SPANSHIFT: CROSS-TOKENIZER POLICY-SHIFT DISTILLATION VIA RAW-TEXT REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-tokenizer distillation enables knowledge transfer across model families, typically through vocabulary mapping or text alignment. Policy-shift distillation focuses on the changes acquired during teacher post-training, but its token-level implementations rely on a shared vocabulary. We introduce SpanShift, which expresses the teacher's policy shift from reinforcement learning (RL) as a reward on student-generated text. The reward is the pre-RL/post-RL log-likelihood difference for the same response, capturing the teacher's change in preference without projecting its distributions into the student vocabulary. Scoring successive text prefixes yields reward increments that sum to the response reward. Their reward-to-go trains each student token using rewards from its current step and subsequent continuation. When the tokenizer is shared, dense supervision over candidate next tokens replaces the sampled current-token term while retaining future reward. We evaluate SpanShift on mathematical reasoning and code generation across three cross-tokenizer student families and a shared-tokenizer setting. SpanShift improves the mathematics and code scores over the strongest baselines by +0.6 and +1.2 on average across tokenizers, and by +1.7 and +1.7 with a shared tokenizer. Ablations show gains from the teacher contrast, reward-to-go, and dense current-token supervision.

1 INTRODUCTION

Knowledge distillation transfers capabilities from a teacher model to a student model, but conventional token-level objectives compare the teacher and student next-token distributions over the same vocabulary. They therefore cannot be applied directly when the two models use incompatible tokenizers (e.g., a Qwen teacher and a Llama student; Yang et al., 2025; Grattafiori et al., 2024). Existing methods handle this mismatch by projecting distributions, using optimal transport, aligning tokens or spans, or scoring responses under the other tokenizer (Boizard et al., 2024; Cui et al., 2024; Zhang et al., 2025; Sreenivas et al., 2026; Dao et al., 2026; Phan et al., 2025). These approaches construct comparable teacher and student scores through a vocabulary mapping or a shared text representation.

Beyond making teacher feedback comparable across vocabularies, distillation also depends on which teacher information is transferred. On-policy distillation (OPD) trains on student-generated responses with dense feedback from a single teacher (Agarwal et al., 2024). Policy-shift distillation draws supervision from the difference between related policies, focusing on the behavioral changes acquired during post-training (Feng et al., 2026; Heo et al., 2026; Yu et al., 2026). For a pre-RL/post-RL teacher pair, this contrast measures how the teacher's preferences change through reinforcement learning (RL). The token-level implementations of these methods use next-token scores in a shared vocabulary, so their supervision cannot be applied directly to students with a different tokenizer. Can this policy shift be transferred without aligning the teacher and student tokens or vocabularies?

Our key observation is that the same student-generated text can be scored by both teachers, even when its token sequence differs across the teacher and student tokenizers. We propose SpanShift, which expresses the teacher's policy shift as a reward on this text. For a response y to prompt x, let π_- and π_+ denote the pre-RL and post-RL teacher policies. The reward is their log-likelihood difference:

$$R_\Delta(x, y) = \log \pi_+(y \mid x) - \log \pi_-(y \mid x). \tag{1}$$

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.

This reward measures the teacher's change in preference for the same response. Since it is computed on text, the student can learn from it using its own tokenizer, without a vocabulary mapping.

To assign this reward across generation steps, SpanShift scores successive response prefixes and computes the change after each student token. An intermediate step may enable a useful continuation even when its immediate reward is small. We therefore use reward-to-go: each token receives the reward accumulated from its own step onward, excluding the score already accumulated in the preceding prefix.

When the teacher and student share a tokenizer, SpanShift can additionally measure the pre-RL/post-RL log-probability change of each candidate next token, as in existing token-level policy-shift distillation (Feng et al., 2026; Heo et al., 2026; Yu et al., 2026). This provides dense token-level policy-shift supervision at the current generation step. SpanShift uses this dense supervision for the current-token component of reward-to-go and retains the reward from later tokens. Figure 1 summarizes the two settings.

We evaluate SpanShift on mathematical reasoning and code generation using a Qwen3 source pair (Yang et al., 2025) and students from the Llama, Gemma, and Mistral families (Grattafiori et al., 2024; Gemma Team et al., 2024; Jiang et al., 2023), alongside a shared-tokenizer Qwen student. SpanShift improves the average mathematics and code scores over the strongest cross-tokenizer baselines by +0.6 and +1.2, respectively. With a shared tokenizer, the gains are +1.7 and +1.7. Ablations show that the pre-RL/post-RL contrast and reward-to-go improve transfer, with further gains from dense current-token supervision in the shared-tokenizer setting.

Our main contributions are:

- We introduce SpanShift, which transfers the teacher's pre-RL/post-RL policy shift through rewards on student-generated text, enabling learning across tokenizers without vocabulary projection or token alignment.
- We derive reward-to-go from successive text prefixes, preserving the response reward while retaining future rewards in each token's update. When the tokenizer is shared, dense candidate supervision replaces the sampled current-token term while retaining the subsequent rewards.
- Experiments on mathematical reasoning and code generation demonstrate higher average scores across three cross-tokenizer student families and in a shared-tokenizer setting. Ablations identify gains from the teacher contrast, reward-to-go, and dense current-token supervision.

2 Related Work

Cross-tokenizer distillation. Cross-tokenizer distillation makes teacher supervision usable across different vocabularies through optimal transport, learned projection, or token and span alignment (Boizard et al., 2024; Cui et al., 2024; Zhang et al., 2025; Chen et al., 2025; Sreenivas et al., 2026; Dao et al., 2026). Text-based approaches compare aligned chunk likelihoods or score student actions with the teacher tokenizer (Minixhofer et al., 2025; Phan et al., 2025). Cross-tokenizer OPD aligns text chunks and redistributes teacher feedback to student tokens (Niu et al., 2026). SimCT extends supervision to short multi-token continuations (Sun et al., 2026), while byte-level methods provide a common representation across tokenizers (Singh et al., 2026). In particular, BPM redistributes teacher probability mass over student tokens through byte-prefix marginalization (Wang et al., 2026). These interfaces support distribution or likelihood matching. SpanShift uses the teacher pair's log-likelihood difference as a scalar reward on student-generated text, without constructing a teacher distribution over the student vocabulary.

On-policy and policy-shift distillation. Sequence-level distillation learns from teacher-generated responses (Kim & Rush, 2016), whereas OPD queries the teacher on student-generated responses (Gu et al., 2024; Agarwal et al., 2024). Policy-shift methods use the change between related policies as supervision. Direct-OPD defines a sequence-level log-ratio and implements a current-token reward objective (Feng et al., 2026). OPD^2 centers the teacher–base delta and gates it by agreement with the OPD signal (Heo et al., 2026). W2S-OPD combines the contrast with the student's initial policy

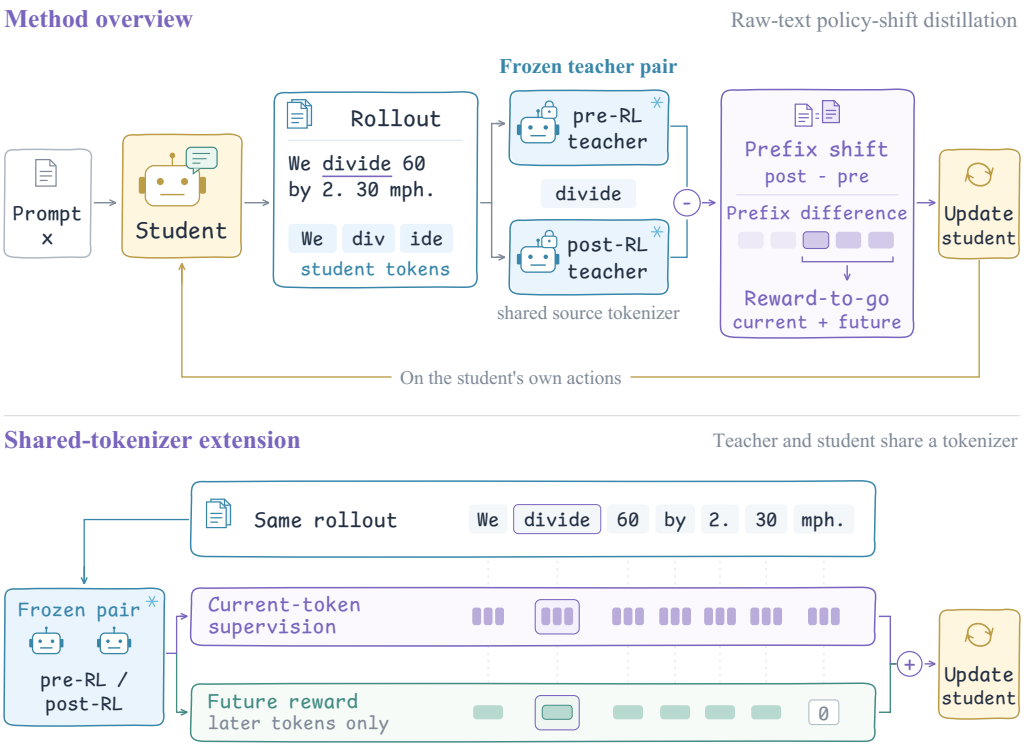


Figure 1: **Overview of SpanShift. Top: cross-tokenizer transfer.** Both frozen teachers score the student's growing response text despite different token boundaries. The pre-RL/post-RL log-likelihood difference gives the prefix shift; adjacent differences give reward increments, whose suffix sums provide reward-to-go. **Bottom: shared-tokenizer extension.** Purple vectors denote dense candidate-token supervision, and green cells denote reward from strictly later tokens in the same rollout. The outlined column links both terms to one generation step. Dense supervision replaces the sampled current-token term; the final position has zero future reward. Token boundaries and colored glyphs are schematic.

to construct a proxy teacher (Yu et al., 2026). OPRD instead uses the teacher shift to rescale a verifier-driven student gradient (Park et al., 2026). These methods differ in how the shift enters optimization; our focus is expressing it on text so that students with different tokenizers can learn from it.

Implicit rewards and credit assignment. Policy log-ratios admit an implicit-reward interpretation under KL-regularized optimization (Rafailov et al., 2023). PRIME learns an implicit reward model from outcome labels, derives dense log-ratio rewards, and uses their future sums for policy optimization (Cui et al., 2025). Its reward model is updated online, whereas SpanShift reuses fixed pre-RL and post-RL teachers. Reward-to-go and value baselines provide the policy-gradient foundation for this credit assignment (Williams, 1992; Schulman et al., 2016). We apply them to teacher contrasts evaluated at student-defined text prefixes, including prefixes whose teacher-side tokenization changes when new text is appended.

3 SpanShift

3.1 Overview

Figure 1 follows a student response from teacher scoring to the student update. We define its response-level policy-shift reward in Section 3.2, derive cross-tokenizer credit assignment in Section 3.3, and introduce dense shared-tokenizer supervision in Section 3.4.

3.2 Response-Level Policy-Shift Reward

Both teachers share tokenizer τ_T and a prompt template. For cross-tokenizer scoring, $z_{1:L} = \tau_T(y)$ encodes the response, and its log-likelihood is

$$\log \pi_{\pm}(y \mid x) = \sum_{j=1}^{L} \log \pi_{\pm}(z_j \mid x, z_{<j}). \tag{2}$$

Subtracting the scores gives the response-level policy-shift reward

$$R_{\Delta}(x, y) = \log \pi_{+}(y \mid x) - \log \pi_{-}(y \mid x). \tag{3}$$

The reward is positive when RL increases the teacher's likelihood of the response. Both teachers score the same text, so the student's tokenization can differ from theirs.

The difference measures how RL changes the teacher's preference for the student's response. We use this policy shift as the training reward and evaluate its ranking of correct and incorrect responses in Section 4.3. Appendix A.1 relates the log-ratio to an underlying reward at a KL-regularized optimum.

3.3 Cross-Tokenizer Credit Assignment

To track the reward during generation, let $y_{\le t}$ be the text after the first t student tokens. Its cumulative policy shift is

$$C_t(x) = \log \pi_{+}(y_{\le t} \mid x) - \log \pi_{-}(y_{\le t} \mid x), \qquad C_0(x) = 0. \tag{4}$$

The scalar reward introduced when the response is extended by token y_t is

$$q_t = C_t - C_{t-1}. \tag{5}$$

Summing these prefix rewards recovers the original response-level reward:

$$\sum_{t=1}^{T} q_t = C_T - C_0 = R_{\Delta}(x, y). \tag{6}$$

Each C_t is evaluated on the complete decoded prefix. Extending the student sequence can change the teacher's tokenization near the end of that prefix, so q_t includes the resulting change in the teacher-side suffix score. The cancellation in Equation 6 holds for these prefix scores without requiring teacher tokens to match student tokens.

An intermediate step may receive little immediate reward yet enable a useful continuation. Training that step with q_t alone omits these later rewards. We use the reward-to-go (Williams, 1992; Schulman et al., 2016):

$$G_t = \sum_{k=t}^{T} q_k = R_{\Delta}(x, y) - C_{t-1}. \tag{7}$$

The full response reward also includes past increments. Subtracting C_{t-1} removes them from the current token's weight while preserving the expected on-policy gradient, because the prefix score is fixed before y_t is sampled. Figure 2 compares the weights; Appendix A.2 gives the derivation.

We fit a value baseline $V_{\psi}(h_t)$ to detached reward-to-go targets G_t, with $h_t = (x, y_{<t})$, by minimizing squared prediction error. The token advantage is $G_t - V_{\psi}(h_t)$, and value predictions are detached in the actor loss. The policy-gradient surrogate is

$$\mathcal{L}_{\text{prefix}} = -\mathbb{E}\left[\sum_{t=1}^{T} \text{sg}(G_t - V_{\psi}(h_t)) \log \pi_{\theta}(y_t \mid h_t)\right] + \lambda \mathcal{L}_{\text{KL}}, \tag{8}$$

where sg denotes stop-gradient and

$$\mathcal{L}_{\text{KL}} = \mathbb{E}\left[\sum_{t=1}^{T} D_{\text{KL}}\left(\pi_{\theta}(\cdot \mid h_t) \,\|\, \pi_0(\cdot \mid h_t)\right)\right]. \tag{9}$$

Here, π_0 is the initial student policy. The KL term limits how far the updated student moves from its own initialization.

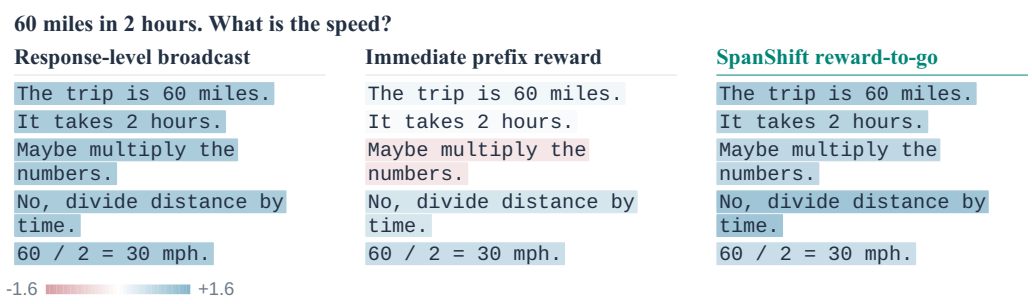


Figure 2: **Schematic comparison of three credit rules.** All columns show the same response, which corrects multiplication to division and obtains 30 mph. **Left:** the final reward R_Δ is repeated. **Middle:** span $j = [a_j, b_j]$ uses its increment $Q_j = \sum_{t=a_j}^{b_j} q_t$, exposing the negative detour and positive correction. **Right:** $G_{a_j} = \sum_{k=j}^{m} Q_k$ includes all remaining increments over the m spans. A negative immediate reward can therefore have positive reward-to-go. Blue and red indicate positive and negative raw rewards on a common scale, before value subtraction.

3.4 Shared-Tokenizer Extension

When the teacher and student share a tokenizer, both teachers score the student's sampled token IDs at the corresponding generation prefixes. Response and prefix likelihoods in this branch are accumulated along that same token sequence. At $h_t = (x, y_{<t})$, the shift for candidate v is

$$d_t(v) = \log \pi_+(v \mid h_t) - \log \pi_-(v \mid h_t), \qquad v \in \mathcal{V}. \tag{10}$$

This shift provides dense current-token supervision (Feng et al., 2026; Heo et al., 2026; Yu et al., 2026).

Following Direct-OPD, let $\mathcal{S}_t = \mathrm{TopK}_v\, \pi_\theta(v \mid h_t)$ be the student's top-k candidates (Feng et al., 2026). Their normalized weights are

$$\bar{\pi}_{\theta t}(v) = \frac{\pi_\theta(v \mid h_t)}{\sum_{u \in \mathcal{S}_t} \pi_\theta(u \mid h_t)}, \qquad v \in \mathcal{S}_t. \tag{11}$$

Along the sampled token sequence, $q_t = d_t(y_t)$. Reward-to-go separates into current and future terms:

$$G_t = d_t(y_t) + H_t, \qquad H_t = G_{t+1} = \sum_{k=t+1}^{T} q_k. \tag{12}$$

We replace the sampled current reward with its dense candidate term and retain H_t on the sampled continuation:

$$\begin{aligned} \mathcal{L}_{\text{shared}} = &-\mathbb{E}\left[\sum_{t=1}^{T} \sum_{v \in \mathcal{S}_t} \mathrm{sg}(\bar{\pi}_{\theta t}(v) d_t(v)) \log \pi_\theta(v \mid h_t)\right] \\ &- \mathbb{E}\left[\sum_{t=1}^{T} \mathrm{sg}(H_t - V_\psi(h_t)) \log \pi_\theta(y_t \mid h_t)\right] + \lambda \mathcal{L}_{\text{KL}}. \end{aligned} \tag{13}$$

Here $H_T = 0$, and the value baseline is fitted to H_t. Full-vocabulary averaging preserves the current term's conditional expectation; normalized top-k evaluation approximates it.

Appendix B details prefix caching, policy updates, and value fitting.

4 Experiments

We evaluate cross-tokenizer and shared-tokenizer transfer on mathematics and code, then examine reward construction, credit assignment, and dense supervision.

4.1 Experimental Setup

Tasks and models. The source pair starts from Qwen3-1.7B in non-thinking mode (Yang et al., 2025). Separate mathematics and code teachers use the GRPO training recipe in Appendix C, with DeepMath-103K level-6 prompts and Eurus-RL-Code, respectively (Shao et al., 2024; He et al., 2025; Cui et al., 2025). Each post-RL teacher is paired with the common pre-RL initialization. Cross-tokenizer students are Llama-3.2-3B-Instruct, Gemma-3-4B-it, and Mistral-7B-Instruct-v0.3 (Grattafiori et al., 2024; Gemma Team et al., 2024; Jiang et al., 2023); the shared-tokenizer student is Qwen3-4B in non-thinking mode. Mathematics evaluation covers AIME 2024, AIME 2025, MATH-500, and OlympiadBench (Hendrycks et al., 2021; He et al., 2024). Code evaluation covers HumanEval+, MBPP+, LiveCodeBench, and BigCodeBench (Liu et al., 2023; Jain et al., 2025; Zhuo et al., 2024).

Baselines. Table 1 includes sequence distillation, mapping, and text-alignment baselines using the post-RL teacher, including cross-tokenizer OPD (Niu et al., 2026) with chunk alignment and token-level credit assignment. SimCT + policy shift and BPM + policy shift are controls constructed here by contrasting scores from both endpoints through the corresponding interface. The shared-tokenizer comparison uses OPD and three policy-shift methods (Table 2), retaining their candidate selection, centering, gating, and reference-policy choices. Component ablations use the SpanShift training procedure.

Implementation and evaluation. Within each student, on-policy methods share initialization, prompts, generation settings, and token budget but generate their own rollouts. Sequence KD's teacher-generated data budget is reported separately. Teacher checkpoints remain fixed. Main tables report final-checkpoint accuracy averaged equally over mathematics benchmarks and pass@1 averaged equally over code benchmarks. Scores use a 0–100 scale; reported gains are absolute score differences. Appendix C gives training and evaluation settings, and Appendix D.1 gives the individual benchmark scores.

4.2 Main Results

Cross-tokenizer transfer. Table 1 compares the same source pair across three student families. The two policy-shift controls isolate text rewards from aligned or mapped supervision using the same source information.

	Llama student		Gemma student		Mistral student	
Method	Math	Code	Math	Code	Math	Code
Student initialization	42.8	49.3	40.5	46.8	43.9	51.0
Sequence KD (Kim & Rush, 2016)	42.4	48.8	40.1	46.6	43.5	50.8
ULD (Boizard et al., 2024)	44.1	51.0	42.3	48.0	45.2	52.6
DSKD (Zhang et al., 2025)	44.8	50.6	42.1	48.9	45.9	52.5
X-Token (Sreenivas et al., 2026)	44.5	51.7	43.0	49.1	45.7	53.3
SRA (Dao et al., 2026)	45.6	51.5	43.7	49.8	46.5	53.5
SimCT (Sun et al., 2026)	45.9	53.1	43.4	50.4	46.8	54.2
BPM (Wang et al., 2026)	46.2	53.4	44.0	50.5	47.1	54.5
Cross-tokenizer OPD (Niu et al., 2026)	46.6	53.8	44.5	51.6	**49.0**	54.9
SimCT + policy shift	46.0	53.5	44.5	50.9	47.6	54.3
BPM + policy shift	46.9	53.6	44.6	51.8	48.8	55.2
SpanShift	**47.8**	**54.7**	**45.8**	**53.1**	48.7	**56.5**

Table 1: **Cross-tokenizer transfer.** Math and Code denote average accuracy and pass@1, respectively. The Qwen source and each student use different tokenizers. The two "+ policy shift" rows are controls built from the same pre-RL and post-RL teachers used by SpanShift.

The average differences over the strongest baseline are +0.6 in mathematics and +1.2 in code. Positive differences occur in 2 of the three students for mathematics and 3 for code. Each comparison selects its strongest baseline within the same student and domain, including both aligned and mapped policy-shift controls.

Shared-tokenizer transfer. Table 2 compares the dense-current extension with OPD and policy-shift baselines under the shared source and student configuration.

Method	Math avg.	Code avg.
Student initialization	45.4	52.1
OPD (Agarwal et al., 2024)	48.3	55.4
Direct-OPD (Feng et al., 2026)	50.7	58.3
OPD^2 (Heo et al., 2026)	<u>51.1</u>	57.9
W2S-OPD (Yu et al., 2026)	50.5	<u>58.7</u>
SpanShift	**52.8**	**60.4**

Table 2: **Shared-tokenizer transfer.** OPD uses the post-RL teacher; policy-shift methods use the same teacher pair. Student initialization, prompts, and rollout budget are shared.

The strongest policy-shift comparisons are OPD^2 for mathematics and W2S-OPD for code. The score differences are +1.7 and +1.7, respectively, under the shared student initialization and rollout budget.

Learning dynamics. Figure 3 compares the credit-assignment variants and the shared-tokenizer signal combinations over training. The curves use the validation protocol in Appendix C. At the final validation checkpoint, cross-tokenizer reward-to-go differs from response-level weighting by +0.7 in mathematics and +1.0 in code, and from immediate-prefix weighting by +0.5 and +0.7. In the shared setting, the combined update differs from current-only by +0.9 and +0.8, and from RTG-only by +0.8 and +0.9, in the same domain order.

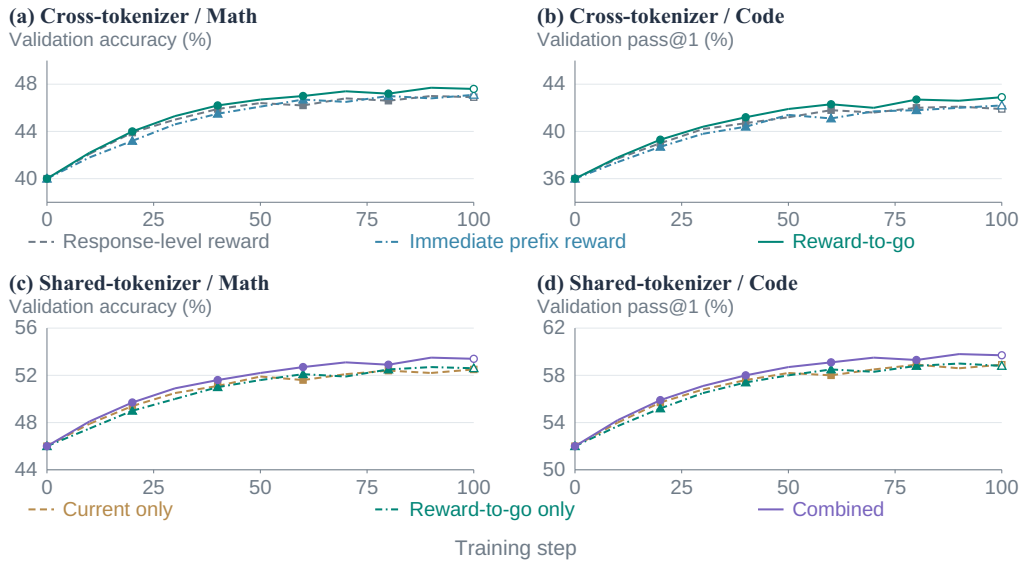


Figure 3: **Learning dynamics of the supervision variants.** Top panels compare response-level, immediate-prefix, and sampled reward-to-go training for a representative Llama-3.2-3B-Instruct configuration; bottom panels compare dense current-only, sampled reward-to-go-only, and combined current-plus-future supervision for Qwen3-4B. Mathematics uses validation accuracy and code uses validation pass@1. The horizontal axis shows student actor-optimizer updates, with the same update settings within each comparison. Points correspond to recorded validation evaluations. The final checkpoint is marked separately.

4.3 Ablation Studies

Reward and credit ablations use one cross-tokenizer student; dense-supervision ablations use the shared-tokenizer student. Each block shares the actor procedure, initialization, and rollout budget.

Each variant generates its own on-policy responses, so the comparison measures training outcomes under the same setup rather than updates on an identical set of trajectories.

Policy-shift reward. Table 3 removes the pre-RL term while retaining reward-to-go. A separate control divides post-RL prefix scores by their teacher-token counts. These variants use the same prefix differences and future sums as SpanShift, isolating the reference-policy and length-normalization choices. Prefix scores use no batch centering. Relative to raw post-RL reward, the policy-shift reward changes the score by +2.7 in mathematics and +1.8 in code. The corresponding differences from length-normalized post-RL reward are +1.3 and +1.0. These comparisons retain reward-to-go and separate the pre-RL reference from response-length normalization.

Setting	Math avg.	Code avg.
Response-reward construction		
Post-RL-only reward + reward-to-go	45.1	52.9
Length-normalized post-RL reward + reward-to-go	46.5	53.7
Policy shift R_Δ + reward-to-go	**47.8**	**54.7**
Cross-tokenizer credit assignment		
Response-level reward	45.8	<u>53.4</u>
Immediate prefix reward	<u>46.7</u>	53.1
Reward-to-go	**47.8**	**54.7**

Table 3: **Cross-tokenizer ablations.** The two blocks compare the response-reward construction and credit-assignment strategies on the same student. Rows within each block use the same training setup.

Reward-to-go. The second block compares weights R_Δ, q_t, and G_t. Comparing R_Δ with G_t removes past increments; comparing q_t with G_t includes future increments. Each variant fits its value baseline to the weight used in its update, with the same value architecture and fitting schedule. With exact value predictions, response-level and reward-to-go advantages coincide (Appendix A.2); with learned predictions, the comparison reflects their fitting and optimization behavior. Reward-to-go differs from response-level weighting by +2.0 in mathematics and +1.3 in code. Relative to immediate-prefix weighting, the differences are +1.1 and +1.6, respectively.

Shared-tokenizer extension. Table 4 removes the sampled future term for current-only training and uses $G_t = q_t + H_t$ for reward-to-go-only training. The combined variant replaces q_t with dense candidate supervision while retaining H_t. Its value function predicts H_t, whereas the RTG-only baseline predicts G_t. Current-only is a component ablation of SpanShift and is evaluated separately from the complete Direct-OPD baseline. The combined update differs from current-only supervision by +2.4 in mathematics and +2.4 in code, and from sampled reward-to-go by +1.2 and +1.4. The former comparison retains future credit, while the latter replaces the sampled current term with dense supervision.

Setting	Current credit	Future credit	Math	Code
Current only	Dense $d_t(v)$	–	50.4	58.0
Reward-to-go only	Sampled q_t	H_t	<u>51.6</u>	<u>59.0</u>
Combined	Dense $d_t(v)$	H_t	**52.8**	**60.4**

Table 4: **Shared-tokenizer ablation.** Reward-to-go uses $G_t = q_t + H_t$. The combined variant replaces the sampled current term with dense candidate supervision and retains H_t. All three variants use the same actor training settings within SpanShift.

Source-score quality. Table 5 compares these scores before training, measuring how often they rank a correct student response above an incorrect response to the same prompt. All scores use the same held-out response pairs. This separates the teacher signal's ranking behavior from the student's subsequent optimization. The same table reports correlations with response length to characterize how each score varies with the amount of generated text. The policy contrast changes pairwise accuracy relative to raw post-RL likelihood by +11.0 for mathematics and +12.5 for code. Against

length-normalized likelihood, the differences are +5.7 and +7.6. The accompanying correlations measure each score's association with response length on the same response pool.

	Pairwise accuracy		Length correlation	
Response score	Math	Code	Math	Code
Post-RL likelihood	58.1	55.8	-0.604	-0.586
Length-normalized post-RL likelihood	63.4	60.7	-0.119	-0.087
Policy shift R_Δ	**69.1**	**68.3**	0.041	0.026

Table 5: **Source-score diagnostics on held-out student responses.** Pairwise accuracy is the percentage of prompt-matched pairs in which a correct response scores above an incorrect one. Length correlation is Pearson's r between the score and response length, measured with the teachers' shared tokenizer and excluding the prompt.

4.4 Analysis

Credit examples. Figure 4 compares response-level weighting and reward-to-go on identical response texts using the same teacher-pair score definition. The left column repeats the response reward; the right shows the remaining reward at each span's start. A return's sign describes the scored continuation, not an individual step's correctness. The responses receive total rewards 1.13 and -0.42, with final answers verified as correct and incorrect, respectively. Span 3 in the first response has immediate reward -0.42 and reward-to-go 0.85; span 4 in the second has values -0.35 and -0.5. Their respective differences are the rewards in the subsequent continuations.

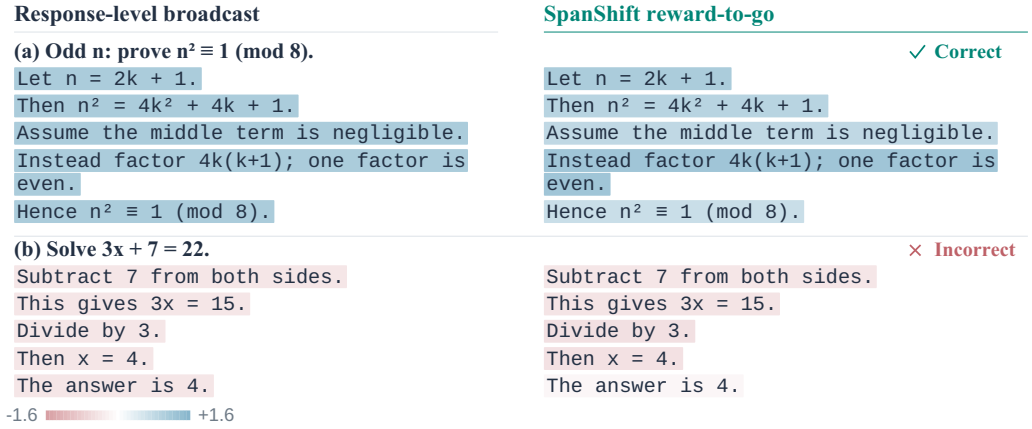


Figure 4: **Credit on identical response texts.** Within each example, the left column repeats R_Δ and the right column displays the remaining reward at each span's start. Blue and red denote positive and negative rewards on a common scale. Both columns use the same response text and teacher-pair score definition. Correctness labels refer to the written final answers. The selection rule is manual construction of the two displayed examples.

5 Conclusion

SpanShift transfers a pre-RL/post-RL teacher contrast across tokenizers through rewards on student-generated text. Prefix differences preserve the response reward and yield per-token reward-to-go. With a shared tokenizer, dense current-token supervision replaces the sampled current term while retaining future reward. Transfer depends on the teacher's policy shift and requires both checkpoints for scoring. The evaluations distinguish reward construction from credit assignment and dense current-token supervision.

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*, 2024. URL https://arxiv.org/abs/2306.13649.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: The universal logit distillation loss for LLMs. *arXiv preprint arXiv:2402.12030*, 2024.

Yijie Chen, Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. Enhancing cross-tokenizer knowledge distillation with contextual dynamical mapping. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8005–8018, 2025. doi: 10.18653/v1/2025.findings-acl.419.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL https://arxiv.org/abs/2502.01456.

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. *arXiv preprint arXiv:2412.14528*, 2024.

Quoc Phong Dao, Hoang Son Nguyen, Pham Khanh Chi, Tung Nguyen, Linh Ngo Van, Nguyen Thi Ngoc Diep, and Trung Le. SRA: Span representation alignment for large language model distillation. *arXiv preprint arXiv:2605.01205*, 2026.

Shiyuan Feng, Huan-ang Gao, Haohan Chi, Hanlin Wu, Zhilong Zhang, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Weak-to-strong generalization via direct on-policy distillation. *arXiv preprint arXiv:2607.05394*, 2026.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Somerstep, et al. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.08543.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025. URL https://arxiv.org/abs/2504.11456.

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 *NeurIPS Datasets and Benchmarks*, 2021.

Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-policy delta distillation. *arXiv preprint arXiv:2607.15161*, 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*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, 2016. doi: 10.18653/v1/D16-1139.

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*, 2023.

Benjamin Minixhofer, Ivan Vulić, and Edoardo M. Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.20083.

Yifan Niu, Han Xiao, Dongyi Liu, Zelong Wang, Dihong Gong, Yasheng Wang, and Jia Li. Breaking the tokenizer barrier: On-policy distillation across model families. *arXiv preprint arXiv:2606.09456*, 2026. URL https://arxiv.org/abs/2606.09456.

Youngrok Park, Sangmin Bae, Hojung Jung, Jongwoo Ko, Yunseon Choi, Young Jin Kim, Pashmina Cameron, Aaron Courville, and Se-Young Yun. Eliciting weak-to-strong generalization with on-policy reverse distillation. *arXiv preprint arXiv:2609.08798*, 2026. URL https://arxiv.org/abs/2609.08798.

Buu Phan, Ashish Khisti, and Karen Ullrich. Cross-tokenizer likelihood scoring algorithms for language model distillation. *arXiv preprint arXiv:2512.14954*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Avyav Kumar Singh, Yen-Chen Wu, Alexandru Cioba, Alberto Bernacchia, and Davide Buffelli. Cross-tokenizer LLM distillation through a byte-level interface. *arXiv preprint arXiv:2604.07466*, 2026. URL https://arxiv.org/abs/2604.07466.

Sharath Turuvekere Sreenivas, Adithyakrishna Venkatesh Hanasoge, Mingyu Yang, Ali Taghibakhshi, Saurav Muralidharan, Ashwath Aithal, and Pavlo Molchanov. X-Token: Projection-guided cross-tokenizer knowledge distillation. *arXiv preprint arXiv:2605.21699*, 2026.

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. URL https://arxiv.org/abs/2605.07711.

Hao Wang, Kun Yuan, Wenlin Zhong, Minglei Zhang, Han Xiao, Ming Sun, and Honggang Qi. Cross-tokenizer on-policy distillation via byte-prefix marginalization. *arXiv preprint arXiv:2607.22334*, 2026. URL https://arxiv.org/abs/2607.22334.

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

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

Fangxu Yu, Weijia Xu, Michael Xu, Tianyi Zhou, and Zinan Lin. Weak-to-strong on-policy distillation. *arXiv preprint arXiv:2607.26246*, 2026. URL https://arxiv.org/abs/2607.26246.

Xue Zhang, Songming Zhang, Yunlong Liang, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. A dual-space framework for general knowledge distillation of large language models. *arXiv preprint arXiv:2504.11426*, 2025.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

STATEMENTS

Reproducibility. Appendices A–C provide the derivations, scoring conventions, and training and evaluation configurations. The result tables separate benchmark scores from their domain averages.

AI use. Generative AI tools assisted with language editing, literature search, methodological discussion, derivation checks, experiment planning, and figure and manuscript preparation. The authors retain responsibility for verifying the mathematical arguments, cited sources, experimental results, and final manuscript.

A PROOFS

A.1 REWARD INTERPRETATION AT A KL-REGULARIZED OPTIMUM

For a fixed prompt, consider a KL-regularized RL objective with reward $r^*(x, y)$, reference policy π_-, and coefficient $\beta > 0$. Its stationarity condition over a normalized distribution, with Lagrange multiplier η, is

$$r^*(x, y) - \beta \left(\log \frac{\pi(y \mid x)}{\pi_-(y \mid x)} + 1 \right) + \eta = 0.$$

Hence $\pi(y \mid x) \propto \pi_-(y \mid x) \exp(r^*(x, y)/\beta)$. Substituting $\pi = \pi_+$ gives

$$\log \frac{\pi_+(y \mid x)}{\pi_-(y \mid x)} = \frac{1}{\beta} r^*(x, y) - \log Z(x), \tag{14}$$

where $Z(x)$ depends only on the prompt. This identity requires a KL-regularized optimum with $\beta > 0$ and reference policy π_- (Rafailov et al., 2023; Feng et al., 2026). Our source-training KL coefficient is 0 (Table 6). For these checkpoints, Equation 3 measures the observed policy change; the reward-recovery identity above is a conditional interpretation, not an assumption of the student update.

A.2 POLICY-GRADIENT JUSTIFICATION FOR REWARD-TO-GO

Write $R = \sum_k q_k$. The score-function identity gives

$$\nabla_\theta \mathbb{E}[R] = \mathbb{E}\left[R \sum_t \nabla_\theta \log \pi_\theta(y_t \mid h_t) \right].$$

For $k < t$, q_k is fixed by the prefix h_t, and $\mathbb{E}[\nabla_\theta \log \pi_\theta(y_t \mid h_t) \mid h_t] = 0$. Removing these past terms leaves $G_t = \sum_{k \geq t} q_k$. The same zero-mean argument permits subtracting any prefix-measurable baseline. Thus, response-level reward and reward-to-go have the same expected on-policy gradient. Reward-to-go excludes past prefix rewards from the weight assigned to each current token.

Immediate-only weighting keeps q_t instead of G_t and drops $\mathbb{E}[\sum_t \sum_{k>t} q_k \nabla_\theta \log \pi_\theta(y_t \mid h_t)]$. These terms generally depend on the current action through the continuation. Thus, conservation of the reward sum does not make immediate-only updates equivalent to the response-level policy gradient.

The choice of value target also matters. For a fixed policy, define $V_R(h_t) = \mathbb{E}[R_\Delta \mid h_t]$ and $V_G(h_t) = \mathbb{E}[G_t \mid h_t]$. Since C_{t-1} is known at h_t,

$$V_G(h_t) = V_R(h_t) - C_{t-1}, \qquad G_t - V_G(h_t) = R_\Delta - V_R(h_t). \tag{15}$$

Exact value baselines therefore yield identical advantages. With learned baselines, the comparison also depends on how well each target is fitted.

A.3 Justification of the shared-tokenizer objective

The shared-tokenizer branch accumulates all likelihoods along the student's sampled token IDs. Consequently, $C_t = \sum_{j=1}^{t} d_j(y_j)$ and $q_t = d_t(y_t)$. Under this convention, the reward-to-go policy gradient contains

$$\mathbb{E} \sum_t [d_t(y_t) + H_t] \nabla \log \pi_\theta(y_t \mid h_t).$$

Conditioned on h_t, the immediate part can be evaluated over candidate next tokens, while the future term remains on the sampled continuation. This yields the two terms in Equation 13. Restricting the first term to the renormalized top-k set is the approximation used in practice. The dense term accounts for the current token, and H_t accounts for the subsequent tokens.

The shared branch scores the sampled token sequence directly. The cross-tokenizer branch scores teacher encodings of the decoded prefixes. Both use a pre-RL/post-RL log-ratio, with the scoring sequence specified by the branch. If decoding and re-encoding changes a sampled token sequence, these two scoring conventions need not assign the same numerical reward.

B Scoring and Value Fitting

Scoring and updates. For each prompt batch, we retain the student's sampled token IDs, log-probabilities, and decoded responses. Cross-tokenizer scoring evaluates the decoded prefixes under both teachers, reusing unchanged teacher-token prefixes and rescoring affected suffixes. Shared-tokenizer scoring instead accumulates shifts along the sampled IDs. Reverse cumulative sums yield G_t and H_t. The actor uses detached value predictions, and the value function is fitted to the corresponding target. When reusing a rollout batch for multiple actor epochs, the old token log-probabilities provide the importance ratios in the PPO clipped update (Schulman et al., 2017).

The scoring configuration specifies model and tokenizer revisions, prompt templates, EOS handling, and truncation. Cached prefix scores are compared with full rescoring, including cases in which appending text changes the teacher-token suffix. Shared-tokenizer scoring follows the token-history condition in Appendix A.3.

Value fitting. The value function predicts the expected remaining reward at each prefix. We fit it with the squared-error objective

$$\mathcal{L}_V(\psi) = \mathbb{E}_t \left[\left(V_\psi(h_t) - \mathrm{sg}(Z_t) \right)^2 \right], \tag{16}$$

where $Z_t = G_t$ for sampled reward-to-go training and $Z_t = H_t$ for the sampled future term of the shared-tokenizer extension. Rewards are undiscounted. Each run fits its value function to the corresponding target and detaches the predictions in the actor loss.

C Full Experimental Protocol

Source-policy training. Table 6 specifies the source-training recipe, starting from Qwen3-1.7B in non-thinking mode. Mathematics prompts come from the level-6 subset of DeepMath-103K (He et al., 2025), and code prompts come from Eurus-RL-Code (Cui et al., 2025). Correct final answers and passed unit tests provide the verifiable rewards for mathematics and code, respectively. Separate post-RL checkpoints are paired with the common initialization. Source validation compares the pre-RL and post-RL task scores on a disjoint prompt split (Table 7). The reward-ranking analysis evaluates their contrast on student responses.

Source-training hyperparameter	Value
Algorithm	GRPO
Initialization	Qwen3-1.7B (non-thinking)
Training data	DeepMath-103K L6 / Eurus-RL-Code
Maximum prompt length	2,048
Maximum response length	16,384
Responses per prompt	8
Global batch size	128
Learning rate	1×10^{-6}
Learning-rate schedule	Constant
Sampling temperature / top-p	1.0 / 1.0
KL coefficient	0
Training steps	500 / 300
Checkpoint selection	Final checkpoint

Table 6: **Training configuration for the mathematics/code source policies.** Paired entries report mathematics and code settings, respectively. Source training and student transfer use separate configurations.

Table 7: **Source-policy validation.** Pre-RL and post-RL checkpoints are evaluated on the same held-out prompts within each domain.

Source policy	Mathematics	Code
Pre-RL	43.1	49.0
Post-RL	51.5	57.4

Student transfer. Table 8 groups student optimization and evaluation settings. Sequence KD uses teacher-generated responses, with its data budget recorded separately from on-policy rollouts.

Method-specific choices. Cross-tokenizer OPD (Niu et al., 2026) uses the post-RL teacher, matching the student initialization, transfer prompts, rollout budget, and evaluation settings in the cross-tokenizer comparison. Its dual-pointer chunk alignment and semantic-prior credit assignment retain the original single-teacher objective. Direct-OPD uses current-token policy-shift rewards and an adaptive KL anchor (Feng et al., 2026). OPD^2 uses centered sampled-token advantages with an agreement condition (Heo et al., 2026). W2S-OPD minimizes reverse KL to a proxy teacher anchored at the initial student (Yu et al., 2026). Candidate supports serve different roles in these methods: dense reward evaluation, centering, or proxy-distribution matching. The implementation configuration records those roles as well as each method's reference policy, reward scale, and normalization.

Matched credit-assignment comparisons. The value targets in Section 4.3 are fitted with Equation 16. Current-only training retains the dense term and student KL regularization, with no sampled future term or future-value baseline. Reward scaling is applied to both returns and value targets.

Post-RL-only controls. To remove the pre-RL reference alone, we define

$$\begin{aligned} C_t^{\text{post}} &= \log \pi_+(y_{\le t} \mid x), \\ q_t^{\text{post}} &= C_t^{\text{post}} - C_{t-1}^{\text{post}}, \qquad G_t^{\text{post}} = C_T^{\text{post}} - C_{t-1}^{\text{post}}. \end{aligned} \tag{17}$$

Table 8: **Student training and evaluation settings.** Paired values denote mathematics/code when the domain configurations differ.

Setting	Value
Training / validation prompt counts	20,000 / 1,000 per domain
Transfer split	Deduplicate, exclude test overlap, then split by prompt ID with seed 42
Student-generated token budget	50 million per run
Maximum prompt / response length	2,048 / 16,384
Prompt batch / responses per prompt	128 / 8
Actor minibatch / epochs per rollout batch	128 responses / 1
Training temperature / top-p	1.0 / 1.0
Actor optimizer / learning rate	AdamW / 5×10^{-7}
Learning-rate schedule / clip range	Constant / 0.2
Student KL / fixed reward scale	0.01 / 1.0
Advantage normalization	Batchwise
Value objective	Squared error
Value architecture	Scalar value head
Value optimizer / learning rate	AdamW / 1×10^{-6}
Value fitting schedule	One pass per rollout batch
Shared-tokenizer candidate-set size	64
Sequence KD response-token budget	50 million response tokens
Evaluation temperature / top-p	0.6 / 0.95
Samples per problem	8
Generation limit / stopping rules	16,384 tokens / EOS or length limit

with $C_0^{\mathrm{post}} = 0$. This control uses the same prefix-scoring and reward-to-go procedure as SpanShift. A separate length-normalized control uses $C_t^{\mathrm{norm}} = \log \pi_+(y_{\le t} \mid x)/L_t$ for nonempty prefixes, where L_t counts teacher response tokens and excludes the prompt. Its empty-prefix score is zero, and increments and returns follow the same differences as Equation 17. We apply no batch centering to either prefix score. Any fixed reward multiplier is applied consistently to rewards and value targets and recorded with the training configuration.

Table 9: **Ablation configuration.** Each entry specifies the scoring or baseline choice used with the token weights in the corresponding block.

Choice	Configuration
Raw post-only prefix score	$\log \pi_+(y_{\le t} \mid x)$
Normalized post-only prefix score	$\log \pi_+(y_{\le t} \mid x)/L_t$
Prefix-score batch centering	None
Fixed reward multiplier	1.0
Value targets: response / immediate / RTG	R_Δ / q_t / G_t
Value target for dense-current plus future	H_t
Shared-tokenizer scoring sequence	Sampled token IDs and prefixes

Evaluation configuration. Mathematics accuracy uses benchmark-specific answer extraction, and code pass@1 uses the benchmark test suites. Each method is evaluated with the same prompt format, decoding settings, and generation limit for that benchmark. Table 8 summarizes the common settings. Prompts follow a fixed zero-shot task prompt within each model's chat template, without benchmark-specific demonstrations. Mathematics answers are extracted with Final boxed answer, else final-answer field and scored by Numeric or symbolic equivalence. Code evaluation uses Isolated worker; 10 s/test, 4 GB memory. Multiple sampled responses estimate average single-response accuracy or pass@1, rather than pass@k for $k > 1$.

Evaluation during training. Each validation record includes the actor-update step and cumulative student-generated token count. Step-based comparisons use the same update settings within a student configuration. The validation protocol is validation accuracy and pass@1, evaluated every 10 actor

updates; the step and budget definitions are one actor-optimizer update per step; the displayed curves cover 100 steps. Main tables use the final checkpoint.

D Additional Experimental Results

D.1 Per-Benchmark Results

Tables 10–13 give the component scores for the methods in the main comparisons. Math and Code averages give equal weight to their four benchmarks. HE+, MBPP+, LCB, and BigCode denote HumanEval+, MBPP+, LiveCodeBench, and BigCodeBench.

Table 10: **Per-benchmark cross-tokenizer results for the Llama student.**

Method	AIME24	AIME25	MATH-500	Olympiad	HE+	MBPP+	LCB	BigCode
Student initialization	20.9	15.2	71.7	63.4	65.0	58.4	35.7	38.1
Sequence KD	21.6	13.8	71.4	62.8	64.9	59.2	36.1	35.0
ULD	23.8	13.9	74.7	64.0	67.6	61.0	38.8	36.6
DSKD	24.3	13.4	75.2	66.3	68.5	59.6	37.9	36.4
X-Token	21.9	14.6	74.8	66.7	65.9	60.5	40.8	39.6
SRA	24.9	16.8	76.8	63.9	66.4	62.3	39.9	37.4
SimCT	24.9	15.6	78.1	65.0	67.7	65.7	39.5	39.5
BPM	24.9	**18.0**	76.6	65.3	68.0	63.2	**42.4**	40.0
Cross-tokenizer OPD (Niu et al., 2026)	25.9	16.3	77.1	67.1	69.4	64.1	42.0	39.7
SimCT + policy shift	24.2	15.6	75.2	69.0	69.8	61.8	42.3	40.1
BPM + policy shift	25.5	16.6	**78.6**	66.9	**70.9**	61.2	41.5	40.8
SpanShift	**26.7**	17.1	77.9	**69.5**	70.0	**66.5**	41.1	**41.2**

Table 11: **Per-benchmark cross-tokenizer results for the Gemma student.**

Method	AIME24	AIME25	MATH-500	Olympiad	HE+	MBPP+	LCB	BigCode
Student initialization	19.1	13.2	69.6	60.1	61.3	57.0	36.5	32.4
Sequence KD	20.0	9.7	70.8	59.9	61.4	57.8	33.8	33.4
ULD	20.1	14.0	72.1	63.0	64.0	56.6	36.7	34.7
DSKD	21.4	12.9	73.6	60.5	67.0	57.4	36.7	34.5
X-Token	22.0	11.4	74.5	64.1	66.6	59.6	34.5	35.7
SRA	23.3	15.9	73.8	61.8	65.6	60.4	37.4	35.8
SimCT	23.0	13.7	73.5	63.4	66.9	60.4	39.2	35.1
BPM	22.9	15.2	74.1	63.8	67.9	59.9	38.9	35.3
Cross-tokenizer OPD (Niu et al., 2026)	24.1	15.4	74.7	63.8	68.2	62.1	38.3	37.8
SimCT + policy shift	**26.4**	14.0	72.6	65.0	66.5	60.0	38.5	38.6
BPM + policy shift	23.5	14.9	**75.5**	64.5	66.1	60.9	**41.7**	38.5
SpanShift	24.9	**18.0**	73.5	**66.8**	**69.5**	**64.2**	39.9	**38.8**

Table 12: **Per-benchmark cross-tokenizer results for the Mistral student.**

Method	AIME24	AIME25	MATH-500	Olympiad	HE+	MBPP+	LCB	BigCode
Student initialization	22.8	14.3	74.2	64.3	68.2	57.8	40.2	37.8
Sequence KD	22.8	15.7	70.9	64.6	65.2	59.5	40.3	38.2
ULD	24.6	17.6	74.7	63.9	67.5	62.8	39.9	40.2
DSKD	26.1	14.3	75.2	68.0	69.3	63.3	40.5	36.9
X-Token	25.7	15.1	75.0	67.0	67.3	64.7	39.8	41.4
SRA	23.3	18.7	75.9	68.1	69.3	62.9	42.1	39.7
SimCT	25.7	16.1	78.2	67.2	70.0	61.0	43.5	42.3
BPM	24.6	18.8	78.1	66.9	**72.2**	62.2	43.9	39.7
Cross-tokenizer OPD (Niu et al., 2026)	**27.6**	**20.2**	78.8	69.4	71.4	64.1	43.7	40.4
SimCT + policy shift	27.0	18.7	76.1	68.6	71.1	66.4	41.4	38.3
BPM + policy shift	27.2	18.7	**80.0**	69.3	70.2	65.8	**44.2**	40.6
SpanShift	27.2	19.8	77.7	**70.1**	70.6	**67.9**	44.1	**43.4**

D.2 Source and Parameter Sensitivity

Tables 14 and 15 vary the source RL run and the shared-tokenizer training parameters, respectively. The source-run comparison uses the Llama mathematics setting, and parameter sweeps use the Qwen mathematics validation split.

Table 13: **Per-benchmark shared-tokenizer results for the Qwen student.**

Method	AIME24	AIME25	MATH-500	Olympiad	HE+	MBPP+	LCB	BigCode
Student initialization	25.1	15.3	76.0	65.2	68.1	60.4	40.7	39.2
OPD	28.3	18.8	76.9	69.2	70.8	65.5	42.6	42.7
Direct-OPD	31.3	19.5	80.7	71.3	73.2	68.4	**48.5**	43.1
OPD^2	31.9	21.2	79.9	71.4	71.8	67.6	46.8	45.4
W2S-OPD	29.0	20.6	**83.0**	69.4	73.9	**70.3**	45.9	44.7
SpanShift	**33.8**	**22.3**	82.3	**72.8**	**76.2**	68.6	47.3	**49.5**

Source run	Source gain	Pairwise acc.	Downstream gain
1	8.4	69.1	0.9
2	7.7	69.5	1.2
3	8.8	67.9	1.2

Table 14: **Variation across source RL runs.** Source gain compares post-RL and pre-RL scores. Pairwise accuracy evaluates R_Δ on held-out responses, and downstream gain compares the student with its strongest matched baseline.

Parameter	Value	Validation score	Policy KL
Student KL	0.005	52.5	0.148
	0.01	53.0	0.091
	0.02	52.2	0.043
Reward scale	0.5	52.4	0.066
	1	53.0	0.091
	2	52.7	0.137
Top-k	32	52.5	0.084
	64	53.0	0.091
	128	52.9	0.093

Table 15: **Sensitivity of the shared-tokenizer objective.** Each block varies one parameter around the default configuration on the validation split.

D.3 Capability Retention and Cost

Table 16 uses the shared-tokenizer Qwen mathematics checkpoints and compares OPD^2 with SpanShift. Table 17 uses the Llama mathematics setting on eight A100 80GB GPUs and includes teacher scoring. Measurement scope: total GPU hours including teacher scoring, per-GPU peak memory, and scorer-to-actor traffic.

Method	IFEval	MMLU-Pro
Student initialization	**78.6**	**48.9**
OPD^2	77.8	48.6
SpanShift	78.1	48.7

Table 16: **General-capability retention.** Scores are measured before and after task-specific transfer using the same evaluation configuration.

Method	GPU hours	Memory (GB)	Scoring ratio	Traffic (GB)
SimCT	147.2	61.2	1.34	43.8
BPM	160.8	63.7	1.49	48.5
BPM + policy shift	287.2	70.4	2.94	92.6
SpanShift	229.6	68.9	2.31	14.2

Table 17: **Transfer efficiency under a matched student-generated token budget.** GPU hours include source scoring and student updates. Memory is the peak usage. Scoring ratio counts source tokens processed after caching per generated student token.