# BREAKING THE TOKENIZER BARRIER: ON-POLICY DISTILLATION ACROSS MODEL FAMILIES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

On-Policy Distillation (OPD) has become a core technique in the post-training of Large Language Models (LLMs) for transferring knowledge from domain experts to student models. However, existing OPD distillation methods restrict teacher and student models to share the same tokenizer within the model series. Current mainstream practice typically employs Supervised Fine-Tuning (SFT) on teacher-generated responses for cross-tokenizer distillation, which fails to capture the rich knowledge embedded in the teacher's probability distribution. In this work, we propose **Cross-Tokenizer On-Policy Distillation (CTOPD)** to enable the standard on-policy distillation method to operate across model families, ensuring that high-fidelity token-level signals can propagate across different tokenizers with a precise token-mapping algorithm. Extensive experiments show that CTOPD is significantly more compute-efficient than baselines on various benchmarks. Our results unlock a broader range of teacher–student pairs for OPD, opening up new avenues for adapting and enhancing interactions between LLMs. The code is available at https://anonymous.4open.science/r/On-Policy-Distill-771C/.

## 1 INTRODUCTION

On-Policy Distillation (OPD) (Zhao et al., 2026; Yang et al., 2026; Hübotter et al., 2026) has become a core technique in the post-training of Large Language Models (LLMs) for transferring knowledge from experts to student models. OPD is known for its computational efficiency and serves as a tool for continual learning. Leading industry models, such as Qwen3 (Yang et al., 2025), MiMo (Team et al., 2026), and GLM-5 (GLM-5-Team et al., 2026), have all integrated OPD into their post-training stage and reported significant improvements. Thinking Machines Lab (Lu & Lab, 2025) also validated the efficacy of OPD: with a mere fraction of the computational cost, they successfully replicated Qwen3's OPD recipe, which demonstrates that "on-policy, dense supervision" is an efficient and effective alternative.

In contrast to off-policy distillation (Jiao et al., 2020; Wang et al., 2020), which trains a student model using static sequences generated by a teacher model and suffers from exposure bias, OPD trains on the student's rollouts and leverages the teacher's per-token log-probabilities as a dense reward signal to refine the student's behavior. However, existing OPD methods (Ko et al., 2025; Kim et al., 2026; Xu et al., 2025) confine this token-level supervision to same-family distillation, where the teacher and student models share an identical tokenizer. When OPD across model families, the misalignment of vocabulary (differing in size, encoding preferences, and the handling of special characters) creates a significant barrier, as raw log-probabilities cannot be directly transferred. Cross-tokenizer OPD remains a challenge that has yet to be thoroughly explored.

Cross-tokenizer alignment has recently become feasible due to the convergence of industrial tokenization standards. Most contemporary language models rely on subword-level tokenization algorithms (Sennrich et al., 2016; Kudo & Richardson, 2018), specifically Byte-Pair Encoding (BPE) (Zouhar et al., 2024), to partition text into discrete tokens. Although different tokenizers employ vocabularies with distinct encoding characteristics (Nawrot et al., 2023), they all share the underlying Unicode/byte-level representation (Xue et al., 2022; Pagnoni et al., 2024). Furthermore, despite the discrepancy in actual token IDs, different tokenizers partition the same underlying text into comparable semantic and character-level fragments, which provides a structural foundation for



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.

synchronization. These commonalities effectively lower the barriers between different vocabulary spaces, making cross-tokenizer distillation possible.

In this work, we address the challenge of cross-tokenizer on-policy distillation and propose an adaptive chunk alignment algorithm and credit assignment framework, named **Cross-Tokenizer On-Policy Distillation (CTOPD)**, that bridges the vocabulary gap across different model families. We propose a dual-pointer chunk alignment algorithm to identify the minimal synchronized chunks between the student and teacher sequences. Building upon this alignment, we introduce a chunk-level credit assignment mechanism based on semantic priors. We derive a closed-form solution that scales the student's log-probabilities, which preserves the student's internal semantic structure while adhering to the guidance of the teacher model. To the best of our knowledge, it is the first work to enable high-fidelity, per-token OPD across different model families. Extensive experiments show that CTOPD is significantly more compute-efficient than off-policy distillation on various benchmarks. Our results unlock a broader range of teacher–student pairs for OPD, opening up new avenues for adapting and enhancing interactions between LLMs.

## 2 Related Work

**On-Policy Distillation** On-Policy Distillation (OPD) optimizes student models by providing dense teacher supervision on self-generated trajectories to have better distribution alignment and performance. In previous works, MiniLLM (Gu et al., 2026) pioneered on-policy refinement by employing a reverse KL objective to enforce mode-seeking behavior. GKD (Agarwal et al., 2024) then established a unified framework by formalizing the sampling distribution as a flexible interpolation between student trajectories and teacher data across diverse divergences. DistiLLM (Ko et al., 2024) later stabilized the paradigm by introducing skewed KL divergences, leading to a more tractable optimization process. Recent research (Song & Zheng, 2026) classifies feedback signals in OPD into logits-based, outcome-based, and self-play rewards. Logits-based signals (Jung et al., 2025; Xu et al., 2026) provide token-level alignment by evaluating student outputs against the teacher's continuous probability distribution. To prevent the computational overhead of full logit, outcome-based methods (Xu et al., 2025; Zhang et al., 2025) use sequence-level reward signals. Moreover, self-play methods (Chen et al., 2024; Zhao et al., 2026) allow models to act as both student and teacher, enabling teacher-free alignment without external intervention. However, these alignments require student and teacher models are the same tokenizer, restricting the applicability of OPD within the model series.

**Cross-Tokenizer Distillation** Recent research on cross-tokenizer distillation can be divided into two categories: logit-level alignment and text-level alignment. Logit-level alignment methods aim to make the logits of teachers and students in different vocabulary spaces comparable by transforming or reconstructing the prediction distribution. ULD (Boizard et al., 2025) and MultiLevelOT (Cui et al., 2025) model cross-tokenizer Logit Alignment as an optimal transport problem, mitigating vocabulary mismatch by transferring probability quality between heterogeneous vocabularys; while CDM (Chen et al., 2025) utilizes entropy-weighted dynamic programming for context-dependent sequence alignment and combines it with dynamic vocabulary mapping to construct a comparable logit space. Text-level alignment methods use text units as a bridge between teachers and students. ALM (Minixhofer et al., 2025) directly uses coarse-grained word chunks as the smallest unit to transmit probabilistic information, while BLD (Singh et al., 2026) aims for fine-grained byte-level alignment, introducing an additional decoder to reconstruct byte-level probabilistic information. However, existing methods are mostly designed for offline distillation strategies and cannot effectively solve the token-level signal propagation problem in OPD. More recently, GOLD (Hugging Face, 2026) extends cross-tokenizer distillation to the online setting by aligning textual spans across different tokenizers and aggregating their associated probabilities. However, how to effectively propagate dense token-level supervision across different tokenizers in OPD remains insufficiently explored.

## 3 Preliminaries

**Notation.** Let $x = \{x_1, \ldots, x_n\}$ denotes an input query and $y = \{y_1, \ldots, y_m\}$ is the response. The prefix of tokens prior to step $t$ is defined as $y_{<t} = (y_1, \ldots, y_{t-1})$. We consider two language models: a student $\pi_\theta$ and a teacher $\pi_T$, each defining a next-token distribution $\pi(\cdot \mid x, y_{<t})$ over a student

vocabulary $\mathcal{V}_S$ or teacher vocabulary $\mathcal{V}_T$. We use $y \sim \pi_\theta(\cdot \mid x)$ to represent a response sampled autoregressively from the student, and $\mathcal{D}_x = \{x^{(i)}\}_{i=1}^N$ denotes the corresponding query dataset.

### 3.1 Tokenization

Tokenization functions fuse bytes into larger (e.g., subword) tokens. Given the text $x$ and the teacher tokenization functions $T(\cdot)$, it encode the text to a set of tokens $T(x) = \{t_1, t_2, \ldots, t_n\}$. Then we have $t_1 \odot t_2 \odot \ldots \odot t_n = x$, where $\odot$ is the concatenation operator. Similarly, the student tokenization function $S(\cdot)$ maps the text to $S(x) = \{s_1, s_2, \ldots, s_m\}$. Furthermore, the tokenization function is injective, and there exists a left inverse $D$ such that $x = D(T(x)), \forall x$. We call $D$ the *detokenization function*. Importantly, $D$ is only a *left* inverse, i.e., there may exist a token sequence $\boldsymbol{t}$ such that $\boldsymbol{t} \neq T(D(\boldsymbol{t}))$ since $T$ is not necessarily bijective. Generally, the vocabulary does not contain "empty tokens", i.e., $|D(t_i)| > 0$ for any single token $t_i \in \mathcal{V}$. This ensures the strict monotonicity of the detokenization function, i.e., $|D(\boldsymbol{t} \odot t')| > |D(\boldsymbol{t})|$. For simplicity, we do not distinguish between teacher $D_T$ and student $D_S$ in the following and denote them both as $D$.

### 3.2 On-Policy Distillation

On-Policy Distillation (OPD) computes supervision on trajectories sampled from the current student $\pi_\theta$. Given a prompt $x \sim \mathcal{D}_x$, the student samples a response $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_T) \sim \pi_\theta(\cdot \mid x)$. Both models are then evaluated on the student-generated prefixes $\hat{y}_{<t}$, yielding two next-token distributions at each step $t$: $\pi_\theta(v \mid x, \hat{y}_{<t})$ and $\pi_T(v \mid x, \hat{y}_{<t})$ for $v \in \mathcal{V}$. In this work, we follow the thinking machines lab (Lu & Lab, 2025) and adopt the token-level reverse KL as the objective function. This is the lightweight variant of OPD, evaluating only the tokens sampled by the student model; it also represents the most common implementation in previous on-policy distillation works (Lu & Lab, 2025; Team et al., 2026; Yang et al., 2025). We set the per-token advantage to the negative reverse KL, the divergence between the student's and teacher's distribution for each token conditioned on the same prior trajectory:

$$\hat{A}_{i,t} = \log \pi_T(\hat{y}_{i,t} \mid x, \hat{y}_{i,<t}) - \log \pi_{\theta_{\text{old}}}(\hat{y}_{i,t} \mid x, \hat{y}_{i,<t}). \tag{1}$$

The reward function minimizes the reverse KL, which pushes the student to approximate the teacher's behavior in every state the student finds itself in. Then we call the RL importance-sampling loss function to perform the training update on the student model:

$$\mathcal{J}_{\text{OPD}} = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_{\theta_{\text{old}}}} \min\left(r_{i,t}(\theta)\, \hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta),\, 1-\epsilon,\, 1+\epsilon\right)\, \hat{A}_{i,t}\right), \tag{2}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$ represents the importance-sampling term, and $\epsilon$ is the clipping hyper-parameter.

## 4 Method

In this section, we introduce the method for cross-tokenizer On-Policy Distillation. The challenge lies in the discrepancy between the student vocabulary and the teacher vocabulary, which prevents a direct token-to-token mapping of supervision signals. We first introduce our proposed chunk alignment algorithm to bridge this gap, followed by a chunk-level credit assignment based on semantic prior.

### 4.1 Adaptive Chunk Alignment

To ensure high-fidelity knowledge transfer, the alignment mechanism must operate at the highest possible resolution. Any alignment that aggregates multiple independent synchronization points into a single block will obscure the dense supervision signals provided by the teacher. Given a student-generated response $y^S = \{s_1, \ldots, s_m\} \sim \pi_\theta(\cdot \mid x)$ and its teacher-tokenized counterpart $y^T = \{t_1, \ldots, t_k\} = T(D(y^S))$, our goal is to identify synchronization points between the two sequences. Due to differing subword boundaries, a single token in one sequence may correspond to a

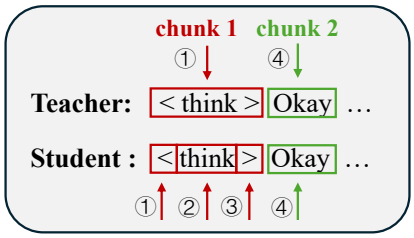


**Figure 1.** Adaptive Chunk Alignment.

partial or multiple tokens in the other. We propose the **Dual-Pointer Chunk Alignment (DPCA)** algorithm, which leverages the deterministic and monotonic properties of the detokenization function $D$ to find the *minimal synchronized chunks*.

**Definition 4.1** (Synchronization Point)**.** An token index pair $(i, j)$ is a *synchronization point* if the decoded prefixes match: $D(y^S_{1:i}) = D(y^T_{1:j})$.

**Algorithm Description.** We maintain two pointers, $p_S$ and $p_T$, representing the current boundaries in the student and teacher sequences, respectively. Starting from any synchronization points $p_S$ and $p_T$ (initialized to 0), we seek the smallest increments $\Delta m, \Delta k \geq 1$ such that:

$$D(s_{p_S}, \ldots, s_{p_S+\Delta m-1}) = D(t_{p_T}, \ldots, t_{p_T+\Delta k-1}). \tag{3}$$

To find these increments, we employ a greedy "catch-up" strategy based on the length of the decoded strings. Within a chunk, let $i$ be the current offset for the student token and $j$ be the offset for the teacher token. We iteratively advance the pointers as follows:

- Let $\text{str}_S = D(\{s_{p_S}, \ldots, s_{p_S+i-1}\})$ and $\text{str}_T = D(\{t_{p_T}, \ldots, t_{p_T+j-1}\})$.
- **If** $|\text{str}_S| < |\text{str}_T|$, we increment $i$ to extend the student's fragment.
- **Else if** $|\text{str}_S| > |\text{str}_T|$, we increment $j$ to extend the teacher's fragment.

The process for a single chunk terminates at the first instance where $\text{str}_S = \text{str}_T$. Upon matching, we define the aligned pair as a *synchronized chunk*, update the base pointers $p_S \leftarrow p_S + i$ and $p_T \leftarrow p_T + j$, and repeat the process until both sequences are exhausted.

**Proposition 4.2** (Completeness and Minimality of DPCA)**.** *The DPCA algorithm ensures: 1.**Completeness**: It identifies the entire set of synchronization points between $y^S$ and $y^T$. 2.**Minimality**: The resulting partitions are minimal synchronized chunks, such that no chunk can be further subdivided into smaller synchronized sub-sequences.*

The *completeness* guarantees that the algorithm never skips a synchronization point. The *minimality* implies that each identified chunk is a minimal closure. Consequently, the resulting partition has found all possible minimal chunks. This is crucial for OPD as it ensures that the teacher's log-probabilities are localized to the smallest possible token clusters, which avoids the ambiguous signal over excessively large chunks. In other words, DPCA ensures the highest partition resolution. Detailed proofs are provided in Appendix E.

### 4.2 Chunk-level Credit Assignment via Semantic Prior

Once the sequences are partitioned into a set of minimal synchronized chunks $\mathcal{P} = \{(\boldsymbol{s}_c, \boldsymbol{t}_c)\}_{c=1}^K$, the teacher's supervision signals should be projected back onto the student tokens. The projection, also referred to as credit assignment, must satisfy the total probability constraint while respecting the student's internal semantic structure.

**Joint Probability Matching.** Since each synchronized chunk $(\boldsymbol{s}_c, \boldsymbol{t}_c)$ represents the same content $x_{chunk} = D(\boldsymbol{s}_c) = D(\boldsymbol{t}_c)$, we assume that the joint probability of generating this content should be equivalent under both the student and teacher distributions:

$$P(\boldsymbol{s}_c \mid x, y^S_{<c}) = P(\boldsymbol{t}_c \mid x, y^T_{<c}), \tag{4}$$

where $y^S_{<c}$ and $y^T_{<c}$ are the prefixes to the current chunk. Under the autoregressive factorization, this equality implies a product of conditional probabilities:

$$\prod_{i=1}^{|\boldsymbol{s}_c|} \pi_\theta(s_{c,i} \mid x, y^S_{<c} \odot s_{c,<i}) = \prod_{j=1}^{|\boldsymbol{t}_c|} \pi_T(t_{c,j} \mid x, y^T_{<c} \odot t_{c,<j}). \tag{5}$$

Applying the logarithm transform, we obtain the constraint that the sum of log-probabilities within a synchronized chunk should be equal:

$$\sum_{i=1}^{|\boldsymbol{s}_c|} \log \pi_\theta(s_{c,i} \mid \ldots) = \sum_{j=1}^{|\boldsymbol{t}_c|} \log \pi_T(t_{c,j} \mid \ldots). \tag{6}$$

**Optimization with Semantic Prior.** In cross-tokenizer OPD, the teacher's density is aggregated over the teacher tokens $\boldsymbol{t}_c$. Let $\mathcal{L}_T^{(c)} = \sum_j \log \pi_T(t_{c,j} \mid \dots)$ be the target log-likelihood for the $c$-th chunk. To distribute this target to student tokens $\boldsymbol{s}_c = \{s_1, \dots, s_m\}$ while preserving the student's semantic prior $\boldsymbol{p} = \{\pi_{\theta_{\text{old}}}(s_i \mid \dots)\}_{i=1}^m$ (i.e., its pre-existing internal distribution), we seek a target distribution $\boldsymbol{q}$ that maintains the maximum directional alignment with the prior $\boldsymbol{p}$:

$$\min_{\boldsymbol{q}} \quad \frac{1}{2} \sum_{i=1}^{m} (\log q_i - k \log p_i)^2 \quad \text{s.t.} \quad \sum_{i=1}^{m} \log q_i = \mathcal{L}_T^{(c)}. \tag{7}$$

where $k$ is the directional scaling factor. By solving this constrained optimization problem with Lagrange multipliers, we obtain a closed-form solution where the target log-probabilities $\boldsymbol{q}$ are distributed proportionally to the student's prior:

$$\log q_i = \frac{\mathcal{L}_T^{(c)}}{\mathcal{L}_S^{(c)}} \log p_i. \tag{8}$$

where $\mathcal{L}_S^{(c)} = \sum_{j=1}^m \log p_j$ is the student's total log-likelihood for the chunk. The target $\log q_i$ is a linearly scaled version of the prior $\log p_i$, where the scaling factor is determined by the ratio of the chunk-level log-probabilities.

**Advantage Computation.** With the target log-probability, the per-token advantage is calculated as:

$$\hat{A}_{i,t} = \log q_i - \log p_i = \left( \frac{\mathcal{L}_T^{(c)}}{\mathcal{L}_S^{(c)}} - 1 \right) \log p_i, \tag{9}$$

**Remark.** This proportional assignment ensures that the relative semantic structure within the chunk is preserved. For instance, tokens that the student initially found more "surprising" (i.e., larger negative log-probabilities) receive a proportionally larger share of the total advantage. Furthermore, this formulation is self-consistent: in the case of 1:1 alignment ($m = 1$), the advantage naturally simplifies to $\hat{A}_{i,t} = \mathcal{L}_T^{(c)} - \mathcal{L}_S^{(c)}$, recovering the standard per-token distillation objective.

### 4.3 Cross-Tokenizer On-Policy Distillation

With the token-level advantages $\hat{A}_{i,t}$ established via chunk-level credit assignment, we integrate them into an on-policy framework to update the student model. Our objective function follow the standard form of OPD, and call the RL importance-sampling loss function:

$$\mathcal{J}_{\text{CTOPD}} = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_{\theta_{\text{old}}}} \min \left( r_{i,t}(\theta)\, \hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta),\, 1-\epsilon,\, 1+\epsilon\right)\, \hat{A}_{i,t} \right), \tag{10}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} | x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} | x, y_{i,<t})}$ represents the importance-sampling term, $\epsilon$ is the clipping hyper-parameter, and $\hat{A}_{i,t}$ is the per-token advantage assigned via the semantic prior (Eq. 9). The ovrall objective ensures that the student model is updated in such a way that, even if its tokenization differs from that of the teacher model, its internal semantic structure remains consistent with the supervision of the teacher model. Training on the student model's own rollouts, the model is able to learn from and correct its unique distributional errors within its own generative context.

## 5 Experiments

### 5.1 Experimental Setup

**Settings.** We consider two heterogeneous teacher–student pairs: **(1)** Qwen3-8B (Yang et al., 2025) → Llama3.1-8B-SFT (Grattafiori et al., 2024), and **(2)** DeepSeek-R1 (Guo et al., 2025) → DeepSeek-R1-Distill-Qwen-7B. Each student is initialized by SFT on 400K OpenThoughts examples using LLaMA-Factory (Zheng et al., 2024). We then train on 20K student-generated trajectories with VeRL (Sheng et al., 2024); a separate teacher service scores those trajectories online. Appendix A gives full hyperparameters.

**Optimization and rollout.** We train both stages with AdamW in bf16 and clip the gradient norm at 1.0. SFT uses a global batch of 128, a learning rate of $1.0 \times 10^{-4}$, and a maximum sequence length

of 16,384 tokens. CTOPD uses a batch of 128 prompts and a learning rate of $1.0 \times 10^{-6}$; prompts are capped at 1,024 tokens and student responses at 16,384 tokens. Rollouts are sampled with $T = 1.0$, top-$p = 1.0$, and top-$k = -1$. These settings are shared across CTOPD and the credit-assignment ablations.

**Implementation pipeline.** During CTOPD, the student generates trajectories and receives all parameter updates, while the frozen teacher only scores them. DPCA aligns the two tokenizations and projects teacher chunk likelihoods to student tokens before optimization; the same pipeline is used for both model pairs.

**Training data.** The training data primarily comes from OpenThoughts (Guha et al., 2025) and DeepMath (He et al., 2025). OpenThoughts covers various tasks (*e.g.*, math, code, science question, and general reasoning), which includes high quality data from 26 sources and uses QwQ-32B (Team, 2025) for all annotations. DeepMath provides more challenging, verifiable, and decontaminated math problems with rich data format and data diversity. We selected 400K samples from OpenThoughts to initialize the student's SFT, and respectively extracted 20K prompts from OpenThoughts and DeepMath for OPD training.

**Baselines.** For the full six-benchmark comparison, we evaluate the SFT initialization, off-policy cross-tokenizer CDM (Chen et al., 2025) and ULD (Boizard et al., 2025), and on-policy GOLD (Hugging Face, 2026). We separately adapt ALM (Minixhofer et al., 2025) to student-policy rollouts in the regime-matched four-benchmark ablation in Section 5.6. Per-benchmark main results are means and standard deviations over three independent runs; the Avg. column reports means only.

**Benchmarks and evaluation.** We evaluated the model's performance on AIME24, AIME25, AIME26 (Zhang & Math-AI, 2026), MATH-500 (Hendrycks et al., 2021), GPQA-Diamond (Rein et al., 2023), and LiveCodeBench (Jain et al., 2024). AIME and MATH-500 were used to measure mathematical reasoning ability, GPQA-Diamond to measure scientific question answering and complex knowledge reasoning ability, and LiveCodeBench to measure code generation and algorithmic reasoning ability. For AIME and MATH-500, we used answer extraction and exact-match judgment; for GPQA-Diamond, we reported multiple-choice accuracy; and for LiveCodeBench, we reported the test case accuracy. More details are in the Appendix B.

**Comparison protocol.** All distillation methods within a teacher–student block start from the same SFT checkpoint, so every reported gain measures the effect of the subsequent distillation stage rather than a stronger initialization. CTOPD always samples from the current student and queries the teacher only to score the resulting trajectory. This protocol keeps the deployment-time model and the training prompt budget fixed while changing the source and granularity of supervision. The two model pairs also test complementary forms of heterogeneity: the first transfers between Qwen and Llama vocabularies, whereas the second uses a much stronger teacher with a Qwen-based distilled student. We therefore report results by pair instead of pooling runs across substantially different starting accuracies.

### 5.2 Main Results

CTOPD is best on every benchmark in both settings. For Qwen3 $\rightarrow$ Llama, it improves the SFT average by $4.9$ points and GOLD by $2.2$; the largest gain over GOLD is on AIME25 ($+4.2$). For DeepSeek-R1 $\rightarrow$ Qwen, it improves SFT by $4.0$ and GOLD by $2.1$ points. The corrected GPQA-Diamond parser yields realistic accuracies rather than the spuriously low values in the earlier evaluation. Consistent gains over GOLD show that on-policy sampling alone does not explain the result: minimal chunk alignment and semantic-prior credit assignment add substantial value.

**Consistency across capabilities.** The gains are not confined to the mathematical data used during distillation. In the Qwen3 $\rightarrow$ Llama setting, CTOPD improves over SFT by $5.0$, $11.7$, $3.3$, and $2.1$ points on the four math benchmarks, but also by $4.0$ points on GPQA-Diamond and $3.2$ points on LiveCodeBench. Relative to the on-policy GOLD baseline, the corresponding gains remain positive on all six tasks, ranging from $1.0$ point on AIME26 to $4.2$ points on AIME25. This breadth is important because it indicates that cross-tokenizer supervision does not merely specialize the student to one answer format. Improvements on science and code suggest that the projected token-level signal remains useful when the generated response contains prose, symbols, and program text with substantially different segmentation patterns.

**Table 1.** Three-run mean ± standard deviation on reasoning, science, and coding benchmarks; Avg. reports the mean only. Best results in each teacher–student block are bold.

| Regime | Method | Math | | | | Science / Code | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | | AIME24 | AIME25 | AIME26 | MATH | GPQA-D | LCB | Avg. |
| **Qwen3-8B → Llama3.1-8B-SFT** | | | | | | | | |
| Init. | SFT | 39.2±0.7 | 30.8±0.5 | 37.5±0.8 | 80.2±0.6 | 29.3±0.4 | 21.3±0.9 | 39.7 |
| Off-policy | CDM | 40.0±0.8 | 31.7±0.5 | 36.7±0.9 | 81.1±0.4 | 30.2±0.6 | 22.3±1.1 | 40.3 |
| | ULD | 39.4±0.5 | 32.3±0.6 | 36.3±0.4 | 80.9±0.3 | 28.4±0.4 | 21.6±0.6 | 39.8 |
| On-policy | GOLD | 42.3±0.7 | 38.3±1.0 | 39.8±0.6 | 81.2±0.4 | 30.5±0.6 | 22.6±1.1 | 42.4 |
| | **CTOPD (ours)** | 44.2±1.2 | 42.5±0.9 | 40.8±1.1 | 82.3±0.7 | 33.3±1.3 | 24.5±0.8 | 44.6 |
| **DeepSeek-R1 → DeepSeek-R1-Distill-Qwen-7B** | | | | | | | | |
| Init. | SFT | 53.8±0.6 | 46.3±0.8 | 50.2±0.5 | 88.4±0.7 | 53.0±0.3 | 25.3±0.9 | 52.8 |
| Off-policy | CDM | 54.8±0.7 | 47.9±0.4 | 52.1±0.6 | 90.1±0.5 | 53.6±0.8 | 26.7±1.0 | 54.2 |
| | ULD | 53.3±0.4 | 46.9±0.3 | 50.8±0.5 | 88.8±0.2 | 53.4±0.6 | 25.0±0.4 | 53.0 |
| On-policy | GOLD | 56.5±0.9 | 48.8±0.8 | 52.7±0.6 | 90.3±0.7 | 53.8±0.4 | 26.1±0.3 | 54.7 |
| | **CTOPD (ours)** | 58.5±1.2 | 52.7±1.1 | 54.4±0.9 | 91.8±0.7 | 55.3±1.0 | 28.0±0.8 | 56.8 |

**Consistency across model pairs.** The DeepSeek-R1 → Qwen experiment starts from a substantially stronger SFT model, yet CTOPD retains a similar advantage. Its gains over SFT are 4.7, 6.4, 4.2, 3.4, 2.3, and 2.7 points from AIME24 through LiveCodeBench, respectively. CTOPD also exceeds GOLD on every task by 1.5–3.9 points. Thus, the benefit persists both across different model families and when the student is already a distilled reasoning model. Across the three independent runs, CTOPD's standard deviations are at most 1.3 points for the first pair and 1.2 for the second, comparable to the baselines. Although not an inferential significance test, consistent mean improvements across all twelve pair–benchmark combinations support the robustness of the comparison.

### 5.3 Compute Cost Analysis

We compare the additional data and FLOPs needed by continued SFT and CTOPD to reach the same AIME24 accuracy from the Llama3.1-8B-SFT checkpoint, using Qwen3-8B as teacher. SFT cost includes teacher rollout and student forward/backward; CTOPD cost includes student rollout, teacher prefill/scoring, and the student update. The SFT curve is fit to four measured checkpoints (200K, 300K, 400K, and 500K examples, with AIME24 accuracies 33.3, 36.8, 38.5, and 40.2). A log-linear fit has $R^2 = 0.9943$ and predicts 43.8 at 800K, close to the measured 44.6, supporting its use for local extrapolation. From SFT-400K, CTOPD reaches 40.5 and 44.4 with only 10K and 20K additional prompts. The fitted SFT curve requires about 471K additional examples to reach 44.4.

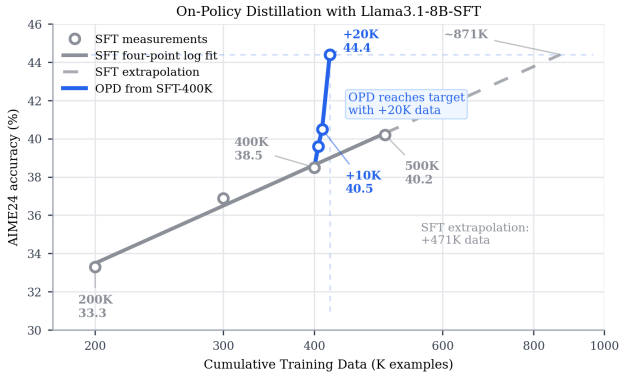


**Figure 2.** Measured AIME24 compute frontier. The SFT log fit uses four checkpoints; 800K is held out for validation.

Table 2 applies the same accounting to both methods. Continued SFT is estimated at about $1.13 \times 10^{21}$ FLOPs, whereas CTOPD uses $4.8 \times 10^{19}$ FLOPs, a 23.6× compute-efficiency improvement. Because one frontier is extrapolated, we report both the fit quality and the held-out checkpoint rather than treating the estimate as a direct measurement.

The efficiency gain is also visible in the data requirement: the local fit assigns continued SFT roughly 471K additional examples, compared with 20K on-policy prompts for CTOPD, again a 23.6× ratio. Moreover, the reduction is balanced across the two sides of training. Teacher compute decreases from $5.8 \times 10^{20}$ to $2.5 \times 10^{19}$ FLOPs, while student compute decreases from $5.5 \times 10^{20}$ to $2.3 \times 10^{19}$ FLOPs. CTOPD therefore does not shift an unreported workload from student to teacher. The

absolute values depend on the 16K-token accounting and local SFT extrapolation, while the 10K- and 20K-prompt CTOPD checkpoints establish the observed frontier.

**Table 2.** Estimated additional compute to reach 44.4 AIME24 from SFT-400K. FLOPs use the Qwen3-8B teacher and Llama3.1-8B student configurations with 16K-token samples.

| Method | AIME24 | Data | Teacher FLOPs | Student FLOPs | CE |
|---|---|---|---|---|---|
| Initialization: SFT-400K | 38.5% | – | – | – | – |
| SFT extrapolated | 44.4% | +471K | $5.8\times10^{20}$ | $5.5\times10^{20}$ | 1.0× |
| **SFT + CTOPD** | **44.4%** | **+20K** | $2.5\times10^{19}$ | $2.3\times10^{19}$ | **23.6**× |

### 5.4 Case Study

Figure 3 illustrates both directions of tokenizer mismatch within a single response. For `120`, Qwen produces three tokens, `1`, `2`, and `0`, whose log-probabilities sum to $-7.12$, while Llama represents the same span with one token of log-probability $-9.21$. After chunk alignment, the teacher likelihood directly yields an advantage of $2.09$ for the Llama token.

Conversely, Qwen represents `...]` as one token with log-probability $-15.91$, whereas Llama splits the same span into `...` and `]`. CTOPD redistributes the teacher chunk likelihood according to the student prior, assigning targets $-7.56$ and $-8.35$ and yielding advantages $-0.63$ and $-0.70$. These examples show that the same alignment-and-projection mechanism naturally handles both many-to-one and one-to-many mismatches while preserving dense token-level supervision. Importantly, the projection operates on synchronized text spans rather than requiring an explicit token-to-token correspondence. This makes the procedure applicable to heterogeneous tokenizer pairs without introducing tokenizer-specific mapping rules.

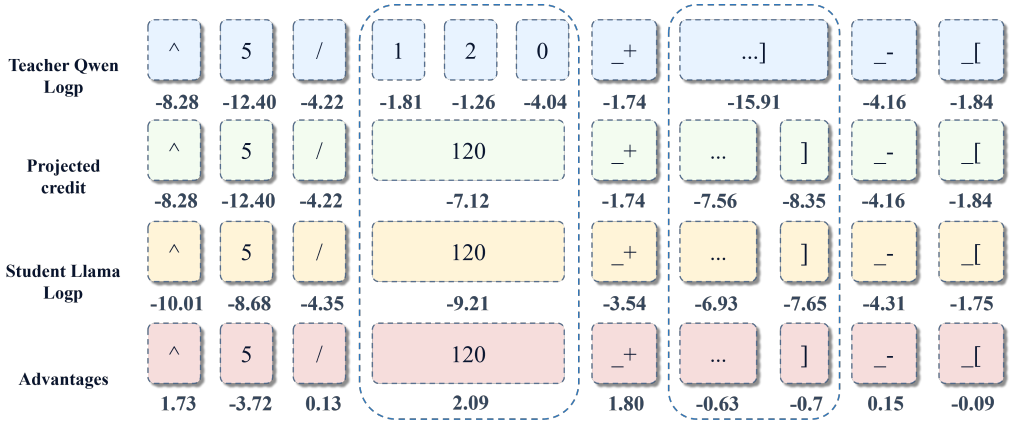


**Figure 3.** Case study of cross-tokenizer credit assignment. From top to bottom, the rows show teacher log-probabilities, projected teacher log-probabilities, student log-probabilities, and the resulting advantages computed as projected teacher log-probabilities minus student log-probabilities. Blue groups are mismatched chunks; gray dashed groups are directly aligned chunks.

### 5.5 Distillation Gain: Same- vs. Cross-Tokenizer OPD

To isolate the effect of tokenizer projection from differences in model strength, we compare each method's improvement over its own SFT initialization on the four math benchmarks shared by the cross-tokenizer results in Table 1 and the same-tokenizer reference in Appendix D. The same-tokenizer setting uses sampled-token OPD, whereas the cross-tokenizer setting uses CTOPD; Top-32 GKD is retained in the appendix as a higher-information reference that requires teacher logits in a shared vocabulary.

**Table 3.** Gain over the corresponding SFT initialization on matched benchmarks. Same-tokenizer results use Qwen3 teacher–student models with sampled-token OPD; cross-tokenizer results use Qwen3-8B → Llama3.1-8B-SFT with CTOPD.

| Benchmark | Same-tokenizer | Cross-tokenizer |
|---|---|---|
| AIME24 | +5.0 | +5.0 |
| AIME25 | +7.9 | **+11.7** |
| AIME26 | **+6.8** | +3.3 |
| MATH | **+2.4** | +2.1 |
| **Avg.** | **+5.5** | **+5.5** |

Both settings improve by $5.5$ points on average. Although the per-benchmark gains differ—cross-tokenizer CTOPD is especially strong on AIME25, while same-tokenizer OPD gains more on AIME26—their matched average is identical. Thus, the alignment and projection step preserves the incremental benefit of dense on-policy supervision despite the vocabulary mismatch.

Because the students differ in absolute capability, Table 3 reports gains rather than raw scores. The matched average indicates no systematic tokenizer-conversion penalty, although per-benchmark behavior differs.

### 5.6 Credit Assignment Ablation and Overhead

Table 4 holds the on-policy rollouts and optimization protocol fixed and changes only the within-chunk credit rule. Uniformly distributing either teacher log-likelihood or chunk advantage improves over SFT, but CTOPD is $0.8$–$0.9$ points better on average. Replacing our rule with on-policy ALM also trails by $1.5$ points. Thus the main gains do not arise solely from moving an existing cross-tokenizer objective on-policy.

**Table 4.** Single controlled credit-assignment ablation run for Qwen3-8B → Llama3.1-8B-SFT on four math benchmarks.

| Credit rule | AIME24 | AIME25 | AIME26 | MATH | Avg. |
|---|---|---|---|---|---|
| SFT initialization | 38.3 | 31.7 | 36.5 | 80.8 | 46.9 |
| Uniform chunk log-likelihood | 42.9 | 42.1 | **41.7** | 82.2 | 52.2 |
| Uniform chunk advantage | 42.5 | 43.1 | 40.8 | 81.9 | 52.1 |
| On-policy ALM | 42.5 | 41.7 | 39.2 | **82.6** | 51.5 |
| **Semantic prior (CTOPD)** | **44.4** | **43.3** | **41.7** | **82.6** | **53.0** |

The alignment is usually near token-level: on Qwen3 → Llama, $92.74\%$ of matched tokens form one-to-one chunks. Of the remaining $7.26\%$, $99.63\%$ of chunks span at most two tokens on either side and $99.99\%$ span at most three. Across tested tokenizer pairs, one-to-one chunks account for $90$–$95\%$, while chunks longer than three tokens are below $0.001\%$. For a 16,384-token student sequence (about 13.7K–16.5K teacher tokens), CPU DPCA takes $304 \pm 20$ ms; even when processed serially for a batch of 64, alignment contributes only $4$–$7\%$ of an end-to-end training step.

The semantic prior is better than both uniform rules on AIME24 and AIME25, ties the best AIME26 result, and jointly reaches the best MATH score. Its advantage is therefore not driven by one benchmark. The modest average gap is expected because the rule acts only on mismatched chunks, which nevertheless accumulate over long reasoning trajectories.

## 6 Conclusion

In this work, we present Cross-Tokenizer On-Policy Distillation (CTOPD), a framework that allows OPD across different model families. We propose an adaptive dual-pointer chunk alignment algorithm and a semantic-prior-based credit assignment mechanism to enable the precise propagation of high-fidelity, token-level signals across different vocabularies. Our results demonstrate that CTOPD is significantly more compute-efficient and effective at capturing dense distributional knowledge. Our work expands the potential for teacher–student pairings and paves the way for more flexible knowledge transfer across diverse model families.

## AI USE STATEMENT

We used generative AI tools to improve the readability and clarity of the manuscript, including language editing and suggestions on presentation. All AI-assisted revisions were reviewed by the authors, who take full responsibility for the final content of this work.

## 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, 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, 2025. URL https://arxiv.org/abs/2402.12030.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.419. URL https://aclanthology.org/2025.findings-acl.419/.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models, 2024. URL https://arxiv.org/abs/2401.01335.

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, 2025. URL https://arxiv.org/abs/2412.14528.

GLM-5-Team, Aohan Zeng, Xin Lv, et al. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models, 2026. URL https://arxiv.org/abs/2306.08543.

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, 2025. URL https://arxiv.org/abs/2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

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, 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. *NeurIPS*, 2021.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Hugging Face. General online logit distillation trainer. TRL documentation, 2026. URL https://huggingface.co/docs/trl/v0.27.0/gold_trainer. Accessed 2026-09-25.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding, 2020. URL https://arxiv.org/abs/1909.10351.

Seongryong Jung, Suwan Yoon, DongGeon Kim, and Hwanhee Lee. Todi: Token-wise distillation via fine-grained divergence control, 2025. URL https://arxiv.org/abs/2505.16297.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms?, 2026. URL https://arxiv.org/abs/2603.24472.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models, 2024. URL https://arxiv.org/abs/2402.03898.

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, 2025. URL https://arxiv.org/abs/2503.07067.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, 2018. URL https://arxiv.org/abs/1808.06226.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching, 2025. URL https://arxiv.org/abs/2503.20083.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6403–6417. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.353. URL http://dx.doi.org/10.18653/v1/2023.acl-long.353.

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, 2024. URL https://arxiv.org/abs/2412.09871.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016. URL https://arxiv.org/abs/1508.07909.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

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

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models, 2026. URL https://arxiv.org/abs/2604.00626.

Core Team, Bangjun Xiao, Bingquan Xia, et al. Mimo-v2-flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Hongling Xu, Qi Zhu, Heyuan Deng, Jinpeng Li, Lu Hou, Yasheng Wang, Lifeng Shang, Ruifeng Xu, and Fei Mi. Kdrl: Post-training reasoning llms via unified knowledge distillation and reinforcement learning, 2025. URL https://arxiv.org/abs/2506.02208.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, and Zhipeng Wang. Paced: Distillation and on-policy self-distillation at the frontier of student competence, 2026. URL https://arxiv.org/abs/2603.11178.

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, 2022. URL https://arxiv.org/abs/2105.13626.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation, 2026. URL https://arxiv.org/abs/2602.12125.

Songming Zhang, Xue Zhang, Tong Zhang, Bojie Hu, Yufeng Chen, and Jinan Xu. Alignditil: Token-level language model alignment as adaptive policy distillation, 2025. URL https://arxiv.org/abs/2503.02832.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models, 2024. URL https://arxiv.org/abs/2403.13372.

Vilém Zouhar, Clara Meister, Juan Luis Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A formal perspective on byte-pair encoding, 2024. URL https://arxiv.org/abs/2306.16837.

# A Details of Experimental Setups

**Table 5.** Main training hyperparameters used for SFT initialization and subsequent OPD training.

| **Hyperparameter** | **SFT** | **CTOPD** |
|---|---|---|
| Training framework | LLaMA-Factory | VeRL |
| Teacher–student pairs | – | Qwen3-8B → Llama3.1-8B-SFT; DeepSeek-R1 → DeepSeek-R1-Distill-Qwen-7B |
| SFT base models | Meta-Llama-3.1-8B | corresponding SFT checkpoint |
| Training data | OpenThoughts | DeepMath and OpenThoughts prompts |
| Global batch size | 128 examples | 128 prompts |
| Maximum sequence length | 16,384 tokens | 1,024 prompt tokens + 16,384 response tokens |
| Learning rate | $1.0 \times 10^{-4}$ | $1.0 \times 10^{-6}$ |
| Optimizer | AdamW | AdamW |
| Gradient clipping | 1.0 | 1.0 |
| Precision | bf16 | bf16 |
| Rollout sampling | – | $T = 1.0$, top-$p = 1.0$, top-$k = -1$ |

# B Details of Dataset and Baseline

The following are the details of the datasets used in the benchmark:

- **MATH** (Hendrycks et al., 2021) consists of 12,500 problems from high school math competitions, measuring the problem-solving ability of machine learning models.
- **LiveCodeBench** (Jain et al., 2024) provides holistic and contamination-free evaluation of coding capabilities of LLMs. We test models on code generation tasks where models are given a problem statement, which includes a natural language description and example tests (input-output pairs), and is tasked with generating a correct solution.
- **AIME** (Zhang & Math-AI, 2026) is a set of math problems from the American Invitational Mathematics Examination.
- **GPQA-Diamond** (Rein et al., 2023) is a comprehensive benchmark that evaluates graduate-level knowledge and reasoning capabilities across 285 disciplines.

For GPQA-Diamond, a single answer parser is used for every method. It recognizes letter-only answers, parenthesized letters, and common verbal forms before applying the same exact-match rule. This corrects the artificially low accuracies produced by the earlier parser without changing model outputs.

# C Robustness of Text Alignment

Before DPCA, student and teacher decodes are reconstructed as normalized UTF-8 strings using Unicode NFC. Special tokens are mapped explicitly (for example, EOS to EOS), and byte-fallback pieces are aligned only after their complete Unicode sequence has been reconstructed. We observed zero alignment-constraint violations in tests covering whitespace, control symbols, byte-fallback tokens, and non-ASCII text.

Table 6 gives two real mismatched chunks. CTOPD preserves the student's relative log-probability profile while rescaling it to the teacher chunk budget, rather than assigning all tokens the same target.

**Table 6.** Examples of semantic-prior credit assignment. Entries list student tokens, their original probabilities, and projected target probabilities.

| Text | Student tokens | Student $p$ | Target $q$ |
|---|---|---|---|
| `surjective` | [`sur`, `jective`] | [0.142, 0.999] | [0.383, 1.000] |
| `convergent` | [`conver`, `gent`] | [0.389, 0.979] | [0.838, 0.996] |

## D SAME-TOKENIZER TOP-$k$ REFERENCE

To contextualize sparse teacher feedback, Table 7 reports a same-tokenizer Qwen3 teacher–student experiment. Top-32 GKD improves over sampled-token supervision, but requires access to teacher logits in the shared vocabulary. It is therefore a reference rather than a direct cross-tokenizer baseline; CTOPD targets the setting in which those logit coordinates do not exist.

**Table 7.** Same-tokenizer Qwen3 reference with mean $\pm$ standard deviation over three runs.

| Method | AIME24 | AIME25 | AIME26 | MATH | Avg. |
|---|---|---|---|---|---|
| SFT | 60.0 | 56.7 | 56.7 | 86.6 | 65.0 |
| Sampled-token OPD | 65.0 | 64.6 | 63.5 | 89.0 | 70.5 |
| Top-32 GKD | 66.0$\pm$0.9 | 65.4$\pm$1.2 | 64.6$\pm$0.7 | 89.8$\pm$0.6 | 71.5 |

## E PROOF

**Theorem E.1** (Completeness and Minimality of DPCA)**.** *The DPCA algorithm ensures: 1.**Completeness**: It identifies the entire set of synchronization points between $y^S$ and $y^T$. 2.**Minimality**: The resulting partition $P^*$ consists of minimal synchronized chunks, such that no chunk within $P^*$ can be further subdivided into smaller synchronized sub-sequences.*

The *completeness* guarantees that the algorithm never skips a synchronization point, regardless of the extent of the vocabulary mismatch. The *minimality* implies that each identified chunk is a minimal closure. Consequently, the resulting partition has the maximal possible number of chunks. This is crucial for OPD as it ensures that the teacher's log-probabilities are localized to the smallest possible token clusters, which avoids the ambiguous signal over excessively large chunks.

*Proof.* We prove this by contradiction and the property of greedy first-match.

*Completeness*: Suppose there exists a synchronization point $(i^*, j^*) \in \mathcal{V}_{\text{sync}}$ that the algorithm fails to identify. Let $(i_r, j_r)$ be the last synchronization point identified by the algorithm such that $i_r < i^*$. During the subsequent search, the DPCA pointers advance based on the catch-up heuristic. Due to the strict monotonicity $|D(\boldsymbol{\tau} \odot \tau')| > |D(\boldsymbol{\tau})|$, the pointers will traverse the state space and eventually reach the indices $(i^*, j^*)$. At this state, the condition $D(y^S_{i_r:i^*}) = D(y^T_{j_r:j^*})$ is satisfied. Since the algorithm terminates the current chunk at the *first* instance this equality holds, it must identify $(i^*, j^*)$ as a synchronization point, contradicting the assumption.

*Minimality*: Since DPCA identifies every synchronization point in $\mathcal{V}_{\text{sync}}$ and partitions the sequences at every such point, any resulting chunk between $(i_k, j_k)$ and $(i_{k+1}, j_{k+1})$ contains no intermediate synchronization points by definition. Thus, each chunk is an atomic (minimal) synchronized unit. □