See Every Block Once: Variance-Reduced Post-Training for Masked Diffusion Language Models

Yunzhen He[1] **Hiroaki Yamagiwa**[2*] **Hidetoshi Shimodaira**[1,3]
[1] Kyoto University [2] SB Intuitions [3] RIKEN AIP
he.yunzhen.25d@st.kyoto-u.ac.jp,
hiroaki.yamagiwa@sbintuitions.co.jp, shimo@i.kyoto-u.ac.jp

Abstract

Policy gradient methods are widely used to align large language model outputs with human preferences, but the log-likelihoods they require cannot be directly computed in Masked Diffusion Language Models (MDLMs). MDLM post-training therefore relies on a surrogate objective based on the Evidence Lower Bound (ELBO), and recent work estimates a blockwise variant of this objective via Monte Carlo methods to support blockwise generation. However, existing estimation schemes randomly sample the block index, so they may evaluate the same block more than once and leave some blocks unevaluated. This introduces variance arising from block selection into the estimates. In this work, we propose a full-block estimator that evaluates every block exactly once and averages the results. We theoretically show that the proposed estimator remains unbiased while eliminating the between-group variance arising from block selection, thereby achieving lower variance than the baseline for the same sample size. In experiments on five benchmarks spanning mathematics, planning, and coding, our method achieves the highest accuracy on four benchmarks and outperforms the random block sampling baseline by an average of 12.79 points.[1]

1 Introduction

Policy gradient methods are widely used to align large language model (LLM) outputs with human preferences (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024). These methods require the log-likelihood of the response, which in autoregressive language models (ARLMs) can be computed as the sum of conditional log-probabilities over individual tokens.

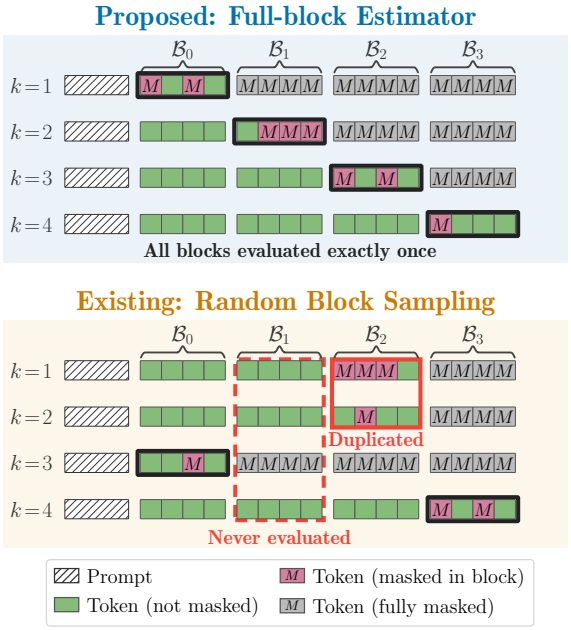


Figure 1: Comparison of block selection in Monte Carlo estimation of the blockwise surrogate objective. For each Monte Carlo sample index k, a block b is sampled, and the corresponding set of token positions is denoted by $\mathcal{B}_b$. For each block b, the masking ratio t_b and the masked positions $\mathcal{M}_{t_b}$ are sampled to compute the blockwise surrogate objective in (1). **Top:** In our method (full-block evaluation), each sample k corresponds to a distinct block, so that every block is evaluated exactly once. **Bottom:** In the existing method (random block sampling), each sample independently selects a block, which may cause the same block to be evaluated multiple times ($\mathcal{B}_2$ is selected for both k=1, 2) or some blocks to remain unevaluated ($\mathcal{B}_1$).

In contrast, Masked Diffusion Language Models (MDLMs), which have recently gained attention, cannot directly compute log-likelihoods. MDLMs generate sequences by randomly masking tokens in a sequence and then progressively unmasking them. The generation process in MDLMs does not uniquely determine the unmasking order, so computing the log-likelihood

*Work done while at Kyoto University.
[1]Code is available at https://github.com/githubhyz/See_Every_Block_Once.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 5836–5857
November 6-10, 2026 ©2026 Association for Computational Linguistics

requires marginalization over all possible orders (Uria et al., 2014). For long sequences, this computational cost is prohibitive.

Consequently, MDLM training cannot directly maximize the log-likelihood of the full response and instead optimizes the Evidence Lower Bound (ELBO), as a surrogate objective (Sahoo et al., 2024). Recent MDLMs also adopt blockwise generation, which partitions sequences into blocks and generates them left to right (Arriola et al., 2025; Nie et al., 2025); a corresponding blockwise surrogate objective has also been proposed (Wang et al., 2026; Sun et al., 2025).

This work focuses on the Monte Carlo estimation of the blockwise surrogate objective. The blockwise surrogate objective is defined as an expectation over the block index b, the masking ratio t_b within the block, and the subset of masked positions $\mathcal{M}_{t_b}$:

$$\mathcal{L}_{\text{Block}} = \mathbb{E}_{b,\, t_b,\, \mathcal{M}_{t_b}} \left[\ell_{b,t_b,\mathcal{M}_{t_b}} \right] \tag{1}$$

(see Section 2.2 for the definition). Here, $\ell_{b,t_b,\mathcal{M}_{t_b}}$ is the token prediction loss at the masked positions within block b. In practice, $\mathcal{L}_{\text{Block}}$ is approximated via Monte Carlo estimation by sampling b, t_b, and $\mathcal{M}_{t_b}$ a small number of times, and the resulting estimation variance can destabilize policy updates. In particular, the existing estimation scheme (Wang et al., 2026) that randomly samples the block index b (random block sampling; Fig. 1, bottom) may select the same block multiple times or leave some blocks entirely unevaluated.

Unlike prior work that focuses on designing policy optimization rules (Zhao et al., 2025; Tang et al., 2026), improving the surrogate objective (Wang et al., 2026), or controlling the bias and variance of likelihood approximation (Zhu et al., 2026), this work examines the design of the Monte Carlo estimator itself while keeping the surrogate objective fixed. Specifically, instead of randomly sampling the block index, we propose a full-block estimator that evaluates every block exactly once and averages the results (Fig. 1, top). Since this approach replaces only the estimator without modifying the blockwise surrogate objective itself, it can be readily integrated into existing MDLM post-training pipelines.

We evaluate our method on five benchmarks (GSM8K, MATH-500, Countdown, Sudoku, and HumanEval) and find that it outperforms baselines on most benchmarks. Furthermore, compared with random block sampling, our method exhibits higher Spearman rank correlation and lower estimation error relative to reference values treated as ground truth, showing that it suppresses the variance arising from block selection at the estimator level. These results indicate that the design of the estimator for the surrogate objective is also a viable avenue for improvement in MDLM post-training.

2 Background

This section first describes Masked Diffusion Language Models (MDLMs), which form the basis of our method, then formulates blockwise generation used in MDLMs, and finally explains the instability of block selection.

2.1 Masked Diffusion Language Models

Masked Diffusion Language Models (MDLMs) apply an absorbing-state diffusion process to discrete text sequences and generate sequences through mask prediction (Sahoo et al., 2024; Nie et al., 2025). In the forward process, each token is independently masked according to a diffusion timestep $t \in (0, 1]$, and in the reverse process the sequence is recovered by predicting the tokens at masked positions (see Appendix C.1 for details).

Notation. Let $\boldsymbol{o}$ denote the response generated by the model for an input prompt $\boldsymbol{c}$. Each response $\boldsymbol{o} = (o^1, \ldots, o^L)$ consists of L tokens over a vocabulary $\mathcal{V}$. We denote by $\boldsymbol{o}_t$ the corrupted sequence obtained by independently replacing each token of $\boldsymbol{o}$ with a mask token[2] $M \notin \mathcal{V}$ with probability depending on t. The MDLM $p_\theta(o^i \mid \boldsymbol{o}_t, \boldsymbol{c})$ predicts the original token at position i given the corrupted sequence $\boldsymbol{o}_t$ and the prompt $\boldsymbol{c}$, where θ denotes the model parameters.

Objective in pre-training. In pre-training, MDLMs maximize the Evidence Lower Bound[3] (ELBO) (Sahoo et al., 2024):

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{o}; \theta) = \mathbb{E}_{t, \boldsymbol{o}_t} \left[\frac{1}{t} \sum_{i:\, o_t^i = M} \log p_\theta(o^i \mid \boldsymbol{o}_t, \boldsymbol{c}) \right], \tag{2}$$

[2]The mask token M serves as the absorbing state in the diffusion process.

[3]The summation in (2) runs only over positions that are masked in the corrupted sequence $\boldsymbol{o}_t$.

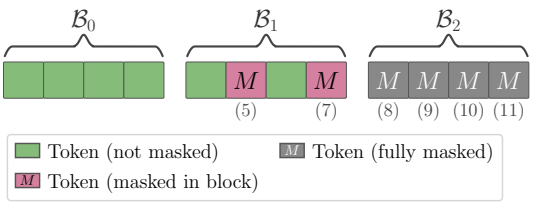


Figure 2: Mask state during the generation of block $b = 1$ for a response $\boldsymbol{o}$ of length $L = 12$ ($C = 3$, $B = 4$). Block $\mathcal{B}_0$ is considered already generated, so its original tokens are retained. In this example, positions 5 and 7 within block $\mathcal{B}_1$ are masked ($\mathcal{M}_{t_1} = \{5, 7\}$), and all positions in block $\mathcal{B}_2$ are masked. The full set of masked positions is $\mathcal{S}_1(\mathcal{M}_{t_1}) = \{5, 7, 8, 9, 10, 11\}$.

where $t \sim \mathrm{Unif}((0, 1])$ and $\boldsymbol{o}_t \sim q(\cdot \mid \boldsymbol{o}, t)$; here $\mathrm{Unif}(X)$ denotes the uniform distribution over a set X.

2.2 Blockwise generation

To achieve better generation quality, recent MDLMs adopt blockwise generation, which partitions a sequence into multiple blocks and generates them sequentially from left to right (Arriola et al., 2025). When blockwise generation is used at inference time, it is desirable to compute the training objective based on the blockwise generation process as well. This work considers reinforcement learning (RL) via policy optimization[4] and formulates a blockwise surrogate objective based on their implementation[5].

Notation. Let B denote the block length and $C = L/B$ the number of blocks[6] when a response $\boldsymbol{o}$ of length L is partitioned into blocks of length B. Let $b \in \{0, \ldots, C-1\}$ be the block index and $\mathcal{B}_b$ the set of token positions belonging to block b ($|\mathcal{B}_b| = B$). We write $t_b \in (0, 1]$ for the masking ratio within block b. Let $\mathcal{M}_{t_b} \subseteq \mathcal{B}_b$ denote the set of positions masked with probability t_b within block b. During the generation of block b, all positions in ungenerated blocks $b' > b$ are also masked in addition to $\mathcal{M}_{t_b}$; we write $\mathcal{S}_b(\mathcal{M}_{t_b})$ for this extended mask set (see Appendix C.5 for the formal definition and Fig. 2 for an example).

[4]Objectives for pre-training (Arriola et al., 2025) (Wang et al., 2026) and Supervised Fine-Tuning (SFT) (Sun et al., 2025) have also been proposed.

[5]Wang et al. (2026) adopt a blockwise masking strategy in their implementation but do not explicitly formulate the corresponding objective (https://github.com/facebookresearch/SPG).

[6]For simplicity, we assume that L is divisible by B.

Blockwise surrogate objective in RL. The number of masked tokens within block b is $\max(1, \lfloor t_b B \rfloor)$, and we write $\Omega_b(t_b)$ for the set of all possible masked-position subsets[7] (see Appendix C.6 for the formal definition). We also define the fraction of masked positions relative to the total sequence length as $\gamma_b(\mathcal{M}_{t_b}) = |\mathcal{S}_b(\mathcal{M}_{t_b})|/L$. The blockwise surrogate objective is then defined as:

$$\mathcal{L}_{\mathrm{Block}}(\boldsymbol{o} \mid \boldsymbol{c}; \theta) := \mathbb{E}_{b, t_b, \mathcal{M}_{t_b}} \left[\ell_{b, t_b, \mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta) \right], \tag{3}$$

where

$$\ell_{b, t_b, \mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta) = \frac{1}{\gamma_b(\mathcal{M}_{t_b})} \sum_{i \in \mathcal{S}_b(\mathcal{M}_{t_b})} \log p_\theta\left(o^i \mid \boldsymbol{o}_{b, \mathcal{M}_{t_b}}, \boldsymbol{c}\right), \tag{4}$$

where $b \sim \mathrm{Unif}(\{0, \ldots, C-1\})$, $t_b \sim \mathrm{Unif}((0, 1])$, and $\mathcal{M}_{t_b} \sim \mathrm{Unif}(\Omega_b(t_b))$. Here, $\boldsymbol{o}_{b, \mathcal{M}_{t_b}}$ denotes the partially masked sequence obtained by replacing the positions in $\mathcal{S}_b(\mathcal{M}_{t_b})$ with M (see Appendix C.5 for the formal per-position definition). This gives the formal definition of (1). The coefficient $1/\gamma_b(\mathcal{M}_{t_b})$ scales the token prediction loss by the number of masked positions, ensuring a consistent loss scale across different masking ratios. We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for policy optimization and use $\mathcal{L}_{\mathrm{Block}}$ as a surrogate for $\log \pi_\theta(\boldsymbol{o}^j \mid \boldsymbol{c})$ in the GRPO objective[8] (see Appendix C.3 for the formulation of GRPO).

2.3 Instability of block selection

Estimation of the blockwise surrogate objective relies on sampling blocks. This work focuses on the finite-sample instability that arises when a small number of blocks are randomly selected.

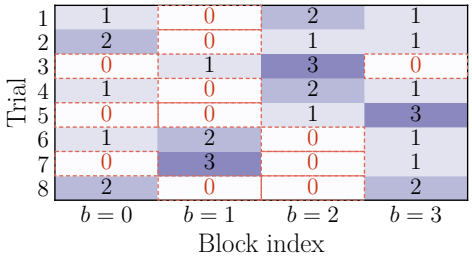
Estimation of the blockwise surrogate objective. A finite-sample Monte Carlo estimate of $\mathcal{L}_{\mathrm{Block}}$ in (3) is obtained by K independent samples:

$$\hat{\mathcal{L}}_{\mathrm{Rand}}(\boldsymbol{o} \mid \boldsymbol{c}; \theta) := \frac{1}{K} \sum_{k=1}^{K} \ell_{b_k, t_k, \mathcal{M}_k}(\boldsymbol{o}, \boldsymbol{c}; \theta), \tag{5}$$

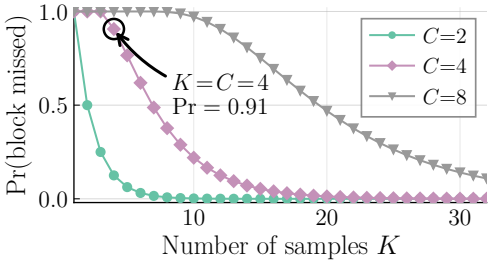
where $b_k \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\{0, \ldots, C-1\})$, $t_k \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}((0, 1])$, and $\mathcal{M}_k \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\Omega_{b_k}(t_k))$. How-

[7]$\lfloor \cdot \rfloor$ denotes the floor function.

[8]Here, $\boldsymbol{o}^j$ denotes the j-th of the g responses $\{\boldsymbol{o}^j\}_{j=1}^g$ generated by the model for an input prompt $\boldsymbol{c}$.



(a) Per-block selection counts over 8 independent trials (C=4, K=4). Cells with 0 indicate missed blocks.



(b) Probability that at least one block is never selected as a function of K for varying C.

Figure 3: Instability of random block sampling with a small number of blocks. (a) Some blocks receive zero selections after K=4 independent draws. (b) The miss probability increases rapidly with C (see Appendix C.7 for a formal derivation).

ever, in practice the number of blocks C is typically small (e.g., $C = 2, 4$, or 8). Consequently, with a finite number of evaluations, the same block may be repeatedly sampled, while other blocks may never be selected. As a result, the variability in block selection increases the instability of the estimate of $\mathcal{L}_{\text{Block}}$.

Block sampling simulation. Figure 3 illustrates the finite-sample instability arising from random block selection. Figure 3a shows the results of simulating 8 independent trials with $C = 4$ and $K = 4$; red-bordered cells indicate blocks that were never selected. In all trials, at least one block remains unselected. Figure 3b shows the probability that at least one block is never selected for $C \in \{2, 4, 8\}$. At $K = C = 4$, $\Pr = 0.91$, indicating that unselected blocks occur with high probability.

3 Proposed Method

We introduce a full-block estimator that evaluates all blocks rather than randomly selecting a subset, while keeping $\mathcal{L}_{\text{Block}}$ itself unchanged (see Fig. 1, top).

Since $\hat{\mathcal{L}}_{\text{Rand}}$ in (5) randomly selects the block index b, the same block may be evaluated more than once. The proposed full-block estimator evaluates every block exactly once and averages the results over all C blocks:

$$\hat{\mathcal{L}}_{\text{Full}}(\boldsymbol{o} \mid \boldsymbol{c}; \theta) := \frac{1}{C} \sum_{b=0}^{C-1} \ell_{b,t_b,\mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta), \quad (6)$$

where $t_b \sim \text{Unif}((0, 1])$ and $\mathcal{M}_{t_b} \sim \text{Unif}(\Omega_b(t_b))$ are sampled independently for each block b.

Indeed, $\hat{\mathcal{L}}_{\text{Full}}$ is an unbiased estimator of $\mathcal{L}_{\text{Block}}$:

$$\begin{aligned} &\mathbb{E}\left[\hat{\mathcal{L}}_{\text{Full}}(\boldsymbol{o} \mid \boldsymbol{c}; \theta)\right] \\ &= \frac{1}{C} \sum_{b=0}^{C-1} \mathbb{E}_{t_b,\mathcal{M}_{t_b}}\left[\ell_{b,t_b,\mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta)\right] \\ &= \mathcal{L}_{\text{Block}}(\boldsymbol{o} \mid \boldsymbol{c}; \theta), \end{aligned}$$

where the last equality follows from the fact that each block b is selected with probability $1/C$ under $b \sim \text{Unif}(\{0, \ldots, C-1\})$ in (3).

4 Theoretical Comparison of Objective Variances

This section theoretically compares the variances of the proposed estimator $\hat{\mathcal{L}}_{\text{Full}}$ in (6) and the baseline $\hat{\mathcal{L}}_{\text{Rand}}$ in (5) in terms of the within-group variance and the between-group variance.

Notation. For each block b, we define the expected loss and the variance of the loss within the block as

$$\begin{aligned} \mu_b &:= \mathbb{E}_{t_b,\mathcal{M}_{t_b}}\left[\ell_{b,t_b,\mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta)\right], \\ \sigma_b^2 &:= \text{Var}_{t_b,\mathcal{M}_{t_b}}\left[\ell_{b,t_b,\mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta)\right], \end{aligned}$$

respectively. We further define the between-group variance, which measures the variability of the expected losses across blocks, and the within-group variance, which is the average of the variances within each block (Muthén, 1991):

$$\sigma^2_{\text{between}} := \text{Var}_{b'}[\mu_{b'}] = \frac{1}{C} \sum_{b=0}^{C-1} (\mu_b - \bar{\mu})^2, \quad (7)$$

$$\bar{\sigma}^2_{\text{within}} := \mathbb{E}_{b'}\left[\sigma^2_{b'}\right] = \frac{1}{C} \sum_{b=0}^{C-1} \sigma_b^2, \quad (8)$$

where $b' \sim \text{Unif}(\{0, \ldots, C-1\})$ and $\bar{\mu} := \frac{1}{C}\sum_{b=0}^{C-1} \mu_b$.

Variance of $\hat{\mathcal{L}}_{\text{Rand}}$. We derive the variance of each sample $\ell_{b_k,t_k,\mathcal{M}_k}$ in $\hat{\mathcal{L}}_{\text{Rand}}$ in (5). The randomness of each sample consists of two stages: the selection of the block index b_k and the sampling of $(t_k, \mathcal{M}_k)$ within the selected block. Applying the law of total variance (Casella and Berger, 2002), $\text{Var}[X] = \text{Var}[\mathbb{E}[X \mid Y]] + \mathbb{E}[\text{Var}[X \mid Y]]$, with $X = \ell_{b_k,t_k,\mathcal{M}_k}$ and $Y = b_k$, we obtain

$$\begin{aligned}\text{Var}[\ell_{b_k,t_k,\mathcal{M}_k}] &= \text{Var}_{b_k}[\mathbb{E}_{t_k,\mathcal{M}_k}[\ell_{b_k,t_k,\mathcal{M}_k} \mid b_k]] \\ &\quad + \mathbb{E}_{b_k}[\text{Var}_{t_k,\mathcal{M}_k}[\ell_{b_k,t_k,\mathcal{M}_k} \mid b_k]] \\ &= \text{Var}_{b_k}[\mu_{b_k}] + \mathbb{E}_{b_k}[\sigma^2_{b_k}] \\ &= \sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}}, \end{aligned} \tag{9}$$

where the last equality holds since $b_k \sim \text{Unif}(\{0, \ldots, C-1\})$. Since the K samples are mutually independent, from (5) and (9) we have

$$\text{Var}\left[\hat{\mathcal{L}}_{\text{Rand}}\right] = \frac{1}{K}\left(\sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}}\right). \tag{10}$$

Variance of $\hat{\mathcal{L}}_{\text{Full}}$. In $\hat{\mathcal{L}}_{\text{Full}}$ in (6), the block index deterministically takes every value $b = 0, \ldots, C-1$ exactly once. The only random variables are $(t_b, \mathcal{M}_{t_b})$ within each block, and these are independent across blocks. From (6) and (8),

$$\text{Var}\left[\hat{\mathcal{L}}_{\text{Full}}\right] = \frac{1}{C^2}\sum_{b=0}^{C-1}\sigma_b^2 = \frac{\bar{\sigma}^2_{\text{within}}}{C}. \tag{11}$$

Variance ratio. From (10) and (11),

$$\frac{\text{Var}\left[\hat{\mathcal{L}}_{\text{Full}}\right]}{\text{Var}\left[\hat{\mathcal{L}}_{\text{Rand}}\right]} = \frac{K}{C}\cdot\frac{\bar{\sigma}^2_{\text{within}}}{\sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}}}. \tag{12}$$

Since the second factor in (12) is at most 1, the variance ratio is at most 1 whenever $K = C$ (i.e., the sample size matches the number of blocks). That is, $\hat{\mathcal{L}}_{\text{Full}}$ remains unbiased for $\mathcal{L}_{\text{Block}}$ while eliminating the between-group variance $\sigma^2_{\text{between}}$ arising from block selection, thereby achieving lower variance when $K = C$.

The variance ratio derived above in (12) is restricted to $K = C$ for simplicity. Appendix D extends the same analysis to arbitrary $K \geq 1$, and the resulting expression (26) shows that the variance ratio is at most 1. The variance reduction of our method ($\text{Var}[\hat{\mathcal{L}}_{\text{Full}}] \leq \text{Var}[\hat{\mathcal{L}}_{\text{Rand}}]$) therefore holds beyond the $K = C$ setting.

5 Experimental Setup

This section evaluates the proposed full-block estimator $\hat{\mathcal{L}}_{\text{Full}}$ in (6) on five benchmarks across three categories: mathematics, planning, and coding. Specifically, we assess (i) whether the difference in estimator alone leads to a performance gap when the sample size matches the number of blocks ($\hat{\mathcal{L}}_{\text{Full}}$ in (6) vs. $\hat{\mathcal{L}}_{\text{Rand}}$ in (5)), and (ii) whether our method achieves competitive performance compared with existing MDLM post-training methods.

5.1 Baselines

We use LLaDA-8B-Instruct (Nie et al., 2025) as the base model. The compared methods are LLaDA-1.5 (Zhu et al., 2026), D1 (Zhao et al., 2025), WD1 (Tang et al., 2026), SPG (Wang et al., 2026), and $\hat{\mathcal{L}}_{\text{Rand}}$ in (5).[9] In particular, the comparison between $\hat{\mathcal{L}}_{\text{Rand}}$ in (5), which samples the block index independently and uniformly, and our method $\hat{\mathcal{L}}_{\text{Full}}$ in (6) serves as an ablation that isolates the effect of the estimator while keeping the objective fixed. See Section 8 for details on each method.

5.2 Tasks

We use five benchmarks spanning the following three task categories. (i) **Mathematics**: GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024). (ii) **Planning**: Countdown (Pan et al., 2025) and Sudoku (Arel, 2025). (iii) **Coding**: HumanEval (Chen et al., 2021). The training data for the coding task are taken from Open R1 (2025).

5.3 Training Details

Model settings. Our method and all compared methods are trained with the same hyperparameter settings. In all experiments, we fine-tune the base model (LLaDA-8B-Instruct) using QLoRA (Dettmers et al., 2023) (4-bit NF4 quantization, LoRA rank $r = 128$; $r = 64$ for the coding task). We use the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 1.5×10^{-6} (constant schedule with warmup), and all tasks are trained for 4,000 steps.

[9] LLaDA-8B-Instruct is one of the largest publicly available instruction-tuned MDLMs, and the compared methods are also evaluated on LLaDA-family MDLMs; we therefore adopt it as the base model in this work. LLaDA-1.5, D1, WD1, and SPG are all standard baselines used in prior work on MDLM post-training.

Method	Blockwise	Benchmark					
		GSM8K	MATH-500	Countdown	Sudoku	HumanEval	Avg.
LLaDA	×	76.27	31.80	19.14	26.56	27.44	36.24
LLaDA-1.5	×	78.70	33.00	18.36	23.64	32.93	37.33
D1	×	81.43†	36.00	36.72	13.21	32.93	40.06
WD1	×	81.20	34.60	44.14	62.25	28.05	50.05
SPG	✓	80.29	36.20†	**70.31**↑3.90	72.73†	29.88	57.88†
$\hat{\mathcal{L}}_{\text{Rand}}$	✓	76.20	34.40	44.14	61.20	34.76†	50.14
$\hat{\mathcal{L}}_{\text{Full}}$ (ours)	✓	**82.26** ▲6.06 / ↑0.83	**38.00** ▲3.60 / ↑1.80	66.41† ▲22.27	**91.39** ▲30.19 / ↑18.66	**36.59** ▲1.83 / ↑1.83	**62.93** ▲12.79 / ↑5.05

Table 1: Performance comparison across benchmarks. Higher scores indicate better performance. LLaDA denotes the base model (LLaDA-8B-Instruct) before fine-tuning. The **Blockwise** column indicates methods that use the blockwise surrogate objective (3). **Bold** denotes the highest accuracy in each column, and † denotes the second highest. ▲ indicates the improvement of $\hat{\mathcal{L}}_{\text{Full}}$ over $\hat{\mathcal{L}}_{\text{Rand}}$, and ↑ indicates the improvement of the highest accuracy over the † value.

Generation settings. During training, the generation length is $L = 256$ (512 for the coding task) and the block length is $B = 64$ (128 for the coding task), yielding $C = 4$ blocks for all tasks. The number of diffusion steps is set to half the generation length, corresponding to an average of 2 tokens unmasked per step.

Objective settings. We use GRPO (Shao et al., 2024) for policy optimization with task-specific rule-based verifiers as reward functions (see Appendix B for more details). $\hat{\mathcal{L}}_{\text{Full}}$, $\hat{\mathcal{L}}_{\text{Rand}}$, and SPG all employ blockwise surrogate objectives, and we set the number of Monte Carlo samples to $K = C = 4$ for all three methods. Under this setting, $\hat{\mathcal{L}}_{\text{Full}}$ evaluates each block exactly once, while SPG and $\hat{\mathcal{L}}_{\text{Rand}}$ independently sample $(b_k, t_k, \mathcal{M}_k)$ K times. D1 and WD1 do not use the blockwise surrogate objective, so this sample size is not applicable.

See Appendix A.1 for further details.

5.4 Evaluation

Following prior work (Wang et al., 2026; Zhao et al., 2025), the evaluation settings for each task are as follows. (i) **Mathematics & Planning**: GSM8K and MATH-500 are evaluated in a 0-shot setting using exact match accuracy between the generated answer and the ground-truth label. For the mathematics tasks, we extract the expression inside `\boxed{}`, normalize the LaTeX representation, and check for mathematical equivalence. For Countdown and Sudoku, correctness is determined by whether the generated answer matches the ground truth. Sudoku is evaluated in a 2-shot setting. (ii) **Coding**: HumanEval is evaluated in a 2-shot setting using Pass@1 (the fraction of problems for which a single generation passes all test cases).

We evaluate each task using the checkpoint from the final training step. See Appendix A.2 for details on inference parameters.

6 Results

Table 1 presents the results. Our method achieves the highest accuracy on four out of five benchmarks and outperforms SPG, the strongest baseline, by 5.05 points in the simple average across all five benchmarks.

We first compare $\hat{\mathcal{L}}_{\text{Full}}$ in (6) and $\hat{\mathcal{L}}_{\text{Rand}}$ in (5) as an ablation that isolates the effect of the estimator. $\hat{\mathcal{L}}_{\text{Full}}$ outperforms $\hat{\mathcal{L}}_{\text{Rand}}$ on all five benchmarks, with an average improvement of 12.79 points. This result shows that, even with the same surrogate objective, the design of the estimator affects the final task performance in the finite-sample regime.

Our method also demonstrates consistent performance compared with all existing methods. On the mathematics tasks, our method achieves the highest accuracy on both GSM8K (82.26) and MATH-500 (38.00). On Sudoku, the performance gap among methods is large; our method (91.39) outperforms the strongest baseline, SPG (72.73), by 18.66 points. On HumanEval, a coding task, our method (36.59) also outperforms all baselines. On the other hand, on Countdown, our method (66.41) outperforms $\hat{\mathcal{L}}_{\text{Rand}}$ (44.14) by 22.27 points but does not surpass SPG (70.31). SPG combines (a) the sandwiched objective and (b) random block sampling. Our method does not use (a), and in-

stead improves (b). The gap on Countdown may therefore be attributable to the effect of (a).

We examine whether these results depend on the specific setting $K = C = 4$ through the additional experiments in Appendix E ($K \in \{2, 6, 8\}$ with $C = 4$ fixed) and Appendix F (doubling to $C = 8$). Our method matches or outperforms $\hat{\mathcal{L}}_{\text{Rand}}$ in every setting.[10] Note that within the settings we examined, the gain of our method is largest in the low-sample regime $K = C$ and generally shrinks as the sample size K grows beyond the number of blocks C (Appendix E).

The next section analyzes the difference between our method and the existing estimator more directly from the perspectives of estimation quality and variance.

7 Empirical Comparison of Objective Estimates

This section quantitatively compares the statistical properties of $\hat{\mathcal{L}}_{\text{Full}}$ in (6) and $\hat{\mathcal{L}}_{\text{Rand}}$ in (5). To this end, we first describe the experimental setup, then evaluate the overall trend of estimation quality using the Spearman rank correlation and RMSE with respect to a high-precision reference estimate. Finally, we verify the theoretical variance ratio derived in (12) using empirical measurements.

7.1 Setup

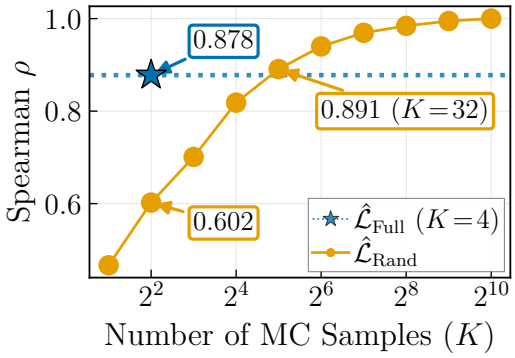
For the five benchmarks described in Section 5.2, we extract $n = 1{,}000$ prompts from the training data and generate one response per prompt using LLaDA-8B-Instruct[11], with the number of blocks set to $C = 4$. Since $\hat{\mathcal{L}}_{\text{Full}}$ evaluates each block exactly once, we set $K = C = 4$ for $\hat{\mathcal{L}}_{\text{Rand}}$ to match the sample size.

In what follows, we refer to $\hat{\mathcal{L}}_{\text{Rand}}$ computed with sample size $K = 1024$ as the high-precision reference estimate $\mathcal{L}_{\text{ref}}$. Although this is not the exact true value, we regard it as an approximate reference with sufficient precision for the comparisons in this section[12].

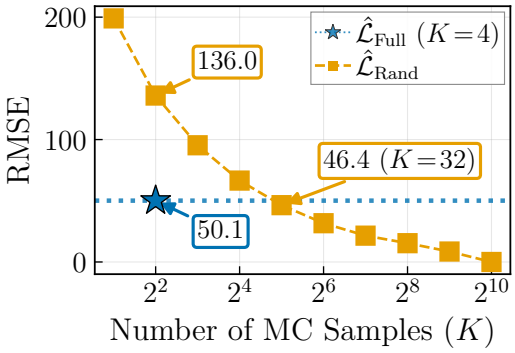
[10] The improvements over $\hat{\mathcal{L}}_{\text{Rand}}$ are as follows: at $K = 2$, +6.66 points on the five-dataset average; at $K = 8$, +0.33 points on the GSM8K/MATH-500 average; and at $C = 8$, +2.92 points on the GSM8K/MATH-500 average.

[11] We use the base model rather than the trained model from Section 6 so that the statistical properties of $\hat{\mathcal{L}}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ are compared under the same conditions.

[12] For example, on GSM8K the Spearman rank correlation between the estimates at $K = 512$ and $K = 1024$ is 0.994, indicating that the ranking is nearly stable and further increasing K has little effect.



(a) Spearman ρ



(b) RMSE

Figure 4: Comparison of estimation quality for $\mathcal{L}_{\text{Block}}$ in (3) between $\hat{\mathcal{L}}_{\text{Full}}$ in (6) and $\hat{\mathcal{L}}_{\text{Rand}}$ in (5) on GSM8K ($C = 4$). $\hat{\mathcal{L}}_{\text{Rand}}$ is computed with varying sample size K. The estimate at $K = 1024$ is used as the high-precision reference estimate $\mathcal{L}_{\text{ref}}$ for computing Spearman ρ and RMSE. Higher ρ and lower RMSE indicate better estimation quality. The star indicates the result of $\hat{\mathcal{L}}_{\text{Full}}$ at $K = C = 4$, which achieves estimation quality roughly comparable to $\hat{\mathcal{L}}_{\text{Rand}}$ at $K \approx 32$.

7.2 Estimation Quality

We evaluate how close the estimates of $\mathcal{L}_{\text{Block}}$ in (3) are to the reference value $\mathcal{L}_{\text{ref}}$ under a finite sample size. As evaluation metrics, we use the Spearman rank correlation coefficient ρ and the Root Mean Squared Error[13] (RMSE) with respect to the reference value. The rank correlation measures how well the relative ordering of the estimates is preserved, and the RMSE measures the magnitude of the estimation error. For $\hat{\mathcal{L}}_{\text{Rand}}$, we compute estimates with sample size $K = 2, 4, 8, \ldots, 1024$ and compare them against

[13] The RMSE for n samples is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathcal{L}}_i - \mathcal{L}_{\text{ref},i}\right)^2},$$

where $\hat{\mathcal{L}}_i$ and $\mathcal{L}_{\text{ref},i}$ are the estimate ($\hat{\mathcal{L}}_{\text{Full}}$ or $\hat{\mathcal{L}}_{\text{Rand}}$) and the reference value for the i-th sample, respectively.

GSM8K	MATH	Countdown	Sudoku	Code
0.122	0.119	0.095	0.076	0.107

Table 2: Variance ratio $\mathrm{Var}[\hat{\mathcal{L}}_{\text{Full}}]$ / $\mathrm{Var}[\hat{\mathcal{L}}_{\text{Rand}}]$ in (12), evaluated numerically for each sample and averaged within each dataset ($K = C = 4$). Smaller values indicate a greater variance reduction effect of $\hat{\mathcal{L}}_{\text{Full}}$.

$\mathcal{L}_{\text{ref}}$.

Figure 4 shows the results on GSM8K. At sample size $K = 4$, $\hat{\mathcal{L}}_{\text{Full}}$ achieves $\rho = 0.878$ and $\text{RMSE} = 50.1$, substantially outperforming $\hat{\mathcal{L}}_{\text{Rand}}$ ($\rho = 0.602$, $\text{RMSE} = 136.0$). This level of accuracy is comparable to $\hat{\mathcal{L}}_{\text{Rand}}$ at $K \approx 32$, demonstrating that merely eliminating the random sampling of the block index substantially improves the estimation quality in the finite-sample regime.

Similar trends are observed on the other datasets (Appendices G.1, G.2); $\hat{\mathcal{L}}_{\text{Full}}$ further exhibits substantially lower tail probability of estimation errors than $\hat{\mathcal{L}}_{\text{Rand}}$ (Appendix G.3).

7.3 Quantitative Evaluation of the Variance Ratio

We quantitatively evaluate the variance ratio derived in (12). We then also discuss the variance ratio at the policy-gradient level.

Estimator level. For each of the $C = 4$ blocks, we independently sample the diffusion timestep 50 times[14] and compute $\ell_{b,t_b,\mathcal{M}_{t_b}}(\boldsymbol{o}, \boldsymbol{c}; \theta)$ in (4). From these sampled values, we compute $\sigma^2_{\text{between}}$ in (7) and $\bar{\sigma}^2_{\text{within}}$ in (8). We then evaluate the right-hand side of (12) with $K = C = 4$ for each sample. Table 2 reports the average of these values within each dataset. The variance ratio ranges from 0.08 to 0.12 across all datasets, indicating that the variance of $\hat{\mathcal{L}}_{\text{Full}}$ in (11) is approximately $1/10$ of the variance of $\hat{\mathcal{L}}_{\text{Rand}}$ in (10).

As shown in (12), when $K = C$, the difference between the variance of $\hat{\mathcal{L}}_{\text{Full}}$ and that of $\hat{\mathcal{L}}_{\text{Rand}}$ lies in the presence or absence of the between-group variance $\sigma^2_{\text{between}}$ arising from block selection. The results in Table 2 show that $\sigma^2_{\text{between}}$ accounts for the majority of $\mathrm{Var}[\hat{\mathcal{L}}_{\text{Rand}}]$. By eliminating this component, $\hat{\mathcal{L}}_{\text{Full}}$ provides a more stable surrogate estimate at the estimator level.

Policy-gradient level. We also examine how far this variance reduction carries over to the policy gradient, which is what actually updates the policy. On GSM8K and MATH-500 at mid-training checkpoints, the estimator-level variance ratio is 0.10–0.16 (Appendix I), whereas the policy-gradient variance ratio is 0.79–0.96 (Appendix H), so the accuracy gains in Section 6 are not necessarily due to reduced gradient variance. Unlike $\hat{\mathcal{L}}_{\text{Rand}}$, which is likely to leave some blocks unselected at each step (Section 2.3), $\hat{\mathcal{L}}_{\text{Full}}$ delivers a learning signal to every block at every step, which we consider another source of its advantage. We do not disentangle how much each of the two factors, estimator-level variance reduction and the learning signal delivered to every block, contributes to the accuracy difference.

[14] For each block b, we independently sample t_b and $\mathcal{M}_{t_b}$ 50 times.

8 Related Work

Masked Diffusion Language Models. Discrete diffusion models originated from D3PM (Austin et al., 2021) and have since evolved through SEDD (Lou et al., 2024), MDLM (Sahoo et al., 2024), and MD4 (Shi et al., 2024). LLaDA (Nie et al., 2025) scaled masked diffusion models to 8B parameters, demonstrating performance comparable to auto-regressive language models and introducing efficient inference via blockwise generation. This work targets the post-training based on blockwise generation and the surrogate objective adopted by LLaDA.

Blockwise Training for MDLM. For pre-training, Arriola et al. (2025) proposed block diffusion, which performs discrete diffusion within blocks and generates across blocks auto-regressively, and introduced a noise schedule that minimizes gradient variance. For SFT, Sun et al. (2025) proposed computing the loss by selecting a single active block, and Wu et al. (2026) proposed converting ARLMs to block diffusion models via a complementary masking strategy.

Reinforcement Learning for MDLM. Within the same GRPO + blockwise surrogate framework as our method, diffu-GRPO (Zhao et al., 2025) proposes one-step log-likelihood estimation via a mean-field approximation, WD1 (Tang et al., 2026) proposes a ratio-free objective, and SPG (Wang et al., 2026) proposes a sandwiched policy gradient. VRPO (Zhu et al., 2026) performs an MDLM variance analysis in the preference-

based setting[15]. RL methods compatible with blockwise generation have also been proposed, including DiRL (Zhu et al., 2025), GDPO (Rojas et al., 2026), and StableDRL (Zhong et al., 2026).

Relation to Classical Sampling Techniques. When $K = C$, $\hat{\mathcal{L}}_{\text{Full}}$ is equivalent to stratified sampling with one sample per stratum (Owen, 2013). The contribution of this work is to introduce the perspective of Monte Carlo estimator design into the blockwise generation structure of MDLM. See Appendix J for discussion, including the extension to importance sampling. We also derive the variance ratio for general K, including $K \neq C$ (Appendix D). Moreover, our method can be viewed as a form of Rao–Blackwellization, in that it computes the expectation over the block index exactly. The variance that our method targets arises from block selection and differs from the variance targeted by the antithetic sampling of VRPO (Zhu et al., 2026).

While these prior studies focus on designing the surrogate objective, approximating the likelihood, or improving the policy optimization rule, this work explicitly examines the missing coverage and the variance arising from block index selection in the Monte Carlo estimator while keeping the blockwise surrogate objective fixed.

9 Conclusion

This work proposed a full-block estimator $\hat{\mathcal{L}}_{\text{Full}}$ that evaluates every block exactly once to address the missing coverage and the estimation variance caused by random block-index sampling in the Monte Carlo estimation of the blockwise surrogate objective for MDLM post-training. The proposed estimator remains unbiased while eliminating the between-group variance from block selection, and it achieved the highest accuracy on four out of five benchmarks, outperforming the random block sampling baseline by an average of 12.79 points. These results indicate that the design of the estimator is also important for improvement in MDLM post-training, in addition to the design of the surrogate objective and the policy optimization rule.

[15] VRPO performs antithetic sampling for a preference-score estimator (the ELBO difference on the DPO preference pair (y_w, y_l)), which presupposes the existence of (y_w, y_l) and a reference policy π_{ref}; an estimator-level comparison between VRPO and our method is therefore not feasible. Likewise, objective-level prior work such as SPG cannot be applied to VRPO.

Limitations

- Due to computational resource constraints, we did not conduct experiments with increased training steps or GRPO group size g.
- Because our estimator changes only how the block index is selected, it is in principle applicable to other policy optimization methods, such as PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2023), as long as they use a blockwise surrogate objective. However, we have not verified its effect in combination with these methods.
- Although LLaDA-8B-Instruct is one of the largest publicly available instruction-tuned MDLMs (Section 5 footnote), generalization to other model sizes and architectures as well as to block counts beyond $C \in \{4, 8\}$ requires further investigation.
- Our evaluation is limited to three task categories (mathematics, planning, and coding) and the effectiveness on other task categories such as open-ended natural language generation and summarization has not been confirmed.
- We evaluate the accuracy of every method only at its final checkpoint and do not compare accuracy at intermediate checkpoints.
- Since $\hat{\mathcal{L}}_{\text{Full}}$ evaluates all C blocks in parallel, GPU VRAM usage scales linearly with C. In our experiments, $C = 4$ was small enough to avoid this issue. For larger C, this $O(C)$ overhead is specific to the parallel-batch implementation and can be relaxed by sequentially accumulating gradients over blocks, which reduces peak VRAM to a single-block footprint at the cost of $C\times$ wall-clock time per iteration; a systematic memory-vs-time trade-off study is left for future work.
- We have not disentangled how much each of the two factors, estimator-level variance reduction and the learning signal delivered to every block, contributes to the accuracy difference between our method and $\hat{\mathcal{L}}_{\text{Rand}}$.
- Our full-block evaluation targets the block index, which is discrete and takes only a few values. Extending the same idea to continuous or high-dimensional random variables,

such as diffusion timesteps and masked positions, requires approximation and is left for future work.

Acknowledgments

This work was partially supported by JSPS KAKENHI JP22H05106, JP23K28045, JP26H02495 (HS), JP25K24366 (HY), JST CREST JPMJCR21N3 (HS), JST Moonshot JPMJMS263I (HS), and JST BOOST JPMJBS2407 (YH).

References

Arel. 2025. Arel's sudoku generator. https://www.ocf.berkeley.edu/~arel/sudoku/main.html. Accessed: 2025-04-08.

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc.

George Casella and Roger L. Berger. 2002. *Statistical Inference*, 2nd edition. Duxbury, Pacific Grove, CA.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Hugging Face. 2025. Open R1: A fully open reproduction of DeepSeek-R1.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

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

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32819–32848. PMLR.

Bengt O. Muthén. 1991. Multilevel factor analysis of class and student achievement components. *Journal of Educational Measurement*, 28(4):338–354.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 50608–50646. Curran Associates, Inc.

Open R1. 2025. verifiable-coding-problems-python_decontaminated-tested-shuffled. https://huggingface.co/datasets/open-r1/verifiable-coding-problems-python_decontaminated-tested-shuffled. Accessed: 2026-02-09.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Art B. Owen. 2013. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. 2026. Improving reasoning for diffusion language models via group diffusion policy optimization. In *International Conference on Learning Representations*, volume 2026, pages 1938–1973.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 130136–130184. Curran Associates, Inc.

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

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

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, pages 103131–103167. Curran Associates, Inc.

Bowen Sun, Yujun Cai, Ming-Hsuan Yang, and Yiwei Wang. 2025. Blockwise SFT for diffusion language models: Reconciling bidirectional attention and autoregressive decoding. *arXiv preprint arXiv:2508.19529*.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. 2026. wd1: Weighted policy optimization for reasoning in diffusion language models. In *International Conference on Learning Representations*, volume 2026, pages 150357–150385.

Benigno Uria, Iain Murray, and Hugo Larochelle. 2014. A deep and tractable density estimator. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 467–475, Beijing, China. PMLR.

Chenyu Wang, Paria Rashidinejad, Andy (DiJia) Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. 2026. SPG: Sandwiched policy gradient for masked diffusion language models. In *International Conference on Learning Representations*, volume 2026, pages 24623–24660.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2026. Fast-dLLM v2: Efficient block-diffusion LLM. In *International Conference on Learning Representations*, volume 2026, pages 128353–128370.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 56729–56762. Curran Associates, Inc.

Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. 2026. Stabilizing reinforcement learning for diffusion language models. In *Forty-third International Conference on Machine Learning*.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11425–11460, San Diego, California, United States. Association for Computational Linguistics.

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyang He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. 2025. DiRL: An efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*.

A Experimental Details

This section provides detailed experimental settings. All methods evaluated in this study—D1, WD1, SPG, $\hat{\mathcal{L}}_{\text{Rand}}$, and $\hat{\mathcal{L}}_{\text{Full}}$—share the following common configuration.

A.1 Training Details

We describe the detailed training configuration. We use LLaDA-8B-Instruct (Nie et al., 2025) as the base model, which is one of the largest publicly available instruction-tuned MDLMs. For memory efficiency, we apply QLoRA (4-bit NF4) quantization (Dettmers et al., 2023). All experiments are conducted using a PyTorch-based framework with Accelerate and DeepSpeed (ZeRO Stage 2) (Rajbhandari et al., 2020) on two NVIDIA GPUs (RTX A6000 or RTX 6000 Ada Generation), with bfloat16 precision. We set the random seed to 42 for all experiments. We follow the prompt templates of Wang et al. (2026) for all tasks.

Hyperparameter	Value
Optimization	
Optimizer	AdamW
Learning rate	1.5×10^{-6}
LR scheduler	constant_with_warmup
Weight decay	0.1
Max gradient norm	0.2
LoRA (QLoRA, 4-bit NF4)	
Rank (r)	128‡
Alpha (α)	64‡
Dropout	0.05
Target modules	All linear layers†
Training	
Number of GPUs	2
Per-device batch size	2‡
Gradient accum. steps	3
Num. generations (g)	4‡
Num. iterations (μ††)	4
Epsilon (clipping)	0.5
Beta (KL coeff.)	0.0
Max steps	4000
Max prompt length	256‡§
Max completion length (L)	256‡

†`q,k,v,o,up,down,gate_proj`.
††GRPO iteration numbers.
‡Code task uses $r = 64$, $\alpha = 32$, $g = 2$, $L = 512$.
Code: per-device batch size 1, max prompt length 1800.
§Sudoku: max prompt length 1000 (2-shot prompts).

Table 3: Common training hyperparameters.

Parameter	Default	Code
Temperature	0.9*	0.9
Block length	64	128
Diffusion steps	128	256
Generation length	256	512
Remasking strategy	low_confidence	

Table 4: Diffusion generation settings during training. Default applies to GSM8K, MATH, Countdown and Sudoku. *Sudoku uses 0.3.

Hyperparameters. Table 3 lists the optimization, LoRA, and training hyperparameters, and Table 4 lists the diffusion generation settings during training. All methods share these hyperparameters.[16]

Datasets. Table 5 lists the training and evaluation data sources for each task. We select five tasks that span diverse domains: mathematical reasoning (GSM8K, MATH), planning and search (Countdown, Sudoku), and code generation (Code). GSM8K (Cobbe et al., 2021) consists of grade-school multi-step arithmetic reasoning problems. For MATH, we use `ankner/math-500` as the training data, which contains competition-level problems in algebra, geometry, number theory, and other topics. Countdown (Pan et al., 2025) requires combining given numbers with arithmetic operations to reach a target value. Sudoku (Wang et al., 2026) is a 4×4 puzzle that requires filling in empty cells while satisfying row, column, and box constraints. For Code, we adopt HumanEval (Chen et al., 2021) as the evaluation benchmark; however, since HumanEval does not provide a training split, we use Verifiable Coding Problems from Open-R1 (Hugging Face, 2025) as the training data, which consists of Python programming problems with automatic test-case verification. Because the Code task requires a longer generation length (512 tokens), we adjust the LoRA rank and the number of generations to accommodate memory constraints (Table 3).

Reward Functions. The design of the reward function for each task is described in Appendix B.

A.2 Evaluation Details

We describe the inference parameters and benchmark details used for evaluation.

Generation Hyperparameters. At inference time, we use greedy decoding (Temperature $= 0.0$) for reproducibility. We adopt the `low_confidence` remasking strategy (Nie et al., 2025), which preferentially remasks low-confidence tokens. For a generation length L, the number of diffusion steps is uniformly set to $L/2$. Table 6 lists the common settings.

Benchmarks. Table 7 lists the benchmarks used for evaluation, along with the number of evaluation samples and the metric for each task.

Correctness Criteria. We describe the correctness criteria for each benchmark. GSM8K: We extract the numerical value from the model response and check for exact match with the ground-truth integer. MATH: We normalize LaTeX expressions and judge mathematical equivalence. For floating-point numbers, we use a tolerance of 10^{-6}. Countdown: We extract the arithmetic expression from the `<answer>` tag of the model response and check two conditions: (i) the numbers used in the expression correspond one-to-one with the given numbers (each number is used exactly once), and

[16] Because of limited computational resources, our training settings, such as the batch size, differ from those of the SPG paper (Wang et al., 2026). Our numbers are therefore not directly comparable to the values reported in that paper, and we retrain all methods under this single setting for comparison.

Task	Train Source	Eval Source
GSM8K	`gsm8k` (train) (Cobbe et al., 2021)	`gsm8k` (test)
MATH	`ankner/math-500`[17]	`MATH-500` (Lightman et al., 2024)
Countdown	`Jiayi-Pan/Countdown-Tasks-3to4`[18]	`Jiayi-Pan/Countdown-Tasks-3to4`
Sudoku	Custom 4×4 (train) (Wang et al., 2026)	Custom 4×4 (test)
Code	Verifiable Coding Problems[19]	HumanEval (Chen et al., 2021)

Table 5: Datasets used for training and evaluation.

Parameter	Value
Temperature	0.0
Remasking strategy	`low_confidence`
Generation length (L)	256, 512
Diffusion steps	$L/2$

Table 6: Inference hyperparameters.

Benchmark	Samples	Metric
GSM8K	1,319	Accuracy
MATH-500	500	Accuracy
Countdown	256	Accuracy
Sudoku	256	Accuracy
HumanEval	164	Pass@1

Table 7: Evaluation benchmarks.

(ii) the evaluation result of the expression matches the target value (tolerance 10^{-5}). Sudoku: We extract the digit string from the `<answer>` tag and interpret it as a 4×4 board. For each empty cell (cells with value 0) in the puzzle, we compare the response with the ground-truth label cell by cell. The metric is the empty-cell accuracy (number of correctly filled cells / total number of empty cells), pooled over all problems. HumanEval: We execute all test cases provided with the dataset and consider a response correct if and only if all test cases pass (functional correctness).

B Reward functions

For GSM8K, MATH, and Code, the reward function is a weighted sum of a correctness reward and a format reward. The format reward is an auxiliary reward that encourages responses to follow prescribed XML tags (`<reasoning>`, `<answer>`). For all tasks except coding task, we use the same reward functions as Wang et al. (2026). For the coding task, we design the reward function based on Hugging Face (2025). We describe the reward function for each dataset below.

GSM8K. The correctness reward extracts the string within the `<answer>` tag from the model response and checks for exact string match with the ground-truth label. A match yields 2.0; a mismatch yields 0.0. The format reward consists of an XML tag count reward, a soft format reward, a strict format reward, and an integer format reward. All reward components are summed with equal weights.

MATH. The correctness reward extracts the expression within the last `\boxed{}` from the model response, normalizes the LaTeX expression via regular expressions, and judges mathematical equivalence. Equivalence yields 2.0; non-equivalence yields 0.0. The format reward encourages the use of `\boxed{}` and `<answer>` tags. The correctness reward and the format reward are summed with equal weights.

Countdown. We extract the arithmetic expression from the `<answer>` tag of the model response. The correctness reward is 0.0 if extraction fails, 0.1 if the expression is extracted but either the given numbers are not each used exactly once or the evaluation result does not match the target value, and 1.0 if all numbers are each used exactly once and the result matches the target.

Sudoku. We extract the digit string from the `<answer>` tag and interpret it as a 4×4 board. For each empty cell, we check for exact string match between the response and the ground-truth label. The correctness reward is the empty-cell accuracy (number of correctly filled cells / total number of empty cells), a continuous value in $[0.0, 1.0]$.

Code. We extract the Python code block from the model response and execute the test cases in a sandboxed environment. Each test case has a timeout of 5 seconds, and the correctness reward is the

[17] https://huggingface.co/datasets/ankner/math-500

[18] https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4

[19] https://huggingface.co/datasets/open-r1/verifiable-coding-problems-python_decontaminated-tested

pass rate (number of passed tests / total number of tests). A format reward is added with a weight of 0.1. The implementation of the reward function and code evaluation follows Open-R1 (Hugging Face, 2025).

C Background Supplement

This section supplements the technical details omitted from the Background (Sections 2.1–2.3). We first describe the formulation of the forward and reverse processes of MDLM (Appendix C.1), then present the general formulation of policy gradients (Appendix C.2) and the intractability of $\log \pi_\theta$ in MDLM (Appendix C.4). We then provide the formal definition of the mask state in blockwise generation (Appendix C.5), the definition of the masked-position set $\Omega_b(t_b)$ (Appendix C.6), and the derivation of the probability that at least one block is never selected, used in Fig. 3b (Appendix C.7).

C.1 Forward / Reverse Process of MDLM

Forward Process. The forward process of MDLM is based on absorbing-state diffusion (Austin et al., 2021) and independently masks each token. In general, a noise schedule α_t defines the masking probability as $1 - \alpha_t$; following Sahoo et al. (2024), we adopt a log-linear schedule $\alpha_t = 1 - t$. Under this schedule, for a diffusion timestep $t \in (0, 1]$, each token o^i is replaced by the mask token M with probability t:

$$q(o_t^i \mid o^i, t) = \begin{cases} 1 - t & \text{if } o_t^i = o^i, \\ t & \text{if } o_t^i = M, \end{cases},$$

where o_t^i denotes the state of position i at timestep t.

Reverse Process. The reverse process of MDLM is realized by predicting the tokens at masked positions via p_θ. In pre-training, the Evidence Lower Bound (ELBO) in (2) is maximized (Sahoo et al., 2024), where $\mathrm{Unif}((0, 1])$ denotes the uniform distribution over the interval $(0, 1]$.

C.2 General Formulation of Policy Gradients

The gradient of the policy gradient objective $\mathcal{J}(\theta)$ in language model post-training is given by

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{c},\boldsymbol{o}}[R(\boldsymbol{c}, \boldsymbol{o}) \nabla_\theta \log \pi_\theta(\boldsymbol{o} \mid \boldsymbol{c})] \quad (13)$$

(Williams, 1992; Schulman et al., 2017), where $\boldsymbol{c} \sim \mathcal{D}$, $\boldsymbol{o} \sim \pi_\theta(\cdot \mid \boldsymbol{c})$, $\mathcal{D}$ is the prompt distribution, and the reward function $R(\boldsymbol{c}, \boldsymbol{o})$ is a scalar-valued function that evaluates the quality of the response $\boldsymbol{o}$ generated under the input $\boldsymbol{c}$.

C.3 Group Relative Policy Optimization (GRPO)

Following Sandwiched Policy Gradient (SPG; Wang et al. 2026), we adopt Group Relative Policy Optimization (GRPO; Shao et al. 2024). Let $\{\boldsymbol{o}^j\}_{j=1}^g$ denote the g responses generated by the model for an input prompt $\boldsymbol{c}$.

Objective for ARLM. Let $\mathcal{D}$ be the prompt distribution, $R(\boldsymbol{c}, \boldsymbol{o})$ the reward function, and π_θ the generation policy with parameters θ. Assuming on-policy updates and omitting clipping[20], the GRPO objective can be written as:

$$\mathcal{J}_{\text{group}}(\theta) = \mathbb{E}_{\boldsymbol{c},\boldsymbol{o}^1,\ldots,\boldsymbol{o}^g}\left[\frac{1}{g}\sum_{j=1}^{g} A^j \log \pi_\theta(\boldsymbol{o}^j \mid \boldsymbol{c})\right], \quad (14)$$

where $\boldsymbol{c} \sim \mathcal{D}$, $\boldsymbol{o}^1, \ldots, \boldsymbol{o}^g \overset{\text{i.i.d.}}{\sim} \pi_{\text{sg}[\theta]}(\cdot \mid \boldsymbol{c})$, and $A^j = R(\boldsymbol{c}, \boldsymbol{o}^j) - \frac{1}{g}\sum_{k=1}^{g} R(\boldsymbol{c}, \boldsymbol{o}^k)$. Here $\text{sg}[\theta]$ denotes the stop-gradient operator, which fixes the parameters at sampling time so that gradients flow only through the current policy π_θ.

Surrogate Objective for MDLM. Prior work (Zhu et al., 2026; Zhao et al., 2025) addresses the intractability of the log-likelihood of MDLM by using the ELBO (2), a lower bound on $\log \pi_\theta(\boldsymbol{o}^j \mid \boldsymbol{c})$, as a surrogate objective. With this surrogate, one can optimize the following objective, which replaces $\log \pi_\theta(\boldsymbol{o}^j \mid \boldsymbol{c})$ in (14) with $\mathcal{L}_{\text{ELBO}}$ in (2):

$$\mathcal{J}_{\text{MDLM}}(\theta) = \mathbb{E}_{\boldsymbol{c},\boldsymbol{o}^1,\ldots,\boldsymbol{o}^g}\left[\frac{1}{g}\sum_{j=1}^{g} A^j \mathcal{L}_{\text{ELBO}}(\boldsymbol{o}^j \mid \boldsymbol{c}; \theta)\right],$$

where $\boldsymbol{c} \sim \mathcal{D}$ and $\boldsymbol{o}^1, \ldots, \boldsymbol{o}^g \overset{\text{i.i.d.}}{\sim} \pi_{\text{sg}[\theta]}(\cdot \mid \boldsymbol{c})$.

Furthermore, when using blockwise generation, $\mathcal{L}_{\text{ELBO}}$ in (2) is replaced by the blockwise surro-

[20] GRPO originally includes importance ratio clipping as in PPO (Schulman et al., 2017), but under on-policy updates the importance ratio equals 1, so clipping can be omitted.

gate objective $\mathcal{L}_{\text{Block}}$ in (3):

$$\mathcal{J}_{\text{Block}}(\theta) = \mathbb{E}_{\boldsymbol{c},\boldsymbol{o}^1,\ldots,\boldsymbol{o}^g}\left[\frac{1}{g}\sum_{j=1}^{g} A^j \mathcal{L}_{\text{Block}}(\boldsymbol{o}^j \mid \boldsymbol{c};\theta)\right]. \tag{15}$$

In practice, since $\mathcal{L}_{\text{Block}}$ is approximated with a finite number of samples, the gradients are computed by replacing $\mathcal{L}_{\text{Block}}$ in (15) with its estimator $\hat{\mathcal{L}} \in \{\hat{\mathcal{L}}_{\text{Full}}, \hat{\mathcal{L}}_{\text{Rand}}\}$.

C.4 Intractability of MDLM's Log-Likelihood

The likelihood of an entire sequence under the MDLM generation process requires marginalization over all possible unmasking orders σ (Uria et al., 2014):

$$\log \pi_\theta(\boldsymbol{o} \mid \boldsymbol{c}) = \log \sum_{\sigma \in S_L} \pi_\theta(\boldsymbol{o}, \sigma \mid \boldsymbol{c}),$$

where S_L is the set of all unmasking orders for a sequence of length L. Since $|S_L|$ is on the order of $L!$, this marginalization is computationally prohibitive.

C.5 Mask State in Blockwise Generation

We define the set of all positions masked during the generation of block b as

$$\mathcal{S}_b(\mathcal{M}_{t_b}) = \mathcal{M}_{t_b} \cup \bigcup_{b'>b} \mathcal{B}_{b'}. \tag{16}$$

This set includes the masked positions within block b as well as all positions of subsequent blocks. The mask state of each position during the generation of block b is then given by

$$o^i_{b,\mathcal{M}_{t_b}} = \begin{cases} o^i & i \notin \mathcal{S}_b(\mathcal{M}_{t_b}), \\ M & i \in \mathcal{S}_b(\mathcal{M}_{t_b}). \end{cases} \tag{17}$$

That is, the tokens in blocks preceding b and the unmasked tokens within block b are retained, while all positions in $\mathcal{S}_b$ are masked.

C.6 Formal Definition of $\Omega_b(t_b)$

We define the set of all possible masked-position subsets within block b under masking ratio t_b as

$$\Omega_b(t_b) := \left\{\mathcal{M} \subseteq \mathcal{B}_b \;\middle|\; |\mathcal{M}| = \max(1, \lfloor t_b |\mathcal{B}_b| \rfloor)\right\}.$$

When L is divisible by B, we have $|\mathcal{B}_b| = B$, and $\Omega_b(t_b)$ is the set of all $\max(1, \lfloor t_b B \rfloor)$-element subsets of $\mathcal{B}_b$.

C.7 Probability That at Least One Block Is Never Selected

We derive the probability, used in Fig. 3b, that at least one block is never selected when sampling K times i.i.d. uniformly from C blocks.

For each block $b \in \{0, \ldots, C-1\}$, let A_b denote the event that block b is never selected in K samples. The desired probability is $\Pr\left(\bigcup_{b=0}^{C-1} A_b\right)$.

For each A_b, $\Pr(A_b) = \left(\frac{C-1}{C}\right)^K$, and for any set of s blocks $\{b_1, \ldots, b_s\}$,

$$\Pr\left(\bigcap_{m=1}^{s} A_{b_m}\right) = \left(\frac{C-s}{C}\right)^K.$$

Applying the inclusion-exclusion principle,

$$\Pr\left(\bigcup_{b=0}^{C-1} A_b\right) = \sum_{s=1}^{C} (-1)^{s+1} \binom{C}{s} \left(\frac{C-s}{C}\right)^K. \tag{18}$$

For example, when $K = C = 4$, (18) gives $\Pr \approx 0.91$, which is consistent with Fig. 3b.

D Variance Analysis for $K \neq C$

The estimator $\hat{\mathcal{L}}_{\text{Full}}$ in (6) handles only $K = C$. We denote its extension to arbitrary $K \geq 1$ by $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ and derive the variance ratio relative to $\hat{\mathcal{L}}_{\text{Rand}}$. Following the notation of Section 4, we additionally define

$$\lambda := \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}}} \in [0, 1] \tag{19}$$

to denote the contribution of the between-group variance to the total variance.

D.1 Definition of $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$

Decompose the sample size K as $K = qC + r$, where $q := \lfloor K/C \rfloor \geq 0$ and $r := K \bmod C \in \{0, \ldots, C-1\}$, and define the estimator as the sum of a deterministic part and a random part:

$$\hat{\mathcal{L}}^{(K)}_{\text{Full}}(\boldsymbol{o} \mid \boldsymbol{c};\theta) := \frac{1}{K}\left[\sum_{b=0}^{C-1}\sum_{j=1}^{q} \ell_b^{(j)} + \sum_{k=1}^{r} \ell_{b_k}\right]. \tag{20}$$

In the first term (the deterministic part), for each block b we sample q pairs $(t_b^{(j)}, \mathcal{M}_{t_b}^{(j)})$ independently and i.i.d., and write the corresponding loss as $\ell_b^{(j)} := \ell_{b,t_b^{(j)},\mathcal{M}_{t_b}^{(j)}}$. In the second term (the random part), we draw r block indices $\{b_1, \dots, b_r\}$ uniformly **without replacement** from $\{0, \dots, C-1\}$. For each b_k, we sample $(t_{b_k}, \mathcal{M}_{t_{b_k}})$ independently and i.i.d., and compute the loss $\ell_{b_k} := \ell_{b_k,t_{b_k},\mathcal{M}_{t_{b_k}}}$. The total number of samples is $qC+r = K$ (only the deterministic part if $r = 0$; only the K without-replacement samples if $q = 0$).

D.2 Derivation of the Variance

The deterministic part $S_{\text{det}} := \sum_{b,j} \ell_b^{(j)}$ and the random part $S_{\text{rand}} := \sum_k \ell_{b_k}$ are independent because they use independent (t, M) sample sequences. We derive each variance.

Variance of the deterministic part. The qC samples $\ell_b^{(j)}$ are all independent, and using the relation $\sum_{b=0}^{C-1} \sigma_b^2 = C\bar{\sigma}^2_{\text{within}}$ from (8), we obtain

$$\text{Var}[S_{\text{det}}] = \sum_{b=0}^{C-1} q\sigma_b^2 = qC\bar{\sigma}^2_{\text{within}}. \tag{21}$$

Variance of the random part. Conditioned on $\mathbf{b} = (b_1, \dots, b_r)$, each $X_k := \ell_{b_k}$ is independent. Applying the law of total variance to $\sum_k X_k$,

$$\begin{aligned}\text{Var}[S_{\text{rand}}] &= \mathbb{E}\left[\sum_{k=1}^{r} \sigma_{b_k}^2 \mid \mathbf{b}\right] + \text{Var}\left[\sum_{k=1}^{r} \mu_{b_k}\right] \\ &= r\bar{\sigma}^2_{\text{within}} + \text{Var}\left[\sum_{k=1}^{r} \mu_{b_k}\right]. \end{aligned} \tag{22}$$

For the second term, using $\text{Var}[\mu_{b_k}] = \sigma^2_{\text{between}}$ and the without-replacement covariance $\text{Cov}[\mu_{b_k}, \mu_{b_{k'}}] = -\sigma^2_{\text{between}}/(C-1)$ for $k \neq k'$, we have

$$\begin{aligned}\text{Var}\left[\sum_{k=1}^{r} \mu_{b_k}\right] &= r\sigma^2_{\text{between}} - \frac{r(r-1)}{C-1}\sigma^2_{\text{between}} \\ &= \frac{r(C-r)}{C-1}\sigma^2_{\text{between}}, \end{aligned} \tag{23}$$

whence

$$\text{Var}[S_{\text{rand}}] = r\bar{\sigma}^2_{\text{within}} + \frac{r(C-r)}{C-1}\sigma^2_{\text{between}}. \tag{24}$$

Setting	Variance ratio vs $\hat{\mathcal{L}}_{\text{Rand}}$
General ($K = qC + r$)	$1 - \lambda + \lambda \cdot \frac{r(C-r)}{K(C-1)}$
$K = 1$	1
$K < C$ ($q = 0, r = K$)	$1 - \lambda \cdot \frac{K-1}{C-1}$
$K = mC$ ($m \geq 1, r = 0$)	$1 - \lambda$

Table 8: Variance ratio of $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ relative to $\hat{\mathcal{L}}_{\text{Rand}}$. Here $K = qC + r$ with $q = \lfloor K/C \rfloor$ and $r = K \bmod C \in \{0, \dots, C-1\}$, and λ is defined in (19). The case $K = C$ corresponds to $K = mC$ with $m = 1$.

Total variance. Since $S_{\text{det}} \perp S_{\text{rand}}$, combining (21) and (24) gives

$$\begin{aligned}\text{Var}\left[\hat{\mathcal{L}}^{(K)}_{\text{Full}}\right] &= \frac{1}{K^2}(\text{Var}[S_{\text{det}}] + \text{Var}[S_{\text{rand}}]) \\ &= \frac{\bar{\sigma}^2_{\text{within}}}{K} + \frac{r(C-r)}{K^2(C-1)}\sigma^2_{\text{between}}. \end{aligned} \tag{25}$$

Together with (10), the variance ratio can be written, with $r := K \bmod C \in \{0, \dots, C-1\}$, as

$$\begin{aligned}\frac{\text{Var}\left[\hat{\mathcal{L}}^{(K)}_{\text{Full}}\right]}{\text{Var}\left[\hat{\mathcal{L}}_{\text{Rand}}\right]} &= 1 - \lambda \cdot \frac{K(C-1) - r(C-r)}{K(C-1)} \\ &= 1 - \lambda + \lambda \cdot \frac{r(C-r)}{K(C-1)} \end{aligned} \tag{26}$$

The second term on the right-hand side is a non-negative correction term that vanishes only when $r = 0$. The variance ratio is therefore at most 1 for all $K \geq 1$.

D.3 Reduction to Special Cases

Expression (26) immediately yields the following special cases, summarized in Table 8.

$K = mC$ ($m \geq 1, r = 0$). Since $r(C-r) = 0$, the variance ratio equals $1 - \lambda$. This is a constant independent of K and m. Hence even as $K \to \infty$, the variance ratio remains at the same value as in the case $K = C$.

$K = C$ ($q = 1, r = 0$). This is the special case of $K = mC$ with $m = 1$. The variance ratio equals $1 - \lambda$, consistent with evaluating (12) in the main text at $K = C$.

$K < C$ ($q = 0, r = K$). From $K(C-1) - K(C-K) = K(K-1)$, the variance ratio equals $1 - \lambda \cdot (K-1)/(C-1)$. It equals 1 at $K = 1$ (matching $\hat{\mathcal{L}}_{\text{Rand}}$) and decreases monotonically in K.

D.4 Behavior of the Correction Term and the Variance Ratio as a Function of K

The correction term $\lambda \cdot r(C-r)/(K(C-1))$ in (26) vanishes only when $r = 0$, so the variance-ratio gap between our method and $\hat{\mathcal{L}}_{\text{Rand}}$ is maximized when K is a multiple of C (i.e., $K = mC$). When K deviates from a multiple of C ($r \neq 0$), the correction term becomes positive and the gap shrinks; for example, with $C = 4$ and $\lambda = 0.90$, the variance ratio is ≈ 0.10 at $K = 4$ but ≈ 0.28 at $K = 5$. Here $\lambda \approx 0.90$ is a representative value consistent with the empirical measurements in Table 2 (variance ratio $1 - \lambda \approx 0.08$–0.12). This occurs because the single random sample added at $K = C + 1$ reintroduces stochastic variation in the block index. Nevertheless, the variance ratio remains at most 1 for all $K \geq 1$.

E Empirical Verification for $K \neq C$

The special cases derived in Appendix D.3 describe the behavior of the variance ratio between $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ for $K \leq C$ and for multiples of C: the ratio equals 1 at $K = 1$ (the two estimators coincide), decreases monotonically to $1-\lambda$ at $K = C$, and remains at the constant value $1 - \lambda$ independently of K and m for $K = mC$ with $m \geq 2$. This section reports training experiments on both sides of the boundary $K = C$ to examine whether the advantage of our method depends on the specific setting $K = C = 4$.

E.1 Left of the Boundary: Empirical Evidence for $K \leq C$

The special case $K \leq C$ ($q = 0$, $r = K$) predicts that the variance ratio decreases monotonically from 1 at $K = 1$ to $1 - \lambda$ at $K = C$. This subsection reports a training experiment at the point $K = 2, C = 4$.

We newly train the baselines (SPG and $\hat{\mathcal{L}}_{\text{Rand}}$) with $K = 2$ and the number of training iterations $I = 8{,}000$, and reuse the result of our method at $K = C = 4$, $I = 4{,}000$ from Table 1. The baselines perform 4,000 updates, as in the main experiment, and in each update they compute two independent $K = 2$ estimates and sum the resulting gradients. Thus, $I = 8{,}000$ is the total number of these estimates. The total number of block evaluations $K \cdot I = 16{,}000$ is matched across the two settings[21], to examine whether the baselines catch up with our method when K is halved and the number of iterations is doubled. All other hyperparameters are identical to those in Section 5 and Appendix A.1; only the baselines' K and I are changed.

As shown in Table 9, $\hat{\mathcal{L}}_{\text{Full}}$ (avg. 62.93) outperforms both $\hat{\mathcal{L}}_{\text{Rand}}$ (56.27) and SPG (60.43) on all five benchmarks, and the advantage is preserved under a matched number of block evaluations.[22]

E.2 Right of the Boundary: Empirical Evidence for $K > C$ (Oversampling)

The setting $K > C$ increases the number of block evaluations per iteration beyond C and thus incurs higher compute cost, so it is not normally used. However, this subsection also examines the $K > C$ regime. The $K = mC$ special case states that the variance ratio is fixed at $1 - \lambda$ independently of K and m. We examine two points with $C = 4$ fixed: a point without the correction term, $K = mC$ ($K = 8$, $m = 2$; the $K = mC$ special case predicts that the variance ratio returns to $1-\lambda$, the same as at $K = C$), and a point with a positive correction term predicted by Appendix D.4 ($K = 6$, $r = 2$; the variance ratio is larger than $1-\lambda$ but still strictly smaller than 1).

Table 10 reports the accuracy comparison between $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ on GSM8K and MATH-500 at the two points. At $K = 8$, the variance ratio between $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ remains at $1 - \lambda$ as predicted by the $K = mC$ theory, while the empirical accuracy gap in Table 10 is smaller than at $K = 4$ (GSM8K $+6.06 \rightarrow +0.46$; MATH-500 $+3.60 \rightarrow +0.20$). At $K = 6$, the accuracy gap on GSM8K is smaller than at $K = 8$. On GSM8K specifically, both methods reach the same accuracy of 80.89 at $K = 6$; here, $\hat{\mathcal{L}}_{\text{Rand}}$ approaches the GSM8K performance level attained by $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ across this K range (within a ± 1-point band around 81%). MATH-500, by contrast, still shows a $+2.20$-point advantage for $\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ at $K = 6$. In summary, within the settings we examined, the gain of our method is largest in the low-sample regime $K = C$ and generally shrinks as the sample size K grows beyond the number of blocks C.

[21] In general, for the proposed method with iteration count I_{Full} and the random block sampling baseline with iteration count I_{Rand}, equalizing the total number of block evaluations corresponds to $C \cdot I_{\text{Full}} = K \cdot I_{\text{Rand}}$, i.e., $I_{\text{Rand}} = (C/K) \cdot I_{\text{Full}}$. For example, if $C = 8$, $K = 2$, then $I_{\text{Rand}} = 4 \cdot I_{\text{Full}}$.

[22] Because the training settings differ from those of Table 1, the baseline values in Table 9 do not match those in Table 1.

Method	GSM8K	MATH-500	Countdown	Sudoku	HumanEval	Avg.
SPG	81.88	35.20	64.84	89.76	30.49	60.43
$\hat{\mathcal{L}}_{\text{Rand}}$	81.58	33.40	54.30	80.96	31.10	56.27
$\hat{\mathcal{L}}_{\text{Full}}$ (ours)	**82.26**	**38.00**	**66.41**	**91.39**	**36.59**	**62.93**

Table 9: Empirical verification for $K < C$. Baselines (SPG and $\hat{\mathcal{L}}_{\text{Rand}}$) are re-trained with $K = 2$, $C = 4$, $I = 8{,}000$, and our method $\hat{\mathcal{L}}_{\text{Full}}$ uses $K = C = 4$, $I = 4{,}000$ (reused from Table 1). Both settings have the same total number of block evaluations $K \cdot I = 16{,}000$.

	$K = 4$		$K = 6$		$K = 8$	
Method	GSM8K	MATH-500	GSM8K	MATH-500	GSM8K	MATH-500
$\hat{\mathcal{L}}_{\text{Rand}}$	76.20	34.40	80.89	34.00	80.97	34.00
$\hat{\mathcal{L}}^{(K)}_{\text{Full}}$ (ours)	**82.26**	**38.00**	80.89	**36.20**	**81.43**	**34.20**

Table 10: Empirical verification for $K > C$ ($C = 4$ fixed, $I = 4{,}000$). The $K = 4$ column is reused from Table 1. The $K = 6$ and $K = 8$ columns are new experiments.

F Generalization to a Doubled Block Count C

The variance-reduction derivation in Section 4 treats C as a general parameter and does not depend on its specific value. In this section, we re-train $\hat{\mathcal{L}}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ on GSM8K and MATH-500 with $C = K = 8$ and compare the results with the main experiment ($C = K = 4$) in Table 11, showing that the advantage of our method is preserved when C is doubled. Code, Countdown, and Sudoku are excluded from this verification due to limited computation time and resources.

As shown in Table 11, $\hat{\mathcal{L}}_{\text{Full}}$ outperforms $\hat{\mathcal{L}}_{\text{Rand}}$ on both GSM8K and MATH-500 at $C = K = 8$ as well, showing that the advantage of our method is preserved when C is doubled.

G Detailed Estimation Quality Analysis

This section extends the estimation quality analysis conducted on GSM8K in Section 7 to all five datasets (GSM8K, MATH-500, Countdown, Sudoku, and Code). The experimental setup is identical to Section 7. We first report the overall statistics of estimation errors (mean, standard deviation, and median) and compare per-sample variances in tables (Appendix G.1). We then present five types of analysis results for each dataset in figures (Appendix G.2): Spearman rank correlation ρ and RMSE with respect to the reference value (estimation quality), tail probability $\Pr(|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}| > \tau)$ (frequency of large estimation errors), the density distribution of estimation errors, and scatter plots of per-sample variances.

G.1 Estimation Error Statistics

Table 12 shows the statistics of the estimation error $|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}|$ across all five datasets. Across all datasets, $\hat{\mathcal{L}}_{\text{Full}}$ achieves substantially lower mean error and median compared with $\hat{\mathcal{L}}_{\text{Rand}}$. The improvement is particularly notable on Sudoku, where the mean error is reduced approximately fourfold from 105.95 to 26.42. On Code, the absolute errors are larger than on the other datasets, which is likely due to the longer generation length ($L = 512$); nevertheless, $\hat{\mathcal{L}}_{\text{Full}}$ improves the mean error of $\hat{\mathcal{L}}_{\text{Rand}}$ by approximately threefold.

Table 13 compares the per-sample variances of the absolute error. The proportion of samples for which this variance is lower for $\hat{\mathcal{L}}_{\text{Full}}$ than for $\hat{\mathcal{L}}_{\text{Rand}}$ is at least 98.9% across all datasets, reaching 100.0% on Sudoku. The ratio of the mean variances ranges from $7.35\times$ to $11.28\times$, and all differences are statistically significant under the Wilcoxon signed-rank test with $p < 10^{-100}$.

G.2 Per-Dataset Details

Fig. 5 presents all results as a single figure, with five datasets as rows and five types of analysis as columns. We describe the first two columns and the last two columns as grouped pairs below; the Error tail column (the third column) is detailed in Appendix G.3.

Estimation quality (Spearman ρ and RMSE). The first two columns plot the Spearman rank correlation ρ and RMSE between each estimator and the reference value $\mathcal{L}_{\text{ref}}$ as a function of the Monte Carlo sample count K. Both quantities compare the K required for $\hat{\mathcal{L}}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ to achieve com-

	$C = K = 4$		$C = K = 8$	
Method	GSM8K	MATH-500	GSM8K	MATH-500
$\hat{\mathcal{L}}_{\text{Rand}}$	76.20	34.40	80.52	31.80
$\hat{\mathcal{L}}_{\text{Full}}$ (ours)	**82.26**	**38.00**	**81.35**	**36.80**

Table 11: Empirical verification of doubling C. Comparison between the main experiment ($C = K = 4$) and the re-trained models with $C = K = 8$. The $C = K = 4$ columns are reused from Table 1. All other hyperparameters are identical to those of the main experiment (LLaDA-8B-Instruct, $I = 4{,}000$). Code, Countdown, and Sudoku are excluded due to limited computation time and resources. **Bold** indicates the higher value within each $C = K$ setting.

		Mean	Std	Median
GSM8K	$\hat{\mathcal{L}}_{\text{Rand}}$	109.32	85.14	90.81
	$\hat{\mathcal{L}}_{\text{Full}}$	36.25	31.82	28.85
MATH	$\hat{\mathcal{L}}_{\text{Rand}}$	118.14	90.89	99.45
	$\hat{\mathcal{L}}_{\text{Full}}$	36.64	33.32	29.18
Countdown	$\hat{\mathcal{L}}_{\text{Rand}}$	131.66	98.14	113.46
	$\hat{\mathcal{L}}_{\text{Full}}$	37.82	32.09	30.90
Sudoku	$\hat{\mathcal{L}}_{\text{Rand}}$	105.95	80.04	92.64
	$\hat{\mathcal{L}}_{\text{Full}}$	26.42	23.79	20.91
Code	$\hat{\mathcal{L}}_{\text{Rand}}$	297.49	233.67	247.40
	$\hat{\mathcal{L}}_{\text{Full}}$	99.69	86.04	80.44

Table 12: Statistics of the estimation error $|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}|$ ($K = 4$, 50 independent repetitions). The reference value is computed with $K = 1024$.

	Full $<$ Rand (%)	Var. ratio
GSM8K	99.5	7.35×
MATH	98.9	7.47×
Countdown	99.7	9.71×
Sudoku	100.0	11.28×
Code	99.6	7.36×

Table 13: Comparison of per-sample variances of the absolute error $|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}|$ ($K = 4$, computed from 50 independent repetitions). **Full $<$ Rand** denotes the proportion of samples for which $\hat{\mathcal{L}}_{\text{Full}}$ has lower variance than $\hat{\mathcal{L}}_{\text{Rand}}$. **Var. ratio** denotes the ratio of the mean variances ($\hat{\mathcal{L}}_{\text{Rand}}/\hat{\mathcal{L}}_{\text{Full}}$). Wilcoxon signed-rank test $p < 10^{-100}$ for all datasets.

parable estimation quality, with the star marker indicating the result of $\hat{\mathcal{L}}_{\text{Full}}$ at $K = C = 4$. On all five datasets, the $\hat{\mathcal{L}}_{\text{Full}}$ star marker lies at a position where $\hat{\mathcal{L}}_{\text{Rand}}$ requires a much larger K (roughly an order of magnitude larger) to reach comparable Spearman rank correlation and RMSE. This means that the effect observed on GSM8K—"making block-index selection deterministic improves estimation quality"—is consistent across all datasets.

Per-prompt error distribution and variance. The last two columns visualize per-prompt error and variance distributions. Error dist. (a) shows the density distribution of the estimation error $|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}|$ (with the vertical axis on a log scale), comparing the heaviness of the error tail. Error dist. (b) plots the per-prompt variances of the absolute error computed from 50 iterations as a scatter plot of $\hat{\mathcal{L}}_{\text{Rand}}$ vs. $\hat{\mathcal{L}}_{\text{Full}}$; points lying below the diagonal $y = x$ indicate smaller per-sample variance for $\hat{\mathcal{L}}_{\text{Full}}$. In Error dist. (a), the $\hat{\mathcal{L}}_{\text{Full}}$ distribution concentrates toward smaller errors on all five datasets, with the heavy tail suppressed. In Error dist. (b), the vast majority of prompts lie below the diagonal $y = x$, showing that the per-sample variance of $\hat{\mathcal{L}}_{\text{Full}}$ is smaller than that of $\hat{\mathcal{L}}_{\text{Rand}}$ on a per-prompt basis. Together, these results indicate that the advantage of our method holds not only at the average level but also at the per-prompt level.

G.3 Tail Probability of Large Estimation Errors

Whereas Spearman ρ and RMSE summarize the overall trend of the error distribution, the tail probability discussed in this subsection directly compares the frequency of rare large estimation errors. Specifically, for each estimator $\hat{\mathcal{L}} \in \{\hat{\mathcal{L}}_{\text{Full}}, \hat{\mathcal{L}}_{\text{Rand}}\}$, we evaluate the tail probability $\Pr(|\hat{\mathcal{L}} - \mathcal{L}_{\text{ref}}| > \tau)$ at each threshold τ. Measurements are computed with $K = 4$ from 50 independent iterations of each prompt, using the absolute error against the reference value. The third column (Error tail) of Fig. 5 shows the results on all five datasets. For example, on GSM8K at $\tau = 100$, $\hat{\mathcal{L}}_{\text{Rand}}$ exceeds the threshold with probability 45.9%, while $\hat{\mathcal{L}}_{\text{Full}}$ does so with only 4.4%; the ratio between the two grows from $1.4\times$ at $\tau = 20$ to $10.5\times$ at $\tau = 100$ as the threshold increases. This shows that the error distribution of $\hat{\mathcal{L}}_{\text{Rand}}$ has a heavier tail, and the same trend holds on the other datasets.

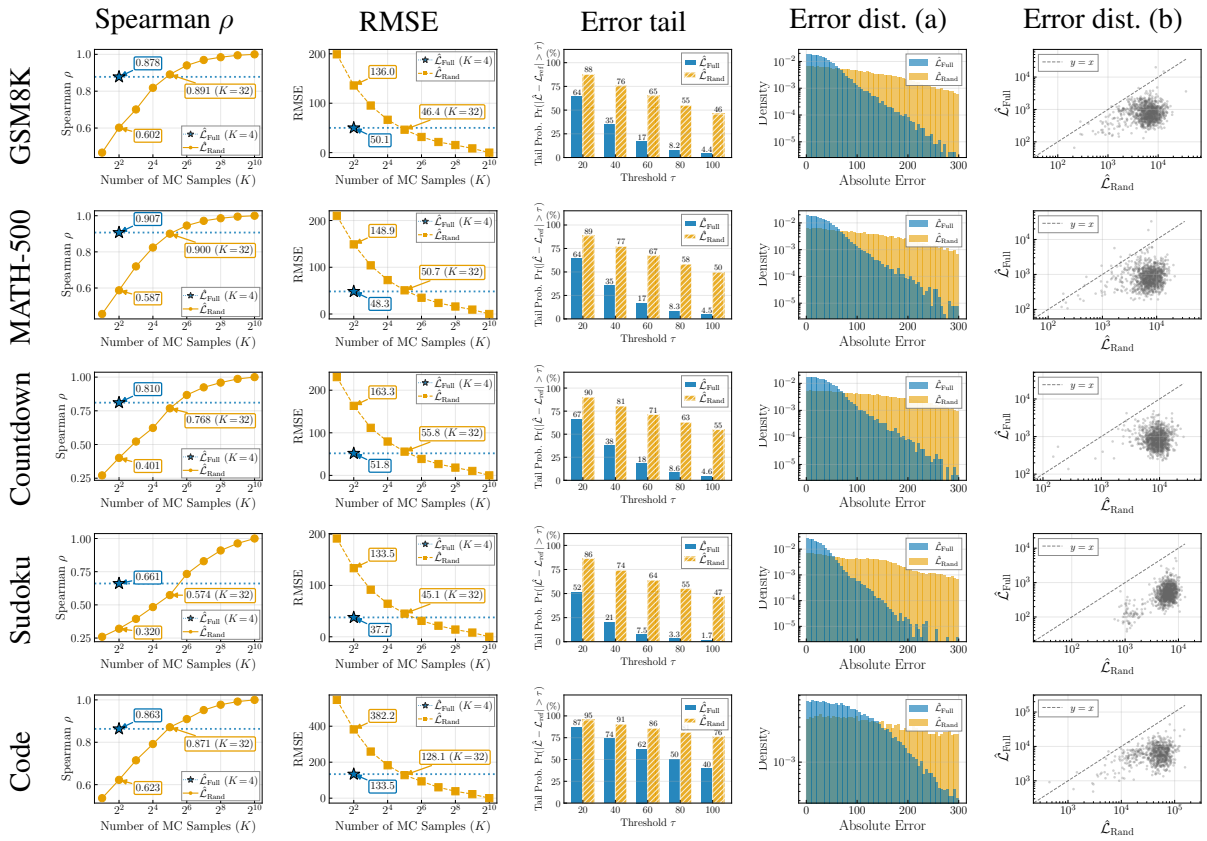


Figure 5: Detailed estimation quality analysis across five datasets. Rows correspond to datasets (GSM8K, MATH-500, Countdown, Sudoku, and Code); columns correspond to the five analysis types described in the introduction of this section.

H Variance Reduction at the Policy-Gradient Level

This section examines, through both a theoretical argument and empirical measurements, how far the variance reduction demonstrated at the estimator level carries over to the between-group variance of the policy gradient. Here, the between-group variance refers to the variance arising from the stochastic selection of the block index $b \in \{0, \ldots, C-1\}$, in analogy with the loss-level definition $\sigma^2_{\text{between}}$ in (7).

H.1 From the Estimator to the Policy Gradient

The variance reduction shown so far in Section 4 and Section 7 is an effect on the estimator $\hat{\mathcal{L}}$ itself. Training stability depends on the variance of the policy gradient $\nabla_\theta \mathcal{J}_{\text{Block}}$, so we examine the relationship between the two. By making block selection deterministic, our method eliminates the between-group variance from both the loss and the gradient. The GRPO advantage A^j differs across responses j, but the same A^j multiplies every block sample within a single response, so it does not affect the between-group variance ratio. In our experiments $\beta = 0$ (Appendix A.1, Table 3), so the KL regularization term also does not contribute. Therefore, the variance reduction shown at the estimator level is preserved for the between-group variance of the policy gradient as well, in the sense that the variance ratio is at most 1.

H.2 Experimental Setup of the Empirical Measurement

The argument above is a theoretical prediction; whether the sample variance of the gradient is actually reduced in line with this prediction during real training is an empirical question that must be examined independently. Table 14, reported in this section, and Table 15, reported in Appendix I, are two analyses that extract different quantities from a common measurement procedure. Specifically, for each training step in $\{1{,}000, 2{,}000, 3{,}000, 4{,}000\}$ of the $\hat{\mathcal{L}}_{\text{Full}}$ run in the main experiment (LLaDA-8B-Instruct, $C = K = 4$), we sample $N = 100$ prompts from

	Training step			
Task	1,000	2,000	3,000	4,000
GSM8K	0.811	0.915	0.889	0.961
MATH-500	0.791	0.926	0.825	0.915

Table 14: Sample variance ratio of the policy gradient $\mathrm{Var}[\hat{\mathcal{G}}_{\text{Full}}]/\mathrm{Var}[\hat{\mathcal{G}}_{\text{Rand}}]$. Computed for the four training checkpoints and the two tasks of the main experiment ($C = K = 4$), aggregating over the remaining prompts after excluding those with $A^j = 0\ \forall j$ from the $N = 100$ samples. Each prompt contributes $S = 50$ gradient samples. A ratio below 1 indicates that the variance of Full is smaller than that of Rand.

	Training step			
Task	1,000	2,000	3,000	4,000
GSM8K	0.115	0.113	0.138	0.134
MATH-500	0.095	0.155	0.104	0.132

Table 15: Estimator variance ratio $1 - \lambda = \bar{\sigma}^2_{\text{within}}/(\sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}})$ ($K = C = 4$) at the mid-training checkpoints of the main experiment. Computed from the variance decomposition (12) of the per-block log-likelihood $\ell_{b,t_b,\mathcal{M}_{t_b}}$ and aggregated over the prompts excluding those with $A^j = 0\ \forall j$. Directly comparable to Table 2 on the base model. Smaller values indicate larger variance reduction.

the GSM8K and MATH-500 evaluation sets at each checkpoint. For each prompt, we generate $g = 4$ responses from the policy (temperature $= 0.9$, identical to the GRPO sampling of the main experiment), compute rewards using the rule-based reward function, and obtain the unnormalized GRPO advantage $A^j = R^j - \bar{R}$. In evaluation mode (dropout off), for each prompt we independently sample $\hat{\mathcal{L}}_{\text{Full}}$ and $\hat{\mathcal{L}}_{\text{Rand}}$ $S = 50$ times, and denote the corresponding policy-gradient samples by $\hat{\mathcal{G}}_{\text{Full}}$ and $\hat{\mathcal{G}}_{\text{Rand}}$ ($\nabla_\theta \mathcal{J}_{\text{Block}}$ computed using the Full and Rand estimators, respectively). We also record the per-block log-likelihood $\ell_{b,t_b,\mathcal{M}_{t_b}}$. Prompts for which all g responses receive the same reward yield $A^j = 0\ \forall j$ and a zero policy gradient; the values reported in Table 14 and Table 15 therefore aggregate over the remaining prompts after excluding these.

H.3 Empirical Variance Ratio of the Policy Gradient

Using the $S = 50$ gradient samples $\hat{\mathcal{G}}_{\text{Full}}$ and $\hat{\mathcal{G}}_{\text{Rand}}$ from the experimental setup, we compute the per-prompt gradient variance and report the ratio $\mathrm{Var}[\hat{\mathcal{G}}_{\text{Full}}]/\mathrm{Var}[\hat{\mathcal{G}}_{\text{Rand}}]$ averaged over the remaining prompts (after exclusion) in Table 14. As shown in Table 14, the ratio is below 1 at every measured checkpoint. However, the ratio ranges from 0.79 to 0.96, which is closer to 1 than the estimator-level ratio of 0.10 to 0.16 measured at the same checkpoints (Table 15).

I Estimator Variance Ratio at Mid-Training Checkpoints

Table 2 in Section 7.3 reports the estimator variance ratio $\mathrm{Var}[\hat{\mathcal{L}}_{\text{Full}}]/\mathrm{Var}[\hat{\mathcal{L}}_{\text{Rand}}]$ in (12) on the base model. In this section, we compute the same quantity for four mid-training checkpoints $\times$ two tasks. The measurement setup is shared with Appendix H.2: using the $S = 50$ samples of $\hat{\mathcal{L}}_{\text{Full}}$, we estimate the per-block variance and mean via the same procedure as in Section 7.3, and report the variance ratio $1 - \lambda = \bar{\sigma}^2_{\text{within}}/(\sigma^2_{\text{between}} + \bar{\sigma}^2_{\text{within}})$ at $K = C = 4$ in Table 15. Directly comparing Table 2 (base model) with Table 15 (mid-training checkpoints) lets us check whether the variance-ratio expression takes a comparable value at mid-training checkpoints.

As shown in Table 15, the variance ratio lies in the range 0.10–0.16 across all checkpoints and both tasks; it is close to the values 0.08–0.12 on the base model in Table 2 and well below 1, so the variance-reduction effect of $\hat{\mathcal{L}}_{\text{Full}}$ is preserved at every measured checkpoint during training.

J Relation to Classical Sampling Techniques

This section elaborates on the relation to classical sampling techniques mentioned in Section 8.

Relation to stratified sampling. Stratified sampling is a classical variance-reduction technique that partitions the population into non-overlapping strata and draws samples independently from each stratum. Our $\hat{\mathcal{L}}_{\text{Full}}$ at $K = C$ is equivalent to stratified sampling with one sample drawn per stratum. The contribution of this work is to introduce the perspective of Monte Carlo estimator design into the blockwise generation structure of MDLM.

Extension to importance sampling. A natural extension of stratified sampling is importance sampling based on block variance. The idea is to concentrate weight on high-variance blocks to lower the overall estimation variance, but in the context of policy optimization a different issue can arise. When the weights concentrate on a small

number of blocks, the policy is barely updated on the other blocks; learning becomes imbalanced, and the policy adapts only to those specific blocks, which risks policy collapse (a phenomenon in which over-adaptation to specific states degrades policy performance). This paper adopts uniform full coverage as a more stable choice, and stabilizing weighted-sampling strategies remains future work.