Learning from Disagreement: Maximum Divergence Knowledge Distillation

Aref Jafari[1,3,†] **Parsa Ashrafi Fashi**[2,*] **Mehdi Rezagholizadeh**[1,†]
Hanieh Asadi Golmankhaneh[3] **Shayan Salehi**[3] **Vikram Appia**[1]
Emad Barsoum[1] **Ali Ghodsi**[3,†]

[1]Advanced Micro Devices Inc. (AMD) [2]NVIDIA
[3]University of Waterloo

Abstract

Knowledge distillation pipelines typically train a student to match a teacher on a fixed corpus of teacher-labeled examples. This leaves a major source of supervision unused: the teacher is a fully queryable model whose output distribution extends far beyond any fixed dataset. We argue that effective distillation should exploit this structure by steering generation toward regions where the student most underestimates the teacher. We propose *Maximum Divergence Knowledge Distillation* (MDKD), a rejection-sampling method for autoregressive language models. At each decoding step, MDKD draws candidate tokens from the teacher and preferentially accepts those the student underweights, constructing trajectories that remain teacher-supported while concentrating training on regions where the student assigns insufficient probability. We show that stochastic MDKD's acceptance rule has an exact distributional characterization: it samples precisely from the teacher's *uncovered mass* $[T - S]_+$, the probability the teacher assigns to tokens the student does not yet cover, with stochastic acceptance probability equal to the teacher–student total variation distance. Standard teacher sampling, by contrast, spends most of its updates on the overlap $\min(T, S)$ already absorbed by the student. This makes MDKD the structural complement of speculative decoding, which exploits the same overlap to accelerate inference. On GSM8K, distilling `Qwen2.5-14B-Instruct` into a base `Qwen2.5-1.5B` student with only $1{,}000$ KD samples (5 epochs) lifts accuracy from 8.26% to 64.90%, outperforming SOTA methods by 10.3 percentage points and recovering 65% of the teacher–student gap. A divergence-gain analysis confirms the mechanism: MDKD sequences raise student cross-entropy by 43.6% while preserving 98% of teacher-sampling accuracy. Across arithmetic reasoning, dialogue summarization, and code generation, MDKD matches or outperforms strong distillation baselines. The code is released at `https://github.com/AMD-AGI/mdkd`.

1 Introduction

Knowledge distillation (KD) transfers capabilities from a large *teacher* model to a smaller *student* by aligning their output distributions [Hinton et al., 2015]. Many common KD pipelines train the student on a fixed corpus of teacher-labeled examples, treating all training inputs uniformly regardless of how well the student already matches the teacher on each one. This passive strategy ignores a critical signal: regions of the output space where the student most underestimates the teacher are precisely where additional training would be most informative.

[†]Corresponding authors: {aref.jafari, mehdi.rezagholizadeh}@amd.com, ali.ghodsi@uwaterloo.ca
[*]Work done while at AMD.

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

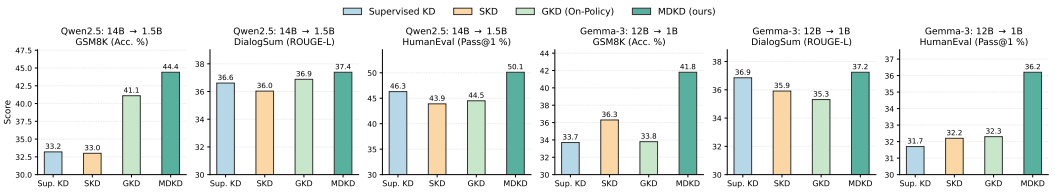


Figure 1: MDKD outperforms supervised KD, SKD, and on-policy KD (GKD) across all tested tasks and across different teacher–student pairs. We distill Qwen2.5-14B-Instruct into a 1.5B-Instruct student and Gemma-3-12B-it into a 1B-it student on three task families: arithmetic reasoning (GSM8K accuracy, %), dialogue summarization (DialogSum ROUGE-L on a 0–100 scale), and code generation (HumanEval Pass@1, %, training on MBPP). All methods share the same 1K-sample training set, identical hyperparameters, and 3 training epochs.

We argue that effective distillation should *actively* construct training data in these regions. We introduce a unified principle: *bias the training distribution toward outputs that the teacher supports but the student underweights.* This principle admits natural instantiations in both continuous and discrete domains. In continuous input spaces (e.g., images), gradient-based perturbation moves inputs in the direction of increasing KD loss, exposing under-trained regions [Jafari et al., 2023]. In discrete input spaces (e.g., token sequences), where gradients over inputs are undefined, a different mechanism is needed.

For autoregressive language models, the distribution-mismatch problem is compounded by *exposure bias* [Bengio et al., 2015]: the student trains on teacher-generated sequences but generates from its own distribution at inference time. Recent methods address this through on-policy generation [Agarwal et al., 2024] or interleaved student–teacher sampling [Xu et al., 2024]. However, none of these methods steer *decoding itself* toward tokens where the teacher–student gap is largest: GKD lets the student explore its own distribution, while SKD corrects the student's worst proposals. What is missing is a generation procedure that actively concentrates supervision on the student's underweighted regions, token by token.

We propose *Maximum Divergence Knowledge Distillation* (MDKD), which instantiates this principle in the discrete autoregressive setting through a simple rejection-sampling rule.[1] At each decoding step, MDKD draws m candidate tokens from the teacher's nucleus-filtered distribution and accepts each with probability $[1 - S(v)/T(v)]_+$; if no candidate is stochastically accepted, a deterministic fallback selects the most divergent candidate. We prove that this rule admits an exact distributional characterization (Theorem 1): the stochastic step samples precisely from the teacher's *uncovered mass* $[T - S]_+$, with acceptance probability equal to the teacher–student total variation distance. Standard teacher sampling, in contrast, spends most of its probability on the overlap $\min(T, S)$ that the student already covers, mass on which an additional training update is wasted. This decomposition exposes a clean duality: speculative decoding [Leviathan et al., 2023] exploits the overlap $\min(T, S)$ to skip teacher computation, while MDKD samples the residual $[T - S]_+$ to avoid wasting distillation updates on tokens the student already covers correctly. The two methods are dual decompositions of teacher sampling along the same partition $T = \min(T, S) + [T - S]_+$, with opposite acceptance polarities.

Contributions.

1. **A unified active-distillation principle** that connects gradient-based input perturbation in continuous domains with token-level rejection sampling in discrete domains, providing a coherent framework for actively constructing KD training data.
2. **MDKD, a simple rejection-sampling rule with an exact distributional characterization.** Theorem 1 shows that MDKD's stochastic step samples from $[T - S]_+$ with acceptance probability equal to the teacher–student total variation distance, revealing MDKD as the

[1] We use "rejection sampling" in the broad sense: at each step, finitely many candidates are drawn from the teacher's nucleus and accepted stochastically with probability α_j; a deterministic argmax fallback selects the most divergent candidate when no stochastic acceptance occurs within the finite candidate set. The fallback is consistent with the divergence-maximization principle and ensures the procedure is well-defined for any finite m.

structural complement of speculative decoding along the same overlap–residual partition. The method requires no auxiliary models, no heuristics, and no reinforcement learning.

3. **Comprehensive experiments** across three task families, arithmetic reasoning (GSM8K, GSM-Plus, UltraInteract-math), dialogue summarization (DialogSum), and code generation (MBPP, HumanEval), showing that MDKD matches or outperforms strong distillation baselines including supervised KD, on-policy KD (GKD), and speculative KD (SKD).
4. **Headline result.** With a `Qwen2.5-14B-Instruct` teacher and only $1{,}000$ KD samples (5 epochs), MDKD lifts a base `Qwen2.5-1.5B` student from 8.26% to 64.90% on GSM8K (Table 7), $+10.3$ pp ahead of the next-best baseline (on-policy KD at 54.59%) and recovering 65% of the teacher–student gap. A divergence-gain analysis (Appendix C.1) confirms the mechanism: MDKD sequences raise student cross-entropy by 43.6% while preserving 98% of teacher-sampling accuracy.

2 Related work

Classical and sequence-level KD. Hinton et al. [2015] introduced KD by training a student to match the teacher's softened output distribution. Kim and Rush [2016] extended this to sequence-level distillation, training the student on teacher-generated sequences with cross-entropy loss. Both approaches operate on a fixed teacher-labeled corpus and provide no mechanism to concentrate supervision where the student most underperforms.

On-policy distillation. GKD [Agarwal et al., 2024] frames distillation as imitation learning [Ross et al., 2011]: the student generates sequences from its own policy, and the teacher provides token-level supervision. This reduces the train–test distribution mismatch, but it does not explicitly target high-divergence regions: the student explores its own distribution, which may overlook tokens where the teacher–student gap is largest. MiniLLM [Gu et al., 2024] takes a related approach, optimizing reverse KL on student-generated sequences to mitigate mode-averaging.

Speculative KD. SKD [Xu et al., 2024] adapts speculative decoding [Leviathan et al., 2023]: the student proposes tokens and the teacher verifies them, replacing poorly ranked tokens with teacher samples. This produces sequences aligned with the student's distribution but corrected toward the teacher. Speculative decoding itself targets the teacher–student overlap $\min(T, S)$ to bypass teacher computation; SKD inherits this overlap-based acceptance rule and uses it to verify student-proposed sequences. MDKD instead targets the complementary residual $[T - S]_+$ via stochastic acceptance, the teacher mass the student does not yet cover, making it the learning-side structural dual of speculative decoding (Theorem 1). The two methods decompose teacher sampling along the same partition with opposite acceptance polarities: speculative decoding accepts on agreement to skip teacher work, while MDKD accepts on disagreement to extract maximum learning signal per query.

Active, hard-example, and adversarial distillation. A broader line of work seeks training data that exposes student weaknesses. Hard-example mining and curriculum learning [Bengio et al., 2009] re-weight existing examples by difficulty; data-free and synthetic-data KD [Micaelli and Storkey, 2019, Fang et al., 2019] generate inputs that maximize teacher–student disagreement, typically via a learned generator or adversarial optimization. Adversarial distillation either trains the student against a discriminator in a minimax game with the teacher [Wang et al., 2018], or perturbs inputs with adversarial attacks to find samples near the teacher's decision boundary and trains the student on them [Heo et al., 2018]. These methods all share MDKD's premise, that the student should be trained where it disagrees most with the teacher, but rely on auxiliary models, generator networks, or differentiable inputs. MDKD achieves the same goal through a closed-form, model-free rejection-sampling rule applied directly to the teacher's own distribution.

3 Method

3.1 Preliminaries

Let $\mathcal{V}$ denote the vocabulary. For a prefix $x_{<i} = (x_1, \ldots, x_{i-1})$, let $T(\cdot \mid x_{<i})$ and $S(\cdot \mid x_{<i}; \theta)$ denote the teacher's and student's next-token distributions, respectively. Classical KD trains the

student by minimizing

$$\mathcal{L}_{\mathrm{KD}}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D}}\Big[\sum_i \ell_D\big(T_i,\, S_i(\theta)\big)\Big], \tag{1}$$

where $T_i := T(\cdot \mid x_{<i})$, $S_i(\theta) := S(\cdot \mid x_{<i};\theta)$, $\mathcal{D}$ is a fixed dataset, and ℓ_D is a divergence measure between two distributions over $\mathcal{V}$. We instantiate ℓ_D with forward KL, reverse KL, or Jensen–Shannon, denoted

$$\ell_{\mathrm{FKL}}(T_i,S_i)=\mathrm{KL}(T_i\,\|\,S_i),\quad \ell_{\mathrm{RKL}}(T_i,S_i)=\mathrm{KL}(S_i\,\|\,T_i),\quad \ell_{\mathrm{JSD}}(T_i,S_i)=\mathrm{JSD}(T_i,S_i). \tag{2}$$

Equation (1) only aligns teacher and student on the observed dataset $\mathcal{D}$. MDKD instead constructs an alternative sequence distribution μ that concentrates probability on regions where T and S disagree, and trains the student on samples from μ in place of $\mathcal{D}$.

3.2 The divergence-maximization principle

The core insight behind MDKD is that training data should be concentrated where the student most underestimates the teacher. In continuous domains, this principle is realized by gradient-based input perturbation: given a sample x, one computes $\nabla_x \ell_D(T,S)$ and perturbs x in that direction to produce a harder training example [Jafari et al., 2023]. In discrete autoregressive settings, inputs are non-differentiable token sequences, so gradient-based perturbation does not apply. MDKD instead realizes the same principle through a rejection-sampling rule applied during sequence generation.

3.3 Divergence-aware token sampling

Procedure. Given a prompt $x_{1:t}$, MDKD generates a continuation one token at a time. At position $i > t$, the teacher's logits are scaled by temperature τ and restricted to a nucleus of cumulative probability p_{nuc}, yielding a filtered distribution $T_\tau(\cdot \mid x_{<i})$ supported on the nucleus. The procedure then draws m candidate tokens i.i.d. with replacement from T_τ and computes, for each candidate $\hat{x}_j$,

$$\alpha_j \;=\; \max\Big(0,\; 1-\tfrac{S_\tau(\hat{x}_j|x_{<i})}{T_\tau(\hat{x}_j|x_{<i})}\Big), \tag{3}$$

where S_τ denotes the student distribution renormalized over the nucleus support of T_τ, so that T_τ and S_τ are both probability distributions on the same set. For each candidate in turn, an independent $r_j \sim \mathrm{Uniform}(0,1)$ is drawn, and the first candidate satisfying $r_j < \alpha_j$ is accepted. If no candidate is stochastically accepted, the candidate with the largest α_j is selected as a deterministic fallback. Generation terminates when an end-of-sequence token is selected or the maximum length n is reached. Algorithm 1 summarizes the procedure.

Interpretation. The acceptance score α_j is large when the student assigns much less probability than the teacher ($S_\tau \ll T_\tau$), biasing generation toward tokens the student underweights. When the two distributions agree ($S_\tau \approx T_\tau$), $\alpha_j \approx 0$ and the candidate is almost certainly rejected. The deterministic fallback ensures that, even when no candidate passes the stochastic test, the selected token is the one where the student most underestimates the teacher among the m candidates, consistent with the divergence-maximization principle. By construction, every accepted token lies in the teacher's nucleus, so the resulting sequences are *teacher-supported* at every position, while the acceptance rule concentrates training signal on the student's underweighted regions.

3.4 Theoretical analysis

The procedure of Section 3.3 admits an exact distributional characterization. Throughout this subsection, T and S refer to T_τ and S_τ as defined above, treated as two probability distributions over their common nucleus support $\mathcal{V}_0 \subseteq \mathcal{V}$ with $T(v) > 0$ for all $v \in \mathcal{V}_0$. We first characterize the stochastic acceptance step in isolation (Theorem 1), then characterize the full selected-token distribution under the finite-candidate algorithm with deterministic fallback (Corollary 2).

Theorem 1 (The stochastic MDKD step samples the teacher's uncovered mass)**.** *Fix a prefix $x_{<i}$ and let T and S be teacher and student next-token distributions over a common support $\mathcal{V}_0$ with $T(v) > 0$ for all $v \in \mathcal{V}_0$. Consider a single proposal $v \sim T$ accepted with probability $\alpha(v) = [1-S(v)/T(v)]_+$. Then:*

Algorithm 1 MDKD generation: divergence-aware response generation

Require: Prompt $x_{1:t}$, teacher T, student snapshot S_{θ^-}, max length n, candidates m, temperature τ, nucleus p_{nuc}

$x \leftarrow x_{1:t}$
for $i = t+1$ **to** n **do**
  $T_\tau(\cdot \mid x) \leftarrow \text{NucleusFilter}\big(\text{softmax}(T_{\text{logits}}(x)/\tau),\ p_{\text{nuc}}\big)$
  $S_\tau(\cdot \mid x) \leftarrow \text{Renormalize}\big(S_{\theta^-}(\cdot \mid x),\ \text{supp}(T_\tau)\big)$ ▷ Renormalize over nucleus support
  Sample $\hat{x}_1, \ldots, \hat{x}_m \sim T_\tau(\cdot \mid x)$ ▷ i.i.d. with replacement
  accepted $\leftarrow$ **false**
  for $j = 1$ **to** m **do**
    $\alpha_j \leftarrow \max\big(0,\ 1 - S_\tau(\hat{x}_j \mid x) \,/\, T_\tau(\hat{x}_j \mid x)\big)$
    Draw $r_j \sim \text{Uniform}(0,1)$
    if $r_j < \alpha_j$ **then**
      $x \leftarrow x \circ \hat{x}_j$; *accepted* $\leftarrow$ **true**; **break**
    end if
  end for
  if not *accepted* **then**
    $j^\star \leftarrow \arg\max_{j \in \{1,\ldots,m\}} \alpha_j$
    $x \leftarrow x \circ \hat{x}_{j^\star}$ ▷ Deterministic fallback
  end if
  if the appended token is $\langle \text{EOS} \rangle$ **then break**
  end if
end for
return x

1. *The probability of stochastic acceptance equals the teacher–student total variation distance:*

$$\Pr[\text{accept}] \;=\; \sum_{v \in \mathcal{V}_0} [T(v) - S(v)]_+ \;=\; \text{TV}(T, S). \tag{4}$$

2. *Whenever* $\text{TV}(T,S) > 0$*, conditioned on acceptance the selected token is distributed as the normalized positive residual:*

$$\Pr[v \mid \text{accept}] \;=\; \frac{[T(v) - S(v)]_+}{\text{TV}(T,S)}. \tag{5}$$

3. *Whenever* $\text{TV}(T,S) < 1$*, conditioned on rejection the proposal is distributed as the normalized teacher–student overlap:*

$$\Pr[v \mid \text{reject}] \;=\; \frac{\min\{T(v), S(v)\}}{1 - \text{TV}(T,S)}. \tag{6}$$

The stochastic MDKD step thus exactly decomposes teacher sampling into an agreement component $\min\{T,S\}$ *(handled by rejection) and a disagreement component* $[T-S]_+$ *(handled by stochastic acceptance), training only on the teacher probability mass that is not already covered by the student.*

Proof sketch. The joint probability of proposal-and-acceptance equals $T(v)\alpha(v) = [T(v) - S(v)]_+$ for every v; summing and using $\sum_v [T(v) - S(v)]_+ = \text{TV}(T,S)$ yields claim 1, after which claims 2–3 follow by conditioning on accept/reject and noting $T(v)(1-\alpha(v)) = \min\{T(v), S(v)\}$. See Appendix A for the full derivation.

Coupling interpretation. Theorem 1 gives MDKD a clean coupling interpretation. Teacher sampling decomposes pointwise into two parts: an overlap component $\min\{T,S\}$, corresponding to tokens already covered by the student, and a residual component $[T-S]_+$, corresponding to teacher-supported mass missing from the student. The MDKD stochastic acceptance rule samples *exactly* from this residual component. Standard teacher sampling, by contrast, spends most of its probability on the overlap, where additional training updates are wasted because the student already matches the teacher. The actual MDKD algorithm uses finite-candidate sampling and a deterministic

fallback, so its full selected-token distribution is a mixture of pure residual sampling (the stochastic step characterized above) and a divergence-biased deterministic selection rule (Corollary 2). Both components push generation toward $[T - S]_+$, the structurally absent part of the teacher distribution, but only the stochastic step samples from the residual exactly.

Corollary 2 (Selected-token distribution under finite candidates)**.** *Suppose the m candidates are drawn i.i.d. with replacement from T, and let $\delta = \mathrm{TV}(T, S)$. Then with probability $1 - (1 - \delta)^m$ at least one candidate is stochastically accepted, and conditioned on this event (whenever $\delta > 0$) the accepted token is distributed according to the residual distribution $R(v) = [T(v) - S(v)]_+/\delta$ (Theorem 1, claim 2). With the complementary probability $(1 - \delta)^m$, no candidate is stochastically accepted and the deterministic fallback selects $\arg\max_j \alpha_j$ from the m candidates, inducing a fallback distribution $F_m(v)$ over the same support. The full selected-token distribution is therefore the mixture*

$$P_{\mathrm{MDKD}}(v) \;=\; \left[1 - (1 - \delta)^m\right] R(v) \;+\; (1 - \delta)^m \, F_m(v). \tag{7}$$

Proof sketch. Each of the m candidates is independently accepted with probability δ by Theorem 1, so the no-accept event has probability $(1 - \delta)^m$; conditioning on the complementary event, every accepted candidate has the same conditional law R, and the law of total probability yields Eq. (7). See Appendix A for details.

Divergence-biased fallback. The fallback distribution F_m is not uniform teacher sampling. By construction the fallback selects the candidate with the largest acceptance score α_j among the m teacher proposals, biasing F_m toward tokens with high (but unrealized) residual mass. Two consequences follow. First, both components of the mixture in Eq. (7) push generation toward $[T - S]_+$; the algorithm is divergence-targeting throughout, with the stochastic step targeting the residual exactly and the fallback targeting it greedily within the candidate set. Second, the mixture weight $1 - (1 - \delta)^m$ is exactly the per-step total variation distance evaluated through the finite-candidate filter: it grows when local teacher–student disagreement is large and shrinks when the student already matches the teacher. Per-position diagnostics (Appendix C.5) show that this mixture weight falls with decode position, consistent with the empirical observation that the fallback dominates increasingly at later positions as local $\mathrm{TV}(T, S)$ shrinks; the effect of varying the candidate pool size m is reported in Appendix C.4. In the deployed configuration the stochastic step fires on only about 10% of token decisions and the fallback handles the rest (Appendix C.1). That frequency is not a free parameter. By Theorem 1 the probability of accepting any one candidate equals the local total variation, so a student that already agrees with the teacher on most of the mass produces a small per-candidate acceptance probability, and the fallback fills every position at which all m candidates are declined. The fallback does not replace the residual-sampling idea. A closed form for F_m (Appendix A) shows that it concentrates on the highest-divergence candidates, and an ablation that disables stochastic acceptance drops accuracy while removing the fallback entirely does not (Appendix J.1). Theorem 1 is therefore an exact description of the stochastic step, which the full algorithm realizes exactly on a minority of positions and approximates, through the divergence-biased fallback, on the rest.

Connection to speculative decoding. Theorem 1 sharpens the connection to speculative decoding [Leviathan et al., 2023]. Speculative decoding accepts a draft proposal v with probability $\min(1, T(v)/S(v))$, which corresponds to sampling from the overlap component $\min\{T, S\}$ with the residual mass handled by a target-side correction, allowing teacher computation to be skipped on tokens where draft and target agree. MDKD's stochastic acceptance rule $[1 - S(v)/T(v)]_+$ is the exact complement: it samples from the residual component $[T - S]_+$ to extract maximum learning signal on tokens where the student does *not* agree with the teacher. The two methods thus form dual decompositions of teacher sampling along the same partition

$$T(v) \;=\; \min\{T(v), S(v)\} \;+\; [T(v) - S(v)]_+,$$

with opposite acceptance polarities and opposite goals: speculative decoding accepts on agreement to skip teacher work; MDKD accepts on disagreement to extract maximum learning signal per teacher query.

3.5 Training objective and algorithm

Given an MDKD-generated sequence $x \sim \mu_{\theta^-}$, where μ_{θ^-} is the sequence distribution induced by Algorithm 1 run against a frozen student snapshot S_{θ^-}, the student is trained by minimizing

$$\mathcal{L}_{\mathrm{MDKD}}(\theta) \;=\; \mathbb{E}_{x\sim\mu_{\theta^-}}\left[\sum_{i=t+1}^{|x|} \ell_D\big(T_i,\, S_i(\theta)\big)\right], \tag{8}$$

where ℓ_D is one of the divergences in Eq. (2). Crucially, gradients are not propagated through the sampling process: μ_{θ^-} acts as a data-generating distribution, while the loss is computed on the resulting sequences using the live student S_θ. The choice of ℓ_D is independent of the generation rule; we evaluate all three variants in the experiments.

Why reverse KL pairs naturally with MDKD. Although MDKD is compatible with any of the divergences in Eq. (2), reverse KL has a particularly natural pairing with the sampling rule. The MDKD generator selects teacher-supported tokens that the student underweights ($S_\tau \ll T_\tau$); reverse KL training penalizes student probability mass placed away from teacher modes on the resulting sequences, making it complementary to the sampling rule. We confirm empirically (Appendix C.2) that reverse KL outperforms forward KL and JSD in the MDKD pipeline. The gain is not an artifact of that choice. On the 14B→1.5B base GSM8K setting of Table 7 (7K samples), re-training every baseline under reverse KL leaves MDKD ahead of the best matched-loss baseline by 6.3 percentage points (71.72% versus 65.45% for SKD). Reverse KL helps GKD and SKD but hurts supervised KD, whose fixed-corpus sequences are off-policy (Appendix J.3).

Training algorithm. The training loop is straightforward: for each mini-batch, the procedure (i) snapshots the live student via $S_{\theta^-} \leftarrow \mathrm{detach}(S_\theta)$, (ii) generates divergence-aware responses with Algorithm 1 against the snapshot, (iii) computes Eq. (8) between teacher and live student over the generated tokens, and (iv) updates θ by gradient descent. The student snapshot is refreshed once per mini-batch. Pseudocode is given in Appendix B.

4 Experiments

We evaluate MDKD across three text-generation task families, math reasoning, dialogue summarization, and code generation, on two architecturally distinct teacher–student pairs. All methods share identical training data, models, optimization budget, and decoding hyperparameters; only the sequence-generation and loss procedures differ.

4.1 Setup

Teacher–student pairs. We use two pairs that span the size and pre-training conventions common in modern open-weight releases: (i) **Qwen2.5-14B-Instruct** → **Qwen2.5-1.5B-Instruct** [Qwen et al., 2025], a $\sim 9\times$ parameter compression with both checkpoints instruction-tuned, and (ii) **Gemma-3-12B-it** → **Gemma-3-1B-it** [Gemma Team, 2025], a $\sim 12\times$ compression on a different base architecture and pre-training corpus. The teachers are used off-the-shelf without further fine-tuning. Each student is first lightly task-SFT'd on the same 1K-sample subset that subsequently serves as the KD training data, providing a shared initialization for every KD method so that the comparison isolates the KD procedure itself. This protocol is distinct from the headline number in the abstract. Table 1 trains instruction-tuned students for 3 epochs. The abstract's $8.26\% \rightarrow 64.90\%$ result instead uses a base (not instruction-tuned) 1.5B student trained for 5 epochs on 1K samples (Table 7). Within each comparison the student identity and the epoch count are identical across methods, so the margins reflect the method rather than a difference in budget or initialization.

Training. All methods share identical training infrastructure: 8×AMD MI300X GPUs, per-GPU batch size 8 (effective batch 64 via DDP through HuggingFace Accelerate [Wolf et al., 2020]), bfloat16 precision, learning rate 1×10^{-5} with cosine schedule and no warmup, gradient clipping at 1.0, seed 42. Each KD method runs for 3 epochs over the 1K-sample training set, with per-epoch checkpointing; the best checkpoint per method is selected by the held-out task metric. At evaluation time, all checkpoints are decoded with vLLM at $T=0.7$, top-$p=0.95$, max length 2,048, seed 42.

Baselines. We compare MDKD against three established sampling-based KD baselines, all using forward KL on the teacher's next-token distribution: **Supervised KD**, **On-Policy KD (GKD)** [Agarwal et al., 2024], **Speculative KD (SKD)** [Xu et al., 2024]. MDKD uses reverse KL with a teacher sampler at $T = 0.7$, nucleus $p_{\text{nuc}} = 0.95$, and $m = 5$ candidates per token. Baseline hyperparameters follow the original SKD repository. Full hardware, reproducibility, and evaluation details are reported in Appendix G.

4.2 Dataset and Evaluation

We evaluate MDKD and the baseline KD methods on three text-generation tasks that span complementary regimes of next-token uncertainty: **arithmetic reasoning**, **dialogue summarization**, and **code generation**. Following the task-specific KD setup of Xu et al. [2024], we operate in the low-data regime by drawing 1K-sample training subsets, which is the realistic budget for downstream task-specific distillation. For all three tasks, the same 1K subset is used both to lightly task-SFT the student (shared initialization) and to train every KD method, so any difference in held-out performance is attributable to the KD procedure rather than to the data.

Arithmetic Reasoning. For arithmetic reasoning, we use the GSM8K dataset [Cobbe et al., 2021], following Xu et al. [2024]. We randomly sample 1K instances from the GSM8K training set (seed 42) to form our input–output pairs (x, y), and evaluate on the full GSM8K test set (1,319 instances). We assess the quality of generated outputs using answer accuracy, computed by a lenient (format-agnostic) symbolic grader so that solutions in either Qwen's or Gemma's preferred chain-of-thought style are scored uniformly.

Dialogue Summarization. For dialogue summarization, we use the DialogSum dataset [Chen et al., 2021b], following Xu et al. [2024], augmented with the SAMSum corpus [Gliwa et al., 2019] to broaden the conversation distribution. We randomly sample 1K dialogue–summary pairs (x, y) from the union as our training set, and evaluate on a held-out test split of 500 dialogues. To assess summary quality, we use the ROUGE-L metric [Lin, 2004].

Code Generation. For code generation, we adopt a cross-distribution evaluation: we train on the MBPP dataset [Austin et al., 2021] (374 program-synthesis problems) and evaluate Pass@1 on HumanEval [Chen et al., 2021a] (164 problems) with execution-based grading. This setup is intentionally cross-distribution, as MBPP and HumanEval differ in problem format, function-naming conventions, and test-case style; it therefore probes whether distillation transfers programming competence beyond the training distribution rather than memorizing surface patterns of MBPP.

4.3 Results

4.3.1 Main Results

Table 1 reports KD performance on six (model-pair, task) cells. **MDKD wins every cell**, on both architecture families and across all three task types, despite the methods sharing identical optimization budgets, batch sizes, and KD training data. The margins on summarization are small, so we repeated the Qwen pair over five seeds (Appendix J.5): MDKD remains first on all three tasks, and its margin over the next-best method exceeds the seed-to-seed standard deviation on each of them; the summarization gain is the smallest in absolute terms. The Gemma pair is reported from the single seed shared with Table 1.

4.3.2 In-Domain Transfer and Regularization

We next ask whether distillation improves only the training task, or whether it preserves broader mathematical competence when the student is evaluated on a different in-domain benchmark. This setting is important because math KD can easily over-specialize: long chain-of-thought data and narrow answer formats may improve the source corpus while damaging nearby tasks. We therefore evaluate two forms of in-domain transfer. First, we train on the long-CoT s1K subset and evaluate on Math-500. Second, we train on UltraInteract-math (UI-math) and evaluate both on held-out UI-math and on GSM8K. Throughout this section, **MDKD** denotes the rejection-sampling method.

Table 1: Main results: KD methods compared across two teacher–student pairs and three task families. We report GSM8K accuracy, DialogSum ROUGE-L on 500 held-out dialogues, and HumanEval Pass@1 (%, execution-based) when training on MBPP. All methods use the same 1K-sample training set, identical hyperparameters, and 3 training epochs. **Bold** marks the best method per column; MDKD is best on every (model, task) cell.

	Qwen2.5-14B-it → Qwen2.5-1.5B-it			**Gemma-3-12B-it → Gemma-3-1B-it**		
Method	**GSM8K** Acc. (%)	**Summ.** ROUGE-L	**HumanEval** Pass@1 (%)	**GSM8K** Acc. (%)	**Summ.** ROUGE-L	**HumanEval** Pass@1 (%)
Supervised KD	33.2	36.61	46.3	33.7	36.85	31.7
SKD	33.0	36.03	43.9	36.3	35.91	32.2
On-Policy KD (GKD)	41.1	36.88	44.5	33.8	35.31	32.3
MDKD (ours)	**44.4**	**37.39**	**50.1**	**41.8**	**37.24**	**36.2**

Table 2: Best Math-500 accuracy after s1K training.

Model / Method	**Math-500 Acc.**
Student-IT	46.80%
SFT Student	33.60%
Supervised KD	40.60%
On-Policy KD (GKD)	47.20%
MDKD	**50.00%**

Table 3: UI-math distillation with 2K samples from the off-the-shelf 14B teacher; GSM8K measures transfer.

Model / Method	**UI-math Dev**	**GSM8K**
SFT student	27.35%	8.57%
Supervised KD	64.07%	44.20%
On-Policy KD (GKD)	69.66%	31.01%
SKD	56.69%	47.08%
MDKD	**70.66%**	**54.59%**

Table 2 shows that s1K training is a difficult regularization test. SFT on only 900 long-CoT examples reduces the 1.5B-Instruct student's Math-500 accuracy from 46.80% to 33.60%, and Supervised KD remains below the off-the-shelf student. GKD is approximately neutral, ending at 47.20%. MDKD is the only method that gives a clear positive transfer result, reaching 50.00% on Math-500, a $+3.20$ pp gain over the original student (full protocol and per-epoch trajectory in Appendix F.1).

The larger UI-math experiment gives the same conclusion under a more controlled shared-initialization protocol. All methods start from the same student, the base `Qwen2.5-1.5B` (not the instruction-tuned variant), SFT'd on UI-math, then consume the same shuffled KD stream. After only 2K KD samples, MDKD is best on both the source-domain dev set and the transfer benchmark: it reaches 70.66% on UI-math and 54.59% on GSM8K (Table 3). The transfer margin is especially large: MDKD is $+7.51$ pp ahead of the next-best GSM8K method, SKD, while also being the top method on UI-math (full protocol, reference accuracies, and teacher ablation in Appendix F.2).

Together, these results suggest that MDKD behaves as a regularized distillation objective rather than a narrow source-task optimizer. On s1K, it avoids the long-CoT overfitting that hurts SFT and Supervised KD. On UI-math, it improves the held-out source task while producing the strongest transfer to GSM8K. The detailed protocols, per-epoch s1K trajectory, and teacher ablations are reported in Appendix F.

4.3.3 Low-to-High Data Distillation Behavior

We additionally study how MDKD behaves as the GSM8K KD budget grows from 1K to 7K samples, with base `Qwen2.5-1.5B` and `Qwen2.5-0.5B` students task-SFT'd on GSM8K as shared initializations. Across both student sizes, MDKD is the only method that consistently improves over the SFT initialization at every per-1K checkpoint, reaching $+10.92$ pp at 1.5B and $+7.43$ pp at 0.5B; several baselines (notably Supervised KD at 1.5B) instead degrade immediately at 1K samples and never recover, indicating that early distillation updates can damage an already competent SFT model. Full per-1K trajectories, the side-by-side accuracy figure, the protocol, and the best-checkpoint summary are reported in Appendix E.

5 Limitations

The main practical limitation of MDKD is its training-time cost. MDKD belongs to the family of *active-generation* KD methods (alongside on-policy KD and SKD) that query the teacher and a student snapshot at decode time during training, which makes it noticeably slower per epoch than supervised KD on a fixed teacher-labeled corpus (Appendix H). This cost is not driven by hyperparameter search: MDKD is empirically robust to the candidate pool size m (Appendix C.4), and a single nucleus threshold $p_{\text{nuc}} = 0.95$ is used in all main-paper experiments without per-task tuning. The remaining bottleneck is the generation procedure itself; integrating MDKD with efficient inference engines such as `vLLM` would substantially reduce this overhead and bring MDKD's wall-clock training cost much closer to that of supervised KD.

Two further limitations qualify the scope of the theory and of the learning signal. First, Theorem 1 characterizes the stochastic acceptance step exactly, but the deployed algorithm realizes that step on only about 10% of token decisions; the rest are handled by the divergence-biased fallback of Corollary 2. The fallback ablation in Appendix J.1 is what connects the theorem to the measured gains: disabling stochastic acceptance drops accuracy, while removing the fallback does not, so the improvement is carried by the residual-sampling step even though that step fires on a minority of positions. Second, MDKD concentrates supervision on tokens where the teacher places mass the student does not. If the teacher is confidently wrong on such a token, and the student happened to be right, MDKD pushes the student toward the teacher's error more aggressively than uniform teacher sampling would. The method therefore assumes a teacher that is the better model on the regions it asks the student to learn.

6 Conclusion

We introduced Maximum Divergence Knowledge Distillation (MDKD), which constructs training sequences for autoregressive language models by preferentially selecting tokens where the student underestimates the teacher. MDKD instantiates a unified active-distillation principle, *train where the student most disagrees*, through a simple rejection-sampling rule with an exact distributional characterization (Theorem 1): the stochastic step samples precisely from the teacher's uncovered mass $[T - S]_+$ with acceptance probability equal to the teacher–student total variation distance, making MDKD the structural complement of speculative decoding along the same overlap–residual partition. Across two teacher–student families (Qwen2.5-14B-Instruct $\rightarrow$ 1.5B-Instruct and Gemma-3-12B-it $\rightarrow$ 1B-it) and three task families (GSM8K, DialogSum, HumanEval-from-MBPP), MDKD wins every (model-pair, task) cell against supervised KD, on-policy KD (GKD), and speculative KD (SKD) under identical training data and optimization budget; it is also the strongest method on in-domain transfer (s1K $\rightarrow$ Math-500, UI-math $\rightarrow$ GSM8K) and the only method that compounds gains over the SFT-warmed initialization in the low-to-high data regime. In the scaling study with a 14B teacher, MDKD lifts a base 1.5B student from 8.26% to 71.72% on GSM8K with 7K samples, recovering 73% of the teacher–student gap. A divergence-gain diagnostic confirms the mechanism: MDKD sequences raise the student's cross-entropy by 43.6% while preserving 98% of teacher-sampling accuracy, providing a stronger learning signal per teacher query.

Acknowledgements and Disclosure of Funding

This work was supported by Advanced Micro Devices, Inc. (AMD), which provided the compute used for all experiments.

Competing interests. The authors declare no competing interests beyond the affiliations listed.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems*, 28, 2015.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A real-life scenario dialogue summarization dataset. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.449. URL https://aclanthology.org/2021.findings-acl.449/.

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

Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

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

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, 2019.

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

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary, 2018. URL https://arxiv.org/abs/1805.05532.

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

Aref Jafari, Mehdi Rezagholizadeh, and Ali Ghodsi. Improved knowledge distillation by utilizing backward pass knowledge in neural networks. *arXiv preprint arXiv:2301.12006*, 2023.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *International Conference on Machine Learning*, 2023.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.

Paul Micaelli and Amos Storkey. Zero-shot knowledge transfer via adversarial belief matching. *Advances in neural information processing systems*, 32, 2019.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi

Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. KDGAN: Knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 775–786, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Wenda Xu, Rujun Han, Zifeng Wang, Long T Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. *arXiv preprint arXiv:2410.11325*, 2024.

A Proofs

This appendix contains the full proofs of Theorem 1 and Corollary 2.

Proof of Theorem 1. A token v is proposed with probability $T(v)$ and accepted with probability $\alpha(v) = [1 - S(v)/T(v)]_+$. For any $v \in \mathcal{V}_0$, the joint probability of proposal-and-acceptance is

$$T(v)\,\alpha(v) \;=\; T(v)\left[1 - \frac{S(v)}{T(v)}\right]_+ \;=\; \big[T(v) - S(v)\big]_+,$$

where the right-hand side covers both cases ($T(v) > S(v)$ and $T(v) \le S(v)$) by direct verification. Summing over v and using the identity $\sum_v [T(v) - S(v)]_+ = \frac{1}{2}\sum_v |T(v) - S(v)| = \mathrm{TV}(T, S)$, which follows because $\sum_v (T(v) - S(v)) = 0$ implies $\sum_v [T(v) - S(v)]_+ = \sum_v [S(v) - T(v)]_+$, yields claim 1. Claim 2 follows immediately by normalizing $T(v)\alpha(v) = [T(v) - S(v)]_+$ by $\Pr[\text{accept}] = \mathrm{TV}(T, S)$, which is well-defined whenever $\mathrm{TV}(T, S) > 0$. For claim 3, observe that

$$T(v)\big(1 - \alpha(v)\big) \;=\; \min\{T(v), S(v)\},$$

again by direct case analysis: if $T(v) > S(v)$, the left-hand side equals $T(v) \cdot S(v)/T(v) = S(v) = \min\{T(v), S(v)\}$; if $T(v) \le S(v)$, it equals $T(v) \cdot 1 = T(v) = \min\{T(v), S(v)\}$. Summing gives $\Pr[\text{reject}] = \sum_v \min\{T(v), S(v)\} = 1 - \mathrm{TV}(T, S)$, and normalizing by this quantity (well-defined whenever $\mathrm{TV}(T, S) < 1$) yields the claim. □

Proof of Corollary 2. By Theorem 1, each of the m candidates is independently accepted with marginal probability δ, so the probability that none is accepted is $(1 - \delta)^m$ and the probability that at least one is accepted is $1 - (1 - \delta)^m$. Conditioned on at least one acceptance, each candidate has the same conditional distribution R (Theorem 1, claim 2), so the distribution of the first accepted candidate equals R regardless of which index j first satisfies $r_j < \alpha_j$. The fallback fires precisely on the complementary event and selects $\arg\max_j \alpha_j$, inducing some distribution F_m over $\mathcal{V}_0$; the law of total probability gives Eq. (7). □

Closed form of the fallback distribution F_m. The mixture weight in Eq. (7) is exact, but it leaves F_m unspecified. F_m does have a closed form. Write $q(v) = \min\{T(v), S(v)\} = T(v)\big(1 - \alpha(v)\big)$ for the joint probability of proposing v and then rejecting it, and write $\delta = \mathrm{TV}(T, S)$, so that $\sum_v q(v) = 1 - \delta$. Group the vocabulary into level sets of the acceptance score, and let $Q(a) = \sum_{u:\alpha(u)=a} q(u)$

be the rejected mass of level a. For a token v define the rejected mass strictly below and up to its level,

$$A_<(v) = \sum_{u:\alpha(u)<\alpha(v)} q(u), \qquad A_\le(v) = A_<(v) + Q\big(\alpha(v)\big).$$

Ties are broken by the lowest candidate index, matching a single left-to-right $\arg\max$. The index j is then the unique maximizer and equals v when the first $j-1$ candidates land strictly below $\alpha(v)$, candidate j is v, and the remaining candidates land at level $\alpha(v)$ or below, with every candidate rejected. Summing over the position j of that first maximizer gives

$$F_m(v) \;=\; \frac{q(v)}{(1-\delta)^m} \cdot \frac{A_\le(v)^m - A_<(v)^m}{Q\big(\alpha(v)\big)}. \tag{9}$$

The level contributions telescope: summing Eq. (9) over v yields one. As m grows, $A_\le^m - A_<^m$ puts increasing mass on the highest level that still carries rejected mass, so the fallback concentrates on the highest-divergence tokens inside the candidate pool. Two degenerate cases are immediate. If $T = S$, then $\alpha \equiv 0$, every token shares one level, and F_m reduces to T itself: with nothing to disagree about, the fallback is ordinary teacher sampling. If the token v is the unique maximizer of α, then $Q(\alpha(v)) = q(v)$ and Eq. (9) simplifies to $\big(A_\le(v)^m - A_<(v)^m\big)/(1-\delta)^m$.

B Training algorithm pseudocode

Algorithm 2 gives the explicit training loop summarized in Section 3.5. For each mini-batch the live student is detached into a frozen snapshot S_{θ^-}, MDKD-Generate (Algorithm 1) is run against the snapshot, the per-token divergence loss is computed between the teacher and the live student, and a single gradient step updates θ. The snapshot is refreshed once per mini-batch.

Algorithm 2 MDKD training loop (in-loop variant)

Require: Student S_θ, teacher T, dataset $\mathcal{D}$, max length n, candidates m, temperature τ, nucleus p_{nuc}, batch size B, learning rate η, divergence ℓ_D

for each mini-batch $\{x^{(i)}\}_{i=1}^{B}$ from $\mathcal{D}$ **do**
  $S_{\theta^-} \leftarrow \text{detach}(S_\theta)$ ▷ Frozen snapshot for generation
  for $i = 1$ **to** B **do**
    $\hat{y}^{(i)} \leftarrow \text{MDKD-GENERATE}(x^{(i)}, T, S_{\theta^-}, n, m, \tau, p_{\text{nuc}})$ ▷ Algorithm 1
  end for
  $\mathcal{L} \leftarrow \dfrac{1}{B}\displaystyle\sum_{i=1}^{B}\sum_{t=|x^{(i)}|+1}^{|\hat{y}^{(i)}|} \ell_D\big(T(\cdot \mid \hat{y}^{(i)}_{<t}),\ S(\cdot \mid \hat{y}^{(i)}_{<t}; \theta)\big)$
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}$ ▷ Gradients flow through S_θ only
end for
return S_θ

C Ablations

MDKD guarantees that each proposed token lies in the teacher's per-step support (nucleus-filtered mass). That property does not, by itself, bound *sequence-level* proximity to a standard teacher rollout: biased choices can compound and shift later teacher conditionals. We therefore isolate key decoding knobs and summarize position-wise statistics on GSM8K problems under a fixed (teacher, student) pair. Unless noted, all configurations share the same chat template and answer extraction as our evaluation protocol, nucleus top-p mass p_{nuc}, the same stop criteria, and a decode budget of up to 512 new tokens per problem. In this run, each configuration uses $N = 512$ problems randomly selected from train set. We report MDKD accuracy (answer correctness) and mean student cross-entropy averaged over generated tokens (difficulty).

C.1 Divergence gain analysis

Before presenting downstream results, we verify MDKD's core premise: that divergence-aware sampling produces sequences that are simultaneously *coherent* and *challenging* for the student. We

Table 4: Divergence gain analysis on GSM8K (512 samples), `Qwen2.5-7B-Instruct` teacher and base `Qwen2.5-0.5B` student. MDKD sequences maintain near-teacher accuracy while increasing the student's cross-entropy loss by 43.6%.

Method	**Accuracy**	**Avg. Length**	**Student CE Loss**
Teacher-only	83.0%	251.9	0.1778
MDKD	81.1%	248.6	0.2553
Delta	−1.9%	−3.3	**+0.0775 (+43.6%)**

Table 5: MDKD ablation study on GSM8K (1,319 test problems), `Qwen2.5-7B-Instruct` teacher and base `Qwen2.5-0.5B` student. Starting from the default config (first row, $\tau = 0.7$, $p_{\text{nuc}} = 0.95$, $m = 5$), we vary the distance metric, temperature, and nucleus threshold. Reverse KL and sharper temperature consistently improve accuracy.

Distance	τ	p_{nuc}	m	**GSM8K Acc. (%)**
Reverse KL	0.7	0.95	5	57.47
Forward KL	0.7	0.95	25	55.95
JSD	0.7	0.95	25	55.12
CE (no KD)	0.7	0.95	5	54.59
Reverse KL	0.2	0.95	5	58.00
Reverse KL	**0.1**	**0.50**	**5**	**59.67**

generate 512 GSM8K responses using (a) standard teacher sampling and (b) MDKD sampling, then measure the student's cross-entropy loss on each set.

In this analysis, we used `Qwen2.5-7B-Instruct` as the teacher and the base `Qwen2.5-0.5B` (no instruction tuning) as the student. As shown in Table 4, MDKD sacrifices only 1.9% accuracy while increasing the student's cross-entropy loss by 43.6%. Per-sample analysis reveals that 399 of 512 samples are solved correctly by both methods; only 26 are correct for the teacher alone and 16 for MDKD alone, indicating minimal quality degradation. Of 127,296 total token decisions, 10.3% were accepted via the stochastic rejection step and 89.7% fell through to the $\arg\max$ fallback. This indicates that MDKD performs a gentle, continuous nudge toward divergent tokens rather than aggressively replacing them, preserving coherence while systematically targeting harder content.

C.2 Hyperparameter analysis

We analyze the impact of MDKD's key design choices through an ablation study on GSM8K (Table 5).

Distance metric. Reverse KL consistently outperforms forward KL, JSD, and cross-entropy. Reverse KL is mode-seeking: it encourages the student to concentrate probability mass on the teacher's top predictions rather than spreading it across the entire distribution. For a capacity-limited student (1.5B parameters), this is more efficient than the mode-covering behavior of forward KL.

Task-dependent tuning. For the main results we use a single MDKD configuration ($\tau = 0.7$, $p_{\text{nuc}} = 0.95$, $m = 5$, reverse KL) across all tasks (Table 6), so that gains over the baselines cannot be attributed to per-task hyperparameter tuning. The GSM8K ablation in Table 5 shows that further task-specific tightening (e.g. a sharper teacher $\tau = 0.1$ with $p_{\text{nuc}} = 0.5$) yields an additional $\sim$2 pp on math reasoning; the main-paper numbers are reported under the default config to keep the comparison hyperparameter-fair. Reverse KL is universally beneficial.

C.3 Scaling to larger teachers

To test how MDKD scales with teacher capacity, we run experiments with Qwen2.5-14B-Instruct as teacher and the base Qwen2.5-1.5B as student on GSM8K. This represents a more challenging distillation scenario: the student starts at 8.26% accuracy (no instruction tuning) with a 94.92% teacher.

Table 6: Per-task hyperparameters used in the main results. Baseline values follow Xu et al. [2024]; MDKD uses a single shared configuration across tasks.

Parameter	GSM8K	Summarization
Baselines (SKD / GKD / Supervised KD)		
Teacher temperature	0.2	0.2
Distance metric	Forward KL	Forward KL
Teacher top-p	1.0	0.5
MDKD (ours)		
Teacher temperature (τ)	0.7	0.7
Distance metric	Reverse KL	Reverse KL
Nucleus top-p	0.95	0.95
Candidates (m)	5	5

Table 7: Scaling experiments: 14B teacher $\rightarrow$ 1.5B student on GSM8K. MDKD's advantage over other methods grows with more data. Results shown at the best epoch for each method.

Method	1K data (5 epochs)	7K data (3 epochs)
SKD	31.77%	39.95%
Supervised KD	35.25%	41.77%
On-Policy KD	54.59%	62.77%
MDKD (ours)	**64.90%**	**71.72%**
Base Student	8.26%	
Base Teacher	94.92%	

Table 7 shows that MDKD's advantage is dramatic with a larger teacher. With 1K training samples, MDKD (64.90%) outperforms On-Policy KD (54.59%) by over 10 points and Supervised KD (35.25%) by nearly 30 points. Scaling to 7K samples improves all methods, but the ranking is preserved: MDKD reaches 71.72%, recovering 73% of the teacher–student gap (per-epoch breakdown in Appendix I). The larger teacher provides a richer probability distribution, amplifying the benefit of MDKD's divergence-aware sampling: with more divergence to exploit, the rejection sampling mechanism selects more informative training tokens.

C.4 Candidate pool size (m)

We run MDKD with fixed $\tau = 0.7$ and $p_{\text{nuc}} = 0.95$, varying only the candidate pool size $m \in \{5, 15, 25, 50\}$. For each m we compare against a teacher-only baseline at the same τ (same nucleus and decode settings; teacher-only does not use MDKD rejection). All experiments use $N = 512$ problems randomly selected from GSM8K train set. We report MDKD vs. teacher-only teacher accuracy and mean student CE (over generated tokens), then paired deltas and the relative CE increase $(\text{CE}_{\text{MDKD}} - \text{CE}_{\text{T-only}}) \,/\, \text{CE}_{\text{T-only}}$.(Table 8)

Results and discussion. Across $m \in \{5, 15, 25, 50\}$, MDKD raises mean student CE by $\sim$26–28% relative to teacher-only CE at $\tau = 0.7$. Teacher accuracy stays near the teacher-only rate for $m \leq 25$ (within $\sim$2 pp), while $m = 50$ shows a larger coherence gap (-2.35 pp vs. T-only). Student CE remains in a narrow band (0.366–0.369): difficulty vs. teacher-only is stable across pool sizes, but very large m trades more teacher accuracy for only a marginal CE change.

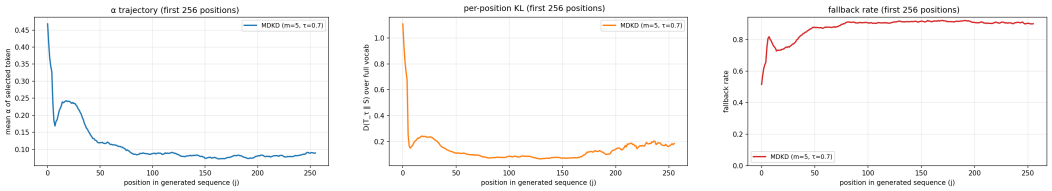
C.5 Per-position diagnostics under default MDKD

We analyze the default MDKD configuration ($m = 5$, $\tau = 0.7$, $p_{\text{nuc}} = 0.95$) over the first 256 decode positions. We report three trajectories: selected-token importance α_{sel}, teacher–student KL $D_{\text{KL}}(T_\tau \| S)$, and fallback rate.

Discussion (Selected-token α). Figure 2a plots α_{sel} only for decode indices $j = 0, \ldots, 255$. Within that window, the curve trends downward as j increases, so accepted-token importance does not grow with depth.

Table 8: MDKD vs. teacher-only by candidate pool size m. Δacc = MDKD acc − T-only acc (pp). ΔCE rel. $= (\mathrm{CE}_{\mathrm{MDKD}} - \mathrm{CE}_{\mathrm{T}})/\mathrm{CE}_{\mathrm{T}}$.

m	MDKD acc (%)	T-only acc (%)	MDKD stud. CE	T-only stud. CE	Δacc	ΔCE
5	92.58	94.34	0.368	0.289	-1.76	$+0.079$ ($+27.3\%$)
15	93.75	94.34	0.366	0.289	-0.59	$+0.077$ ($+26.6\%$)
25	93.16	94.34	0.367	0.289	-1.18	$+0.078$ ($+27.0\%$)
50	91.99	94.34	0.369	0.289	-2.35	$+0.080$ ($+27.6\%$)



(a) A2: α_{sel} (b) A3: $D_{\mathrm{KL}}(T_\tau \| S)$ (c) A4: fallback rate

Figure 2: Per-position diagnostics (first 256 positions), MDKD default configuration ($m = 5$, $\tau = 0.7$, $p_{\mathrm{nuc}} = 0.95$, $N = 512$).

Discussion (teacher–student KL). Figure 2b shows $D_{\mathrm{KL}}(T_\tau \| S)$ over the same $j = 0, \ldots, 255$ range. Visually, KL is relatively flat without a dominant early spike; combined with the headline result that MDKD raises student CE vs. teacher-only at matched τ, this supports interpreting "harder training" primarily through CE on MDKD-selected tokens, rather than through a dramatic change in full-vocab KL shape.

Discussion (Fallback dynamics). Figure 2c shows per-position fallback over $j = 0, \ldots, 255$. Fallback is frequent throughout the plotted range and tends to increase with j within that window.

The clearest take-away is student cross-entropy (CE): under matched configurations, MDKD trajectories yield higher mean CE than teacher-only rollouts, so the student is trained on harder targets. MDKD accuracy stays in a similar band in our sweeps, so this difficulty gain does not come with a large coherence collapse. The per-position panels (Fig. 2) describe behavior for the steps: α_{sel} tends to decrease over positions, KL has no extreme early-only spike in the plot, and fallback is frequent and trends upward with j in the same window. Together with higher headline CE under MDKD, the straightforward reading is stronger student supervision via harder targets, with sequential plots.

D Discussion

When does MDKD help most? MDKD shows the largest gains on tasks with structured, multi-step outputs (math reasoning) and with larger teacher–student capacity gaps (14B→1.5B vs. 7B→1.5B). This suggests MDKD is most valuable when the output space is large enough that the student has substantial room for disagreement.

Teacher-sourced versus student-sourced rollouts. MDKD deliberately draws its candidates from the teacher's nucleus, so every accepted token is teacher-supported. That choice preserves coherence: the divergence-gain analysis finds only a 1.9 percentage point drop in answer accuracy against a 43.6% rise in student cross-entropy. Sampling the rollout from the student instead, and reweighting it by total variation against the teacher, is close in spirit to on-policy KD, which we already compare against. The distinction shows up when the student is weak. In the 0.5B low-to-high setting (Appendix E), on-policy KD falls 4.02 percentage points below its SFT initialization, because the 0.5B student's own rollouts are malformed, and the loss on them pushes it further off-format; MDKD's teacher-sourced candidates avoid that failure. A hybrid that keeps student rollouts but scores them by total variation is a natural next step. We expect the trade-off to be coherence, which teacher-sourced sampling buys, against exposure-bias reduction, which student-sourced sampling buys.

Efficiency considerations. MDKD requires m teacher forward passes per token (to sample candidates) plus one student forward pass (to evaluate them). With $m = 5$, this makes generation approximately 2–3$\times$ slower than standard teacher generation. However, MDKD is significantly faster than SKD, which requires iterative propose-verify loops and does not support batched generation. In our experiments, MDKD training took $\sim$2m 45s per epoch on GSM8K versus $\sim$11m 15s for SKD.

E Additional Details for Low-to-High Data Distillation

Experimental protocol. Both experiments use the same deterministic pipeline: students are SFT'd for 3 epochs on GSM8K, then serve as shared initialization for all four KD methods. Each method runs one epoch on the shuffled 7,473-sample train split, checkpointing every 1,000 samples. All methods consume the identical shuffled stream (`set_seed(42) + random.shuffle`), ensuring apples-to-apples data-exposure comparisons at every 1K boundary.

The two settings differ in:

- **1.5B student:** `Qwen2.5-1.5B` base $\xrightarrow{\text{SFT}}$ 53.30%; teacher is the off-the-shelf `Qwen2.5-14B-Instruct` (94.69%).
- **0.5B student:** `Qwen2.5-0.5B` base $\xrightarrow{\text{SFT}}$ 25.93%; teacher is the SFT'd 14B (94.24%).

Table 9: Best-checkpoint accuracy for both student scales. Δ is gain over the SFT initialization; Gap% is the fraction of the teacher–student gap recovered. The teacher–student gap is 41.39% at 1.5B and 68.31% at 0.5B, so absolute gains are not directly comparable across student sizes.

	1.5B Student			0.5B Student		
Method	**Acc.**	Δ	**Gap%**	**Acc.**	Δ	**Gap%**
MDKD	**64.22**	+10.92	26.4	**33.36**	+7.43	10.9
On-Policy KD (GKD)	56.79	+3.49	8.4	21.91	−4.02	−5.9
Speculative KD (SKD)	42.99	−10.31	−24.9	23.65	−2.28	−3.3
Supervised KD	18.27	−35.03	−84.7	26.31	+0.38	0.6

Detailed findings, 1.5B student. (i) MDKD is the only method that survives the SFT tuned init "valley." At 1K samples, every baseline has already dropped below the 53.30% SFT init: Sup-KD by 35%, SKD by 23%, and GKD by 4.5%. Only MDKD remains above the init (55.42%, +2.1%).

(ii) Supervised KD undergoes catastrophic forgetting. Sup-KD drops from 53.30% to 18.27% within the first 1K samples and continues to drift down through 7K (16.98%). The mechanism is teacher–task mismatch: the off-the-shelf 14B teacher's output distribution is sharp and slightly off-format relative to the gold answers. Forward-KL distillation on these peaked, off-policy continuations actively *unlearns* the student's SFT-induced format. This is cross-validated in the 0.5B setting where an SFT'd teacher eliminates the collapse entirely.

(iii) MDKD's advantage compounds monotonically. At every per-1K checkpoint, MDKD closes more of the teacher–student gap than any other method: +2.1% at 1K, +7.2% at 2K (next best GKD: +0.9%), +10.9% at 7K (next best GKD: +3.5%).

(iv) Off-the-shelf teacher is sufficient. SFT'ing the 14B teacher on GSM8K yields 94.31% (a *loss* of 0.38% from 94.69%), validating the use of the non-SFT teacher for KD.

Detailed findings, 0.5B student. (i) Supervised KD does *not* collapse with an SFT'd teacher. Sup-KD here stays within $\pm$1.6% of the 25.93% init at every checkpoint. This controlled flip of a single variable (off-the-shelf $\rightarrow$ SFT'd teacher) confirms that the 1.5B catastrophic collapse was caused by teacher–task mismatch, not by Sup-KD itself.

(ii) On-Policy KD inverts to the worst method (−4.02%). With a 0.5B student producing malformed rollouts, the FKL signal on student-sampled trajectories steers the student further off-format. This shows that on-policy KD's advantage depends on sufficient student capacity to produce mostly-coherent rollouts.

(iii) Capacity ceiling at ∼33–34%. The 0.5B student plateaus from 4K samples onward, suggesting the model's effective capacity for GSM8K has been reached under this training budget. The relative gap closed shrinks from 26.4% (1.5B) to 10.9% (0.5B) even though the qualitative method ranking is preserved.

F Additional Details for In-Domain Transfer and Regularization

F.1 s1K → Math-500

Protocol. The s1K experiment uses `Qwen2.5-1.5B-Instruct` as the student and `Qwen2.5-14B-Instruct` as the teacher. Training data consists of 900 long chain-of-thought math examples from s1K; evaluation uses Math-500. Runs use 3 epochs, effective batch size 64, learning rate 1×10^{-5}, warmup 0.1, and max-new-tokens 10,000. MDKD uses the same GSM8K-selected configuration, $\tau = 0.1$, $p_{\text{nuc}} = 0.5$, $m = 5$, with reverse KL. GKD and Supervised KD use forward KL with teacher temperature 0.2.

The comparison has one protocol caveat: MDKD and GKD initialize from the off-the-shelf 1.5B-Instruct student, while Supervised KD initializes from the s1K SFT student. This means the Supervised KD number partly reflects damage from the SFT step itself. For this reason, the main text emphasizes the regularization question: whether each training procedure can avoid degrading the original 1.5B-Instruct student's Math-500 performance.

Table 10: Math-500 accuracy by epoch on the s1K → Math-500 transfer task. "crashed" means the training run terminated before producing a valid checkpoint.

Method	Init	Epoch 1	Epoch 2	Epoch 3
Supervised KD	SFT student (33.60%)	40.40%	37.80%	40.40%
On-Policy KD (GKD)	1.5B-Instruct (46.80%)	47.20%	45.00%	*crashed*
MDKD	1.5B-Instruct (46.80%)	48.40%	49.20%	**50.00%**

MDKD improves monotonically across epochs, from 48.40% to 50.00%. GKD initially matches the base student but then degrades and becomes numerically unstable. This trajectory is the clearest evidence in this experiment that MDKD's token-selection rule acts as a regularizer under long-CoT distillation.

F.2 UltraInteract-math → GSM8K

Protocol. UI-math is built from UltraInteract `Math_CoT` and `Math_PoT`, parent-deduplicated to 10,000 training instances. The student is `Qwen2.5-1.5B`, first SFT'd for 3 epochs on the same UI-math corpus and then used as the shared KD initialization. Each KD method trains for one epoch over the shuffled 10K split with checkpoints every 1K samples. Hyperparameters are shared across methods: learning rate 1×10^{-5}, effective batch size 64, `INP_LENGTH` $= 1024$, and `MAX_NEW_TOKENS` $= 1024$. MDKD uses $\tau = 0.7$, $p_{\text{nuc}} = 0.95$, $m = 5$, with reverse KL.

Evaluation uses two math benchmarks. UI-math dev contains 501 problems and is graded with symbolic equality. GSM8K contains 1,319 test problems and is graded with strict numeric matching under greedy decoding. GSM8K is never used for UI-math KD training, so it measures transfer within the math domain.

Table 11: Reference accuracies for the UI-math transfer experiment.

Model	UI-math Dev	GSM8K
Base Student (1.5B, no FT)	0.60%	5.61%
SFT Student (1.5B + UI-math SFT)	27.35%	8.57%
Base Teacher (14B-Instruct)	83.43%	94.69%
SFT Teacher (14B + UI-math SFT, 10K)	71.06%	90.30%

Table 12: KD methods at 2K UI-math samples using the off-the-shelf 14B teacher. Best per row is in **bold**.

Budget	Eval	Sup-KD	GKD	SKD	MDKD
2K	UI-math	64.07%	69.66%	56.69%	**70.66%**
2K	GSM8K	44.20%	31.01%	47.08%	**54.59%**

At 2K samples, MDKD leads on both metrics. The 2K MDKD checkpoint also reaches within 0.40 pp of the 14B SFT teacher's UI-math dev accuracy, despite using only one fifth of the UI-math training data for KD.

Teacher ablation. We also repeat the first 1K UI-math KD step using a 14B teacher that was SFT'd on UI-math. This ablation is useful for understanding teacher choice, but it is not the main transfer result because all methods are substantially weaker than with the off-the-shelf teacher.

Table 13: KD methods after 1K UI-math samples using the SFT'd 14B teacher. Best per column is in **bold**.

Method	UI-math Dev	GSM8K
Supervised KD	50.30%	8.19%
On-Policy KD (GKD)	48.70%	8.87%
SKD	49.50%	**13.27%**
MDKD	**52.50%**	8.04%

The off-the-shelf teacher gives higher UI-math accuracy for every method at the 1K budget: MDKD 66.27% vs 52.50%, GKD 67.86% vs 48.70%, and Supervised KD 64.27% vs 50.30%. This supports using the broader instruction-tuned teacher for distillation rather than first specializing the teacher to the source corpus.

G Additional experimental details

Hardware. All experiments were run on nodes with 8 AMD Instinct MI300X GPUs (192 GB HBM3 each) using PyTorch with ROCm, HuggingFace Transformers, and Accelerate for distributed data parallel training.

Reproducibility. Seeds are fixed at 42 via `set_seed(42)` from HuggingFace Accelerate, covering Python `random`, NumPy, and PyTorch CUDA RNG. Results are approximately reproducible but not bit-for-bit deterministic due to nondeterministic CUDA kernels and DDP all-reduce operations.

Evaluation. All models are evaluated with a single sampled generation per problem using vLLM. GSM8K uses the `math_equal` symbolic grader for numerical equivalence. Summarization uses the `rouge-score` Python package. HumanEval uses execution-based Pass@1 grading on the official test cases.

H Training time comparison

Table 14: Training time per epoch on GSM8K (7B→1.5B, 1K samples, 8 GPUs).

Method	Time / Epoch	Relative
Supervised KD	∼25 s	1.0×
On-Policy (GKD)	∼1 m 30 s	3.6×
MDKD (ours)	∼2 m 45 s	6.6×
SKD	∼11 m 15 s	27.0×

MDKD is approximately 4× faster than SKD per epoch. This speedup comes from three factors: (1) full batch parallelism during generation, (2) KV-cache reuse across decoding steps, and (3) no wasted computation from rejected token chunks.

I Scaling experiments: per-epoch accuracy

Table 15: Per-epoch accuracy for the 14B→1.5B scaling experiment on GSM8K with 7K training data.

Method	Epoch 1	Epoch 2	Epoch 3
MDKD	**71.72%**	70.66%	70.13%
On-Policy KD	62.77%	60.88%	60.12%
Supervised KD	38.51%	41.55%	**41.77%**
SKD	36.47%	**39.95%**	39.88%

On-policy methods (MDKD, On-Policy KD) peak at epoch 1 with 7K data, while off-policy methods (Supervised KD, SKD) continue improving through later epochs. This suggests that on-policy methods learn more efficiently per step but are more sensitive to overfitting with more passes over the data.

J Analyses of the acceptance rule, the loss, and seed variance

This appendix collects the analyses that separate the sampling rule from the fallback, from the choice of divergence, and from seed noise. Unless a table says otherwise, MDKD uses reverse KL.

J.1 Fallback policy

Corollary 2 writes the deployed sampler as a mixture of residual sampling and a deterministic fallback. To test which component carries the gain, we vary only the rule used when no candidate is stochastically accepted, and hold everything else fixed: a 14B teacher, a base 1.5B student, 1K GSM8K examples, reverse KL, 3 epochs, $m=10$, $\tau=1.0$, and $p_{\mathrm{nuc}}=1.0$. Setting τ and p_{nuc} to 1 removes nucleus truncation, so the comparison isolates the acceptance rule. These absolute accuracies are therefore not comparable to Table 7, which uses $\tau=0.7$, $p_{\mathrm{nuc}}=0.95$, and $m=5$. The no-fallback variant resamples until some candidate is accepted, which is the pure residual sampler of Theorem 1 and is much slower, because a low acceptance rate forces many redraws. The fallback-only variant disables stochastic acceptance and always takes $\arg\max_j \alpha_j$.

Table 16: GSM8K accuracy when only the fallback policy changes. The stochastic step is present in every row except the last. The 14B→1.5B base student is trained for 3 epochs on 1K examples with reverse KL, $m=10$, $\tau=1.0$, and $p_{\mathrm{nuc}}=1.0$.

Fallback policy	**GSM8K acc. (%)**
$\arg\max \alpha$ (default MDKD)	70.20
Random among the m candidates	68.88
Draw a fresh teacher sample	71.42
No fallback (resample until acceptance)	71.19
Fallback only (stochastic acceptance off)	65.55

Every variant that keeps the stochastic step lands in a tight band, 68.9 to 71.4%. Removing the fallback entirely, at the price of repeated resampling, does not hurt (71.19 versus 70.20). Disabling stochastic acceptance does hurt (65.55). The gain therefore comes from the residual-sampling step, not from the particular fallback used to fill the remaining positions. The default $\arg\max$ rule is kept because it matches the pure residual sampler within 1 percentage point and avoids the resample loop.

J.2 Acceptance rate as a readout of disagreement

Theorem 1 says the per-candidate acceptance probability equals $\mathrm{TV}(T, S)$. A low acceptance rate is then the expected reading for a student that already agrees with the teacher, not a sign that the rule is idle. We check this by freezing the teacher at Qwen2.5-14B-Instruct and changing only the student, running MDKD as an inference-time diagnostic on $N=512$ GSM8K problems with $m=10$ and $p_{\mathrm{nuc}}=0.95$. Temperature scales only the teacher (Algorithm 1), so $\tau=1.0$ is the point at which identical distributions must produce zero acceptance.

Table 17: Share of positions filled by the stochastic step, against a fixed 14B-Instruct teacher at $\tau=1.0$. The complement is the fallback. The three students span no disagreement, a competent same-family student, and a randomly initialized student.

Student	**Accept (%)**	**Fallback (%)**
14B-Instruct ($T \equiv S$)	0.00	100.00
1.5B-Instruct	15.35	84.65
1.5B, random initialization	100.00	0.00

Acceptance moves from exactly 0% (no residual, over roughly 153,000 token decisions) to 100% as the disagreement grows. At the deployed temperature $\tau=0.7$, the 1.5B-Instruct student fills 10.23% of positions by stochastic acceptance, in line with the roughly 10% rate measured for the smaller pair in Appendix C.1.

A low rate also does not mean a low effect on the sequence, because the next-token distribution is conditional and an early change reroutes everything after it. Accepts concentrate at the start of the sequence, which is the same pattern as the rising fallback curve in Appendix C.5. On the GSM8K problem "Jeff and Brad divide $100; Jeff gets 4 times as much as Brad. How much does Jeff get?", the teacher and a 1.5B-Instruct student open with nearly the same correct solution ("let B be Brad's share, so Jeff gets $4B$"). MDKD, with about 15.8% of its tokens accepted and those accepts concentrated in the opening, follows a different but still correct derivation. The extra learning signal measured by the 43.6% cross-entropy gap is carried by that short prefix.

J.3 Loss-controlled baselines

The main baselines use forward KL, following their original papers, while MDKD uses reverse KL. Table 18 removes that confound by retraining every baseline under reverse KL on the Table 7 setting (14B teacher, base 1.5B student, 7K GSM8K examples). The forward-KL column is the one already reported in Table 7.

Table 18: GSM8K accuracy under a matched divergence. Same 14B→1.5B base, 7K-sample setting as the right-hand column of Table 7. MDKD is trained only with reverse KL.

Method	Forward KL	Reverse KL
Supervised KD	41.77	31.10
On-Policy KD (GKD)	62.77	65.26
SKD	39.95	65.45
MDKD (ours)	—	**71.72**

Reverse KL helps the two methods whose sequences are drawn on-policy or interleaved (GKD and SKD), and it hurts supervised KD, whose fixed teacher continuations are off-policy relative to a mode-seeking loss. Under the matched reverse-KL objective, MDKD remains 6.27 percentage points above the best baseline (SKD at 65.45%). The sampling rule, not the loss, accounts for that margin.

Total variation can be used as a training loss as well as a sampling criterion, and the two choices are independent: Theorem 1 uses $\mathrm{TV}(T, S)$ only to decide which tokens to accept, while ℓ_D in Eq. (8) is chosen separately. We do not report a pure total-variation loss trained on ordinary teacher samples in this version.

J.4 MiniLLM and Adaptive KL

We add two further baselines on GSM8K, both adapted from their released implementations to this task. MiniLLM is the closest reverse-KL method: it minimizes reverse KL on student-generated sequences. Adaptive KL blends forward and reverse KL with a loss-side schedule rather than a sampling rule. MiniLLM's auxiliary pretraining-corpus language-modeling term is omitted, because that corpus is outside the 1K and 7K task-specific setting used here.

Table 19: GSM8K accuracy of MiniLLM and Adaptive KL against MDKD. The Table 1 setting is the instruction-tuned 1.5B student; the Table 7 setting is the base 1.5B student with the off-the-shelf 14B teacher.

Method	Table 1 (1.5B-it)	Table 7 (1.5B base)
MiniLLM	39.8	67.81
Adaptive KL	26.2	19.70
MDKD (ours)	**44.4**	**71.72**

MDKD is ahead in both settings. Adaptive KL, which changes only the loss and not the sampling distribution, falls well below the KD baselines of Table 1.

J.5 Five-seed rerun of the Qwen main comparison

Table 1 is a single seed. We reran the Qwen pair (14B-Instruct $\rightarrow$ 1.5B-Instruct) over five seeds and report the mean and the standard deviation across seeds. This is the seed-to-seed spread, not a confidence interval, and it is not a standard error. The Gemma pair was not part of this sweep and remains the single run in Table 1. Means need not reproduce the single seed of Table 1; the comparison of interest is the ranking and the margin relative to the spread.

Table 20: Qwen2.5-14B-Instruct $\rightarrow$ 1.5B-Instruct, same protocol as the left half of Table 1, mean $\pm$ standard deviation over 5 seeds.

Method	**GSM8K**	**DialogSum ROUGE-L**	**HumanEval**
Supervised KD	33.3 ± 1.9	36.67 ± 0.36	46.59 ± 0.27
SKD	38.2 ± 2.7	36.40 ± 0.48	44.91 ± 0.92
On-Policy KD (GKD)	39.3 ± 2.4	36.48 ± 0.43	45.56 ± 0.70
MDKD (ours)	$\mathbf{45.5 \pm 2.5}$	$\mathbf{37.51 \pm 0.24}$	$\mathbf{49.65 \pm 0.61}$

MDKD is first on all three tasks, and on each one its margin over the next-best method is larger than the seed-to-seed standard deviation of either method. On GSM8K the margin over GKD is 6.2 points, against standard deviations of 2.4 to 2.5. On HumanEval the margin over supervised KD is 3.1 points, against standard deviations below 1. On DialogSum the next-best method is supervised KD, and the margin is 0.84 ROUGE-L, against standard deviations of 0.24 (MDKD) and 0.36 (supervised KD). The summarization gain is therefore consistent across seeds but small in absolute terms.

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 introduction clearly state the proposed MDKD method, its theoretical characterization, and its empirical contributions, which are supported by the theoretical analysis and experiments in Sections 3 and 4.

   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 5 (Limitations) discusses the training-time cost of MDKD relative to supervised KD (quantified in Appendix H), the gap between the exact theory (Theorem 1 characterizes only the stochastic acceptance step, which fires on about 10% of token decisions) and the deployed algorithm with its deterministic fallback, and the risk of amplifying teacher errors when the teacher is confidently wrong on residual tokens. The scope of the seed-variance analysis (five seeds for the Qwen pair only) is stated in Appendix J.5.

   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: Theorem 1 and Corollary 2 are stated with their full assumptions in Section 3.4, each followed by a proof sketch. Complete proofs, and the closed form of the fallback distribution F_m, are given in Appendix A.

   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: The algorithm is fully specified in Section 3 (Algorithm 1) and Appendix B (Algorithm 2). Models, datasets, splits, training and evaluation protocols are given in Section 4.1–4.2; hyperparameters in Table 6; and hardware, seeds, and evaluation details in Appendix G. Protocols for the additional experiments are in Appendices E, F, and J.

   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: The code is publicly released at `https://github.com/AMD-AGI/mdkd`. All datasets (GSM8K, DialogSum, SAMSum, MBPP, HumanEval, s1K, MATH-500, UltraInteract) and all teacher and student models (Qwen2.5, Gemma-3) are publicly available; data subsampling uses fixed seeds as described in Section 4.2 and Appendix G.

   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 4.1–4.2 specifies the models, training subsets and test splits, optimizer settings (learning rate, schedule, batch size, gradient clipping, epochs), checkpoint selection, and evaluation metrics. Per-method hyperparameters are listed in Table 6, and the rationale for the shared MDKD configuration is given in Appendix C.2.

   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 the main comparison on the Qwen2.5-14B-Instruct → 1.5B-Instruct pair, Appendix J.5 (Table 20) reports the mean and standard deviation over five training seeds for all methods on all three tasks. The variability captured is the training seed; the reported spread is the sample standard deviation across seeds (1-sigma), not a standard error or confidence interval, and no normality assumption is made. The Gemma pair in Table 1 and the remaining experiments (Tables 2–19) are single runs with seed 42, as stated in Section 4.3.1 and Appendix J.5.

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: Appendix G describes the hardware (nodes of 8 AMD Instinct MI300X GPUs, 192 GB each) and software stack, and Appendix H (Table 14) reports per-epoch training time for each method.

   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 research uses publicly available models and benchmark datasets and does not involve human subjects, private data, or deceptive data collection.

   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: [N/A]

    Justification: The paper presents a general methodological contribution in knowledge distillation and does not target a specific real-world deployment or application domain with direct societal impact.

    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 a new high-risk pretrained model, scraped dataset, or generative system requiring additional safeguards for controlled deployment.

    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: The paper properly cites the existing models, datasets, and baseline methods used in the experiments, including the Qwen2.5 and Gemma-3 models and the GSM8K, DialogSum, SAMSum, MBPP, and HumanEval datasets, and uses them within their published terms of use.

    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: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, or model checkpoint as a research asset.

    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 experiments 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, crowdsourcing, or data collection requiring IRB approval or equivalent ethical review.

    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 are not part of the core method development; they were used only for grammatical editing and language refinement of the manuscript.

    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.