DISTILLATION WITH ERRORS: EFFICIENT ON-POLICY DISTILLATION VIA SPARSE UNBIASED GRADIENT ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) mitigates the distribution mismatch between training and inference by using teacher supervision on trajectories generated by the student. However, standard full-distribution OPD backpropagates dense gradients over the entire vocabulary, incurring substantial computational cost in the output-head backward pass. To reduce this cost, we introduce **Distillation with Errors (DWE)**, an efficient OPD algorithm based on sparse unbiased gradient estimation. Our key insight is that distillation learns to match the teacher distribution over the course of training, so each step need not use the exact full-vocabulary gradient as long as the full-distribution gradient is preserved in expectation. Based on this insight, DWE exploits the zero-sum structure of the logit gradient to form two sampling distributions from its positive and negative parts. Sampling one coordinate from each distribution yields a paired sparse update whose expectation matches the original dense gradient. To reduce estimation error, DWE uses output-head geometry to guide the pairing. It then applies stratified sampling over K pairs to reduce sampling variance. We theoretically establish the unbiasedness of the resulting estimator. Our experiments show that DWE reduces output-head backward time by up to 96.8% and total output-head time by up to 37.6%. Over 100 training updates, DWE reduces mean recorded training-step time by 2.31% while achieving average performance comparable to standard full-distribution reverse-KL distillation across 13 mathematics, code, and science benchmarks.

1 INTRODUCTION

On-policy distillation (OPD) (Agarwal et al., 2024) has become an effective approach for transferring capabilities from a stronger teacher to a student language model. By using teacher supervision on trajectories generated by the student, OPD mitigates the distribution mismatch between training and inference. However, standard full-distribution OPD backpropagates dense gradients over the entire vocabulary, resulting in substantial computational cost in the output head.

Existing approaches to efficient OPD broadly fall into two categories. Normalized top-k methods (Li et al., 2026) restrict distribution matching to a subset of tokens, while TA-OPD (Huang & Wei, 2026) aggregates the remaining probability mass into a single tail category. These approaches reduce computation but lose information from the full distribution. In contrast, sampling-based methods such as Sparse Logit Sampling (Anshumann et al., 2025) and sample-corrected TA-OPD (Huang & Wei, 2026) randomly select tokens and reweight their contributions so that the estimated gradient matches the full-distribution gradient in expectation. However, sampling only sparsifies the gradient estimation, while the output-head backward pass remains dense. Can we further reduce output-head backward computation while preserving the full-distribution gradient in expectation?

To address this question, we propose **Distillation with Errors (DWE)**, an efficient on-policy distillation algorithm based on sparse unbiased gradient estimation. Our key insight is that distillation aims to match the teacher distribution over the course of training, rather than to reproduce the exact full-vocabulary gradient at every step. This is because each token position provides only a local view of the teacher distribution at the current state, while the student learns from such supervision across many states. This allows the exact per-step gradient to be replaced by a cheaper sparse estimate, provided that the full-distribution gradient is preserved in expectation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Based on this insight, DWE turns each dense logit gradient into a sparse update over only a few vocabulary coordinates. DWE first decomposes the exact gradient into its positive and negative parts. Because the gradient sums to zero, the total positive and negative contributions are equal, so the two parts can be normalized into two sampling distributions. DWE then samples one coordinate from each distribution in proportion to its gradient magnitude, yielding a paired update whose expectation equals the original dense gradient. To reduce sampling error, DWE uses output-head geometry to guide which positive and negative coordinates are paired, and stratified sampling to draw K such pairs. This leaves at most $2K$ nonzero gradient entries, allowing the output-head backward pass to operate only on the corresponding rows.

Our experiments show that DWE reduces OPD training cost while maintaining average downstream performance comparable to full-distribution reverse-KL distillation. At the output head, DWE reduces backward time by up to 96.8% and total forward-and-backward time by up to 37.6%. Over 100 on-policy updates, it reduces mean recorded training-step time by 2.31% relative to Reverse KL. Across 13 mathematics, code, and science benchmarks, DWE achieves an average score of 38.2, compared with 38.9 for Reverse KL and 37.4 for TA-OPD.

Our main contributions are as follows.

1. We propose DWE, an efficient on-policy distillation algorithm based on sparse unbiased gradient estimation. DWE turns each dense logit gradient into a sparse update with at most $2K$ nonzero entries while preserving the full-distribution gradient in expectation. We theoretically establish the unbiasedness of the resulting estimator.
2. We extensively evaluate DWE in terms of efficiency and downstream performance. The results show that DWE reduces output-head backward time by up to 96.8%, total output-head time by up to 37.6%, and mean recorded training-step time by 2.31%, while maintaining average performance comparable to full-distribution reverse-KL distillation across 13 benchmarks.

2 PRELIMINARIES

On-policy distillation. OPD (Agarwal et al., 2024) trains a student policy π_θ to match a fixed teacher π_{T} on responses generated by the student. Let $\mathcal{D}$ denote the prompt distribution. Given a prompt $x \sim \mathcal{D}$, the student autoregressively samples a response $y = (y_1, \ldots, y_T) \sim \pi_\theta(\cdot \mid x)$, where T is the response length. At position t, let $y_{<t} = (y_1, \ldots, y_{t-1})$ be the response prefix generated so far, and write s_t for the context formed by appending this prefix to prompt x. At this context, the student and teacher induce next-token distributions $p_t(v) = \pi_\theta(v \mid s_t)$ and $q_t(v) = \pi_{\mathrm{T}}(v \mid s_t)$ over their shared vocabulary $\mathcal{V}$, whose size is V.

Let ℓ_t denote the token-level divergence used for distillation. For a sampled response, OPD minimizes the average token-level loss

$$\mathcal{L}(\theta; x, y) = \frac{1}{T} \sum_{t=1}^{T} \ell_t. \tag{1}$$

Recent OPD methods, including normalized top-k OPD (Li et al., 2026) and TA-OPD (Huang & Wei, 2026), commonly use reverse KL. We use it below to analyze the output-head computation addressed by DWE. The same algorithm also applies to forward KL. The token-level reverse-KL loss is

$$\ell_t = D_{\mathrm{KL}}(p_t \| q_t) = \sum_{v \in \mathcal{V}} p_t(v) \log \frac{p_t(v)}{q_t(v)}. \tag{2}$$

Output-head gradients. At context s_t, the student maps its hidden state $h_t \in \mathbb{R}^d$ to logits $z_t = W h_t$ with the linear output head $W \in \mathbb{R}^{V \times d}$, and obtains $p_t = \mathrm{softmax}(z_t)$. For reverse KL, let $g_t = \nabla_{z_t} \ell_t$. The component at vocabulary item v is

$$g_{t,v} = p_t(v) \left(\log \frac{p_t(v)}{q_t(v)} - \ell_t \right). \tag{3}$$

By the chain rule, this logit gradient propagates to the hidden state and output weights as

$$\nabla_{h_t} \ell_t = W^\top g_t, \qquad \nabla_W \ell_t = g_t h_t^\top. \tag{4}$$

For N response-token positions, computing these dense output-head products requires $O(NVd)$ arithmetic. DWE reduces this cost by replacing the dense logit gradients with sparse estimates.

3 DISTILLATION WITH ERRORS

DWE is an efficient on-policy distillation algorithm that reduces output-head backward computation by replacing dense logit gradients with sparse estimates. Our key insight is that each update need not use the exact full-vocabulary gradient, as long as the estimate matches it in expectation. DWE therefore constructs a sparse estimate for each token position and backpropagates only through the selected output rows.

DWE constructs this estimate in three steps. First, it decomposes the zero-sum logit gradient into positive and negative components and represents it with sparse paired updates. Second, it uses output-head geometry to guide the pairing, targeting the error propagated to the hidden state. Third, it uses stratified sampling to draw K pairs and control sampling variance. Averaging these updates yields at most $2K$ nonzero coordinates, enabling sparse output-head backpropagation.

Figure 1 illustrates the procedure. We consider one token context s and write $g = \nabla_z \ell \in \mathbb{R}^V$ for the exact logit gradient and $\widehat{g}$ for its estimate.

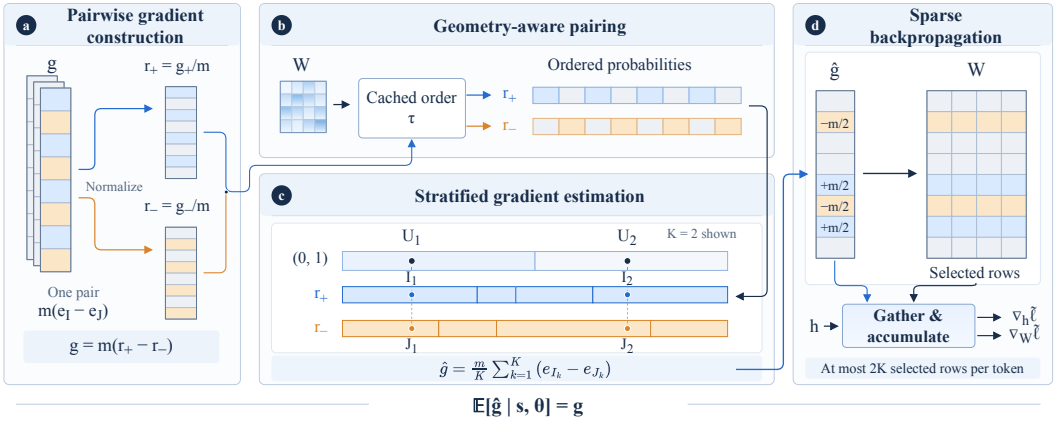

Figure 1: Overview of DWE. The positive and negative parts of the logit gradient are normalized into two sampling distributions. Output-head geometry guides their pairing, and stratified sampling selects K pairs, yielding at most $2K$ active output rows per token. The resulting estimator matches the full gradient in expectation. The illustration uses $K = 2$. Blue and orange denote positive and negative contributions, respectively.

3.1 PAIRWISE GRADIENT CONSTRUCTION

Pairwise gradient construction turns each dense logit gradient into a two-sparse random update while preserving the original gradient in expectation. The construction relies on the zero-sum property of softmax gradients.

Lemma 1 (Zero-sum logit gradients)**.** *Let $p = \mathrm{softmax}(z)$ and let $\ell(q, p)$ be differentiable in p, with q fixed. Let $g = (g_1, \ldots, g_V)^\top = \nabla_z \ell(q, p) \in \mathbb{R}^V$ denote the logit gradient. Then*

$$\sum_{v=1}^{V} g_v = 0.$$

The property follows from the shift invariance of softmax: adding the same constant to every logit leaves the output distribution unchanged. The proof is given in Appendix A.1.

We write each coordinate's positive contribution as $(g_+)_v = \max(g_v, 0)$ and its negative magnitude as $(g_-)_v = \max(-g_v, 0)$. These vectors satisfy $g = g_+ - g_-$. By Lemma 1, their total magnitudes are equal. Since $|g_v| = (g_+)_v + (g_-)_v$, this common magnitude is

$$m = \sum_v (g_+)_v = \sum_v (g_-)_v = \frac{1}{2}\|g\|_1, \tag{5}$$

where $\|g\|_1 = \sum_v |g_v|$ is the total absolute gradient magnitude. For $g \neq 0$, normalizing the two parts gives

$$r_+ = \frac{g_+}{m}, \qquad r_- = \frac{g_-}{m}. \tag{6}$$

Both vectors are nonnegative and sum to one. Thus, r_+ and r_- are probability distributions over the positive and negative coordinates, respectively.

Consider random coordinates I and J satisfying $I \sim r_+$ and $J \sim r_-$. Assigning $+m$ to I and $-m$ to J gives the two-sparse update $m(e_I - e_J)$, where e_i is the i-th standard basis vector. For any vocabulary coordinate v,

$$\begin{aligned} \mathbb{E}[(m(e_I - e_J))_v] &= m \Pr(I = v) - m \Pr(J = v) \\ &= m\big((r_+)_v - (r_-)_v\big) \\ &= (g_+)_v - (g_-)_v = g_v. \end{aligned} \tag{7}$$

Thus, each sampled pair contains only two nonzero coordinates, and its expectation matches the original dense gradient. For $g = 0$, we use the zero update. However, a single sampled update can still contain substantial stochastic error.

3.2 Geometry-Aware Pairing

Geometry-aware pairing uses output-head geometry to choose which positive and negative coordinates are sampled together, with the aim of reducing hidden-state gradient error.

The distributions r_+ and r_- determine how often each coordinate is selected, but they do not determine how positive and negative coordinates are paired. We describe their joint sampling by a coupling.

Definition 1 (Coupling)**.** A coupling γ of r_+ and r_- is a joint distribution of (I, J) whose marginals satisfy $I \sim r_+$ and $J \sim r_-$.

Any coupling with these marginals preserves the expected logit gradient in Eq. (7). Different couplings, however, can produce different hidden-state gradient errors after the sparse update is propagated through the output head.

Let $w_i^\top$ denote row i of the output-head matrix W. For a sampled pair $(I, J) \sim \gamma$, the sparse update produces the hidden-state gradient

$$W^\top m(e_I - e_J) = m(w_I - w_J).$$

The exact hidden-state gradient is $W^\top g$. Since the sparse logit update is unbiased, its expected hidden-state gradient is also $W^\top g$. Its mean-squared error is therefore

$$\begin{aligned} \mathbb{E}_\gamma\left[\left\|m(w_I - w_J) - W^\top g\right\|_2^2\right] &= m^2 \mathbb{E}_\gamma\left[\|w_I - w_J\|_2^2\right] \\ &\quad - \|W^\top g\|_2^2. \end{aligned} \tag{8}$$

For fixed marginals, the second term is independent of the coupling. Therefore, reducing the expected squared distance between paired output rows reduces the hidden-state gradient error.

DWE approximates this distance-based pairing with a cached one-dimensional ordering of the vocabulary. At initialization, we center the student output-head rows and estimate their leading principal direction. We project each row onto this direction and sort the vocabulary by its projection score. The resulting order τ provides a one-dimensional proxy for the output-head geometry.

Let $F_{r_+,\tau}$ and $F_{r_-,\tau}$ denote the cumulative distributions of r_+ and r_- along this order. We use the same quantile $u \in (0, 1)$ for both distributions:

$$I(u) = \tau\left(F_{r_+,\tau}^{-1}(u)\right), \qquad J(u) = \tau\left(F_{r_-,\tau}^{-1}(u)\right). \tag{9}$$

This construction pairs positive and negative coordinates at the same cumulative-probability level along the geometry-derived order.

For $U \sim \mathrm{Unif}(0,1)$, the length of each coordinate's CDF interval equals its probability mass. Hence, $I(U) \sim r_+$ and $J(U) \sim r_-$. The shared-quantile construction therefore defines a valid coupling while preserving the original marginals and expected gradient.

Even with geometry-aware pairing, a finite number of sampled pairs still introduces sampling variance.

3.3 Stratified Gradient Estimation

Stratified gradient estimation reduces this remaining variance by spreading K draws across the full quantile interval instead of sampling all pairs independently from the same interval.

Using the geometry-aware pairing above, define the paired update $f(u) = m(e_{I(u)} - e_{J(u)})$. For $U \sim \mathrm{Unif}(0,1)$, the shared-quantile coupling preserves the marginals of r_+ and r_-, so the dense gradient can be written as

$$g = \mathbb{E}[f(U)] = \int_0^1 f(u)\,\mathrm{d}u. \tag{10}$$

To estimate this integral with K pairs, DWE divides the quantile interval into K equal strata. It draws one quantile independently from each stratum:

$$U_k \sim \mathrm{Unif}\left(\frac{k-1}{K}, \frac{k}{K}\right), \qquad k = 1, \ldots, K. \tag{11}$$

Each quantile determines a pair $(I_k, J_k) = (I(U_k), J(U_k))$. Averaging the corresponding updates gives the DWE estimator

$$\widehat{g} = \frac{m}{K} \sum_{k=1}^{K} (e_{I_k} - e_{J_k}). \tag{12}$$

Each realization contains at most $2K$ nonzero coordinates, and each paired update sums to zero. We prove in Appendix A that the resulting estimator is unbiased for the full gradient. We also show that, for a fixed coupling and sample budget, stratified sampling has no larger covariance than independent sampling.

3.4 Sparse Backpropagation

Sparse backpropagation converts the $2K$-sparse gradient estimate into actual output-head computation savings. Because $\widehat{g}$ contains at most $2K$ nonzero coordinates, both output-head backward products involve only the corresponding output rows.

Substituting $\widehat{g}$ for the dense logit gradient gives

$$W^\top \widehat{g} = \frac{m}{K} \sum_{k=1}^{K} (w_{I_k} - w_{J_k}), \qquad \widehat{g} h^\top = \frac{m}{K} \sum_{k=1}^{K} (e_{I_k} - e_{J_k}) h^\top. \tag{13}$$

The first expression computes the hidden-state gradient by gathering only the selected rows of W. The second updates only the corresponding rows of the output-head gradient.

DWE still evaluates the full OPD objective and its exact logit gradient before sampling. The computational savings come from using the sparse estimate in these output-head backward products.

The unbiasedness of $\widehat{g}$ also carries to the student parameters. For a fixed context and parameter state, let $J_\theta = \partial z / \partial \theta$ denote the logit Jacobian. Backpropagation with $\widehat{g}$ gives the parameter gradient $J_\theta^\top \widehat{g}$. By linearity,

$$\mathbb{E}[J_\theta^\top \widehat{g} \mid s, \theta] = J_\theta^\top \mathbb{E}[\widehat{g} \mid s, \theta] = J_\theta^\top g = \nabla_\theta \ell. \tag{14}$$

The same result extends by linearity to a fixed minibatch. The corresponding proof is provided in Appendix A.

3.5 Complexity Analysis

The computational saving comes from the output-head backward products. Across N supervised positions, dense hidden-state and output-weight gradients require $O(NVd)$ arithmetic. Since DWE uses at most $2K$ output rows per position, the corresponding sparse computation requires $O(NKd)$. The two dense products use $2NVd$ multiply-adds, compared with at most $4NKd$ for DWE, giving a reduction factor of $V/(2K)$ for the output-head backward products.

Both teacher and student still evaluate their full-vocabulary output distributions, and constructing the sampling distributions requires vocabulary-wide operations. The end-to-end gain therefore depends on how much of the training cost is spent in the output-head backward pass.

4 Experiments

Our experiments evaluate whether DWE reduces the computational cost of OPD while preserving distillation performance. We examine output-head and end-to-end training efficiency, downstream performance, and the accuracy–cost tradeoff controlled by the sampling budget K.

4.1 Experimental Setup

Models and data. We use Qwen3-1.7B as the student and Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the teacher, with thinking disabled throughout training and evaluation. Training uses approximately 100k prompts sampled equally from OpenMathReasoning (Moshkov et al., 2025), OpenCodeReasoning (Ahmad et al., 2025), and OpenScienceReasoning-2 (NVIDIA, 2025). We use only the prompts from these datasets and generate student responses on-policy during training.

Methods. We compare DWE with a standard full-distribution baseline and a representative efficient OPD method. *Reverse KL* serves as the full-distribution baseline, computing the exact reverse-KL objective and its dense logit gradient over the entire vocabulary. *TA-OPD* (Huang & Wei, 2026) reduces computation by matching the teacher's top-k tokens together with a single category for the remaining probability mass, with $k = 16$. DWE uses the same full-distribution objective as Reverse KL but constructs a sparse gradient estimate from $K = 4$ sampled pairs per token for output-head backpropagation. All three methods start from the same student checkpoint and are trained on the same prompts for 100 updates with a global batch size of 256 on two NVIDIA H100 GPUs. Appendix B provides configuration details and component ablations.

Evaluation. We evaluate the final checkpoints on 13 benchmarks covering mathematics, code, and science, with the initial student and teacher as references. For mathematics, we use AIME25 (MAA, 2025), AMC23 (MAA, 2023), HMMT25 (MathArena, 2025), MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), OlympiadBench (He et al., 2024), and ReasoningGym Math (Stojanovski et al., 2025). For code, we use CodeContests (Li et al., 2022), Codeforces (Open-R1 Team, 2025), LiveCodeBench v5 (Jain et al., 2024), and ReasoningGym Algorithm (Stojanovski et al., 2025). For science, we use GPQA Diamond (Rein et al., 2023), SuperGPQA (P Team et al., 2025), and SciBench (Wang et al., 2023). We report single-answer accuracy under matched decoding settings and native task scores for ReasoningGym.

4.2 Output-Head Efficiency

We measure output-head forward and backward time on fixed student-generated responses from the Reverse KL checkpoint at update 100, comparing Reverse KL with DWE at 1K, 4K, and 8K token positions per GPU. Each configuration uses 10 warmup iterations followed by 50 timed repetitions, measured with CUDA events. Table 1 reports the average runtimes and the resulting speedup.

DWE reduces output-head backward time by **90.3%** at 1K tokens and **96.8%** at 8K tokens. After accounting for gradient construction and sampling overhead, total output-head speedups are $\mathbf{1.47}\times$, $\mathbf{1.57}\times$, and $\mathbf{1.60}\times$ at 1K, 4K, and 8K tokens, respectively. At 8K tokens, total time falls by **37.6%**. The larger relative savings at higher token counts are consistent with amortizing sparse-backward overhead over more positions.

Table 1: Output-head runtime and speedup.

	Reverse KL			DWE			
Tokens	Forward	Backward	Total	Forward	Backward	Total	**Speedup**
1K	9.84	16.9	26.7	16.5	**1.63 (−90.3%)**	18.1	**1.47×**
4K	37.5	69.6	107	65.1	**3.13 (−95.5%)**	68.2	**1.57×**
8K	74.2	141	215	130	**4.56 (−96.8%)**	134	**1.60×**

Runtimes are in milliseconds. Forward time includes teacher and student output projections, loss computation, and, for DWE, gradient construction and sampling. Parentheses give backward-time reductions relative to Reverse KL, computed before rounding. Speedup uses unrounded total forward-and-backward time (1K = 1,024).

4.3 Training Efficiency

We examine whether the output-head savings translate into lower training-step cost under the setup in Section 4.1. Over 100 on-policy updates, we record training-step time, response-token throughput, and the full-distribution reverse KL. Table 2 summarizes the recorded training-step time.

Table 2: Recorded training-step time over 100 updates.

Method	**Time/update** (s)	**Cumulative time** (h)	**Reduction** (%)
Reverse KL	361	10.0	–
TA-OPD	360	9.99	0.42
DWE	**353**	**9.80**	**2.31**

Timing covers all 100 training updates, including the 10 learning-rate warmup updates, and excludes optimizer updates and checkpoint saving outside the recorded training-step calls. Reductions and their ratio are computed from unrounded times relative to Reverse KL.

DWE reduces mean recorded training-step time from 361 s to 353 s, a **2.31%** reduction, compared with 0.42% for TA-OPD. This time saving is $5.44\times$ that of TA-OPD.

Figure 2 compares the response-token throughput and full-distribution reverse KL of DWE and Reverse KL over the 100 updates.

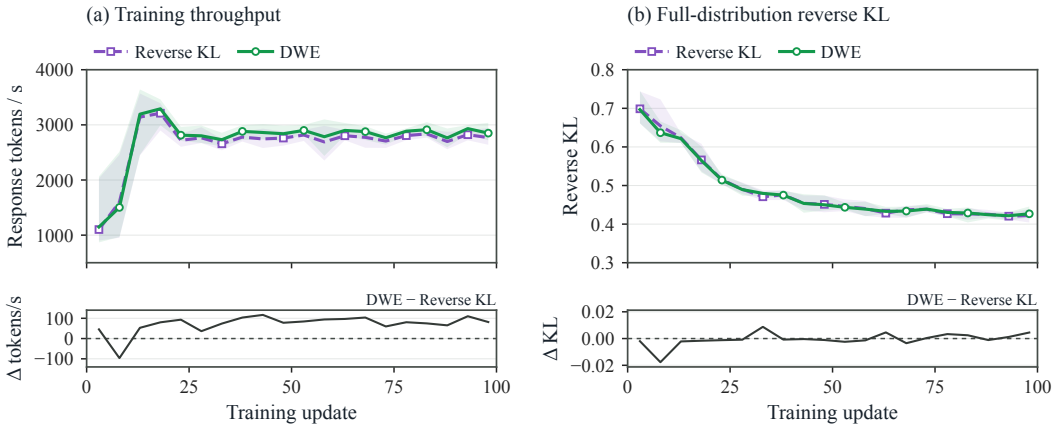

Figure 2: Response-token throughput and full-distribution reverse KL for DWE and Reverse KL over 100 on-policy updates. Each method is evaluated on its own on-policy responses. Curves aggregate consecutive blocks of five updates, with shading showing the within-block range. Lower panels show DWE minus Reverse KL at matching blocks.

DWE achieves an aggregate response-token throughput of **2,790 tokens/s**, compared with 2,720 for Reverse KL. These efficiency gains are accompanied by similar loss trajectories for DWE and Reverse KL, with mean reverse KL values of 0.424 and 0.421, respectively, over the final ten updates.

4.4 Distillation Performance

We assess whether the computational savings preserve downstream performance using the final checkpoints after 100 updates. Table 3 compares the three methods on the 13 benchmarks, alongside the initial student and teacher.

Table 3: Distillation performance across 13 reasoning benchmarks. Higher is better.

Benchmark	Reference models		Distillation baselines		Ours
	Initial	Teacher	Reverse KL	TA-OPD	DWE
MATH500	69.6	90.2	**82.5**	82.0	**82.4**
AIME25	0.0	36.7	**25.3**	22.0	**24.0**
AMC23	40.0	92.5	73.0	70.5	**74.5**
HMMT25	3.3	36.7	16.7	13.3	**20.0**
OlympiadBench	24.3	44.2	36.4	36.9	**37.5**
RG-Math	83.7	93.6	**91.7**	90.8	**90.0**
CodeContests	2.4	18.2	9.1	**9.7**	**6.7**
Codeforces	1.5	5.7	**6.4**	1.8	**5.3**
LiveCodeBench v5	5.4	22.8	**19.0**	17.6	**16.0**
RG-Algorithm	19.5	63.9	**36.4**	28.4	**32.4**
GPQA Diamond	35.9	59.1	44.4	**49.5**	**40.4**
SuperGPQA	26.3	37.8	28.5	28.8	**29.8**
SciBench	26.4	57.4	36.6	35.3	**37.3**
Average	26.0	50.7	**38.9**	37.4	**38.2**

Distilled-student scores on MATH500, AIME25, and AMC23 average five decoding seeds. All other entries, including initial-student and teacher scores, use one seed. Scores are on a 0–100 scale, using native graded scores for ReasoningGym (RG) and accuracy for other benchmarks. Best distilled-student scores are shaded gray for baselines and darker blue for DWE. The blue column identifies DWE. Average is the unweighted mean over all 13 benchmarks.

Across the 13 benchmarks, DWE achieves an average score of **38.2**, compared with 38.9 for Reverse KL and 37.4 for TA-OPD. From an initial-student average of 26.0, DWE gains **12.1 points**, compared with 12.9 points for Reverse KL. It therefore retains most of the average gain from full-distribution distillation while reducing recorded training-step time.

DWE improves over the initial student on all 13 benchmarks and achieves the highest distilled-student score on **five benchmarks** across mathematics and science. It closely matches Reverse KL on MATH500 (82.4 versus 82.5) and exceeds TA-OPD by **6.7 points on HMMT25** (20.0 versus 13.3) and **4.0 points on RG-Algorithm** (32.4 versus 28.4).

4.5 Sampling Budget

The preceding experiments use $K = 4$. We vary $K \in \{1, 2, 4, 8, 16\}$ to evaluate gradient estimation accuracy and output-head cost (Table 4). We use the initial student and 1,024 fixed response positions per domain, keeping model weights and vocabulary ordering fixed. We compare estimated hidden-state and output-weight gradients with their exact counterparts using relative MSE, $\|\widehat{A} - A\|_F^2 / \|A\|_F^2$, where A is the full-distribution gradient, $\widehat{A}$ its estimate, and $\|\cdot\|_F$ the Frobenius norm. Runtimes average 100 measurements after 10 warmups.

Increasing K consistently reduces both gradient errors. From $K = 1$ to $K = 4$, hidden-state and output-weight relative MSE fall from 0.194 and 0.183 to 0.0219 and 0.0210, respectively, while total output-head time stays near 18.2 ms. Increasing K further to 16 lowers the errors but raises backward time from 1.67 ms to 2.64 ms and total time to 19.2 ms. Thus, K controls the tradeoff

Table 4: Gradient accuracy and output-head cost across sampling budgets.

	Relative MSE ($\times 10^{-2}$)		**Time (ms)**	
Pairs K	Hidden state	Output weights	Backward	Total
1	19.4	18.3	1.62	18.3
2	6.59	6.29	1.65	18.3
4 (default)	2.19	2.10	1.67	18.2
8	0.689	0.660	2.05	18.7
16	0.212	0.201	2.64	19.2

Errors are means over 128 random seeds after averaging the three domains. Total time includes teacher and student forward projections, gradient construction, sampling, and backward computation.

between gradient accuracy and computational cost. We use $K = 4$ in this work, achieving good distillation performance at low computational cost.

5 Related Work

Existing methods for improving OPD efficiency mainly fall into two categories: vocabulary truncation and sampling.

Vocabulary truncation. Normalized top-k OPD (Li et al., 2026) retains only a subset of high-probability tokens for distribution matching. TA-OPD (Huang & Wei, 2026) aggregates the remaining probability mass into a single tail category. Teacher Top-K Local Support Matching (Fu et al., 2026) restricts reverse-KL matching to the teacher's top-K local support. These methods reduce computation by involving only part of the vocabulary, but lose information from the full distribution. DWE also uses only a few vocabulary entries in each output-head backward pass, but recovers the full-distribution gradient in expectation.

Sampling. Sampling-based methods select a subset of tokens and reweight their contributions to estimate the distillation gradient. Sparse Logit Sampling (Anshumann et al., 2025) uses importance-weighted samples from the teacher distribution. Sample-corrected TA-OPD (Huang & Wei, 2026) samples from the student distribution to estimate the residual tail contribution. Both can preserve the full-distribution gradient in expectation, but leave the student output-head backward pass dense. DWE instead samples from the logit gradient itself, so each update backpropagates through at most $2K$ selected output rows.

Limitations

DWE preserves the full-distribution gradient only in expectation. Individual updates therefore remain stochastic and can follow a different optimization trajectory from full-distribution distillation. Increasing K reduces the variance of each gradient estimate. Using more OPD steps also gives the zero-mean sampling errors more opportunities to average out across training.

DWE reduces computation only in the output-head backward pass. The full teacher and student distributions are still required, while generation, output-head forward computation, and backbone computation remain unchanged. End-to-end gains are therefore larger when this backward pass accounts for a substantial fraction of training cost, as in large-vocabulary OPD with long sequences or large token batches.

6 Conclusion

In this work, we introduced Distillation with Errors (DWE), an efficient on-policy distillation method that reduces output-head backward computation through sparse updates while preserving the full-distribution gradient in expectation. Our experiments demonstrate that DWE effectively reduces training cost while maintaining performance comparable to full-distribution distillation, providing a practical solution for efficient OPD.

AI USE STATEMENT

We used generative AI tools to refine our research ideas, assist with method implementation, and draft and polish the manuscript throughout the research process. All AI-generated outputs were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, et al. OpenCodeReasoning: Advancing data distillation for competitive coding. *arXiv preprint arXiv:2504.01943*, 2025. URL https://arxiv.org/abs/2504.01943.

Anshumann, Mohd Abbas Zaidi, Akhil Kedia, Jinwoo Ahn, Taehwak Kwon, Kangwook Lee, Haejun Lee, and Joohyung Lee. Sparse logit sampling: Accelerating knowledge distillation in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18085–18108, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.885. URL https://aclanthology.org/2025.acl-long.885/.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026. doi: 10.48550/arXiv.2603.25562. URL https://arxiv.org/abs/2603.25562.

Chaoqun He, Renjie Luo, Yuzhuo Bai, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-policy delta distillation. *arXiv preprint arXiv:2607.15161*, 2026. URL https://arxiv.org/abs/2607.15161.

Huipeng Huang and Hongxin Wei. Tail-Aware Top-k On-Policy Distillation. *arXiv preprint arXiv:2608.14728*, 2026. URL https://arxiv.org/abs/2608.14728.

Naman Jain, King Han, Alex Gu, et al. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026. doi: 10.48550/arXiv.2604.13016. URL https://arxiv.org/abs/2604.13016v2.

Yujia Li, David Choi, Junyoung Chung, et al. Competition-level code generation with AlphaCode. *arXiv preprint arXiv:2203.07814*, 2022. URL https://arxiv.org/abs/2203.07814.

Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

MAA. American Mathematics Competitions (AMC), 2023, 2023. URL https://maa.org/student-programs/amc/.

MAA. American Invitational Mathematics Examination (AIME), 2025, 2025. URL https://maa.org/maa-invitational-competitions/.

MathArena. HMMT February 2025 Benchmark, 2025. URL https://huggingface.co/datasets/MathArena/hmmt_feb_2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, et al. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025. URL https://arxiv.org/abs/2504.16891.

NVIDIA. OpenScienceReasoning-2, 2025. URL https://huggingface.co/datasets/nvidia/OpenScienceReasoning-2.

Open-R1 Team. Codeforces Dataset, 2025. URL https://huggingface.co/datasets/open-r1/codeforces.

P Team, Xinrun Du, Yifan Yao, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025. URL https://arxiv.org/abs/2502.14739.

David Rein, Betty Li Hou, Asa Cooper Stickland, et al. GPQA: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, et al. REASONING GYM: Reasoning environments for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025. URL https://arxiv.org/abs/2505.24760.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, et al. SciBench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023. URL https://arxiv.org/abs/2307.10635.

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

A Proofs and Derivations

The sparse construction in Section 3 relies on the zero-sum logit gradient and preserves its expectation under paired sampling. We derive the required gradient identities, prove unbiasedness, and establish the variance reduction from stratification. Throughout this section, the prefix s, student parameters θ, and teacher distribution are fixed. Expectations are over the sampling randomness, and covariance comparisons use the same coupling and budget K.

A.1 Supporting Derivations

The zero-sum property ensures that positive and negative gradient mass can be paired. We first prove this property for differentiable softmax losses, then derive the reverse- and forward-KL gradients used by DWE.

Proof of Lemma 1. Adding a common constant c to the logits leaves the softmax probabilities unchanged. With q fixed, the loss $\ell(q, \mathrm{softmax}(z + c\mathbf{1}))$ is therefore constant in c. Differentiating gives

$$0 = \frac{\mathrm{d}}{\mathrm{d}c}\ell(q, \mathrm{softmax}(z + c\mathbf{1}))\Big|_{c=0} = \sum_{v=1}^{V} g_v, \tag{15}$$

where $\mathbf{1}$ is the all-ones vector. □

Reverse KL. For $\ell = D_{\mathrm{KL}}(p\|q) = \sum_j p_j \log(p_j/q_j)$, we have $\partial\ell/\partial p_j = \log(p_j/q_j) + 1$. Using the softmax derivative $\partial p_j/\partial z_i = p_j(\delta_{ij} - p_i)$, where δ_{ij} is the Kronecker delta, gives

$$\begin{aligned} g_i &= \sum_j p_j(\delta_{ij} - p_i)\left(\log\frac{p_j}{q_j} + 1\right) \\ &= p_i\left(\log\frac{p_i}{q_i} + 1\right) - p_i\sum_j p_j\left(\log\frac{p_j}{q_j} + 1\right) \\ &= p_i\left(\log\frac{p_i}{q_i} - \ell\right). \end{aligned} \tag{16}$$

This is the logit gradient used in Eq. (3).

Forward KL. For $\ell = D_{\mathrm{KL}}(q\|p)$, the same softmax derivative gives $g_i = -\sum_j q_j(\delta_{ij} - p_i) = p_i - q_i$. Thus $\sum_i g_i = 0$, and the common positive and negative mass is $m = \frac{1}{2}\sum_i |p_i - q_i| = D_{\mathrm{TV}}(p, q)$. The sampling construction therefore applies to forward KL as well.

A.2 Unbiasedness

The paired update in Section 3.1 is unbiased for any valid coupling. Here we show that the shared-quantile construction and stratification preserve this property, and extend the result to minibatch parameter gradients.

Theorem 1 (Unbiasedness)**.** *Conditional on the context s and current parameters θ, the estimator in Eq.* (12) *satisfies*

$$\mathbb{E}[\widehat{g} \mid s, \theta] = g. \tag{17}$$

Proof. Fix a vocabulary order τ. The generalized inverse CDF $F^{-1}(u) = \min\{j : F(j) \geq u\}$ returns a position in the ordered vocabulary. If token i has ordered position j, it is selected by $I(U) = \tau(F^{-1}_{r_+,\tau}(U))$ precisely when U lies in the corresponding CDF interval, whose length is $(r_+)_i$. Hence $\Pr(I(U) = i) = (r_+)_i$ for $U \sim \mathrm{Unif}(0, 1)$; likewise, $\Pr(J(U) = i) = (r_-)_i$. Zero-probability entries have empty intervals, and the choice of interval endpoints has probability zero. The shared quantile therefore defines a coupling with the required marginals. For $f(u) = m(e_{I(u)} - e_{J(u)})$,

$$\int_0^1 f(u)\,\mathrm{d}u = m\big(\mathbb{E}[e_{I(U)}] - \mathbb{E}[e_{J(U)}]\big) = m(r_+ - r_-) = g. \tag{18}$$

Let $A_k = ((k-1)/K, k/K)$. Each interval has length $1/K$, so a uniform sample U_k within A_k satisfies $\mathbb{E}[f(U_k)] = K \int_{A_k} f(u)\,\mathrm{d}u$. Averaging across the K strata yields

$$\begin{aligned}\mathbb{E}[\widehat{g}] &= \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[f(U_k)] \\ &= \sum_{k=1}^{K}\int_{A_k} f(u)\,\mathrm{d}u = g.\end{aligned} \tag{19}$$

The equality holds for the average over strata; individual strata need not preserve the full marginals. When $g = 0$, the estimator is zero and the result holds directly. □

Minibatch gradients. Let $\mathcal{B}$ be a fixed minibatch of sampled prompt-response pairs, and let ω denote DWE's sampling randomness. Write $\widehat{\nabla_\theta \mathcal{L}_\mathcal{B}}$ for the parameter gradient formed using the DWE estimator and $\mathcal{L}_\mathcal{B}$ for the full-distribution OPD loss on this batch. The token-loss normalization and minibatch weights are fixed given $\mathcal{B}$, so Eq. (14) and linearity give

$$\mathbb{E}_\omega\left[\widehat{\nabla_\theta \mathcal{L}_\mathcal{B}} \mid \mathcal{B}, \theta\right] = \nabla_\theta \mathcal{L}_\mathcal{B}. \tag{20}$$

A.3 Variance Reduction

We compare stratified sampling with independent draws from the same coupling at the same budget K. The variance reduction follows from eliminating the randomness in how many samples fall in each stratum.

Theorem 2 (Variance reduction)**.** *For a fixed coupling and sample budget K, let $\widehat{g}_{\mathrm{strat}}$ denote the estimator in Eq.* (12)*, and let $\widehat{g}_{\mathrm{iid}}$ average K independent uniform quantile draws. Then*

$$\mathrm{Cov}(\widehat{g}_{\mathrm{strat}} \mid s, \theta) \preceq \mathrm{Cov}(\widehat{g}_{\mathrm{iid}} \mid s, \theta). \tag{21}$$

Here $\preceq$ denotes the positive-semidefinite order.

Proof. Let $X = f(U)$ for $U \sim \mathrm{Unif}(0,1)$, and let S index the stratum containing U. Each stratum has probability $1/K$. Define its conditional mean and covariance as

$$\begin{aligned}\mu_k &= \mathbb{E}[X \mid S = k], \\ \Sigma_k &= \mathrm{Cov}(X \mid S = k).\end{aligned}$$

The law of total covariance separates variation within strata from variation between their means:

$$\mathrm{Cov}(X) = \frac{1}{K}\sum_{k=1}^{K}\Sigma_k + \mathrm{Cov}(\mu_S). \tag{22}$$

For K independent draws from the full coupling, averaging divides this covariance by K. Stratification instead takes one independent draw X_k from the conditional distribution of X given $S = k$ in each stratum. The two estimators satisfy

$$\begin{aligned}\mathrm{Cov}(\widehat{g}_{\mathrm{iid}}) &= \frac{1}{K^2}\sum_{k=1}^{K}\Sigma_k + \frac{1}{K}\mathrm{Cov}(\mu_S), \\ \mathrm{Cov}(\widehat{g}_{\mathrm{strat}}) &= \frac{1}{K^2}\sum_{k=1}^{K}\Sigma_k.\end{aligned}$$

Subtracting gives the covariance improvement

$$\mathrm{Cov}(\widehat{g}_{\mathrm{iid}}) - \mathrm{Cov}(\widehat{g}_{\mathrm{strat}}) = \frac{1}{K}\mathrm{Cov}(\mu_S) \succeq 0. \tag{23}$$

Thus stratification has no larger covariance than iid sampling from the same coupling. It removes the component caused by random allocation of samples across strata. □

B Experimental Details and Additional Results

This appendix specifies the implementation and evaluation settings behind Section 4 and examines the contributions of DWE's two variance-reduction components.

B.1 Experimental Details

Training configuration. The runs in Section 4.1 use training seed 42. The learning rate reaches 5×10^{-6} after 10 warmup updates, then follows cosine decay to one tenth of this value. Student responses are sampled at temperature 0.7 with at most 8,192 generated tokens. Training uses BF16 and DeepSpeed ZeRO-2. DWE processes 256 token positions per chunk and retains the initial vocabulary order throughout training. With output-head weights replicated across ranks, selected-row contributions are accumulated locally in a dense weight-gradient buffer before standard gradient synchronization. TA-OPD uses a tail-probability floor of 10^{-6}.

Evaluation protocol. We evaluate the models using benchmark adapters from OPD[2] (Heo et al., 2026) (`2dac53f`)[1] and Evalchemy (`ce5cea9`).[2] We use their benchmark prompts and scoring rules with matched settings across models and the coverage described below. Generation uses temperature 0.2 for CodeContests and 0.7 for other benchmarks, with a limit of 32,768 new tokens and thinking disabled. Our decoding-repeat schedule uses seeds 0–4 for trained checkpoints on MATH500, AIME25, and AMC23, and seed 0 for other entries. Each repeat generates one answer per question, and reported scores average single-answer correctness across repeats. Each method has one training run.

Benchmark coverage. For SuperGPQA, we select 50 questions per discipline with selection seed 0, giving 650 questions across 13 disciplines. For RG-Math and RG-Algorithm, we use five tasks each, with 20 generated instances per task at data seed 0. These question-selection settings are shared across models. GPQA Diamond uses all 198 questions in the Diamond split, and HMMT25 uses the February 2025 competition. For the code tasks, Codeforces uses 453 non-interactive problems with editorials removed from the prompts. CodeContests evaluates all 165 test problems against their public, private, and generated tests. SciBench includes all 692 problems, with numeric-answer formatting normalized before scoring. ReasoningGym retains its native graded scores rather than binary accuracy.

Training statistics. To construct Figure 2, we group consecutive, non-overlapping blocks of five updates. Block throughput is total response-token count divided by total recorded step time, and block KL is the mean of the five losses. Shaded bands show the range of individual updates within each block. Difference curves subtract Reverse KL from DWE at matching blocks. Since the two methods generate their own responses, these differences include variation in the generated sequences.

B.2 Component Ablation

Section 4.5 studies the effect of the sampling budget. Here we fix $K = 4$ to determine how geometry-aware pairing and stratified sampling each affect gradient error. We evaluate alternative estimators on the same states from the final DWE checkpoint, so differences reflect the sampling construction at a fixed model state. The geometry order is the original one cached during training. The checkpoint generates eight responses per domain at temperature 0.7, capped at 512 tokens each. We select 1,024 evenly spaced response-token positions per domain and hold the resulting states and weights fixed across configurations. Relative MSE follows Section 4.5 and is averaged over 128 sampling seeds and then equally across the three domains.

Table 5 compares independent positive and negative draws with shared-quantile sampling under random and geometry-derived orders. The random order is one fixed permutation. For shared-quantile sampling, we either draw K independent uniform quantiles or draw one quantile from each of K equal strata.

[1] `https://github.com/naver-ai/opd2/tree/2dac53f`

[2] `https://github.com/mlfoundations/Evalchemy/tree/ce5cea9`

Table 5: Component ablation at $K = 4$ on fixed states from the final DWE checkpoint.

		Relative MSE ($\times 10^{-2}$)	
Pairing	**Sampling**	Hidden state	Output weights
Independent marginals	i.i.d.	7.66	7.24
Random order	i.i.d.	7.66	7.19
Geometry order	i.i.d.	7.62	7.25
Random order	Stratified	3.65	3.35
Geometry order (DWE)	Stratified	3.55	3.35

Stratification provides most of the error reduction on these fixed states. With geometry fixed, it lowers hidden-state and output-weight relative MSE by 53.4% and 53.8%. Under stratified sampling, geometry further reduces hidden-state error by 2.76% relative to the fixed random order, with nearly unchanged output-weight error.