# ByteDistill: Cross-Tokenizer Distillation via Chunk-wise Byte-Level Distribution Alignment

**Yang Chen** **Xianqi Yu** **Shaowei Yao**
**Fuyu Lv** **Dan Ou** **Haihong Tang**
Taobao & Tmall Group of Alibaba
yisheng.cy@alibaba-inc.com
{xianqi.yxq,yaoshaowei,fuyu.lfy,oudan.od,piaoxue}@taobao.com

## Abstract

Teacher and student language models often use different tokenizers, so their next-token probabilities cannot be compared directly. ByteDistill expresses both predictions over the same 256 byte values. It groups overlapping teacher and student token spans into chunks covering the same text and, within each chunk, sums token probabilities according to the next UTF-8 byte. At an observed token boundary, the ending probability is assigned to the next token's byte distribution. Byte-Level Distribution Alignment (BLDA) then compares the resulting conditional byte distributions with KL divergence, preserving each model's tokenizer and token history. In off-policy distillation from Qwen3-8B to Gemma-2-2B, ByteDistill reaches $49.27\%$ average Pass@1 on GSM8K and MATH, exceeding SFT by $5.40$ points and CTLS by $1.78$ points. It also achieves $38.65$ ROUGE-L on DialogSum with 1,500 training examples and improves Countdown accuracy for all three students in on-policy distillation. The project repository is available at `https://github.com/Piplup0924/ByteDistill`.

## 1 Introduction

Knowledge distillation trains a student language model to match a teacher's predictions. Learning from the teacher's full next-token distribution can provide more information than learning from a single target token [Hinton et al., 2015, Kim and Rush, 2016, Gu et al., 2023]. The comparison can use fixed training responses or responses sampled from the student during training, as in on-policy distillation [Agarwal et al., 2023]. In either case, a token-level loss assumes that the two models assign probabilities to the same tokens.

Different tokenizers break this correspondence between tokens. The word `playing`, for example, may be split into `[play][ing]` by one tokenizer and represented as `[playing]` by another. The models then predict different text units at different positions in the sequence. Equal vocabulary sizes would not resolve the mismatch: corresponding softmax entries would still refer to different events. This problem commonly arises when the teacher and student come from different model families and use different subword vocabularies [Sennrich et al., 2016, Kudo and Richardson, 2018].

Both tokenizations of `playing` decode to the same sequence of UTF-8 bytes, providing a common basis for comparing the models. At the same position in this text, we can compare their probabilities for the next byte. For example, the candidate tokens `play`, `plane`, and `please` all begin with `p`, so each contributes to the probability of that first byte. Summing these contributions gives byte probabilities directly from each model's native softmax.

**ByteDistill** uses this construction for cross-tokenizer distillation. Overlapping token spans are grouped into shared text chunks. At each byte position within a chunk, the models' predictions

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

are converted to 256-way byte distributions and compared with KL divergence. Token boundaries require a further step. After `play`, the teacher may advance to its next token while the student remains inside `playing`. ByteDistill combines the probability of continuing a longer candidate token with the probability of ending the current token and predicting a byte from the next state. The resulting objective is **Byte-Level Distribution Alignment** (BLDA).

In off-policy distillation from `Qwen3-8B` to `Gemma-2-2B`, ByteDistill reaches $49.27\%$ average zero-shot Pass@1 on GSM8K and MATH, compared with $43.87\%$ for SFT and $47.49\%$ for CTLS. On DialogSum, it achieves $38.65$ ROUGE-L with 1,500 student training examples. In on-policy Countdown training, it improves accuracy over the SFT initialization and the compared distillation methods for all three students, whose tokenizer overlap with the teacher ranges from $0.999$ to $0.063$.

Our contributions are:

1. A method for comparing teacher and student next-byte distributions while keeping their original tokenizers and token histories.
2. A linear-time chunking algorithm and a byte-probability construction that handles interleaved token boundaries. We show that the construction gives a normalized distribution at every valid comparison position.
3. Experiments on mathematical reasoning, low-resource summarization, and on-policy Countdown training, including additional student models and Chinese mathematical reasoning.

## 2 Related Work

**Distillation and byte-level modeling.** Knowledge distillation transfers a teacher's predictions to a student [Hinton et al., 2015, Kim and Rush, 2016, Gu et al., 2023, Agarwal et al., 2023]. Most language models predict subword tokens produced by BPE or SentencePiece [Sennrich et al., 2016, Kudo and Richardson, 2018]. Token-free models instead operate on characters or bytes [Clark et al., 2022, Xue et al., 2022, Tay et al., 2022, Yu et al., 2023]. ByteDistill keeps the original subword models and uses bytes only when computing the distillation loss.

**Cross-tokenizer distillation.** Existing approaches align the models' probabilities or representations across their token spaces. Universal Logit Distillation compares sorted token probabilities, Multi-Level Optimal Transport aligns token and sequence distributions, and Contextual Dynamical Mapping learns a context-dependent mapping between vocabularies [Boizard et al., 2025, Cui et al., 2024, Chen et al., 2025].

Likelihood-based methods compare the probability of the same text under the two tokenizations. ALM uses binarized likelihood comparisons [Minixhofer et al., 2025], while CTLS scores text across tokenizers through the recursive structure of BPE [Phan et al., 2026]. DSKDv2 aligns hidden representations and selects matching text spans through exact token alignment [Zhang et al., 2025]. ByteDistill compares the conditional distribution of the next byte at a shared position in the decoded text. Summing native token probabilities produces these distributions even at positions where the teacher and student observe different tokens.

**On-Policy Distillation (OPD).** On-policy distillation queries the teacher's predictions on contexts generated by the current student [Gu et al., 2023, Agarwal et al., 2023]. GOLD extends this training setting across model families through sequence alignment and partial token mappings [Patiño et al., 2025]. ByteDistill uses the decoded student responses to locate shared byte positions for training. Each model retains its own token history when computing a byte distribution, and reverse KL compares the two distributions at each of these positions.

## 3 Background

**Notation.** Let $\mathcal{B} = \{0, \ldots, 255\}$ denote the UTF-8 byte alphabet and $\Delta_{256}$ the set of probability distributions over $\mathcal{B}$. A tokenizer vocabulary $\mathcal{V}$ is partitioned into textual tokens $\mathcal{V}_{\text{txt}}$ and structural control tokens $\mathcal{V}_{\text{spc}}$ such as EOS, padding, or chat-template markers. Each textual token $v \in \mathcal{V}_{\text{txt}}$ has a deterministic byte realization $\text{bytes}(v) = (b_1, \ldots, b_L)$ with $b_i \in \mathcal{B}$. Structural tokens control decoding but are not assigned text bytes. A token sequence $\mathbf{e} = (e_1, \ldots, e_N)$ deterministically

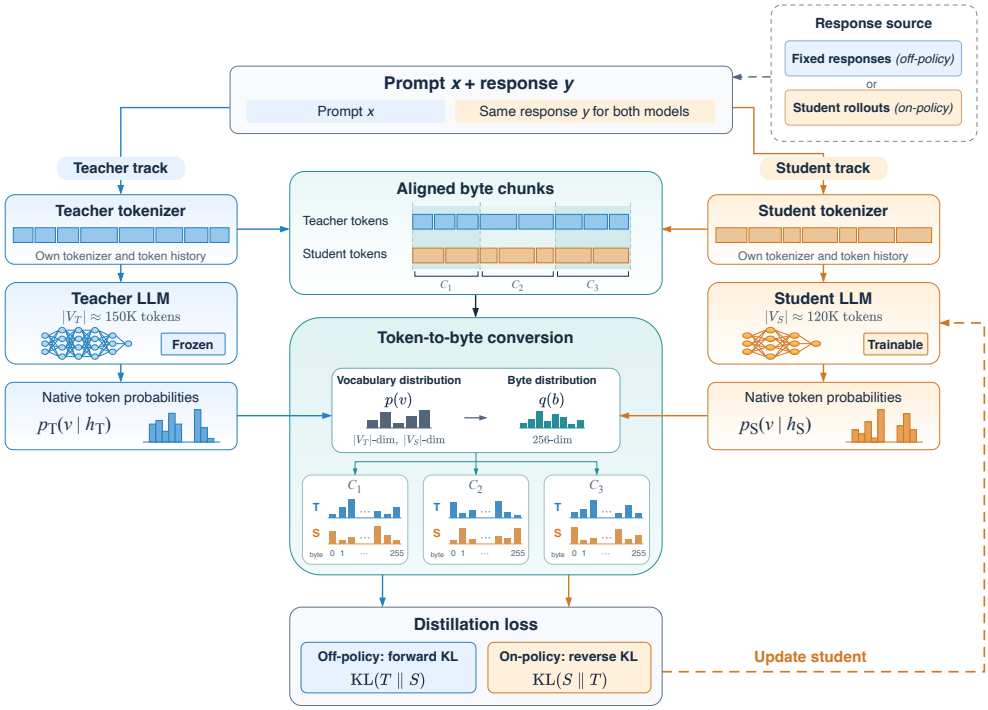



Figure 1: ByteDistill training pipeline. Teacher and student vocabulary distributions are mapped to 256-dimensional next-byte distributions and compared at matching byte positions within aligned chunks. Training uses fixed responses and forward KL off-policy, or student rollouts and reverse KL on-policy. Only the student is updated.

decodes to the concatenated byte sequence $\mathrm{bytes}(\mathbf{e})$. We use 1-based indexing, $\mathbf{e}_{<i}$ for the token prefix before position $i$, $\beta \oplus b$ for appending byte $b$ to a byte prefix $\beta$, and $\epsilon$ for the empty byte prefix. For byte strings $\gamma$ and $w$, we write $\gamma \sqsubseteq w$ when $\gamma$ is a prefix of $w$ (including $\gamma = w$), and $\gamma \sqsubset w$ for strict prefix inclusion.

**Vocabulary Misalignment.** Given its token history $h$, a language model with parameters $\theta$ predicts $p_\theta(v \mid h)$ for each token $v \in \mathcal{V}$. When the teacher and student share a tokenizer, distillation can minimize $D_{\mathrm{KL}}(p_T(\cdot \mid h) \| p_S(\cdot \mid h))$. With different tokenizers, corresponding entries refer to different text fragments, so a cross-tokenizer loss must express both models' predictions over the same set of outcomes before comparing them.

## 4 Methodology

ByteDistill aligns the two tokenizations by the text they decode to, then compares their predictions in byte space. We first form shared chunks and identify the token state for each byte position (Section 4.1). From these states, we derive next-byte distributions (Section 4.2), whose KL divergence defines the training loss. Figure 1 shows how these steps fit into the training pipeline.

The models continue to process their own token sequences. Even at the same byte position, their predictions may come from different token histories: one model may have completed a token while the other is partway through a longer one. Their matched byte prefixes can differ for the same reason. We therefore retain each model's history and prefix when converting its next-token probabilities, so that the resulting distributions refer to the same next-byte outcomes.

### 4.1 Building Aligned Byte Chunks

We start with teacher and student tokenizations of the same response. Each token occupies an interval of byte positions. We group tokens whose intervals overlap across the two tokenizations;

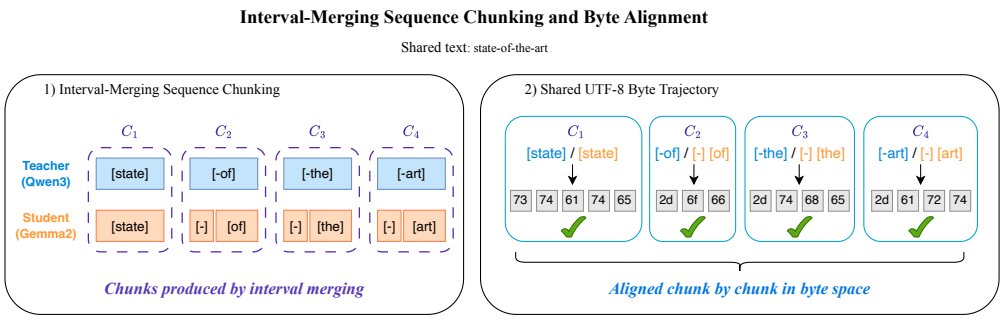



Figure 2: Building aligned byte chunks. The Qwen3 (teacher) and Gemma2 (student) tokenizations of *"state-of-the-art"* have different boundaries. Grouping overlapping token spans produces four chunks $C_1, \ldots, C_4$, each covering the same substring on both sides.

each connected group forms a *chunk*. The teacher and student tokens in a chunk cover exactly the same substring, although their internal boundaries can differ.

For example, Qwen3 tokenizes `Simplifying` as `[S]` `[impl]` `[ifying]`, while Gemma-2 uses `[Simp]` `[lifying]`. Their internal boundaries are at byte positions $\{1, 5\}$ and $\{4\}$, respectively. The overlapping spans connect all five tokens into one chunk covering the eleven bytes. Figure 2 shows a response with several chunks; Appendix E.1 lists all state pairs for `Simplifying`.

Chunk construction requires a single pass through the ordered token intervals. We join overlapping teacher and student tokens, then advance the pointer whose interval ends first. Before retaining a component, we check that it contains tokens from both models and that they cover the same byte span. Full tokenizations of the same response satisfy these checks. In an audit of 80 responses, we retained all 20,411 components and covered all 55,602 bytes (Appendix F). Algorithm 1 gives the procedure.

For each byte position, we then identify the native token state from which to compute its distribution. Let $[s_C, e_C)$ denote the span of a chunk, with $t \in [s_C, e_C)$ marking the position just before byte $t$ is generated. The state $(h, \beta)$ records the token history $h$ that produced the softmax and the byte prefix $\beta$ matched within its candidate tokens. We select this state separately for each model because a shared position may fall inside one model's token and at the end of the other's.

*State selection.* Each side applies the following rules:

- At the chunk start $t = s_C$, use INSIDE: $h$ is the history before the first chunk token, and $\beta = \epsilon$.
- At an internal token boundary $s_C < t < e_C$, use BOUNDARY: if token $e_k$ ends at $t$, select $(h, \beta) = (\mathbf{e}_{<k}, \text{bytes}(e_k))$.
- At any other position, use INSIDE: $h$ is the history before the containing token $e_k$, and $\beta$ is the part of that token before $t$.

*Boundary state selection.* At the end of the observed token `play`, we use its original state $(h, \texttt{play})$, whose softmax also contains longer candidates such as `player` and `playful`. Their continuation probabilities would be lost if we moved directly to $(h \oplus \texttt{play}, \epsilon)$. Instead, we use that next state's distribution only for the probability assigned to ending `play`, as defined below. This combines continuing within a longer token with starting a new token in a single comparison. Adding the next state as another loss term would count the same next-byte position twice, since its contribution is already included through the ending probability.

$$\underbrace{\texttt{[play]}}_{h,\,\beta=\texttt{play}} \Big|_t \underbrace{\texttt{[next token]}}_{h\oplus\texttt{play},\,\beta=\epsilon} \qquad \text{select the left boundary state at } t.$$

### 4.2 Computing Next-Byte Probabilities

Consider the response `playing`, tokenized as `[play][ing]` by the teacher and `[playing]` by the student. After the prefix `play`, the student's observed token continues. The teacher has two sources of next-byte probability: longer candidate tokens that continue after `play`, and the token `play` itself followed by a new token. We compute these two contributions separately.

**Continuing or ending a token.** Fix a model state $(h, \beta)$ with next-token probabilities $p_\theta(v \mid h)$. For each byte $b$, let $\mathbf{r}_b$ be the total probability of textual tokens that begin with $\beta \oplus b$. Let $e$ be the total probability of tokens that end exactly at $\beta$:

$$\mathbf{r}_\theta^{\mathcal{V}}(h,\beta)_b = \sum_{\substack{v \in \mathcal{V}_{\text{txt}}: \\ \beta \oplus b \sqsubseteq \text{bytes}(v)}} p_\theta(v \mid h), \qquad e_\theta^{\mathcal{V}}(h,\beta) = \sum_{\substack{v \in \mathcal{V}_{\text{txt}}: \\ \text{bytes}(v) = \beta}} p_\theta(v \mid h), \tag{1}$$

The total probability of tokens matching the prefix is $z_\theta^{\mathcal{V}}(h,\beta) = \mathbf{1}^\top \mathbf{r}_\theta^{\mathcal{V}}(h,\beta) + e_\theta^{\mathcal{V}}(h,\beta)$.

**Handling token boundaries.** Inside an observed token, we normalize the continuation probabilities $\mathbf{r}$. At an observed boundary, ending the token does not emit a byte. We therefore assign its probability $e$ to the next token's byte distribution, $\mathbf{q}_{\text{next}}$, and add that contribution to $\mathbf{r}$.

Writing $\mathbf{r} = \mathbf{r}_\theta^{\mathcal{V}}(h,\beta)$ and $e = e_\theta^{\mathcal{V}}(h,\beta)$, the next-byte distribution is

$$\hat{q}_\theta^{\mathcal{V}}(b \mid h, \beta) = \begin{cases} \dfrac{\mathbf{r}_b}{\mathbf{1}^\top \mathbf{r}}, & \textsc{Inside}, \\ \dfrac{\mathbf{r}_b + e\, \mathbf{q}_{\text{next},b}}{\mathbf{1}^\top \mathbf{r} + e}, & \textsc{Boundary},\ \text{bytes}(e_k) = \beta. \end{cases} \tag{2}$$

Here, $\mathbf{q}_{\text{next}}$ is computed by applying the INSIDE branch at $(h \oplus e_k, \epsilon)$, after consuming the observed token $e_k$. Thus the boundary branch is a weighted combination of continuing within a longer candidate token and starting the next token. The text may be a training target, a teacher-generated response, or a student rollout; in each case we follow its observed tokenization. We refer to this choice as *observed-path closure*. Appendix C describes the alternative that sums over other compatible token histories.

A comparison position is valid when both models have positive normalization denominators. The BOUNDARY branch additionally requires a valid next-state byte distribution whenever $e > 0$. At a boundary, the observed token must also be the unique textual token with that byte string after canonicalization. Appendix D specifies these checks, including duplicate byte strings and unavailable next states. The resulting distribution is non-negative and sums to one (Proposition 1, proved in Appendix F.1), so both models define valid distributions on the same byte alphabet for the KL loss. Algorithm 2 gives the vectorized implementation.

**Text and control tokens.** The sums in Eq. (1) include textual tokens only, so BLDA compares probabilities conditional on continuing the text stream. Control tokens such as EOS and chat-template markers have no shared byte representation. Their behavior is learned through the SFT loss in off-policy training and carried over from SFT initialization and the prompt template in on-policy training.

**From token probabilities to a KL term.** Table 1 gives illustrative native distributions for the response `playing`. At the empty prefix, all candidates beginning with `p` contribute to that byte: the teacher assigns total probability $0.85$ and the student $0.90$. The remaining probabilities, $0.15$ and $0.10$, are assigned to `d`. Deeper in the tree, only tokens matching the observed byte prefix contribute.

At the selected teacher boundary state $(h, \beta = \texttt{play})$, the original softmax assigns $0.30$ to `play`, $0.18$ to `player`, and $0.12$ to `playful`. Thus $e_T = 0.30$, $r_T(\texttt{e}) = 0.18$, and $r_T(\texttt{f}) = 0.12$. Suppose the next teacher state predicts `ing` with probability one. The ending contribution then goes to byte `i`, giving

$$\hat{q}_T(\texttt{i}, \texttt{e}, \texttt{f}) = \frac{(0.30, 0.18, 0.12)}{0.60} = (0.50, 0.30, 0.20).$$

The student's observed token `playing` continues at this position. Its matching candidates `playing`, `player`, and `playful` have probabilities $0.40$, $0.12$, and $0.08$. The INSIDE branch gives

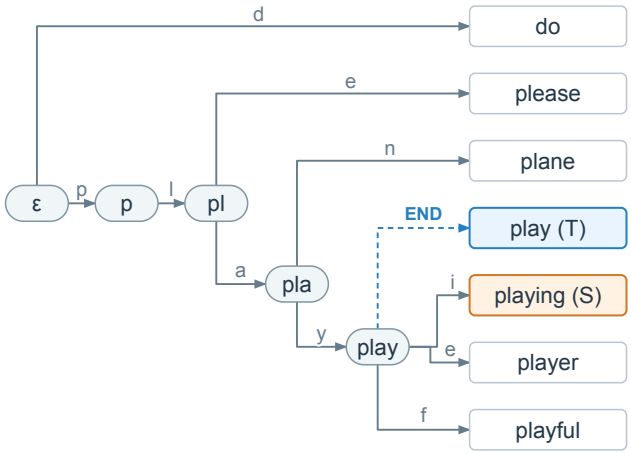
$$\hat{q}_S(\texttt{i}, \texttt{e}, \texttt{f}) = \frac{(0.40, 0.12, 0.08)}{0.60} = (2/3, 1/5, 2/15).$$

Table 1: Illustrative native next-token distributions and their byte-prefix tree at the start of `playing`. T and S denote the teacher and student. The dashed END branch denotes token-ending mass; solid branches are labeled by the next byte.

**(a) Token probabilities**

| Token | T | S |
|---|---|---|
| `do` | 0.15 | 0.10 |
| `please` | 0.10 | 0.15 |
| `plane` | 0.15 | 0.15 |
| `play` | 0.30 | – |
| `playing` | – | 0.40 |
| `player` | 0.18 | 0.12 |
| `playful` | 0.12 | 0.08 |
| Total | 1.00 | 1.00 |

**(b) Byte-prefix tree**




Both distributions now refer to the same three next bytes. Their forward KL is $0.058892$ nats before smoothing. Appendix E.2 expands the calculation at all seven byte positions, whose mean KL is $0.011512$ nats. The observed tokenizations select which branch to use; the probabilities come from all matching candidate tokens in the softmax.

**Implementation.** We represent each vocabulary on the GPU as a byte matrix $\mathbf{B} \in \mathbb{Z}^{|\mathcal{V}|\times L_{\max}}$ and a token-length vector $\boldsymbol{\ell}$. A prefix mask selects candidates beginning with $\beta$, and their lengths distinguish tokens that end at the prefix from those that continue. We read each continuing token's next byte from $\mathbf{B}$ and scatter-add its probability into the corresponding entry of $\mathbf{r}$. Summing the ending-token probabilities gives $e$. At a boundary, we also use the next observed state's byte distribution. Algorithm 2 gives the complete procedure.

### 4.3 Training Losses

**Off-Policy Distillation.** Let $\mathcal{G}(C)$ contain the byte positions in chunk $C$ that are valid for both models. At each such position $t$, Eq. (2) gives a teacher distribution $\hat{\mathbf{q}}_T^t$ and a student distribution $\hat{\mathbf{q}}_S^t$ over the same 256 bytes. We smooth both distributions as $\tilde{\hat{\mathbf{q}}} = (1-\varepsilon)\hat{\mathbf{q}} + \varepsilon\mathbf{u}$, where $\mathbf{u}$ is uniform over $\mathcal{B}$. BLDA is their average forward KL divergence:

$$\mathcal{L}_{\text{BLDA}} = \frac{1}{N_{\text{off}}} \sum_{C\in\mathcal{C}} \sum_{t\in\mathcal{G}(C)} D_{\text{KL}}\left(\tilde{\hat{\mathbf{q}}}_T^t \,\|\, \tilde{\hat{\mathbf{q}}}_S^t\right), \qquad N_{\text{off}} = \sum_{C\in\mathcal{C}} |\mathcal{G}(C)|. \tag{3}$$

Examples with $N_{\text{off}} = 0$ are skipped. The complete objective is $(1-\alpha)\mathcal{L}_{\text{SFT}} + \alpha\mathcal{L}_{\text{BLDA}}$, where $\mathcal{L}_{\text{SFT}}$ is the student's next-token negative log-likelihood on the target sequence under its own tokenization.

**On-Policy Distillation.** For each prompt $x$, we sample a response $y$ using the current student parameters $\bar{\theta}$. We retokenize the decoded response with both tokenizers and construct its aligned chunks $\mathcal{C}(y)$ as before. Following standard on-policy distillation [Gu et al., 2023, Agarwal et al., 2023], we minimize the reverse KL $D_{\text{KL}}(\tilde{\hat{\mathbf{q}}}_S^t \,\|\, \tilde{\hat{\mathbf{q}}}_T^t)$ over the resulting byte positions:

$$\mathcal{L}_{\text{OnPolicy}} = \mathbb{E}_{y\sim\mu_{\bar{\theta}}(\cdot|x)}\left[\frac{1}{Z(y)} \sum_{\substack{C\in\mathcal{C}(y)\\ t\in\mathcal{G}(C)}} D_{\text{KL}}\left(\tilde{\hat{\mathbf{q}}}_S^t \,\|\, \tilde{\hat{\mathbf{q}}}_T^t\right)\right], \qquad Z(y) = \sum_{C\in\mathcal{C}(y)} |\mathcal{G}(C)|. \tag{4}$$

Gradients flow through the smoothed student distributions $\tilde{\hat{\mathbf{q}}}_S^t$ but not through sampling or retokenization; rollouts with $Z(y) = 0$ receive zero weight. On-policy distillation trains the student on

contexts visited under its current generation distribution, using the same byte-level comparison as the off-policy loss.

# 5 Experiments

We evaluate ByteDistill on mathematical reasoning, dialog summarization, and Countdown. Reasoning uses fixed training data, while summarization tests distillation with limited student training data; both use forward KL. Countdown uses student-generated responses and reverse KL, with three teacher–student pairs spanning different levels of tokenizer overlap.

## 5.1 Off-Policy Distillation

### 5.1.1 Setup

**Mathematical Reasoning.** We use `Qwen3-8B` [Yang et al., 2025] as the teacher and `Gemma-2-2B` [Gemma Team, 2024] as the student. We train the teacher on 1M examples from `OpenMathInstruct-2` [Toshniwal et al., 2024] using SFT, then evaluate zero-shot Pass@1 accuracy on GSM8K [Cobbe et al., 2021] and MATH [Hendrycks et al., 2021]. Student distillation uses 320K training examples and the same training budget as ALM (see Hyperparameters below).

**Low-Resource Dialog Summarization.** We evaluate summarization on DialogSum [Chen et al., 2021]. DialogSum contains 12,460 training dialogues and 2,000 validation/test dialogues. We use the full training set for teacher SFT with `Qwen3-8B`, but restrict student distillation with `Gemma-2-2B-IT` to a random subset of 1,500 training examples.

**Hyperparameters.** For mathematical reasoning, all student methods use batch size 64 and 5,000 steps, matching the ALM budget; for DialogSum, we train for 3 epochs. All runs use AdamW [Loshchilov and Hutter, 2019] with learning rate $5 \times 10^{-6}$, weight decay $10^{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, cosine scheduling with warmup ratio 0.1, and bfloat16 precision. Each math run takes approximately 8 hours on 16 NVIDIA H20 GPUs (96 GB each); each DialogSum run $\approx 0.7$ hours. Unless otherwise stated, the training loss uses $\alpha = 0.8$ in the off-policy objective $(1 - \alpha)\mathcal{L}_{\text{SFT}} + \alpha\mathcal{L}_{\text{KD}}$, where $\mathcal{L}_{\text{KD}}$ is the method-specific distillation loss. We train with prompt length 1024, maximum response lengths of 1024 for math and 128 for DialogSum, byte-distribution smoothing $\varepsilon = 10^{-6}$, and greedy decoding at evaluation time.

**Baselines.** We compare SFT, DSKDv2 [Zhang et al., 2025], ALM [Minixhofer et al., 2025], and CTLS [Phan et al., 2026]. SFT uses target tokens without distributional distillation. DSKDv2 aligns representations and exact token spans; ALM and CTLS compare text likelihoods across tokenizers. ALM also provides the reference data and training budget for our math setting. We use the authors' DSKDv2 implementation with the recommended intersection thresholds. CTLS is reproduced from its paper and public repository using the reported default configuration, as its training code is not fully released.

Within each off-policy setting, all methods use the same teacher, student, training data, and optimization budget.

### 5.1.2 Math and Summarization Results

Tables 2 and 3 report zero-shot Pass@1 for math and ROUGE-L [Lin, 2004] for summarization, alongside gains over SFT and the strongest cross-tokenizer baseline.

On GSM8K and MATH (Table 2), ByteDistill achieves $49.27\%$ average Pass@1, exceeding SFT by $5.40$ points and CTLS, the strongest compared baseline, by $1.78$ points. Its gains over CTLS are $1.75$ points on GSM8K and $1.80$ points on MATH. The objectives differ in how much of the teacher's prediction they retain. DSKDv2's exact-token filtering limits supervision to matched spans, leaving fewer comparisons when token boundaries diverge. ALM compares binarized likelihoods; BLDA preserves the relative probabilities of possible next bytes. This gives the student access to the teacher's preferred continuations throughout a reasoning trace, even where the observed tokens differ. CTLS recovers useful supervision through cross-tokenizer likelihood scoring; ByteDistill also retains how probability is distributed among the possible next bytes at each shared text position.

Table 2: Zero-shot mathematical reasoning results. Pass@1 (%); gain columns show margin over SFT and over the best cross-tokenizer baseline (CTLS). Base teacher is pre-task-SFT and not used for distillation; Task-SFT teacher is the distillation target. All student entries report mean ± std. over three seeds; Avg. and gain columns are computed from mean values.

| | | Pass@1 (%) ↑ | | | Avg. gain ↑ | |
|---|---|---|---|---|---|---|
| **Model** | **Method** | **GSM8K** | **MATH** | **Avg.** | **vs SFT** | **vs CTLS** |
| *Teacher reference* | | | | | | |
| Qwen3-8B | Base teacher (pre-task-SFT) | 93.40 | 70.56 | 81.98 | – | – |
| | Task-SFT teacher (distillation target) | 93.86 | 71.00 | 82.43 | – | – |
| *Student methods* | | | | | | |
| Gemma-2-2B-Base | SFT | 59.06 $\pm 0.58$ | 28.68 $\pm 0.32$ | 43.87 | 0.00 | -3.62 |
| | DSKDv2 | 43.59 $\pm 0.71$ | 19.36 $\pm 0.46$ | 31.48 | -12.39 | -16.01 |
| | ALM | 57.77 $\pm 0.63$ | 28.54 $\pm 0.39$ | 43.16 | -0.71 | -4.33 |
| | CTLS | 63.38 $\pm 0.44$ | 31.60 $\pm 0.35$ | 47.49 | +3.62 | 0.00 |
| | **ByteDistill (Ours)** | **65.13** $\pm 0.52$ | **33.40** $\pm 0.41$ | **49.27** | **+5.40** | **+1.78** |

Table 3: Low-resource DialogSum results (1,500 student examples). ROUGE-L; gain columns show margin over SFT and over the best cross-tokenizer baseline (ALM). Teacher rows follow the same convention as Table 2. All student entries report mean ± std. over three seeds; gain columns are computed from mean values.

| | | ROUGE-L ↑ | Gain ↑ | |
|---|---|---|---|---|
| **Model** | **Method** | | **vs SFT** | **vs ALM** |
| *Teacher reference* | | | | |
| Qwen3-8B | Base teacher (pre-task-SFT) | 21.55 | – | – |
| | Task-SFT teacher (distillation target) | 38.25 | – | – |
| *Student methods* | | | | |
| Gemma-2-2B-IT | SFT | 33.44 $\pm 0.42$ | 0.00 | -4.30 |
| | DSKDv2 | 23.62 $\pm 0.79$ | -9.82 | -14.12 |
| | ALM | 37.74 $\pm 0.31$ | +4.30 | 0.00 |
| | CTLS | 36.49 $\pm 0.46$ | +3.05 | -1.25 |
| | **ByteDistill (Ours)** | **38.65** $\pm 0.34$ | **+5.21** | **+0.91** |

On DialogSum (Table 3), ByteDistill reaches $38.65$ ROUGE-L with 1,500 student training examples. This exceeds ALM by $0.91$, CTLS by $2.16$, and SFT by $5.21$. Each reference summary supplies one wording for a dialog, whereas the teacher's distribution also assigns probability to alternative continuations. BLDA groups these continuations by their next bytes, allowing the student to learn their relative probabilities even where its token boundaries differ from the teacher's. The student can thus learn from more than the single reference wording in its 1,500 labeled examples.

**CoT length analysis.** We group test problems by the length of the teacher's chain of thought (CoT) and measure how much of its accuracy the student retains. We call the student's accuracy divided by the teacher's accuracy *teacher-knowledge retention*. ByteDistill ranks highest in every group on both datasets. In the longest GSM8K group ($[140, \infty)$ words), retention is $39.0\%$ for ByteDistill, $35.0\%$ for CTLS, and $29.2\%$ for SFT. Appendix B gives the full results. Preserving supervision across mismatched token spans lets the student learn teacher preferences throughout an extended reasoning trace, helping explain the larger margin on long GSM8K traces.

### 5.1.3 Additional Students and Chinese Reasoning

**Student size and initialization.** We keep the task-SFT Qwen3-8B teacher and evaluate MiniCPM4-0.5B [MiniCPM Team, 2025], a smaller student from another model family, and Gemma-2-2B-IT, which starts from instruction-tuned weights. All five methods use the same 320K examples, training hyperparameters, and zero-shot greedy evaluation as the main math setting.

ByteDistill leads on both benchmarks for both students (Table 4). Its averages of $43.11\%$ for MiniCPM4-0.5B and $56.60\%$ for Gemma-2-2B-IT exceed CTLS by $1.02$ and $1.90$ points, respec-

Table 4: Mathematical reasoning with smaller and instruction-tuned students: Pass@1 (%). Avg. is the mean of GSM8K and MATH. The shared task-SFT teacher scores 93.86/71.00 (Avg. 82.43). Initial denotes the student before task training.

| | MiniCPM4-0.5B | | | Gemma-2-2B-IT | | |
|---|---|---|---|---|---|---|
| Method | GSM8K | MATH | Avg. | GSM8K | MATH | Avg. |
| Initial | 50.19 | 24.80 | 37.50 | 62.47 | 3.90 | 33.19 |
| SFT | 53.60 | 29.06 | 41.33 | 65.73 | 33.16 | 49.45 |
| DSKDv2 | 39.80 | 19.10 | 29.45 | 37.23 | 11.12 | 24.18 |
| ALM | 53.75 | 29.66 | 41.71 | 68.92 | 35.66 | 52.29 |
| CTLS | 54.21 | 29.96 | 42.09 | 72.18 | 37.22 | 54.70 |
| **ByteDistill** | **55.49** | **30.72** | **43.11** | **73.46** | **39.74** | **56.60** |

tively. The gains extend the main result to a smaller model family and to an instruction-tuned student.

**Chinese mathematical reasoning.** HighMATH [Liu et al., 2025] contains Chinese high-school mathematics across eight subjects. We use the 4,399 publicly released problems, stratified by problem type (`type`) and difficulty level (`level`) with random seed 42 into 3,399 training and 1,000 held-out test problems. The two sets are disjoint at the problem level. Qwen3-8B generates the training responses. Every student method starts from Gemma-2-2B-IT and uses the same response set and matched training settings.

Table 5: Accuracy on the 1,000 held-out Chinese HighMATH problems (%). We use the split stratified by problem type and difficulty described above.

| Model / method | Accuracy |
|---|---|
| Teacher (Qwen3-8B) | 56.5 |
| Initial student (Gemma-2-2B-IT) | 4.9 |
| SFT | 12.6 |
| DSKDv2 | 16.1 |
| ALM | 16.3 |
| CTLS | 15.7 |
| **ByteDistill** | **17.4** |

On this Chinese test set, ByteDistill reaches $17.4\%$ accuracy, exceeding ALM by $1.1$ points and SFT by $4.8$ points (Table 5). The strongest baseline differs across the evaluations: CTLS leads the other baselines on GSM8K and MATH, while ALM leads on DialogSum and HighMATH. ByteDistill exceeds the corresponding strongest baseline in each setting.

### 5.2 On-Policy Distillation

#### 5.2.1 Setup

**Task and Dataset.** We evaluate on-policy distillation using GOLD's setup [Patiño et al., 2025] on `HuggingFaceTB/Countdown-Task-GOLD`. In Countdown, the model must combine the given numbers using the four elementary arithmetic operations to reach a target value, placing its final answer inside `<answer>...</answer>` tags. We use the official 30.4k-example training split and 10k-example test split, keeping GOLD's prompts and evaluator unchanged.

**Models and Tokenizer Overlap.** We use `Qwen3-4B-Instruct-2507` [Yang et al., 2025] as the teacher; it obtains $68.56\%$ test accuracy with the same evaluator. Following GOLD, we evaluate three students: `Qwen2.5-1.5B-Instruct` [Qwen et al., 2024], `Llama-3.2-1B-Instruct` [Grattafiori et al., 2024], and `Gemma-3-1B-IT` [Gemma Team, 2025]. These students cover three levels of tokenizer overlap with the teacher: near-equivalent tokenization for Qwen2.5, moderate mismatch for LLaMA, and severe mismatch for Gemma.

We quantify overlap with the string-level Jaccard index used by GOLD. For each tokenizer, we construct the set of decoded token strings and compute its intersection-over-union with the teacher's

Table 6: Test accuracy (%) on `Countdown-Task-GOLD`. Tokenizer IoU denotes string-level Jaccard overlap with the teacher tokenizer. Each online method starts from the same SFT checkpoint. ΔSFT is ByteDistill minus SFT Init.; ΔOnline is ByteDistill minus the stronger of DSKDv2 and GOLD for the same student. Reporting conventions follow Table 2.

| Model | Tok. IoU | Base | SFT Init. | +DSKDv2 | +GOLD | +ByteDistill | ΔSFT | ΔOnline |
|---|---|---|---|---|---|---|---|---|
| *Teacher reference: Qwen3-4B-Instruct-2507, accuracy 68.56%.* | | | | | | | | |
| Qwen2.5-1.5B-Instruct | 0.999 | 4.98 | 67.42 | 65.62 $\pm 0.47$ | 67.61 $\pm 0.31$ | **68.23** $\pm 0.26$ | **+0.81** | **+0.62** |
| Llama-3.2-1B-Instruct | 0.640 | 4.12 | 63.03 | 62.22 $\pm 0.38$ | 62.38 $\pm 0.52$ | **64.12** $\pm 0.44$ | **+1.09** | **+1.74** |
| Gemma-3-1B-IT | 0.063 | 3.18 | 59.99 | 60.27 $\pm 0.29$ | 59.47 $\pm 0.61$ | **60.59** $\pm 0.36$ | **+0.60** | **+0.32** |
| **Mean over students** | – | 4.09 | 63.48 | 62.70 | 63.15 | **64.31** | **+0.83** | **+0.89** |

decoded-token set. The resulting overlaps are $0.999$ for `Qwen2.5-1.5B-Instruct`, $0.640$ for `Llama-3.2-1B-Instruct`, and $0.063$ for `Gemma-3-1B-IT`. The near-unity Qwen overlap comes from the tokenizer design: `Qwen3` extends `Qwen2.5` with four additional control tokens, `<think>`, `<tool_response>`, `</tool_response>`, and `</think>`.

**Training Setup.** We use GOLD's dataset and model suite with our reverse-KL byte-level objective. Following Li et al. [2026], we first initialize each student by SFT on the same 30.4k examples, then start all online distillation methods from that student's checkpoint. Unless otherwise stated, the on-policy stage runs for one epoch with warmup ratio $0.1$, learning rate $1 \times 10^{-7}$, temperature $1$, maximum prompt/response lengths $512/2048$, and 16 NVIDIA H20 GPUs (96 GB each); each run takes $\approx 3$ hours. DSKDv2, GOLD, and ByteDistill therefore receive the same SFT initialization for each student.

### 5.2.2 Countdown Results

All three students improve over their SFT initialization after ByteDistill training (Table 6): by $0.81$ points for Qwen2.5, $1.09$ for Llama, and $0.60$ for Gemma. Mean accuracy across students reaches $64.31\%$, compared with $63.48\%$ for SFT Init., $63.15\%$ for GOLD, and $62.70\%$ for DSKDv2. Each student's online methods begin at the same SFT checkpoint, so the comparison captures what they gain from subsequent training on student-generated responses.

ByteDistill exceeds the stronger online baseline for each student by $0.62$, $1.74$, and $0.32$ points, respectively, with the largest gain for Llama at tokenizer IoU $0.640$. When tokenizers differ, BLDA can compare positions that exact-token alignment would omit. Even for the nearly matched Qwen pair, it changes the learning target from next-token identity to next-byte continuation, drawing supervision from positions inside tokens as well as at their boundaries. This finer-grained target helps explain why Qwen also benefits, despite having little tokenizer mismatch.

## 6 Conclusion

ByteDistill lets teacher and student models keep their own tokenizers while comparing their predictions over the same byte alphabet. Overlapping token spans define shared text chunks; within these chunks, we aggregate token probabilities by their next byte and account for token endings along the observed tokenization. The resulting BLDA loss improves student performance on mathematical reasoning, summarization, and Countdown training with student-generated responses.

## 7 Limitations

BLDA aligns textual byte probabilities; it does not align control tokens such as chat-template markers. It follows the observed tokenization rather than summing over all compatible token histories. The current byte-prefix aggregation has not been optimized with caching or fused kernels, and the throughput–fidelity trade-off has not been measured. Beyond the Chinese HighMATH evaluation, broader multilingual tasks, including Santhali and Shan with heavier byte fallback, code generation, and longer on-policy training runs remain to be studied.

## Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their constructive feedback. This work was supported solely by Taobao & Tmall Group of Alibaba, with which all authors are affiliated.

## 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. *arXiv preprint arXiv:2306.13649*, 2023.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: the universal logit distillation loss for llms. *Transactions on Machine Learning Research*, 2025.

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*, pages 8005–8018, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.419. URL `https://aclanthology.org/2025.findings-acl.419/`.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.449.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 2022. arXiv:2103.06874.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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:2412.14528v2*, 2024.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

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

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

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

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-2012.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016v2*, 2026.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Yan Liu, Minghui Zhang, Bojian Xiong, Yifan Xiao, Yinong Sun, Yating Mei, Longyu Zeng, Jingchao Yang, Yang Wang, and Deyi Xiong. HighMATH: Evaluating math reasoning of large language models in breadth and depth. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10241–10253. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.542. URL https://aclanthology.org/2025.findings-emnlp.542/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. arXiv:1711.05101.

MiniCPM Team. MiniCPM4: Ultra-efficient LLMs on end devices, 2025. URL https://arxiv.org/abs/2506.07900.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. *arXiv preprint arXiv:2503.20083v4*, 2025.

Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Unlocking on-policy distillation for any model family, 2025.

Buu Phan, Ashish Khisti, and Karen Ullrich. Cross-tokenizer likelihood scoring algorithms for language model distillation. *Proceedings of ICLR*, 2026. arXiv:2512.14954.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1162.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*, 2022. arXiv:2106.12672.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

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. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi: 10.1162/tacl_a_00461.

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

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

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. doi: 10.48550/arXiv.2504.11426. URL `https://arxiv.org/abs/2504.11426`.

# A Pseudocode

Algorithm 1 constructs aligned byte chunks by merging overlapping token intervals. Algorithm 2 computes the vectorized observed-path next-byte distribution used by BLDA. For efficient GPU computation of Eq. (2), we flatten the vocabulary into a byte matrix $\mathbf{B} \in \mathbb{Z}^{|\mathcal{V}| \times L_{\max}}$ with length vector $\boldsymbol{\ell}$ and a textual-token mask. Vectorized prefix masks and scatter-add operations then aggregate continuation and boundary mass, as detailed in Algorithm 2.

**Algorithm 1:** Building Aligned Byte Chunks

**Input:** Teacher textual tokens $\mathbf{t} = [t_1, \ldots, t_M]$ with byte intervals $\mathcal{I}_T$; Student textual tokens $\mathbf{s} = [s_1, \ldots, s_N]$ with byte intervals $\mathcal{I}_S$
**Output:** Set of chunks $\mathcal{C} = \{C_1, \ldots, C_K\}$

`// Stage 1: Interval Mapping`
**foreach** *token* $v$ *in* $\mathbf{t}$ *(resp.* $\mathbf{s}$*)* **do**
  Decode $v$ to bytes; record interval $[\text{start}_v, \text{end}_v)$;
**end**
`// Stage 2: Intersection Search via Dual Pointers`
Initialize UF $\leftarrow$ UnionFind$(M + N)$;
$i \leftarrow 1,\ j \leftarrow 1$;
**while** $i \leq M$ ***and*** $j \leq N$ **do**
  **if** $start_{t_i} < end_{s_j}$ ***and*** $start_{s_j} < end_{t_i}$ **then**
    UF.Union$(t_i,\ s_j)$ ; `// Link overlapping tokens`
  **end**
  **if** $end_{t_i} \leq end_{s_j}$ **then**
    $i \leftarrow i + 1$;
  **else**
    $j \leftarrow j + 1$;
  **end**
**end**
`// Stage 3: Input-validity Checks and Chunk Construction`
`// Complete partitions of the same response pass both checks`
$\mathcal{C} \leftarrow \emptyset$;
**foreach** *component* $G$ *in UF.Components*() **do**
  **if** $G \cap \mathbf{t} = \emptyset$ ***or*** $G \cap \mathbf{s} = \emptyset$ **then**
    **continue**;
  **end**
  $[b^T_{\min}, b^T_{\max}) \leftarrow [\min_{v \in G \cap \mathbf{t}} \text{start}_v,\ \max_{v \in G \cap \mathbf{t}} \text{end}_v)$;
  $[b^S_{\min}, b^S_{\max}) \leftarrow [\min_{v \in G \cap \mathbf{s}} \text{start}_v,\ \max_{v \in G \cap \mathbf{s}} \text{end}_v)$;
  **if** $[b^T_{\min}, b^T_{\max}) \neq [b^S_{\min}, b^S_{\max})$ **then**
    **continue**;
  **end**
  Add chunk $(G \cap \mathbf{t},\ G \cap \mathbf{s},\ [b^T_{\min}, b^T_{\max}))$ to $\mathcal{C}$;
**end**
**return** $\mathcal{C}$;

**Algorithm 2:** Compute Observed-Path Next-Byte Distribution. Byte columns of $\mathbf{B}$ are 0-indexed. The prefix length $g$ ranges in $\{0, \ldots, L_{\max}\}$; column $g$ is read only when $c > 0$, which implies that at least one continuing token has length $> g$ and hence $g < L_{\max}$. When $g = L_{\max}$ the boundary branch can still be valid via $\hat{\mathbf{q}} = \mathbf{q}_{\text{next}}$ (i.e. $c = 0, e > 0$).

**Input:** Softmax probabilities $\mathbf{p} \in \Delta_{|\mathcal{V}|}$; byte prefix $\beta$ of length $g$; flattened byte matrix $\mathbf{B} \in \mathbb{Z}^{|\mathcal{V}| \times L_{\max}}$; byte lengths $\boldsymbol{\ell} \in \mathbb{Z}^{|\mathcal{V}|}$; textual-token mask $\mathbf{m}_{\text{txt}} \in \{0,1\}^{|\mathcal{V}|}$; mode $\in \{\text{INSIDE}, \text{BOUNDARY}\}$ indicating whether the current state is an interior or observed-boundary offset; next-state byte distribution $\mathbf{q}_{\text{next}} \in \Delta_{256}$ at observed boundaries, or NONE otherwise

**Output:** Observed-path byte distribution $\hat{\mathbf{q}} \in \Delta_{256}$, or INVALID

**if** $g > 0$ **then**
  $\mathbf{M}_{\text{pfx}} \leftarrow \mathbf{m}_{\text{txt}} \wedge \bigwedge_{r=0}^{g-1} (\mathbf{B}_{:,r} = \beta_r)$ ; `// Textual prefix match`
**else**
  $\mathbf{M}_{\text{pfx}} \leftarrow \mathbf{m}_{\text{txt}}$;
**end**
$\mathbf{M}_{\text{cont}} \leftarrow \mathbf{M}_{\text{pfx}} \wedge (\boldsymbol{\ell} > g)$ ; `// Tokens that continue with a byte`
$\mathbf{M}_{\text{end}} \leftarrow \mathbf{M}_{\text{pfx}} \wedge (\boldsymbol{\ell} = g)$ ; `// Tokens that end at` $\beta$
$c \leftarrow \sum_i \mathbf{p}_i \mathbf{M}_{\text{cont},i}$ ; `// Continuation mass` $(\mathbf{1}^\top \mathbf{r})$
$e \leftarrow \sum_i \mathbf{p}_i \mathbf{M}_{\text{end},i}$ ; `// Token-ending mass`
$\boldsymbol{m} \leftarrow \mathbf{0}^{256}$;
**if** $c > 0$ **then**
  $\mathbf{c}_{\text{byte}} \leftarrow \mathbf{B}_{\mathbf{M}_{\text{cont}}, g}$ ; `// Next-byte values for continuing tokens`
  $\boldsymbol{m} \leftarrow \texttt{scatter_add}(\boldsymbol{m}, \mathbf{c}_{\text{byte}}, \mathbf{p}_{\mathbf{M}_{\text{cont}}})$ ; `// Aggregate continuation mass into r`
**end**
**if** *mode* $=$ INSIDE **then**
  `// Interior offset: token known to continue, condition on continuation`
  **if** $c \leq 0$ **then**
    **return** INVALID;
  **end**
  **return** $\boldsymbol{m}/c$;
**end**
`// Boundary offset: fold token-ending mass into next observed state`
$a \leftarrow c + e$;
**if** $a \leq 0$ **then**
  **return** INVALID;
**end**
**if** $e > 0$ ***and*** $\mathbf{q}_{\text{next}} =$ NONE **then**
  **return** INVALID ; `// Token-ending mass but no next state`
**end**
$\hat{\mathbf{q}} \leftarrow \boldsymbol{m}/a$;
**if** $e > 0$ **then**
  $\hat{\mathbf{q}} \leftarrow \hat{\mathbf{q}} + (e/a)\, \mathbf{q}_{\text{next}}$ ; `// Sums to 1 by construction`
**end**
**return** $\hat{\mathbf{q}}$;

## B CoT Length Retention Analysis

We group test problems by the word count of the `Qwen3-8B` teacher's chain of thought. The intervals are $[0, 80)$, $[80, 140)$, and $[140, \infty)$ for GSM8K, and $[0, 100)$, $[100, 200)$, and $[200, \infty)$ for MATH. Every method is evaluated on the same problems within each interval. We report *teacher-knowledge retention*, defined as $\text{Acc}_{\text{student}}/\text{Acc}_{\text{teacher}}$ within that interval and expressed as a percentage.

Retention decreases with teacher CoT length for all methods on both datasets (Figure 3). ByteDistill has the highest retention in every interval. Its gains over CTLS are $1.6$, $1.6$, and $4.0$ points in the

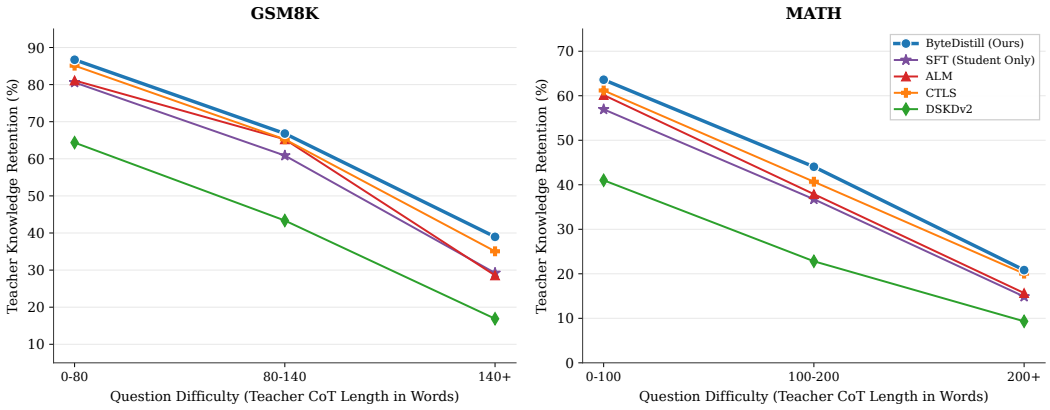



Figure 3: Teacher knowledge retention $\mathrm{Acc}_{\mathrm{student}}/\mathrm{Acc}_{\mathrm{teacher}}$ after grouping problems by teacher CoT length. Each interval contains the same problems for every method.

three GSM8K intervals, and 2.4, 3.4, and 0.8 points in the three MATH intervals. Relative to SFT, the GSM8K gains are 6.1, 6.0, and 9.8 points.

On GSM8K, the largest gain occurs in the longest interval: ByteDistill retains 39.0% of teacher accuracy, compared with 35.0% for CTLS, 28.6% for ALM, and 29.2% for SFT. On MATH, the largest gain over CTLS occurs in the middle interval (3.4 points), followed by the shortest (2.4) and longest (0.8).

The corresponding per-bin problem counts are shown in Figure 4.

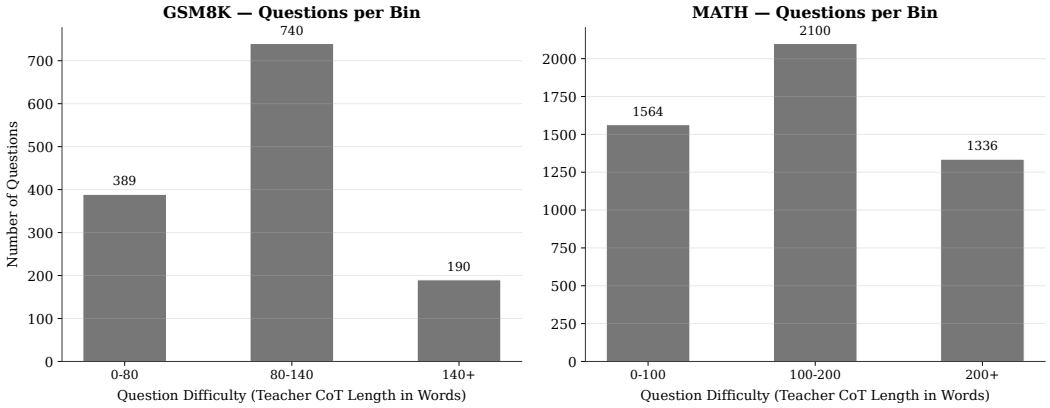



Figure 4: Problem counts for the teacher CoT length intervals on GSM8K and MATH. Since bins are assigned once from teacher outputs, these counts apply to all methods.

## C Full Boundary-Marginalization Closure

The observed-path closure in Eq. (2) follows the unique tokenization path recorded in the trajectory when marginalizing token boundaries. The ideal observable next-byte distribution $\bar{q}_{\theta,\star}^{\mathcal{V}}(b \mid h, \beta)$ instead marginalizes over every textual token $v$ whose byte realization equals $\beta$, allowing the recursion

to leave the observed path. Writing $z_\theta = \mathbf{1}^\top \mathbf{r}_\theta + e_\theta$ for the total prefix mass at $(h, \beta)$,

$$\bar{q}^{\mathcal{V}}_{\theta,\star}(b \mid h, \beta) = \begin{cases} \dfrac{\mathbf{r}^{\mathcal{V}}_\theta(h,\beta)_b}{\mathbf{1}^\top \mathbf{r}^{\mathcal{V}}_\theta(h,\beta)}, & \text{interior offset,} \\ \dfrac{\mathbf{r}^{\mathcal{V}}_\theta(h,\beta)_b + \displaystyle\sum_{v \in \mathcal{V}_{\mathrm{txt}}:\, \mathrm{bytes}(v)=\beta} p_\theta(v \mid h)\, \bar{q}^{\mathcal{V}}_{\theta,\star}(b \mid h \oplus v, \epsilon)}{z^{\mathcal{V}}_\theta(h,\beta)}, & \text{boundary offset.} \end{cases} \tag{5}$$

The interior branch conditions on continuation, as in the observed-path construction. At a boundary, rather than advancing along a single observed token $e_k$, the recursion considers every $v$ with $\mathrm{bytes}(v) = \beta$. It weights each candidate by its probability $p_\theta(v \mid h)$ and proceeds to the next state $(h \oplus v, \epsilon)$ that it induces. When $\beta$ is not a token boundary on any continuation, the inner sum is empty and the recursion terminates immediately.

Evaluating the boundary sum in Eq. (5) requires a next-state distribution for every compatible token $v$. BLDA uses the state reached by the observed token $e_k$. Under the uniqueness condition $\mathrm{bytes}(e_k) = \beta$, the sum has one term: its probability is $e^{\mathcal{V}}_\theta(h, \beta)$, and its next state is $(h \oplus e_k, \epsilon)$. Applying the interior branch at that state gives the boundary formula in Eq. (2), which is used in both training settings.

# D Validity and Edge Cases for $\mathcal{G}(C)$

Section 4.2 defines when Algorithm 2 returns a valid distribution. The edge cases below determine whether an offset $t$ enters $\mathcal{G}(C)$. We check them independently at the teacher and student native states $(h_a(t), \beta_a(t))$.

**Interior offsets.** An interior offset is admissible iff the continuation mass is strictly positive on both sides, i.e. $\mathbf{1}^\top \mathbf{r}^{\mathcal{V}}_\theta(h, \beta) > 0$ (equivalently, $c > 0$ in the notation of Algorithm 2). At a strict observed-interior offset, the observed token itself continues past $\beta$, so $c > 0$ holds by construction. The admissibility check guards against numerical edge cases where the model assigns negligible mass to the observed token (e.g. due to underflow), causing $c \approx 0$ despite the observed token continuing past $\beta$; such offsets are dropped from $\mathcal{G}(C)$.

**Boundary offsets.** At a boundary offset, the next observed state must also supply a valid byte distribution whenever the ending mass is nonzero. If $e^{\mathcal{V}}_\theta(h, \beta) > 0$ but the next state $(h \oplus e_k, \epsilon)$ does not yield a valid byte distribution (for example, because the next observed token is structural or that side has no admissible $e_k$), the boundary fold in Eq. (2) is undefined and the offset is excluded. The offset also requires $\mathbf{1}^\top \mathbf{r}^{\mathcal{V}}_\theta(h, \beta) + e^{\mathcal{V}}_\theta(h, \beta) > 0$ (equivalently, $a = c + e > 0$ in Algorithm 2) so that the denominator is well defined.

**Structural tokens.** Offsets at which the next observed token on either side belongs to $\mathcal{V}_{\mathrm{spc}}$ are excluded from $\mathcal{G}(C)$ entirely. Structural tokens have no canonical tokenizer-invariant byte realization; in off-policy training they are supervised by $\mathcal{L}_{\mathrm{SFT}}$, while in on-policy training their formatting behavior is inherited from the SFT initialization and the prompt template. The same rule applies to chat-template markers and other formatting symbols.

**Duplicate-byte realizations.** The boundary case of Eq. (2) requires the observed token $e_k$ to be the unique textual token in $\mathcal{V}_{\mathrm{txt}}$ with $\mathrm{bytes}(e_k) = \beta$. The evaluated tokenizer vocabularies satisfy this after canonicalizing special and added tokens; rare duplicate textual realizations are canonicalized to a single representative or excluded from $\mathcal{G}(C)$.

We apply these rules separately to each side. An offset enters $\mathcal{G}(C)$ only when both teacher and student pass all checks at their own native states. Every term in Eqs. (3) and (4) therefore compares two well-defined elements of $\Delta_{256}$.

# E Worked Examples of Byte-Level Alignment

## E.1 Interleaved Token Boundaries

The word `Simplifying`, taken from an OpenMathInstruct-2 response, provides a many-to-many alignment example. Its UTF-8 bytes are `53 69 6D 70 6C 69 66 79 69 6E 67`. Qwen3-8B tokenizes it as `[S]` `[impl]` `[ifying]`, whereas Gemma-2-2B uses `[Simp]` `[lifying]`.

The teacher's internal boundaries are $\{1, 5\}$ and the student's are $\{4\}$; there is no shared internal boundary. The overlapping intervals therefore form one connected component, giving the shared chunk $[0, 11)$.

| Byte interval | $[0, 1)$ | $[1, 4)$ | $[4, 5)$ | $[5, 11)$ |
|---|---|---|---|---|
| Decoded text | `S` | `imp` | `l` | `ifying` |
| Teacher tokens | `[S]` | `[impl]` | | `[ifying]` |
| Student tokens | `[Simp]` | | `[lifying]` | |

Let $H$ and $H'$ be the teacher and student token histories before this chunk. Table 7 lists every selected state, with I and B denoting INSIDE and BOUNDARY. At $t = 1$ and $t = 5$, only the teacher uses B; at $t = 4$, only the student uses B. BLDA maps the independently selected states onto the same byte alphabet at each offset.

Table 7: Complete state selection for `Simplifying`. Each offset selects one state on each side. $\epsilon$ is the empty byte prefix.

| $t$ | Byte | Teacher $(h, \beta)$ | Mode | Student $(h, \beta)$ | Mode |
|---|---|---|---|---|---|
| 0 | `S` | $(H, \epsilon)$ | I | $(H', \epsilon)$ | I |
| 1 | `i` | $(H, \texttt{S})$ | B | $(H', \texttt{S})$ | I |
| 2 | `m` | $(H \oplus \texttt{S}, \texttt{i})$ | I | $(H', \texttt{Si})$ | I |
| 3 | `p` | $(H \oplus \texttt{S}, \texttt{im})$ | I | $(H', \texttt{Sim})$ | I |
| 4 | `l` | $(H \oplus \texttt{S}, \texttt{imp})$ | I | $(H', \texttt{Simp})$ | B |
| 5 | `i` | $(H \oplus \texttt{S}, \texttt{impl})$ | B | $(H' \oplus \texttt{Simp}, \texttt{l})$ | I |
| 6 | `f` | $(H \oplus \texttt{S} \oplus \texttt{impl}, \texttt{i})$ | I | $(H' \oplus \texttt{Simp}, \texttt{li})$ | I |
| 7 | `y` | $(H \oplus \texttt{S} \oplus \texttt{impl}, \texttt{if})$ | I | $(H' \oplus \texttt{Simp}, \texttt{lif})$ | I |
| 8 | `i` | $(H \oplus \texttt{S} \oplus \texttt{impl}, \texttt{ify})$ | I | $(H' \oplus \texttt{Simp}, \texttt{lify})$ | I |
| 9 | `n` | $(H \oplus \texttt{S} \oplus \texttt{impl}, \texttt{ifyi})$ | I | $(H' \oplus \texttt{Simp}, \texttt{lifyi})$ | I |
| 10 | `g` | $(H \oplus \texttt{S} \oplus \texttt{impl}, \texttt{ifyin})$ | I | $(H' \oplus \texttt{Simp}, \texttt{lifyin})$ | I |

## E.2 A Complete Prefix-Tree Aggregation Example

We expand the `playing` example from Section 4.2. The prompt is *Convert "play" to its -ing form*; the response bytes are `70 6C 61 79 69 6E 67`. The observed teacher tokenization is `[play]` `[ing]` and the student tokenization is `[playing]`. Table 1 in the main text gives the illustrative native softmaxes and byte-prefix tree used below.

At $t = 0$, summing the initial `p` subtree gives $0.30 + 0.18 + 0.12 + 0.15 + 0.10 = 0.85$ for the teacher and $0.40 + 0.12 + 0.08 + 0.15 + 0.15 = 0.90$ for the student. The remaining masses, $0.15$ and $0.10$, go to `d`.

At $t = 2$, the prefix is `pl`; candidates outside that subtree no longer contribute. The teacher's `a` and `e` branches have masses $0.75$ and $0.10$, so their normalized probabilities are $0.75/0.85$ and $0.10/0.85$. The student's corresponding probabilities are $0.75/0.90$ and $0.15/0.90$.

At $t = 4$, the student remains inside `playing`. Its continuation masses for $(\texttt{i}, \texttt{e}, \texttt{f})$ are $(0.40, 0.12, 0.08)$, giving $\hat{q}_S = (2/3, 1/5, 2/15)$ after division by $0.60$.

The teacher reaches the observed boundary after `play`: its continuation masses are $r_T(\texttt{e}) = 0.18$ and $r_T(\texttt{f}) = 0.12$, and its token-ending probability mass is $e_T = 0.30$. Suppose its next native state predicts `ing` with probability one, so that $q_{\text{next}}(\texttt{i}) = 1$. The boundary branch gives

$$\hat{q}_T(b) = \frac{r_T(b) + 0.30\, q_{\text{next}}(b)}{0.18 + 0.12 + 0.30}, \qquad \hat{q}_T(\texttt{i}, \texttt{e}, \texttt{f}) = (1/2, 3/10, 1/5).$$

Table 8 completes the construction along all seven bytes. The empty prefix $\epsilon$ is neither EOS nor the smoothing coefficient $\varepsilon$.

Table 8: Next-byte distributions for the toy response `playing`, before uniform smoothing. Only nonzero entries are shown; probabilities are rounded to four decimals. After the teacher consumes `play`, its history advances for $t = 5, 6$.

| $t$ | Byte | Teacher: prefix, mode; $\hat{q}_T$ | Student: prefix, mode; $\hat{q}_S$ |
|---|---|---|---|
| 0 | `p` | $\epsilon$, I; {`p` : .85, `d` : .15} | $\epsilon$, I; {`p` : .90, `d` : .10} |
| 1 | `l` | `p`, I; {`l` : 1} | `p`, I; {`l` : 1} |
| 2 | `a` | `pl`, I; {`a` : .8824, `e` : .1176} | `pl`, I; {`a` : .8333, `e` : .1667} |
| 3 | `y` | `pla`, I; {`y` : .80, `n` : .20} | `pla`, I; {`y` : .80, `n` : .20} |
| 4 | `i` | `play`, B; {`i` : .50, `e` : .30, `f` : .20} | `play`, I; {`i` : .6667, `e` : .20, `f` : .1333} |
| 5 | `n` | `i`, I; {`n` : 1} | `playi`, I; {`n` : 1} |
| 6 | `g` | `in`, I; {`g` : 1} | `playin`, I; {`g` : 1} |

Using natural logarithms and the unrounded probabilities, the nonzero forward KL terms occur at $t = 0, 2, 4$ and equal $0.012235$, $0.009457$, and $0.058892$ nats, respectively. Their mean over the seven byte offsets is $0.011512$ nats. Training applies the uniform smoothing defined in Section 4.

# F Chunk Coverage Audit and Distribution Validity

**Input-validity checks.** Algorithm 1 skips a connected component if it contains textual tokens from only one side or if the two sides span different byte intervals. When both interval lists partition the same response bytes completely, neither condition can occur: every nonempty token interval overlaps at least one interval on the other side, and a component can end only at a boundary shared by the two partitions.

We audited 80 responses: 40 from the GSM8K test set and 40 reference responses from OpenMathInstruct-2. These responses contain 55,602 UTF-8 bytes. With the Qwen3-8B and Gemma-2 tokenizers, Algorithm 1 produced 20,411 connected components. No component was one-sided or had mismatched spans. All 20,411 components were retained, covering all 55,602 response bytes: component retention and byte coverage were both 100% on this sample.

Thus, neither input-validity check introduces approximation or loss of byte coverage at the chunk-construction stage in the audited responses.

## F.1 Normalization of the Observed-Path Distribution

**Proposition 1** (Well-definedness of BLDA)**.** *For any offset selected by $\mathcal{G}(C)$, the construction in Eq.* (2) *returns a probability distribution over the* 256 *UTF-8 bytes, i.e. $\hat{q}_\theta^{\mathcal{V}}(\cdot \mid h, \beta) \in \Delta_{256}$.*

*Proof.* In the INSIDE case, $\mathcal{G}(C)$ requires $\mathbf{1}^\top \mathbf{r} > 0$, so dividing the non-negative continuation vector $\mathbf{r}$ by its $\ell_1$ norm yields a valid distribution.

In the BOUNDARY case, let $c = \mathbf{1}^\top \mathbf{r}$ and $a = c + e$. $\mathcal{G}(C)$ admits the offset only when $a > 0$, and if $e > 0$, Algorithm 2 provides a valid $\mathbf{q}_{\text{next}} \in \Delta_{256}$. The numerator $\mathbf{r} + e\,\mathbf{q}_{\text{next}}$ is non-negative and sums to $c + e = a$, so $(\mathbf{r} + e\,\mathbf{q}_{\text{next}})/a \in \Delta_{256}$. This also covers the degenerate case $c = 0$, $e > 0$, where the output reduces to $\mathbf{q}_{\text{next}}$. □

# G Additional Evaluation Details

## G.1 Smaller and Instruction-Tuned Students

Section 5.1.3 reports the MiniCPM4-0.5B and Gemma-2-2B-IT results in Table 4. Both use the task-SFT Qwen3-8B teacher from Table 2. MiniCPM changes the student size and model family, while Gemma-2-2B-IT changes the initialization relative to the base student.

Each configuration uses the same 320K training examples, 5,000 steps, batch size 64, and learning rate $5 \times 10^{-6}$ with cosine scheduling and warmup ratio 0.1. Evaluation uses zero-shot greedy

decoding. We rerun SFT, DSKDv2, ALM, CTLS, and ByteDistill with these matched settings for each student.

### G.2 Chinese Mathematical Reasoning on HighMATH

HighMATH [Liu et al., 2025] evaluates Chinese high-school mathematics across eight subjects. The paper describes 5,293 problems. We use the version from the official public repository[1], which contains 4,399 records across eight subject files.

The public release does not provide an official training split. We therefore stratify all 4,399 problems by problem type (`type`) and difficulty level (`level`), using a fixed random seed of 42 to obtain 3,399 training problems and 1,000 held-out test problems. The two sets are disjoint at the problem level. We will include the data-preparation script in our open-source code release.

Qwen3-8B generates a response for each training problem. All student methods start from Gemma-2-2B-IT and train on this fixed response set under matched settings. The held-out 1,000 problems are used only for evaluation. The results are reported in Table 5.

[1] `https://github.com/tjunlp-lab/HighMATH`

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state three contributions (conceptual reformulation, the BLDA objective, and empirical evaluation), and each is supported in the paper: the formulation in Section 4, Proposition 1 for well-definedness, and the off-policy and on-policy results in Sections 5.1–5.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the intended scope of ByteDistill and practical considerations for applying the same byte-level interface in broader training settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The single theoretical result, Proposition 1 (well-definedness of BLDA), is stated with its assumptions and proved in Appendix F.1; the full boundary-marginalization closure derivation is given in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5 specifies the teacher/student model checkpoints, datasets, optimizer, learning rate, batch size, sequence length, and evaluation metrics. The appendix gives pseudocode for constructing aligned chunks and computing observed-path next-byte distributions, along with the full boundary-marginalization closure and validity rules for edge cases. Section 4 describes how these components enter the off-policy and on-policy losses. The supplemental code release provides a runnable implementation and configuration files for reproducing the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a code release with usage documentation as supplemental material. The repository includes the BLDA implementation, training scripts for both off-policy and on-policy distillation, and instructions to reproduce the main experimental results. All datasets (GSM8K, MATH, DialogSum, Countdown, HighMATH) and teacher/student model checkpoints are publicly available under their respective licenses. Appendix G.2 describes our HighMATH split; we will include the data-preparation script in our open-source code release.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and the appendix specify the train/eval splits, optimizer (AdamW), learning-rate schedule, batch size, sequence length, distillation temperature, KL weighting, and evaluation protocols (Pass@1 for math, ROUGE-L for DialogSum, success rate for Countdown).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For student distillation in Tables 2, 3, and 6, we report the mean $\pm$ standard deviation over three seeds. The standard deviations summarize variation across random seeds, including rollout sampling in on-policy training.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 reports GPU memory per device (96 GB), the number of GPUs per run (16), batch size, training steps, sequence lengths, and approximate wall-clock time per run for each experimental configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The work uses publicly released models and datasets under their original licenses and conducts no human-subjects research.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Positive impacts include improving the efficiency of transferring capabilities to smaller models, which may reduce deployment cost. Negative impacts are inherited from the underlying generative models: cheaper distillation could also lower the cost of transferring harmful capabilities. The paper does not release new models or datasets and evaluates only existing public benchmarks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release new pretrained models, datasets, or generative systems with high misuse potential; it studies a distillation objective trained on existing public corpora.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All teacher/student models (Qwen3, Qwen2.5, Gemma-2, Gemma-3, Llama-3.2, MiniCPM4) and datasets (GSM8K, MATH, DialogSum, Countdown, HighMATH) are cited in Section 5 and Appendix G; they are used under their respective publicly stated licenses (Apache-2.0, Gemma Terms of Use, Llama Community License, MIT).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplemental material includes the released code and a README describing environment setup, configuration details, and step-by-step examples for reproducing both off-policy and on-policy distillation experiments. The code is released under the Apache-2.0 license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects research, so no IRB review was required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs (Qwen, Gemma, Llama families) are the subject of study as teachers and students in distillation; no LLM was used as an original or non-standard methodological component beyond standard writing assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.