	92.0	21.8	34.1	17.4	34.6	38.7	57.1
JSD	32.5	56.7	17.0	27.3	81.7	93.4	40.6	87.7	18.1	30.2	12.6	28.6	33.8	54.0
RKL	37.9	56.7	17.8	36.4	59.7	86.2	32.5	75.5	15.6	26.9	7.6	22.6	28.5	50.7

H.1 Per-Benchmark Scores for the Divergence Comparison

Across the three pairs, divergence choice moves six-benchmark avg@8 by three to five points (Table 2), a swing comparable to the main text's margins over the baselines. Table H.1 traces each aggregate back to its six benchmark scores at the same selected checkpoints. Two patterns explain most of the movement. First, MATH-500 compresses stable arms into a 6.0-point band (78.8–84.8 avg@8), whereas AIME 2026 spreads by as much as 12.5 points within one block; contest mathematics and code therefore determine most of the divergence ranking. Second, FKL gains primarily on code. For P1, the three divergences lie within 1.6 points in mathematics average (46.1–47.7) and within 1.3 on AIME, but FKL leads the code average by 3.7–4.8 points, with the widest gap on TACO (13.3 against 8.6 and 10.5). On P2, its lead spans both halves and is anchored by AIME (35.4 against 26.7 and 28.3).

P3 admits no single divergence ranking because the arms trade benchmarks. JSD posts the block's best mathematics average (43.7; AIME 32.5 against FKL's 25.4) and its worst HumanEval+ average (40.6 against 67.1), yet HumanEval+ pass@8 remains nearly intact (87.7 against 92.0). The arm still solves most problems in at least one sample but does so less consistently, indicating a consistency loss rather than a capability loss; Table H.5 quantifies the same trade with paired intervals. The flagged RKL row shows a different failure: collapse appears unevenly across benchmarks. At its selected pre-collapse peak, RKL still posts the block's best AIME score (37.9), while MATH-500 has already fallen to 59.7 against 81.7–81.8 for its siblings. The larger 500-problem set exposes degradation not yet sampled by the 30-problem contest, and the code half trails throughout (7.6 on TACO).

The main text notes the BPM margins are widest on the contests: the best arm beats each pair's strongest AIME 2026 baseline by +1.7 (P1, SimCT at 35.8), +12.1 (P2, SeqKD at 23.3), and +7.1 (P3, SeqKD at 18.3). The +12.1 is wide partly because P2's strongest AIME baseline is anomalously weak on that benchmark; the pair itself is not harder, since AIME 2026 is the same test across all three pairs.

H.2 Method-Agnostic Distributional Closeness

We evaluate distributional closeness from four complementary directions, each computed from the two models alone and averaged over student rollouts on held-out mathematics. *Teacher bits-per-byte* is the teacher's byte-normalized negative log-likelihood of student-generated text. *Shared-mass ratio* measures teacher probability on shared-surface tokens at student-visited states, a concentration that prior on-policy-distillation work associates with success (Section 4). *Byte top-1 agreement* records how often the models' most likely next tokens begin with the same byte at shared boundaries. The reverse direction, *student bits-per-byte on teacher text*, evaluates the student's negative log-likelihood of teacher-written solutions and is not directly optimized by any on-policy arm.

The direction of evaluation matters. Teacher bits-per-byte, shared mass, and byte agreement all reward a student the teacher finds predictable, so low-entropy repetition can improve them without

Table H.2: Distributional closeness across all three pairs, averaged over student rollouts on held-out mathematics at the checkpoint selected by six-benchmark avg@8. Unlabeled BPM uses the default JSD ($\beta=\frac{1}{2}$). Arrows indicate the nominal closer-to-teacher direction; student BpB evaluates teacher-written text and is not directly optimized by any on-policy arm. P3 baselines are selected at step 24, before their collapse.

Method	Tea. BpB ↓	Shared ↑	Byte agr. ↑	Stu. BpB ↓
P1 (Qwen3-32B teacher)				
Qwen3.5-2B (base)	0.328	0.955	0.945	0.251
SimCT	0.205	0.995	0.963	0.267
ULD	0.162	0.968	0.975	0.272
GOLD	0.172	0.985	0.972	0.264
BPM (FKL)	0.259	0.993	0.965	0.263
BPM (JSD)	0.229	0.993	0.965	0.254
BPM (RKL)	0.217	0.992	0.964	0.254
P2 (GLM-Z1-9B teacher)				
Qwen3.5-2B (base)	0.449	0.930	0.933	0.242
SimCT	0.155	0.952	0.977	0.248
ULD	0.231	0.933	0.963	0.241
GOLD	0.226	0.954	0.969	0.238
BPM (ours)	0.250	0.943	0.964	0.237
P3 (MiniMax-M2.7 teacher)				
Qwen3.5-2B (base)	0.269	0.893	0.957	0.282
SimCT	0.183	0.916	0.970	0.279
ULD	0.201	0.896	0.966	0.281
GOLD	0.195	0.924	0.972	0.279
BPM (FKL)	0.329	0.928	0.957	0.295
BPM (JSD)	0.295	0.934	0.963	0.288
BPM (RKL)	0.205	0.934	0.973	0.282

faithful transfer. Student bits-per-byte on teacher text cannot obtain the same advantage from collapse and therefore provides the complementary fidelity check. On the two pairs whose baselines train to comparable depth, this reverse-direction measure ranks BPM first among the distilled arms (Table H.2). Because a single-pair lead can fall within decoding noise, the claim rests on the repeated ordering rather than any one margin.

P3 requires a different interpretation. Its baselines are selected at step 24, one sixth of the schedule and immediately before collapse; every reverse-direction value lies within 0.014 bits of the base, and baseline gains are at most 0.003. The column therefore measures distance from initialization rather than transfer on this pair. Teacher-facing metrics reverse their ordering: BPM arms trained to step 99 without collapse achieve the highest shared-mass and byte-agreement values, whereas the most degenerate baselines lead those columns on P1 and P2. The two directions exchange leaders precisely where the baselines remain closest to initialization, consistent with what the four metrics measure.

Baseline collapse timeline. The collapse timeline exposes the weakness of the three teacher-facing quantities. SimCT enters runaway repetition by iteration 24 on MiniMax-M2.7 and after iteration 74 on GLM-Z1-9B, while remaining stable on the student's own model family. As repetition lowers output entropy, the student becomes easier for the teacher to predict and can appear closer under teacher bits-per-byte or shared mass. Reverse-direction student bits-per-byte is what separates this predictable collapse from faithful transfer.

P2 SimCT checkpoint recovery. The selected P2 SimCT checkpoint predates a checkpoint-retention incident and was recovered by replaying the identical recipe from initialization; the replayed and surviving checkpoints then entered the same checkpoint-selection rule as every other arm.

TACO-test benchmark construction. The competitive-programming test set in the main results is built from the official TACO test split (BAAI/TACO, dataset revision `d593ed0a`), restricted

Table H.3: Per-benchmark expansion of the GLM-Z1-9B data ablation in Table 3: avg@8 and pass@8 (%) under the original mix, the harder-math mix, and the expanded corpus. Each block header gives the selected checkpoint, and All averages the six benchmarks.

	AIME 2026		HMMT 2026		MATH-500		HumanEval+		LiveCodeBench		TACO		All (6)	
	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8	avg@8	pass@8
Original mix (selected checkpoints, as in the divergence table)														
FKL	35.4	63.3	21.2	36.4	84.8	95.4	68.8	91.4	22.3	33.5	16.5	36.4	41.5	59.4
JSD	26.7	56.7	17.8	30.3	84.7	95.2	69.6	90.2	19.4	32.4	13.6	29.7	38.6	55.8
RKL	28.3	50.0	20.5	33.3	78.8	94.8	65.3	89.0	16.1	30.2	11.8	30.4	36.8	54.6
Harder-math mix (all three select step 149*)*														
FKL	28.3	63.3	18.2	27.3	83.3	95.8	67.3	88.3	22.3	34.1	17.1	33.9	39.4	57.1
JSD	29.2	60.0	19.3	33.3	82.9	94.0	66.9	87.7	20.6	34.6	14.7	32.5	38.9	57.0
RKL	27.1	60.0	18.9	27.3	82.0	94.2	66.9	88.3	18.1	31.9	11.4	28.6	37.4	55.1
Expanded corpus (FKL and RKL select step 149*, JSD step* 224*)*														
FKL	33.3	60.0	21.6	33.3	84.4	95.8	67.6	92.0	21.2	35.2	15.9	34.3	40.7	58.4
JSD	30.8	63.3	19.7	33.3	81.1	94.0	68.9	90.8	19.4	34.6	13.6	33.6	38.9	58.3
RKL	31.7	50.0	21.2	36.4	79.7	94.4	65.2	87.1	17.3	28.6	10.7	27.2	37.6	53.9

to the official EASY and MEDIUM difficulty labels (400 of 1,000 problems), with three further exclusions, each from official metadata or a disclosed screen: 75 special-judge problems listed in the TACO repository's `output_spj.jsonl` (our harness compares outputs exactly, so special-judge problems would grade as false negatives), 21 problems with image assets or empty test suites, and 21 problems whose statements near-duplicate our training pool under the same n-gram screens used for mix assembly (raw 13-gram containment ≥ 0.7, raw 8-gram ≥ 0.6, or NFKC 6-gram ≥ 0.60; all removals are Codeforces near-duplicates and are listed in the build manifest). The final set holds 283 problems (149 EASY, 134 MEDIUM) across five judges, each with up to 40 executable test cases. Because the training mix never touches the TACO test split (the pool is built from the train shards only), this set measures in-distribution transfer to unseen problems.

Training-mix decontamination. The 20,000-problem training mix is screened against all six evaluation benchmarks in two stages. The first stage drops verbatim leaks at pool construction, including problems that reappear in the February 2026 competitions, under the same n-gram containment thresholds used for the TACO-test screen above. The second stage re-screens at mix assembly, where no further removals were needed because the pool was already clean, and a final audit of all 20,000 assembled prompts finds no residual leak.

Collapse-probe sample sizes. The whitespace-collapse measurements in Section 4.3 report the following probe sizes, all on the controlled no-mask arm of the P2 pair: the indentation-target probe covers 957 indentation rows and the off-realized-mass decomposition 477; the margin probe spans 80 programs with 1,799 indentation positions (22.5 per program on average); the graded pass- and error-rate diagnostics use 128 generations per checkpoint at the training-rollout temperature 1.0.

Data-mixture ablation: construction and per-benchmark detail. The harder-math variant of Table 3 replaces the mathematics half with harder competition-style problems and refreshes 1,250 of the 10k code problems. All three arms select step 149 under the same rule, and Table H.3 expands their six-benchmark aggregates. FKL's -2.1 change is concentrated on the contests (AIME $35.4 \to 28.3$, HMMT $21.2 \to 18.2$), while its code average barely moves ($35.9 \to 35.6$), indicating that the easier original mathematics half aided contest transfer. JSD and RKL remain within ± 2.5 per benchmark with no consistent direction; JSD even improves on AIME ($26.7 \to 29.2$). Response lengths are nearly unchanged between the mixes, with medians reported in Section H.3.

The expanded-corpus variant grows the training set from 20k to 35,527 prompts ($1.78\times$): the mathematics half doubles ($+$3k from the same competition source, $+$7k from DeepMath-103K (He et al., 2025)) and the code half grows by 5.5k (TACO, KodCode (Xu et al., 2025), and curated candidates), with zero contamination against all six evaluation benchmarks. Training runs the same protocol for 277 steps. Scores plateau from step 99 onward with no late collapse: checkpoints were evaluated every 25 steps through the terminal save at step 274, the six-benchmark average fluctuates within ± 1.5 points of each arm's peak, and median AIME response lengths stay at 15–18k tokens throughout, in contrast to the collapsing 20k baselines of Table H.4.

H.3 Divergence Instability and Length Behavior

Response length helps distinguish learned stopping from degeneration. We use it here to examine three observations from the main results: contest length-cap behavior after stop transfer, the late collapse of the pure mode-seeking arm, and the effect of the data-mixture ablation on generation length.

Contest length-cap behavior. Table H.4 reports, for every arm at its selected checkpoint, the median and mean response length and the fraction of samples that ran to the $27{,}648$-token cap on the two contests. The main text's headline reads off the first block: the undistilled base hits the AIME cap on 97% of samples, and BPM-FKL cuts that to 34%, the lowest of any distilled P1 arm (SimCT 50%, ULD 49%, GOLD 62%, SeqKD 61%). On GLM-Z1-9B the effect is strongest: BPM-FKL stops early enough that only 8% of AIME samples hit the cap. Two limits temper this. P1 ULD posts the shortest medians of its block and one of its lowest scores. Its brevity is degeneration rather than learned stopping. And on MiniMax-M2.7 the benefit does not arrive: every P3 arm, BPM included, still caps at the median (cap fractions 62–90% for the BPM arms), and the teacher itself runs to the cap on 55% of its HMMT samples. On this pair a verbose teacher's end-of-turn decision is a weaker lever. Where the median pins to the cap but the mean sits lower, more than half the samples cap while a minority stop early; we read the median as the typical response.

Late RKL collapse. On the MiniMax-M2.7 pair the RKL ($\beta{=}1$) arm peaks early and then collapses. After posting the block's best AIME 2026 score at step 99, it degrades over the next fifty steps, and by step 155 over 99% of its contest responses run to the $27{,}648$-token cap, with repetition filling 56% of their tokens. The FKL and $\beta{=}\frac{1}{2}$ arms on the same pair stay stable past their selected checkpoints. The instability therefore needs pure mode-seeking and the most dissimilar pair together. Its selected step-99 row in Table H.4 already shows the highest BPM cap fractions of the block, the collapse announcing itself in length before it reaches accuracy.

Data-mixture lengths. Response lengths under the two training mixes are close. Harder-math contest medians sit at 15–22k tokens against the original mix's 15.6–23.7k, and MATH-500 sits near 1.8k under both. Neither mix moves the two code benchmarks, whose medians sit at the cap under both. Because the selected original-mix checkpoints are already below the cap on the contests, the mixture is not a length lever at these checkpoints. The truncation behavior in this section comes from the objective.

H.4 Bootstrap Confidence Intervals for the Headline Margins

We attach uncertainty to each within-pair margin with a paired bootstrap over problems. For every pair, the selected checkpoints of the best BPM arm and its comparator baseline are re-generated under the same protocol: 8 samples per problem, temperature 0.6, and identical graders. Each problem then contributes an avg@8 for both arms from one harness run. We resample problems with replacement $10{,}000$ times, recompute the margin, and report the 2.5th and 97.5th percentiles. For the Avg column, each resample draws problem indices separately within all six benchmarks and aggregates them jointly.

These problem-level intervals reflect variability in the fixed set of eight regenerated samples; they do not estimate variation across training runs or checkpoint selection, since each arm is trained once and its checkpoint is already chosen. Re-generation redraws all samples, so its point estimates differ from the original runs in Table 1 by sampling noise. The largest observed per-benchmark deviation is 6.3 points on the 30-problem AIME 2026, which remains within one standard error. The same limitation applies to the closeness intervals reported above.

Table H.5 presents four comparisons: the best arm against SimCT on P1 and against ULD on P2 and P3, plus JSD against ULD on P3 as an alternate-divergence check for the most dissimilar pair. On P1 and P3 the comparator is the pair's strongest baseline; on P2 it is the strongest distributional baseline, as SeqKD ranks higher there in Table 1.

Table H.4: Contest response lengths at selected checkpoints: median and mean tokens, and cap%, the share of eight samples per problem that reach the 27,648-token limit. Pair headers give teacher statistics.

	AIME 2026			HMMT 2026		
Arm (step)	median	mean	cap%	median	mean	cap%
Qwen3.5-2B (base)	27648	27551	97	27648	27359	95
P1 (Qwen3-32B teacher; teacher: median 11588/18116*, cap* 10/19%*)*						
SimCT (155)	27648	19762	50	27648	22947	66
ULD (155)	11684	15554	49	4664	11875	34
GOLD (99)	27648	19341	62	27648	21086	71
SeqKD (191)	27648	20960	61	27648	22548	64
BPM (FKL, 155)	21398	19406	34	27640	22011	50
BPM (JSD, 155)	23922	19898	44	27648	22452	55
BPM (RKL, 155)	21238	19103	38	27648	22116	54
P2 (GLM-Z1-9B teacher; teacher: median 6075/10712*, cap* 0/3%*)*						
SimCT (49)	27648	22027	71	27648	23942	77
ULD (49)	15161	15575	22	18007	17716	25
GOLD (49)	19144	18132	35	26027	21598	48
SeqKD (182)	18122	17849	45	27648	20594	54
BPM (FKL, 149)	15584	14687	8	18595	17713	14
BPM (JSD, 155)	17568	17206	27	21364	19501	36
BPM (RKL, 149)	19428	18421	42	23671	20954	46
P3 (MiniMax-M2.7 teacher; teacher: median 21476/27648*, cap* 41/55%*)*						
SimCT (24)	27648	27648	100	27648	27448	98
ULD (24)	27648	25072	82	27648	26005	91
GOLD (24)	27648	26708	95	27648	26993	96
SeqKD (111)	27648	27170	98	27648	26599	95
BPM (FKL, 99)	27648	21632	62	27648	23535	75
BPM (JSD, 99)	27648	21499	64	27648	23792	78
BPM (RKL, 99)	27648	24710	82	27648	25609	90

Table H.5: Paired-bootstrap avg@8 margins (percentage points) with 95% intervals from 10,000 problem-level resamples. Positive values favor the first method named; bold intervals exclude zero.

	P1: FKL−SimCT	P2: FKL−ULD	P3: FKL−ULD	P3: JSD−ULD
AIME 2026	$+0.8$ $[-5.0, +7.1]$	$\mathbf{+11.3}$ $[+5.0, +17.9]$	$\mathbf{+13.8}$ $[+6.3, +21.3]$	$\mathbf{+10.4}$ $[+1.7, +19.6]$
HMMT 2026	$+2.7$ $[-3.0, +8.3]$	$\mathbf{+4.9}$ $[+0.8, +9.9]$	$\mathbf{+7.6}$ $[+1.9, +14.8]$	$\mathbf{+6.4}$ $[+1.9, +11.7]$
MATH-500	-0.1 $[-1.6, +1.4]$	$\mathbf{+2.8}$ $[+1.1, +4.5]$	$\mathbf{+3.0}$ $[+0.9, +5.0]$	$\mathbf{+4.3}$ $[+2.3, +6.3]$
HumanEval+	$\mathbf{+4.5}$ $[+0.8, +8.3]$	$+2.4$ $[-1.9, +6.5]$	-0.7 $[-5.1, +3.8]$	$\mathbf{-23.9}$ $[-28.8, -18.9]$
LCB	$\mathbf{+12.2}$ $[+9.2, +15.4]$	$\mathbf{+6.8}$ $[+4.2, +9.7]$	$\mathbf{+3.7}$ $[+1.8, +5.7]$	$+1.3$ $[-0.5, +3.2]$
TACO	$\mathbf{+10.1}$ $[+7.9, +12.3]$	$\mathbf{+3.1}$ $[+0.9, +5.4]$	$\mathbf{+4.2}$ $[+1.8, +6.6]$	$+0.1$ $[-2.2, +2.3]$
Avg	$\mathbf{+5.0}$ $[+3.3, +6.7]$	$\mathbf{+5.2}$ $[+3.6, +6.9]$	$\mathbf{+5.3}$ $[+3.4, +7.2]$	-0.2 $[-2.2, +1.8]$

I Case Studies

Three case studies ground the paper's central mechanisms in raw material from the GLM-Z1-9B pair. Cases 1 and 2 read the release files of the two models. Case 3 draws on the collapse probe of Section 4.3: 128 code generations per checkpoint at the rollout temperature, graded by the probe harness behind Figure 3c.

Case 1: a spanning token at every reasoning boundary. Every thinking-mode response closes its reasoning span with the same bytes. The two release tokenizers read those bytes differently. Shading marks the token boundaries; each cell sits under exactly the bytes it covers:

Bytes	`</think>\n\n```python\n`
Teacher (GLM-Z1-9B)	`</think>\n\n```python\n`
Student (Qwen3.5-2B)	`</think>\n\n```python\n`

The student token `</think>` covers the teacher tokens `</` and `think` entirely, plus the first byte of the third. This is the spanning case of Section 3.3 with residual prefix $\mathbf{r} = >$. The scatter of Equation (4) alone would give `</think>` near-zero mass at every reasoning boundary. The chain correction of Equation (6) supplies the teacher's realized-path probability instead. On ordinary code and prose lines the two segmentations mostly coincide; the divergence concentrates at special tokens and whitespace.

Case 2: the teacher's stop token is not its declared `eos`. The stop extension and the bridge of Equation (8) rest on the stop sets $\mathcal{S}^{\mathrm{t}}$ and $\mathcal{S}^{\mathrm{s}}$. The release files of the pair show what a single declared field would miss:

	Teacher (GLM-Z1-9B)	Student (Qwen3.5-2B)
Declared `eos` field	`<|endoftext|>`	`<|im_end|>`
Turn end in the chat template	`<|user|>`	`<|im_end|>`
Collected stop set	`<|endoftext|>`, `<|user|>`, `<|observation|>`	`<|im_end|>`, `<|endoftext|>`

The student's field is truthful: Qwen3.5 closes every turn with its declared `<|im_end|>`. The teacher's field is not. The GLM-Z1 chat template appends no end token to the assistant turn; in a rendered conversation the assistant's content runs directly into the `<|user|>` that opens the next turn. The declared `<|endoftext|>` never appears there. A collector that trusts the `eos` field alone waits for a token the teacher does not emit. The teacher's stopping signal never arrives. The three-source collection of Section 3.4 recovers all three teacher ids, ϕ maps each of them to the student's realized stop token s^*, and the bridge of Equation (8) sums the teacher's stopping mass over the full set.

Case 3: the whitespace collapse in one problem. The two BPM-JSD arms of Figure 3 differ only in the whitespace-row mask. At step 149, on one dynamic-programming problem from the probe, the unmasked arm generates:

BPM-JSD without the whitespace-row mask, step 149

```
def robot_climb_ways(steps: int) -> int:
     if steps == 1:
          return 1
     if steps == 2:
          return 2
       current = 0
        prev2 = 1
        prev1 = 2
         for _ in range(steps - 2):
              current = prev1 + prev2
              prev2 = prev1
               prev1 = current
          return prev1
```

The algorithm is correct. The indentation drifts from five spaces to seven, eight, and nine, and the block fails with an `IndentationError` at line 6. Across the probe, 108 of 128 blocks at this checkpoint fail the same way (84%, the terminal point of Figure 3c). The masked arm at the same step answers the same problem with:

BPM-JSD with the whitespace-row mask, step 149

```
def robot_climb_ways(steps: int) -> int:
    """
    :param steps: An integer representing the number of steps to the top.
    :return: An integer representing the number of distinct ways to reach
             the top.
    """
    if steps <= 2:
        return steps if steps == 1 else 2
    a, b = 1, 2
    for _ in range(3, steps + 1):
        a, b = b, a + b
    return b
```

Four-space indentation holds throughout, the block compiles, and the tests pass. Across the probe, the masked arm produces zero indentation-broken blocks. Both excerpts are verbatim generations from the two checkpoints.