CD^{4}LM: Consistency Distillation and aDaptive Decoding for Diffusion Language Models

Anonymous ACL submission

Abstract

Diffusion language models (DLMs) accelerate generation by decoding multiple tokens in parallel, but pushing them to **fast decoding often sacrifices accuracy**. We attribute this to a static-to-dynamic mismatch: training optimizes local denoising under fixed schedules, while fast inference skips steps and encounters irregular mask patterns. Existing accelerators largely address only one side of this problem. Training-free inference methods speed up decoding without changing the model, leaving the mismatch unresolved; training-side distillation methods adapt the model, but often tie the student to a specific decoding trajectory, limiting composability and still suffering accuracy loss. We propose **CD^{4}LM**, a composable framework for accurate DLM acceleration. Its core training component, *Discrete-Space Consistency Distillation (DSCD)*, trains a trajectory-invariant student on diverse masked states, reducing the mismatch without fixing a decoding policy. We pair DSCD with a lightweight *Confidence-Adaptive Decoder (CAD)*, while keeping the trained student compatible with external inference-side accelerators. On GSM8K, **CD^{4}LM** preserves accuracy while achieving a 5.18× speedup; stacking **CD^{4}LM** with per-step cache accelerators further pushes speedup to 9.73× at teacher-level accuracy, suggesting a scalable path for combining training-side robustness with future decoding advances.

1 Introduction and Related Work

Diffusion language models (DLMs) such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) offer a compelling alternative to autoregressive (AR) language models (Achiam et al., 2023; Grattafiori et al., 2024) by generating text through iterative denoising rather than strict left-to-right prediction. Because each denoising step can update multiple masked tokens in parallel, DLMs promise to replace the sequential dependency of AR decoding with a smaller number of parallel refinement steps. This promise is especially attractive for structured generation tasks such as mathematical reasoning and code synthesis (Hendrycks et al., 2021; Austin et al., 2021b; Chen, 2021), where long outputs and multi-step derivations make decoding latency a practical bottleneck.

In practice, the parallelism promised by DLMs only materializes when the number of function evaluations (NFE) is aggressively reduced, and pushing DLMs to low-NFE decoding causes substantial accuracy degradation. We attribute this to a *static-to-dynamic mismatch*: training optimizes local denoising under fixed corruption schedules, while fast inference skips intermediate states and commits multiple tokens at once, inducing irregular mask patterns far from those seen during training. As a result, the model is then asked to refine off-schedule masked configurations for which its conditional predictions and confidence estimates are poorly calibrated.

Existing DLM accelerators tackle this from two largely separate directions, but neither simultaneously delivers speed, accuracy, and composability. *Training-free inference accelerators*—including key-value (KV) caching (Liu et al., 2025; Jiang et al., 2025), confidence thresholding (Wu et al., 2025; Sahoo et al., 2024) and parallel unmasking heuristics (Israel et al., 2025; Christopher et al., 2025; Wei et al., 2025; Wang et al., 2025)—are plug-and-play across backbones, but the mismatch remains because the base model is unchanged, capping attainable speedup whenever accuracy must be preserved. *Training-side distillation methods* adapt the model itself for faster generation, but they typically train the student along a particular trajectory (Deschenaux and Gulcehre, 2025; Zhang et al., 2026), certainty schedule (Chen et al., 2025b), attention pattern (Kim et al., 2025), rollout order (Qian et al., 2026), or decoding policy. This creates a *trajectory-locked* student that becomes

difficult to combine with inference-side accelerators inducing different intermediate states, and a noticeable accuracy gap to the teacher typically remains.

We propose that DLM acceleration should be designed around *composability*: a training-side accelerator should improve robustness to fast decoding without prescribing how decoding must be performed, turning training-side acceleration into a reusable layer that the inference loop can choose, adapt, or stack.

Following this principle, we introduce **CD^{4}LM**, a composable framework for accurate DLM acceleration. Its training component, Discrete-Space Consistency Distillation (DSCD), trains a student to be *trajectory-invariant* via teacher-subset masking. The student observes a more heavily masked view, while the teacher conditions on a lighter masked view that contains a superset of the student's visible information. The student is trained to match the teacher's conditional posterior while remaining anchored to a ground-truth reconstruction objective. This exposes the student to a broad family of off-schedule mask patterns and reduces the mismatch without binding the model to any particular decoding trajectory.

We pair DSCD with a lightweight Confidence-Adaptive Decoder (CAD), which uses the student's confidence to decide how many tokens to commit at each denoising step: High-confidence tokens are finalized early, uncertain tokens remain masked for further refinement, which converts fixed schedules into a dynamic risk-efficiency trade-off. Because DSCD preserves full bidirectional attention and hard-codes no rollout sequence, the same student also drops into external inference-side accelerators such as KV caching and parallel decoding.

Empirically, **CD^{4}LM** reaches 77.6% on GSM8K at a 5.18× wall-clock speedup (Table 1). Stacking the same student on top of per-step cache accelerators further pushes the frontier: **CD^{4}LM**+dLLM-Cache (Liu et al., 2025) reaches higher speed and **CD^{4}LM**+d^{2}Cache (Jiang et al., 2025) reaches higher accuracy, with both composed configurations Pareto-dominating the strongest standalone training and inference-side accelerators we evaluate in Table 1. A single **CD^{4}LM** student also supports both block-diffusion and pure-diffusion decoding (Appendix G.1). We also report transfer to DiffuCoder (Gong et al., 2025) in Appendix G.3.

Our contributions are as follows.

1. *A composability-first formulation.* We frame DLM acceleration around the static-to-dynamic mismatch and argue that training-side acceleration should serve as a composable robustness layer rather than a replacement for inference-side acceleration.

2. *A trajectory-invariant training and adaptive decoding framework.* We propose **CD^{4}LM** = DSCD+CAD. DSCD trains a student over diverse teacher-subset masked states without committing to a fixed decoding path; CAD uses confidence-guided commitment to allocate denoising compute adaptively. We further give a theoretical view of DSCD as a Bayes-optimal projection of the teacher's belief state onto the student's information set, with a Rao–Blackwellized posterior-risk analysis of the teacher-subset target.

3. *Strong accuracy-latency trade-offs and empirical composability.* **CD^{4}LM** reaches 77.6% on GSM8K at 5.18× speedup. Stacking with reproduced per-step cache accelerators further pushes speedup to **9.73**× (**CD^{4}LM**+dLLM-Cache, 77.6%) and accuracy to **78.0%** (**CD^{4}LM**+d^{2}Cache, 8.96×), with both composed configurations Pareto-dominating every standalone baseline reported in Table 1. A single distillation pass also supports both block-diffusion and pure-diffusion decoding (Appendix G.1). Transfer to DiffuCoder is reported in Appendix G.3.

An extended discussion of related work, covering AR and DLMs, training objectives and distillation for DLMs, inference-side parallel decoding and caching, and DLMs for code and mathematical reasoning, is provided in Appendix H.

2 Method

CD^{4}LM is designed to resolve the structural misalignment between fixed diffusion training schedules and the need for flexible, budget-aware inference. To bridge this gap, we propose a framework built upon an absorbing-state discrete diffusion backbone (formal definitions and full notation provided in Appendix A). Our approach consists of two core components: A DSCD scheme (Sect. 2.1) that trains a student to be approximately trajectory-invariant along the teacher's masking process, and a CAD policy (Sect. 2.2) that dynamically allocates NFE across tokens and instances. Taken together,

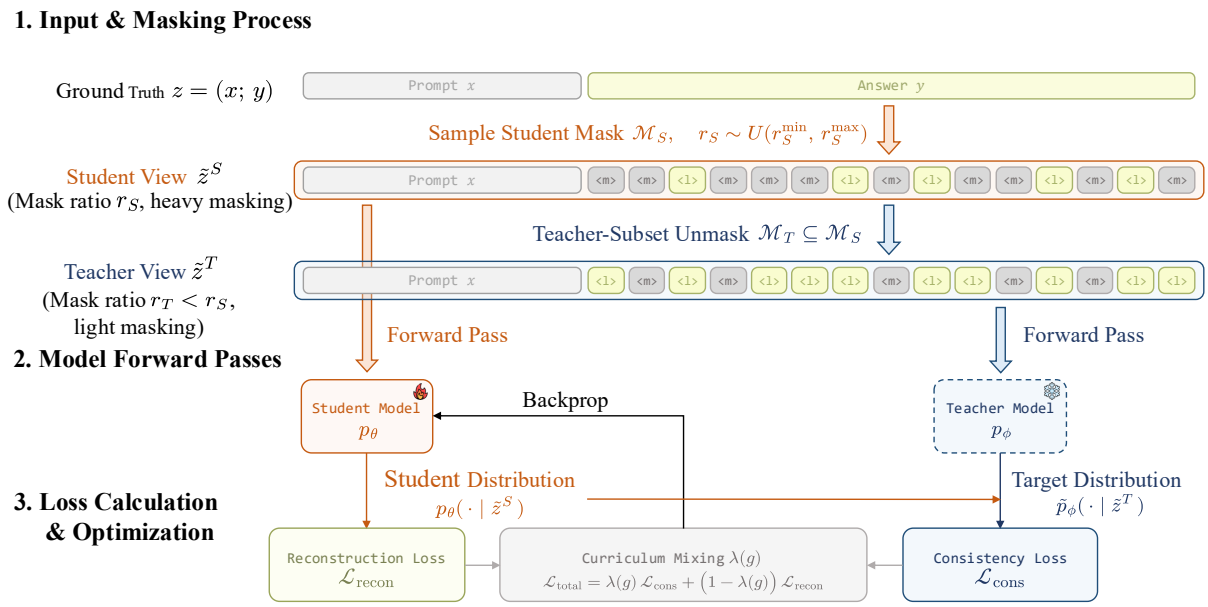

Figure 1: **Overview of the DSCD pipeline.** Teacher-subset masking ($\mathcal{M}_T \subseteq \mathcal{M}_S$) lets the teacher condition on a strict superset of the student's context. The student is trained jointly under $\mathcal{L}_{\text{recon}}$ (ground-truth anchor) and $\mathcal{L}_{\text{cons}}$ (alignment with the teacher's soft targets), with curriculum weight $\lambda(g)$ governing the transition from distillation to refinement.

these modules decouple the trained model from any fixed diffusion schedule and enable flexible low-NFE decoding. Furthermore, because no specific decoding pattern is embedded in the student, the resulting model is directly composable with inference-side accelerators (Sect. 4.1).

2.1 Discrete-Space Consistency Distillation

Given a training example consisting of a prompt x and a target response y (concatenated as $z = [x; y]$), our goal is to train a student model p_θ to predict the clean z from a corrupted state $\tilde{z}$ in a few steps, without being tied to any particular diffusion schedule. Unlike the teacher p_ϕ which follows a fixed denoising schedule, we train p_θ to be approximately *trajectory-invariant* along the absorbing diffusion process of the teacher.

Ideal trajectory invariance. For a clean sequence z and its absorbing diffusion trajectory $\{\tilde{z}_t\}_{t\in[0,1]}$, generated by the absorbing masking process (defined in Appendix A), an ideal consistency-style student would satisfy

$$p_\theta(z \mid \tilde{z}_t) \;\approx\; p_\theta(z \mid \tilde{z}_{t'}) \;\approx\; p_\phi(z \mid \tilde{z}_t, t) \quad \text{for all } t, t' \in [0, 1]. \tag{1}$$

Unlike the teacher p_ϕ, which conditions on explicit time t, the student p_θ learns to infer the effective noise level directly from the masked input $\tilde{z}$, enabling time-agnostic inference. This requirement is too strong to enforce directly: It couples all time points on the trajectory and requires matching full sequence distributions. In practice, DSCD implements a weaker but tractable surrogate that enforces *pairwise* consistency between stochastic views of the same z and anchors them to the data distribution.

Paired teacher-subset masking. For each training example, we first sample a student mask ratio $r_S \sim \mathcal{U}(r_S^{\min}, r_S^{\max})$ and then set a lighter teacher ratio $r_T = r_S \cdot u$, where $u \sim \mathcal{U}(u_{\min}, u_{\max})$ with $u_{\max} < 1$ (thus $r_T < r_S$). We convert each ratio to its diffusion timestep using the same noise schedule, yielding t_S and t_T. Let $n_S = \lfloor L_y r_S \rfloor$ and $n_T = \lfloor L_y r_T \rfloor$. We form the student masked set $\mathcal{M}_S$ by uniformly sampling (without replacement) n_S target positions from $\{L_x + 1, \ldots, L\}$. To construct the teacher mask, we then uniformly sample a subset $\mathcal{M}_T \subseteq \mathcal{M}_S$ of size n_T. This nested masking makes the teacher informationally richer (it conditions on a strict superset of the student's visible context). Hence, the consistency loss is evaluated on $\mathcal{M}_S$. The overall training pipeline is depicted in Fig. 1.

At a high level, DSCD then combines a reconstruction loss and a Kullback-Leibler (KL)

divergence-based consistency loss computed on $\mathcal{M}_S$ to (i) anchor the student to the data distribution at its own masked positions and (ii) align the student with the teacher's softer predictions under the same corruption pattern, yielding a practical relaxation of the trajectory-invariance property in (1). This constraint ensures the teacher always conditions on a superset of the student's information, providing a lower-variance distillation target (theoretical proof provided in Appendix C).

Training Objective. We minimize a joint objective $\mathcal{L}_{\text{total}} = \lambda(g)\mathcal{L}_{\text{cons}} + (1-\lambda(g))\mathcal{L}_{\text{recon}}$, where $\lambda(g)$ follows a cosine curriculum to transition from distillation to refinement. $\mathcal{L}_{\text{recon}}$ anchors the student to the ground truth, while $\mathcal{L}_{\text{cons}}$ aligns the student with the temperature-scaled teacher distribution $\tilde{p}_\phi(\cdot|\tilde{z}^T)_i \propto \exp(l_\phi^{(i)}/\tau)$ using standard τ^2 scaling (Hinton et al., 2015):

$$\mathcal{L}_{\text{recon}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathbb{E}_{r_S,\mathcal{M}_S} \left[\frac{1}{|\mathcal{M}_S|}\sum_{i\in\mathcal{M}_S} -\log p_\theta(z_i \mid \tilde{z}^S)\right]. \quad (2)$$

$$\mathcal{L}_{\text{cons}}(\theta) = \tau^2 \cdot \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathbb{E}_{r_S,r_T,\mathcal{M}_S,\mathcal{M}_T} \left[\frac{1}{|\mathcal{M}_S|}\sum_{i\in\mathcal{M}_S} \mathrm{KL}\big(\tilde{p}_\phi(\cdot \mid \tilde{z}^T)_i \big\| p_\theta(\cdot \mid \tilde{z}^S)_i\big)\right]. \quad (3)$$

This formulation ensures the student learns trajectory-invariant refinement (via $\mathcal{L}_{\text{cons}}$) while retaining generation quality (via $\mathcal{L}_{\text{recon}}$), see Appendix B for theoretical analysis. For a complete algorithmic description of the training procedure, including the details of mask sampling and the curriculum schedule, readers can refer to the pseudocode provided in Appendix E.1.

Relation to trajectory-locked distillation. Concurrent distillation methods also constrain the student to a specific decoding pattern: dParallel (Chen et al., 2025b) aligns the student along the teacher's certainty schedule, d3LLM (Qian et al., 2026) distills the teacher's pseudo-decoding order, and T3D (Zhang et al., 2026) applies reverse-KL distillation to teacher rollouts. In contrast, DSCD supervises the student on random teacher-subset masks (Eq. 3), which projects the teacher's conditional posterior onto every possible student information σ-algebra (Appendix B). This formulation makes the student robust to downstream decoding policies that induce off-schedule mask patterns, including the cache-based policies evaluated in Sect. 4.1.

2.2 Confidence-Adaptive Decoding

After distillation, we discard the teacher and decode using the student model only. Our CAD is a generic unmasking policy. In this work, we instantiate it within the LLaDA block diffusion framework. Pure diffusion is recovered as a special case by setting $b = L_{\text{gen}}$. Fig. E.2 in Appendix E.2 contrasts CAD against fixed-schedule block diffusion, pure diffusion, and AR decoding. The same appendix gives a step-by-step illustration of a single CAD step.

Block diffusion schedule. Unlike autoregressive models that generate variable-length sequences by dynamically appending tokens, diffusion-based decoding operates on a pre-allocated, fixed-size canvas. We set a maximum target horizon L_{gen} (e.g., matching the training sequence length or a system budget) and pad the initial state with masks. The actual sequence length is determined dynamically during decoding via the end-of-sequence (EOS) blocking mechanism. Given L_{gen} and block size b, we partition positions into contiguous blocks $\{\mathcal{B}_j\}_{j=1}^{J}$, where $\mathcal{B}_j = \{(j-1)b+1, \ldots, \min(jb, L_{\text{gen}})\}$ and $J = \lceil L_{\text{gen}}/b \rceil$. Decoding proceeds left-to-right with an active block index j. Tokens from finished blocks are cached (frozen), while future blocks remain masked until activated. Note that while our primary experiments use full-attention recomputation to isolate the algorithmic gains of DSCD, this block-wise formulation is structurally compatible with approximate KV caching schemes (Wu et al., 2025).

State, masked set, and eligible set. Let $\tilde{z}^{(s)} \in \mathcal{V}^{L_{\text{gen}}}$ denote the partially-masked state at decoding step s and let $\mathcal{M}^{(s)} = \{i \in [L_{\text{gen}}] : \tilde{z}_i^{(s)} = \texttt{m}\}$ be the masked set. CAD operates on the *eligible set* within the active block:

$$\mathcal{E}^{(s)} = \mathcal{M}^{(s)} \cap \mathcal{B}_j, \quad (4)$$

which reduces to $\mathcal{E}^{(s)} = \mathcal{M}^{(s)}$ when $b = L_y$ (pure diffusion).

Confidence-adaptive set selection. We treat the decoding process as a dynamic risk-efficiency trade-off. The full decision cycle is illustrated in Appendix E.2 (Fig. E.1). Starting from a partially masked state (Step 1), the student model predicts distributions for all masked positions in a

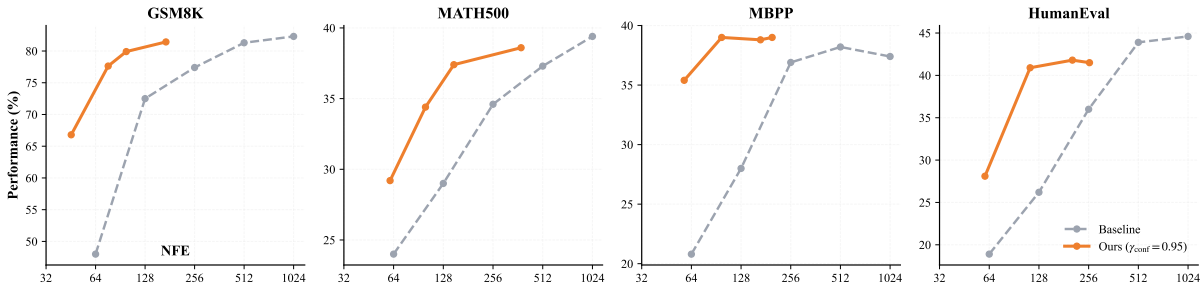

Figure 2: **Accuracy–compute trade-off under different decoding budgets.** We plot performance (%) versus the average NFE across benchmarks. The dashed gray curve denotes the LLaDA-8B-Instruct baseline evaluated under increasing NFE budgets, while the solid orange curve denotes our CAD with the given threshold ($\gamma_{\text{conf}} = 0.95$). Each marker corresponds to one decoding budget. Across all four benchmarks, our method yields a consistently better trade-off, achieving comparable or higher performance at substantially lower NFE.

single forward pass (Step 2). We then compute a confidence score $c_i^{(s)}$ for each eligible position $i \in \mathcal{E}^{(s)}$ (Step 3), using the maximum probability $c_i^{(s)} = \max_v p_\theta(v \mid \tilde{z}^{(s)})_i$ as a proxy for correctness. Based on these scores, we identify a candidate set $\widehat{\mathcal{S}}^{(s)} = \{i \in \mathcal{E}^{(s)} : c_i^{(s)} \geq \gamma_{\text{conf}}\}$ containing tokens where the model's certainty outweighs the risk of error (Step 4). To stabilize the dynamic trajectory, we clamp the commit size via

$$\begin{aligned} k^{(s)} &= \operatorname{clip}\big(|\widehat{\mathcal{S}}^{(s)}|,\ k_{\min},\ k_{\max}\big), \\ \mathcal{S}^{(s)} &= \operatorname{TopK}\big(\{c_i^{(s)}\}_{i \in \mathcal{E}^{(s)}},\ k^{(s)}\big), \end{aligned} \tag{5}$$

and finally unmask the predictions in $\mathcal{S}^{(s)}$ while keeping lower-confidence tokens masked for future refinement (Step 5). This greedy policy approximates the optimal stopping rule for per-step decoding risk (see derivation in Appendix D).

To prevent degenerate short outputs, we also apply an EOS blocking mechanism (implementation details in Appendix D.4). Our controller introduces negligible overhead compared to the model forward pass (complexity analysis in Appendix D.6). For the complete pseudocode of the CAD algorithm, which details the interaction between block-wise diffusion and the dynamic acceptance policy, readers can refer to Appendix E.2.

3 Experimental Setup

We discuss the experimental setup next.

3.1 Models and Training

Teacher and student. We use LLaDA-8B-Instruct (Nie et al., 2025) as the teacher. The student is initialized from the teacher's pretrained weights. All experiments use BF16 precision and a maximum context length of 1024.

Training data. We train our student model on a 200k-sample subset of OpenCodeInstruct (Ahmad et al., 2025), a comprehensive instruction-following code generation dataset. To ensure high-quality distillation, we filter for samples with valid solutions and reserve 5% (10k samples) for validation. For mathematical reasoning experiments, we use the GSM8K (Cobbe et al., 2021) training split and use the same training protocol.

3.2 Benchmarks

Code generation. We evaluate the model in a zero-shot setting on HumanEval (Chen, 2021) and three-shot on MBPP (Austin et al., 2021b), and also report results on the stricter variants: HumanEval+ and MBPP+. We report functional correctness using pass@1 and pass@5.

Mathematical reasoning. We report accuracy on GSM8K (Cobbe et al., 2021) and MATH500 (Hendrycks et al., 2021) in a zero-shot setting.

Evaluation framework. All evaluations are implemented based on the open-sourced DAEDAL's evaluation codebase (Li et al., 2025), which standardizes prompting, post-processing, and metric computation for LLaDA-style DLMs.

3.3 Training Protocol

We use a global batch size of 64 on 8×AMD MI250 GPUs, AdamW with learning rate 5×10^{-6}, cosine decay, and 10% warmup, and train for three epochs. Detailed hyperparameters and prompt templates are provided in Appendix E.3.

Method	Acc. (%) ↑	Avg. NFE ↓	Tokens/s (Speedup) ↑
LLaDA-8B-Instruct baselines (fixed tokens-per-step)			
LLaDA tps=1	77.4	256.0	7.5 (1.00×)
LLaDA tps=2	74.8	128.0	14.9 (1.99×)
LLaDA tps=4	43.6	64.0	29.7 (3.96×)
LLaDA tps=8	21.2	32.0	59.5 (7.93×)
Training-based accelerators†			
dParallel (Chen et al., 2025b)	76.7	30.0	64.0 (8.53×)
CDLM (Kim et al., 2025)	73.9	57.7	33.3 (4.44×)
d^3LLM (Qian et al., 2026)	74.2	36.8	52.2 (6.96×)
Inference-based accelerators			
Fast-dLLM (Wu et al., 2025)	75.8	84.1	32.4 (4.30×)
d^2Cache (Jiang et al., 2025)	77.4	256	29.4 (3.90×)
dLLM-Cache (Liu et al., 2025)	76.9	256	31.7 (4.25×)
Ours (training + inference)			
DSCD + CAD (Our CD^{4}LM)	77.6	76.3	**38.7 (5.18×)**
DSCD + Fast-dLLM	77.2	**70.5**	42.0 (5.60×)
Ours + d^2Cache	**78.0**	76.5	67.0 (8.96×)
Ours + dLLM-Cache	77.6	76.6	**72.7 (9.73×)**

Table 1: **GSM8K (zero-shot) accuracy-efficiency across acceleration paradigms** ($L_{\text{gen}} = 256$, block size $b = 32$). Methods are grouped by acceleration paradigm. The tps next to a method denotes the preset tokens per step; methods without a annotation use adaptive or method-specific scheduling. Speedup is wall-clock relative to the LLaDA tps=1 baseline. Methods marked † are taken directly from the original papers under their reported settings; their tokens/s and speedup are extrapolated from the reported NFE relative to the LLaDA baseline. The remaining baselines are reproduced under the same harness as our method. Sequential ($b = 1$) and pure-diffusion ($b = L_{\text{gen}}$) variants of LLaDA and our method are reported in Appendix G.1.

3.4 Inference Protocols

Generation paradigms. We consider three generation paradigms: (i) Sequential ($b = 1$), (ii) Block Diffusion (block size $b = 32$), and (iii) Pure Diffusion ($b = L_{\text{gen}}$).

For mathematical reasoning benchmarks, we use block diffusion with $b = 32$, which is the best-performing configuration in our tuning process and is used throughout the main mathematical experiments. We use pure diffusion for code generation. Unless specified otherwise, we use a maximum generation length of $L_{\text{gen}} = 256$.

Sampling for pass@k. We compute pass@1 using greedy $\tau_{\text{samp}} = 0$ decoding and pass@5 using sampling with temperature $\tau_{\text{samp}} = 1.0$ (other decoding knobs follow the DAEDAL defaults for each benchmark).

Speed measurement. We report wall-clock speedup under the same hardware and evaluation harness, measured relative to the corresponding LLaDA-8B-Instruct baseline configuration for each benchmark with a batch size of 1.

Inference-side baselines. For composability analysis (Sect. 4.1), we additionally reproduce Fast-dLLM (Wu et al., 2025), d^2Cache (Jiang et al., 2025), and dLLM-Cache (Liu et al., 2025) under the same harness and default settings, so that the comparison baselines and the DSCD-composed configurations differ only in the inference-side accelerator.

4 Results and Analysis

Next, we present our experimental results and their analysis.

4.1 Main Results

We examine how four families of DLM acceleration trade off accuracy against speed on GSM8K (Table 1), then verify that the headline configuration transfers across the broader code and mathematics suite (Table 2).

Acceleration paradigms on GSM8K. Table 1 groups accelerators into four paradigms, and the comparison exposes a consistent pattern.

(i) Naive step reduction collapses accuracy. Increasing tokens-per-step on the unmodified teacher cuts NFE proportionally but accuracy degrades from 77.4% at tps=1 to 74.8% at tps=2 and finally 21.2% at tps=8: with the base model unchanged,

Task	Benchmark	LLaDA-8B-Ins	Ours	Speedup
Code	HumanEval (pass@1 %)	38.7 256	**40.9** 113.2	3.30×
	HumanEval (pass@5 %)	51.2 256	**52.4** 113.9	3.26×
	HumanEval-plus (pass@1 %)	31.7 256	**32.9** 115.0	3.25×
	HumanEval-plus (pass@5 %)	42.7 256	**43.9** 113.3	3.27×
	MBPP-plus (pass@1 %)	**48.7** 256	47.9 108.0	3.51×
	MBPP-plus (pass@5 %)	**68.8** 256	67.7 108.8	3.41×
Math	MATH500 (Acc %)	37.3 512	**38.6** 148.3	5.33×
Avg.		45.6 292.6	**46.3** 117.2	3.62×

Table 2: **Overall performance on code and mathematics benchmarks.** We report pass@1/5 on code benchmarks and accuracy on MATH500. The nfe next to each score denotes the Avg. NFE for that setting. Speedup is measured by wall-clock time relative to LLaDA-8B-Instruct under the same evaluation protocol. **Avg.** denotes the unweighted mean across the listed benchmarks.

speedup from the teacher alone is bought at near-total accuracy loss.

(ii) Training-based methods reduce but do not eliminate the accuracy gap. dParallel, CDLM, and d^{3}LLM are designed around specific decoding patterns and recover most of the accuracy lost under naive reduction, but their reported accuracies remain 0.7-3.5 points below the teacher (76.7%, 73.9%, 74.2%).

(iii) Reproduced inference-side accelerators remain below the combined frontier. Cache-based accelerators (d^{2}Cache, dLLM-Cache) and the parallel-decoding-plus-cache method Fast-dLLM avoid retraining and achieve 3.9-4.3× speedup in our harness; Fast-dLLM reaches its highest reported speedup in this group at 75.8% accuracy.

(iv) DSCD+CAD preserves accuracy and composes cleanly with caching. **CD^{4}LM** reduces NFE to 76.3 while retaining 77.6% accuracy at 5.18× speedup. Because DSCD's supervision is decoupled from any specific inference policy, **CD^{4}LM** stacks directly on cache-based per-step accelerators: **CD^{4}LM**+d^{2}Cache and **CD^{4}LM**+dLLM-Cache combine NFE reduction (from CAD) with per-step compute reduction (from caching), reaching **8.96**× (78.0%) and **9.73**× (77.6%) wall-clock speedup and Pareto-dominating every other entry in Table 1. The composed gain is sub-multiplicative rather than the product of the two axes, since CAD has already compressed the schedule and leaves fewer steps over which the per-step cache can amortize. The same composition transfers to MBPP, where DSCD+Fast-dLLM Pareto-improves the standalone cache on both pass@1 (33.8% → 38.7%) and speedup (3.07× → 3.29×, Appendix G.2).

Across the seven settings in Table 2, **CD^{4}LM** reduces mean NFE from 292.6 to 117.2 (60% less) and yields a 3.62× end-to-end speedup while slightly improving the unweighted average score from 45.6 to 46.3, with per-benchmark gains of +2.2pt pass@1 at 3.30× on HumanEval and +1.3pt accuracy at 5.33× on MATH500.

Configuration	Acc (%)	NFE	Speedup
DSCD composes with inference accelerators			
DSCD only (fixed)	**78.3**	256.0	1.00×
DSCD + CAD (Ours)	77.6	76.3	5.18×
DSCD + Fast-dLLM	77.2	70.5	**5.60**×
SFT-based training is constrained			
SFT only (fixed)	77.6	256.0	1.00×
SFT + CAD	77.6	95.4	4.15×
SFT + Fast-dLLM	76.9	83.7	4.73×

Table 3: **Decoding flexibility and training effectiveness on GSM8K** ($L_{\text{gen}} = 256$, $b = 32$). *Group 1* shows that the same DSCD student operates with both CAD (training-aware decoding) and Fast-dLLM (KV caching) without retraining. *Group 2* replaces DSCD with plain supervised fine-tuning while holding the decoder fixed. Under both compared decoders, DSCD achieves a higher speedup than SFT at similar accuracy (CAD: 4.15× vs. 5.18×; Fast-dLLM: 4.73× vs. 5.60×).

DSCD improves confidence-guided decoding efficiency. Table 3 isolates DSCD's contribution by holding the decoding policy fixed (CAD or Fast-dLLM, mutually exclusive commit policies) and swapping the training objective. Under either decoder, the DSCD student preserves teacher-level accuracy at a higher speedup than the SFT student (CAD: 5.18× vs. 4.15×; Fast-dLLM: 5.60× vs. 4.73×). These matched-decoder comparisons are consistent with DSCD better supporting confidence-guided commitment, although we do not directly measure confidence calibra-

tion. Under the diagnostic global-denoising setting ($b{=}L_{\text{gen}}$), SFT+CAD reaches 38.2% at 3.13×, whereas DSCD+CAD reaches 54.7% at 4.91× (Appendix F).

Consistent efficiency gains. Fig. 2 visualizes the accuracy-compute frontier across all four benchmarks: at the plotted budgets, our CAD ($\gamma_{\text{conf}} = 0.95$) shifts the frontier upward and leftward versus the LLaDA baseline across diverse domains without per-benchmark tuning. On GSM8K, sweeping $\gamma_{\text{conf}} \in [0.85, 0.99]$ further produces a Pareto-improved frontier over the plotted baseline points (Appendix G.5). The ablation in Appendix F additionally shows that, under diagnostic global denoising, DSCD+CAD exceeds SFT+CAD in both accuracy (54.7% vs. 38.2%) and speedup (4.91× vs. 3.13×).

4.2 Mechanism of Efficiency

We analyze per-sample compute distributions in Appendix G.6 (Fig. G.2). Across the four plotted tasks, CAD terminates at lower mean NFE than the fixed baseline budgets (means of 76.3, 148.3, 97.7, and 113.2 versus fixed budgets of 256, 512, 256, and 256, respectively).

Translating step reduction to wall-clock speedup. Our wall-clock speedup can exceed the NFE-ratio reduction: on GSM8K, **CD^{4}LM** uses 76.3 rather than 256.0 NFE (3.36× fewer) while reaching 5.18× measured wall-clock speedup (Table 1). Thus, NFE reduction alone does not account for the measured latency gain; the observed throughput also reflects per-step execution cost.

Illustrative unmasking trajectory. The per-step unmasking trajectory in Appendix G.8 (Fig. G.3) provides a qualitative example of a "scaffold-first, fill-later" pattern: in one HumanEval generation, several syntactic tokens are committed early, while parts of the arithmetic and conditional logic are committed at later steps. This example illustrates the type of non-left-to-right refinement permitted by CAD; it is not a population-level analysis of planning behavior.

Backbone generality. Applied to DiffuCoder-Instruct (Gong et al., 2025) with no architecture-specific modifications, the same DSCD+CAD pipeline improves MBPP pass@1 from 73.1% to 73.7% with a 3.25× wall-clock speedup (Appendix G.3), providing evidence of transfer to a second absorbing-state DLM backbone.

5 Conclusion

We presented **CD^{4}LM**, a training-side DLM accelerator that commits to no decoding pattern: DSCD trains a trajectory-invariant student via a Rao-Blackwellized teacher-subset objective (Martingale-projection view, posterior-risk analysis), and CAD exploits this invariance for dynamic confidence-adaptive decoding. The result reaches teacher-level accuracy on GSM8K (77.6% at 5.18× speedup) and transfers to DiffuCoder-Instruct without modification. Because DSCD's supervision is decoupled from any specific inference policy, stacking it with per-step cache accelerators is orthogonal: **CD^{4}LM**+dLLM-Cache and **CD^{4}LM**+d^{2}Cache reach the table's highest speedup (**9.73**× at 77.6%) and highest accuracy (**78.0%** at 8.96×) respectively, Pareto-dominating every standalone training- or inference-side accelerator we evaluate.

Outlook. Decoupling training from any specific decoding pattern is a productive design discipline: Stacking DSCD with future inference-side accelerators and pairing trajectory-invariant supervision with on-policy refinement are natural next steps, with **CD^{4}LM** providing a stable training-side layer beneath the evolving inference-side ecosystem.

Limitations and Discussion

We close by acknowledging where **CD^{4}LM** trades standalone speed for composability, the structural assumptions inherited from its block-diffusion backbone, and the scope of our evaluation.

Position relative to trajectory-locked distillation. Trajectory-invariant supervision trades some standalone speed for composability. Among the training-side baselines reported in Table 1, dParallel (Chen et al., 2025b) and d^{3}LLM (Qian et al., 2026) have higher reported/extrapolated standalone speedups than **CD^{4}LM** (8.53× and 6.96× vs. 5.18×), but lower accuracy (76.7% and 74.2% vs. 77.6%). **CD^{4}LM** reaches higher speedups only once composed with reproduced cache accelerators. Whether these composability gains generalize beyond the three evaluated inference-side accelerators (Fast-dLLM, d^{2}Cache, and dLLM-Cache) remains open.

Static canvas constraints. Like the LLaDA backbone and most block-diffusion models, our approach relies on a pre-defined maximum sequence length L_{gen}. Although CAD handles variable-length outputs via EOS blocking, computation is allocated for the fixed generation canvas, which can waste work on short outputs and limits the maximum generated length.

Evaluation scope. Our experiments cover LLaDA-8B-Instruct and one DiffuCoder-Instruct transfer setting, evaluated on code and mathematical reasoning tasks. We do not test larger models or open-ended generation. CAD uses maximum token probability as a proxy for correctness; its behavior when many continuations are valid, such as in dialogue or creative writing, remains untested.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. 2025. OpenCode-Instruct: A large-scale instruction tuning dataset for code LLMs. *arXiv preprint arXiv:2504.04030*.

Masaki Asada and Makoto Miwa. 2025. Addressing the training-inference discrepancy in discrete diffusion for text generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7156–7164.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. 2025. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. *arXiv preprint arXiv:2509.25188*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Tianqi Chen, Shujian Zhang, and Mingyuan Zhou. 2025a. DLM-One: Diffusion language models for one-step sequence generation. *arXiv preprint arXiv:2506.00290*.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. 2025b. dParallel: Learnable parallel decoding for dLLMs. *arXiv preprint arXiv:2509.26488*.

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. 2025. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12042–12059.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Justin Deschenaux and Caglar Gulcehre. 2025. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, volume 2025, pages 5007–5045.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. DiffuCoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ishaan Gulrajani and Tatsunori B. Hashimoto. 2023. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715.

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. 2025. MDPO: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. 2025. Reinforcing the diffusion chain of lateral thought with diffusion language models. *arXiv preprint arXiv:2505.10446*.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2025. Accelerating diffusion LLMs via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. 2025. d^2Cache: Accelerating diffusion-based LLMs via dual adaptive caching. *arXiv preprint arXiv:2509.23094*.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. 2025. CDLM: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, and 1 others. 2025. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. 2025. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *arXiv preprint arXiv:2508.00819*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-LM improves controllable text generation. *arXiv preprint arXiv:2205.14217*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.

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. *arXiv preprint arXiv:2502.09992*.

Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. 2026. d3LLM: Ultra-fast diffusion LLM using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*.

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

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *arXiv preprint arXiv:2303.01469*.

Pascal Vincent. 2011. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. 2025. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. 2025. Accelerating diffusion large language models with slowfast: The three golden principles. *arXiv preprint arXiv:2506.10848*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2024. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Vladimir Pavlovic, and 1 others. 2026. T3D: Few-step diffusion language models via trajectory self-distillation with direct discriminative optimization. *arXiv preprint arXiv:2602.12262*.

Huaisheng Zhu, Zhengyu Chen, Shijie Zhou, Zhihui Xie, Yige Yuan, Zhimeng Guo, Siyuan Xu, Hangfan Zhang, Vasant Honavar, and Teng Xiao. 2025. Simple denoising diffusion language models. *arXiv preprint arXiv:2510.22926*.

A Preliminaries

We review the absorbing-state discrete diffusion formulation adopted from LLaDA (Nie et al., 2025) to establish the notation used in our theoretical analysis.

Data representation. Let $\mathcal{D} = \{(x, y)\}$ be a corpus of conditional generation tasks, where $x = (x_1, \ldots, x_{L_x})$ is the prompt (e.g., a coding problem or mathematics question) and $y = (y_1, \ldots, y_{L_y})$ is the target sequence (e.g., the solution program or derivation). We concatenate them into a single sequence $z = (x; y) = (z_1, \ldots, z_L)$, with $L = L_x + L_y$. The base vocabulary $\mathcal{V}$ is augmented with a special absorbing mask token $\mathtt{m} \notin \mathcal{V}$, and an end-of-sequence token <EOS>, giving the extended vocabulary: $\mathcal{V}^+ = \mathcal{V} \cup \{\mathtt{m}, \texttt{<EOS>}\}$. For chain-of-thought supervision, we place the entire reasoning trace together with the final answer in the target region ($i > L_x$); hence, the model explicitly learns to refine both intermediate reasoning and final answers. For code completion benchmarks, the natural-language problem description and function signature belong to the prompt region, while the full solution body is treated as target tokens.

Forward absorbing process. We adopt the absorbing discrete diffusion framework for text (Austin et al., 2021a; Shi et al., 2024; Sahoo et al., 2024; Nie et al., 2025), which models corruption as stochastic erasure rather than additive noise. Let $t \in [0, 1]$ be a continuous noise level with a monotone schedule $\alpha(t) \in [0, 1]$ satisfying $\alpha(0) = 0$ and $\alpha(1) \approx 1$. The forward process $q_t(\tilde{z} \mid z)$ gradually destroys information in the target region while preserving the prompt:

$$q_t(\tilde{z} \mid z) = \prod_{i=1}^{L} q_t(\tilde{z}_i \mid z_i)$$

$$q_t(\tilde{z}_i \mid z_i) = \begin{cases} \delta_{\tilde{z}_i = z_i}, & i \le L_x, \\ (1 - \alpha(t))\delta_{\tilde{z}_i = z_i} + \alpha(t)\delta_{\tilde{z}_i = \mathtt{m}}, & i > L_x, \end{cases} \quad (6)$$

where δ is the Kronecker delta. Once a token is replaced by $\mathtt{m}$, it remains masked for all larger t; hence, the forward chain is an absorbing Markov process. We denote the masked set by $\mathcal{M}(\tilde{z}) = \{i : \tilde{z}_i = \mathtt{m}\}$. In practice, we do not simulate this Markov chain step-by-step. Given z and t, we instead sample a Bernoulli mask $b \in \{0, 1\}^{L_y}$ with $\Pr(b_j = 1) = \alpha(t)$ for target positions and set $\tilde{z} = \mathrm{Mask}(z, b)$, which matches $q_t(\tilde{z} \mid z)$ in (6). In all experiments, we reuse the continuous-time masking schedule from LLaDA and adopt the same choices of $\alpha(t)$, the sampling distribution $\rho(t)$ over t, and the weighting function $w(t)$.

Diffusion teacher and ELBO-style objective. A DLM parameterized by ϕ defines a reverse model that predicts clean tokens at the masked positions of a corrupted sequence:

$$p_\phi(z \mid \tilde{z}, t) = \prod_{i \in \mathcal{M}(\tilde{z})} p_\phi(z_i \mid \tilde{z}, t), \quad (7)$$

where each $p_\phi(z_i \mid \tilde{z}, t)$ is a categorical distribution over $\mathcal{V}^+$. We follow LLaDA (Nie et al., 2025) and parameterize p_ϕ with a Transformer that maps $(\tilde{z}, t)$ to logits over the vocabulary at all positions; unmasked positions are simply copied from the input. For absorbing discrete diffusion, the log-likelihood admits an evidence lower bound whose dominant term reduces to a time-weighted masked cross-entropy (Austin et al., 2021a; Sahoo et al., 2024; Shi et al., 2024). We adopt the simplified continuous-time objective used in masked DLMs:

$$\mathcal{L}_{\text{teacher}}(\phi) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{E}_{t\sim\rho(t)} \mathbb{E}_{\tilde{z}\sim q_t(\cdot|z)} \Bigg[\frac{w(t)}{|\mathcal{M}(\tilde{z})|} \cdot \sum_{i \in \mathcal{M}(\tilde{z})} -\log p_\phi(z_i \mid \tilde{z}, t) \Bigg], \quad (8)$$

where $\rho(t)$ and $w(t)$ are inherited from LLaDA. Optimizing (8) yields a high-quality diffusion teacher, but sampling requires iterating over a long schedule $t_1 > \cdots > t_T$, leading to high decoding latency. In practice, p_ϕ is instantiated as the publicly released LLaDA checkpoint of the corresponding size and kept frozen throughout student training; we do not modify its architecture, training objective, or noise schedule. All improvements reported in this article, therefore, come from the distilled student and our decoding policy, rather than from retraining the teacher.

Notation. We summarize the main symbols in the theoretical analysis and their meanings.

- p_θ, p_ϕ = Student model and frozen Teacher model
- $\tilde{z}^S, \tilde{z}^T$(Sect. 2.1) $\leftrightarrow$ S, T (Appendix B) = masked input views
- $\mathcal{M}_S, \mathcal{M}_T$ = masked position sets for student and teacher
- $\lambda(g), \tau$ = Curriculum schedule and distillation temperature

- $\mathcal{M}^{(s)}$ (Sect. 2.2) = masked set at decoding step s
- $\mathcal{E}^{(s)}$ = eligible set = $\mathcal{M}^{(s)} \cap \mathcal{B}_j$ (active block intersection)
- $\mathcal{S}^{(s)}$ = commit set (positions to unmask at step s)
- $c_i^{(s)}$ = confidence score = $\max_v p_\theta(v \mid \tilde{z}^{(s)})_i$
- γ_{conf} = Confidence threshold for CAD acceptance
- $k_{\min}, k_{\max}$ = Trust region bounds for commit size $|\mathcal{S}^{(s)}|$

B A Martingale Projection View of DSCD

While DSCD is empirically motivated, it possesses a rigorous theoretical grounding. By enforcing *teacher-subset masking* ($\mathcal{M}_T \subseteq \mathcal{M}_S$), we effectively construct a stochastic filtration of information where the teacher always holds a strictly finer information set than the student. In this section, we formally prove that minimizing $\mathcal{L}_{\text{cons}}$ is equivalent to training the student to approximate the Martingale projection (Vincent, 2011) of the teacher's belief state. This guarantees that the student learns the expected trajectory of the teacher, satisfying the statistical definition of trajectory invariance.

B.1 Setup and Information Ordering

Let $(\Omega, \mathcal{F}, P)$ be the probability space that supports all random variables in our construction. Let $Z \in \mathcal{Z}$ denote the clean (ground-truth) token sequence. The absorbing masking process of LLaDA defines, for each masking pattern $\mathcal{M}$, a partially masked sequence $X^{(\mathcal{M})}$ obtained by replacing a subset of token positions in Z with the mask token $\texttt{m}$.

To simplify notation in this theoretical analysis, we denote the random variables corresponding to the student view $\tilde{z}^S$ and teacher view $\tilde{z}^T$ (defined in Sec. 2.1) simply as S and T. We define:

$$S = X^{(\mathcal{M}_S)}, \qquad T = X^{(\mathcal{M}_T)}.$$

We write

$$\mathcal{F}_S = \sigma(S), \qquad \mathcal{F}_T = \sigma(T)$$

for the σ-algebras generated by the corresponding observed (unmasked) tokens.

In our nested masking design, the teacher always sees a superset of the tokens visible to the student. Concretely, if $\mathcal{M}_S$ and $\mathcal{M}_T$ denote the random sets of *masked* positions for the student and teacher, we enforce

$$\mathcal{M}_T \subseteq \mathcal{M}_S,$$

so that every token unmasked for the student is also unmasked for the teacher. Equivalently, the teacher's observation T is a measurable function of (S, Z), and the associated σ-algebras are ordered as $\mathcal{F}_S \subseteq \mathcal{F}_T$.

Definition 1 (Information ordering)**.** *We say that the teacher is informationally richer than the student, and write $S \preceq T$, if*

$$\mathcal{F}_S \subseteq \mathcal{F}_T,$$

i.e., every event that is measurable with respect to the student state S is also measurable with respect to the teacher state T. Under our nested masking scheme $\mathcal{M}_T \subseteq \mathcal{M}_S$, we have $S \preceq T$ by construction.

B.2 Consistency as Projection onto Student Information

We now formalize how DSCD can be viewed as projecting the teacher's conditional distribution onto the information available to the student. Let $Z \in \mathcal{Z}$ denote the clean (ground-truth) sequence, and let S and T be the student and teacher states as in Definition 1, with $\mathcal{F}_S = \sigma(S)$ and $\mathcal{F}_T = \sigma(T)$ satisfying $S \preceq T$.

Given a student state $S = s$, the DSCD objective compares the student distribution $p_\theta(\cdot \mid s)$ against the teacher distribution $p_\phi(\cdot \mid T)$ averaged over the teacher states T that are compatible with $S = s$. Formally, the DSCD consistency loss can be written as

$$\mathcal{L}_{\text{DSCD}}(\theta) = \mathbb{E}_S \mathbb{E}_{T|S}\big[D_{\text{KL}}(p_\phi(\cdot \mid T) \| p_\theta(\cdot \mid S))\big], \tag{9}$$

where the outer expectation is taken over the student states induced by the diffusion process and masking policy, and the inner expectation is taken over the corresponding teacher states. For later use, we define the teacher-induced *projected* distribution

$$Q(z \mid S = s) := \mathbb{E}_{T|S=s}[p_\phi(z \mid T)], \tag{10}$$

i.e., $Q(\cdot \mid s)$ is obtained by averaging the teacher's conditional distributions over all teacher states T consistent with the student state $S = s$. Equivalently, $Q(\cdot \mid S)$ is the conditional expectation of $p_\phi(\cdot \mid T)$ with respect to the σ-algebra $\mathcal{F}_S$.

Proposition 1 (DSCD as projection onto $\mathcal{F}_S$)**.** *Assume the student model class $\{p_\theta(\cdot \mid S)\}$ is rich enough to represent any conditional distribution over $\mathcal{Z}$ for each $S = s$. Then the DSCD loss (9) is minimized by*

$$p^*_\theta(z \mid S = s) = Q(z \mid S = s) = \mathbb{E}_{T|S=s}[p_\phi(z \mid T)], \quad \forall s \in \mathcal{S}. \tag{11}$$

In particular, $p^*_\theta(\cdot \mid S)$ coincides with the conditional expectation of the teacher's conditional distribution $p_\phi(\cdot \mid T)$ onto the coarser σ-algebra $\mathcal{F}_S$.

Proof. By expanding the KL divergence in (9), we obtain

$$\begin{aligned}
\mathcal{L}_{\text{DSCD}}(\theta) &= \mathbb{E}_{S,T}[D_{\text{KL}}(p_\phi(\cdot \mid T) \| p_\theta(\cdot \mid S))] && (12)\\
&= \mathbb{E}_{S,T}\left[\sum_{z\in\mathcal{Z}} p_\phi(z \mid T) \log \frac{p_\phi(z \mid T)}{p_\theta(z \mid S)}\right] && (13)\\
&= \underbrace{\mathbb{E}_{S,T}\left[\sum_{z\in\mathcal{Z}} p_\phi(z \mid T) \log p_\phi(z \mid T)\right]}_{\text{constant w.r.t. } \theta} - \mathbb{E}_{S,T}\left[\sum_{z\in\mathcal{Z}} p_\phi(z \mid T) \log p_\theta(z \mid S)\right]. && (14)
\end{aligned}$$

The first term does not depend on θ and can be dropped. Thus, minimizing (9) is equivalent to maximizing

$$\mathbb{E}_{S,T}\left[\sum_{z\in\mathcal{Z}} p_\phi(z \mid T) \log p_\theta(z \mid S)\right] = \mathbb{E}_{S}\left[\sum_{z\in\mathcal{Z}} \mathbb{E}_{T|S}[p_\phi(z \mid T)] \log p_\theta(z \mid S)\right]. \tag{15}$$

For each fixed student state $S = s$, the inner objective becomes

$$\sum_{z\in\mathcal{Z}} Q(z \mid s) \log p_\theta(z \mid s), \tag{16}$$

which is maximized when $p_\theta(\cdot \mid s) = Q(\cdot \mid s)$. Therefore, $p^*_\theta(\cdot \mid S) = Q(\cdot \mid S)$. □

This proposition shows that DSCD does not force the student to reproduce the teacher's full diffusion trajectory. Instead, the student learns the teacher's conditional distribution after integrating out the extra information available only to the teacher, i.e., after projecting onto the coarser information σ-algebra $\mathcal{F}_S$. This implies that the student learns to directly predict the *expectation* of the teacher's multi-step denoising result, effectively compressing the diffusion trajectory into a single forward pass.

C Rao-Blackwellized Posterior-Risk Analysis of Teacher-Subset Masking

We theoretically justify our choice of the *teacher-subset masking* scheme ($\mathcal{M}_T \subseteq \mathcal{M}_S$) not merely as a heuristic, but as an information-nesting design rooted in statistical decision theory. Intuitively, if $\mathcal{M}_T$ were independent of $\mathcal{M}_S$, the teacher might lack access to tokens visible to the student, causing it to marginalize over "blind" positions and produce high-variance targets. By enforcing $\mathcal{M}_T \subseteq \mathcal{M}_S$, we ensure the teacher always conditions on a strict superset of the student's information. In this section, we formalize the guarantee that justifies the nested construction: As the teacher's information set is refined while retaining the student's visible tokens, the posterior residual risk of the teacher target cannot increase. Independent masking has no such nesting guarantee, because it may hide tokens that the student already observes.

C.1 Problem Setup

Let z be the ground truth sequence. The student observes a partial view $\tilde{z}^S$ with mask $\mathcal{M}_S$. The student's goal is to minimize the divergence from a target distribution provided by the teacher. Let the teacher's view be $\tilde{z}^T$ with mask $\mathcal{M}_T$. The teacher's prediction is a random variable $Y = p_\phi(\cdot \mid \tilde{z}^T)$, which serves as the regression target for the student.

We compare two masking schemes:

- **Teacher-subset masking:** $\mathcal{M}_T \subseteq \mathcal{M}_S$. The teacher observes everything the student observes, plus additional tokens $\Delta = \mathcal{M}_S \setminus \mathcal{M}_T$.
- **Independent masking:** $\mathcal{M}_T$ is sampled independently. Crucially, there exists a non-empty set of "blind" tokens $\mathcal{B} = \mathcal{M}_T \cap (\mathcal{V} \setminus \mathcal{M}_S)$ that are *visible to the student* but *masked for the teacher*.

C.2 Posterior Risk Reduction via Rao-Blackwellization

We now make the risk-reduction claim precise. Let U be a random target quantity that the teacher aims to approximate (e.g., a one-hot encoding of the ground-truth token or a sufficient statistic thereof). For any information σ-algebra $\mathcal{G}$, the Bayes-optimal predictor based on $\mathcal{G}$ is the conditional expectation

$$Y^{(\mathcal{G})} = \mathbb{E}[U \mid \mathcal{G}].$$

The student observes a state $\tilde{z}^S$ and the corresponding σ-algebra $\mathcal{F}_S = \sigma(\tilde{z}^S)$. In the teacher-subset scheme, the teacher observes all information available to the student plus additional tokens. Denote a nested teacher information set by $\mathcal{F}_T$, with $\mathcal{F}_S \subseteq \mathcal{F}_T$.

While the frozen teacher model p_ϕ is not the true Bayes-optimal posterior of the data, the conditional-expectation calculation clarifies why the nested design is statistically preferable: It keeps the teacher target compatible with the student's observed tokens and only adds information beyond the student view.

The following lemma is a conditional version of the Rao-Blackwell theorem.

Lemma 1 (Conditional Rao-Blackwell risk reduction)**.** *Let $U \in L^2(\Omega, \mathcal{F}, P)$ and let $\mathcal{G} \subseteq \mathcal{H} \subseteq \mathcal{F}$ be σ-algebras. Define*

$$Y_{\mathcal{G}} = \mathbb{E}[U \mid \mathcal{G}], \qquad Y_{\mathcal{H}} = \mathbb{E}[U \mid \mathcal{H}].$$

Then for any σ-algebra $\mathcal{K} \subseteq \mathcal{G}$,

$$\mathbb{E}\big[\|U - Y_{\mathcal{H}}\|_2^2 \mid \mathcal{K}\big] \le \mathbb{E}\big[\|U - Y_{\mathcal{G}}\|_2^2 \mid \mathcal{K}\big] \quad \textit{almost surely (a.s.)}$$

with equality iff $Y_{\mathcal{G}} = Y_{\mathcal{H}}$ a.s.

Proof. Fix $\mathcal{K} \subseteq \mathcal{G}$. Since $\mathcal{G} \subseteq \mathcal{H}$, the tower property gives $\mathbb{E}[Y_{\mathcal{H}} \mid \mathcal{G}] = Y_{\mathcal{G}}$. The orthogonality of conditional expectations in L^2 yields the Pythagorean identity

$$\mathbb{E}\big[\|U - Y_{\mathcal{G}}\|_2^2 \mid \mathcal{K}\big] = \mathbb{E}\big[\|U - Y_{\mathcal{H}}\|_2^2 \mid \mathcal{K}\big] + \mathbb{E}\big[\|Y_{\mathcal{H}} - Y_{\mathcal{G}}\|_2^2 \mid \mathcal{K}\big].$$

The second term is nonnegative and vanishes iff $Y_{\mathcal{H}} = Y_{\mathcal{G}}$ a.s., proving the claim. □

We now instantiate Lemma 1 with

$$\mathcal{K} = \mathcal{F}_S, \quad \mathcal{G} = \mathcal{F}_S, \quad \mathcal{H} = \mathcal{F}_T.$$

By construction, $\mathcal{F}_S \subseteq \mathcal{F}_T$, and therefore

$$\mathbb{E}\big[\|U - \mathbb{E}[U \mid \mathcal{F}_T]\|_2^2 \mid \mathcal{F}_S\big] \le \mathbb{E}\big[\|U - \mathbb{E}[U \mid \mathcal{F}_S]\|_2^2 \mid \mathcal{F}_S\big].$$

The inequality is strict whenever the extra tokens revealed to the teacher carry information about U beyond the student view.

In words, teacher-subset masking *Rao-Blackwellizes* the target relative to the student's information set: It preserves all tokens visible to the student and adds only extra information from the same clean sequence. Independent masking lacks this monotone information structure; it can create teacher targets that are blind to tokens the student observes, which is precisely the instability mode diagnosed in Appendix F.3.

D Theoretical Analysis of Confidence-Adaptive Decoding (CAD)

This appendix provides (i) a decision-theoretic motivation for CAD as a greedy solver of a per-step risk-efficiency trade-off with trust-region and liveness constraints, and (ii) basic guarantees on termination, step bounds, and computational overhead.

This derivation shows that confidence thresholding is optimal for an additive risk-efficiency objective, and that under cardinality constraints, the optimal solution is a confidence-sorted prefix. CAD implements a greedy approximation with explicit trust-region and liveness constraints.

D.1 Notation and Eligible Set

Let the target length be L_{gen}. At decoding step s, the current partially-masked state is $\tilde{z}^{(s)} \in \mathcal{V}^{L_{\text{gen}}}$, and the masked index set is $\mathcal{M}^{(s)} = \{i \in [L_{\text{gen}}] : \tilde{z}_i^{(s)} = \texttt{m}\}$. For block diffusion with block size b, define blocks

$$\mathcal{B}_j = \{(j-1)b+1, \ldots, \min(jb, L_{\text{gen}})\}, \quad j = 1, \ldots, J, \; J = \lceil L_{\text{gen}}/b \rceil,$$

and maintain an active block index j (decoded left-to-right). CAD operates on the *eligible set*

$$\mathcal{E}^{(s)} \;=\; \mathcal{M}^{(s)} \cap \mathcal{B}_j, \tag{17}$$

i.e., masked positions within the active block. Pure diffusion is recovered as a special case by setting $b = L_{\text{gen}}$ (a single block), in which case $\mathcal{E}^{(s)} = \mathcal{M}^{(s)}$.

D.2 Decision-theoretic Derivation (Per-step Objective)

At step s, the model produces a confidence score for each eligible position $i \in \mathcal{E}^{(s)}$, $c_i^{(s)} \in [0, 1]$. We interpret $c_i^{(s)}$ as a proxy for the probability that the current argmax token at position i is correct:

$$c_i^{(s)} \approx \mathbb{P}(\text{correct}_i \mid \tilde{z}^{(s)}). \tag{18}$$

Let $\mathcal{V}^{(s)} \subseteq \mathcal{E}^{(s)}$ denote the *valid* candidates at step s (e.g., after excluding padding or other structurally invalid positions; by default $\mathcal{V}^{(s)} = \mathcal{E}^{(s)}$). The decision variable is a commit set $\mathcal{S} \subseteq \mathcal{V}^{(s)}$.

We define two opposing per-step objectives: (i) **risk**: the expected number of incorrect tokens committed at this step, $\mathcal{R}(\mathcal{S}) = \sum_{i \in \mathcal{S}} (1 - c_i^{(s)})$, and (ii) **efficiency**: the number of committed tokens, $\mathcal{E}(\mathcal{S}) = |\mathcal{S}|$. We combine them into a net cost

$$\mathcal{L}(\mathcal{S}) \;=\; \mathcal{R}(\mathcal{S}) - \lambda_{\text{trade}}\, \mathcal{E}(\mathcal{S}) \;=\; \sum_{i \in \mathcal{S}} \big(1 - c_i^{(s)} - \lambda_{\text{trade}}\big), \tag{19}$$

where $\lambda_{\text{trade}} > 0$ controls the risk-speed trade-off and is *unrelated* to the training loss-mixing schedule $\lambda(g)$. Because (19) is additive across positions, the unconstrained minimizer commits all positions whose per-token contribution is negative:

$$i \in \mathcal{S}^* \iff 1 - c_i^{(s)} - \lambda_{\text{trade}} < 0 \iff c_i^{(s)} > 1 - \lambda_{\text{trade}}. \tag{20}$$

Defining the confidence threshold $\gamma_{\text{conf}} = 1 - \lambda_{\text{trade}}$ yields the basic *confidence thresholding* rule.

D.3 Trust-region and Liveness Constraints, and the CAD Rule

In practice, we impose two cardinality constraints: *trust region* ($|\mathcal{S}| \le k_{\max}$) to prevent committing too many tokens based on local confidence estimates, and *liveness* ($|\mathcal{S}| \ge k_{\min}$) to guarantee progress whenever possible. This yields the constrained problem

$$\min_{\mathcal{S} \subseteq \mathcal{V}^{(s)}} \mathcal{L}(\mathcal{S}) \quad \text{s.t.} \quad k_{\min} \le |\mathcal{S}| \le k_{\max}. \tag{21}$$

Sorting candidates by decreasing confidence produces an ordering $i_{(1)}, \ldots, i_{(|\mathcal{V}^{(s)}|)}$ with $c_{i_{(1)}}^{(s)} \ge \cdots \ge c_{i_{(|\mathcal{V}^{(s)}|)}}^{(s)}$. Under (21), the optimum is always a prefix of this ordering. CAD adopts a simple greedy approximation: Count how many positions within the trust region exceed γ_{conf},

$$n_{\text{conf}} \;=\; \left| \left\{ j \le \min(k_{\max}, |\mathcal{V}^{(s)}|) : c_{i_{(j)}}^{(s)} \ge \gamma_{\text{conf}} \right\} \right|, \tag{22}$$

then clamp the committed token count and select the top-$k^{(s)}$ positions:

$$k^{(s)} \;=\; \min\Big(\max(k_{\min}, n_{\text{conf}}),\; \min(k_{\max}, |\mathcal{V}^{(s)}|) \Big), \qquad \mathcal{S}^{(s)} = \{i_{(1)}, \ldots, i_{(k^{(s)})}\}. \tag{23}$$

Equations (22)-(23) match the CAD update implemented in the main text.

D.4 Monotonic Progress and Step Bound

Termination guarantee. Within the active block $\mathcal{B}_j$, CAD operates in an *absorbing* manner: once a position is committed (i.e., <mask> is replaced by a concrete token), it is never remasked. Let $\mathcal{E}^{(s)} = \mathcal{M}^{(s)} \cap \mathcal{B}_j$ denote the set of unresolved (masked) positions in the active block at step s. The clamping mechanism in Eq. (23) enforces $k^{(s)} \geq 1$ whenever $\mathcal{E}^{(s)} \neq \emptyset$ (typically $k^{(s)} \geq k_{\min}$). Consequently, CAD guarantees strict monotonicity in the reduction of the mask set:

$$|\mathcal{M}^{(s+1)} \cap \mathcal{B}_j| < |\mathcal{M}^{(s)} \cap \mathcal{B}_j|. \tag{24}$$

This implies that any block of size $|\mathcal{B}_j|$ must be fully resolved after at most $N_{\text{step}} = \lceil |\mathcal{B}_j|/k_{\min} \rceil$ function evaluations. This yields a deterministic per-block latency upper bound, ruling out the non-terminating refinement loops often observed in non-absorbing iterative decoding.

EOS blocking and safety. To prevent premature termination, we apply an **EOS blocking** strategy with a threshold β_{EOS}. For any position i selected for commitment, if the top-ranked prediction is the EOS token but its confidence $c_i^{(s)} < \beta_{\text{EOS}}$, we suppress the EOS and instead commit the next most likely non-EOS candidate:

$$\hat{z}_i = \operatorname*{arg\,max}_{v \in \mathcal{V} \setminus \{\text{EOS}\}} p_\theta(v \mid \tilde{z}^{(s)})_i.$$

The EOS token is only committed when it is both top-ranked and sufficiently confident ($c_i^{(s)} \geq \beta_{\text{EOS}}$), or when the decoding reaches a hard stopping condition (e.g., all blocks resolved or global $S_{\max}$ reached). Crucially, because EOS blocking only alters *which* concrete token is chosen—but still commits a token—it preserves the monotonicity property derived above.

The following statements formalize the monotonicity and yield the step bound.

Lemma 2 (Monotonicity within the active block)**.** *Assume $\mathcal{E}^{(s)} \neq \emptyset$ and $k_{\min} \geq 1$. After one CAD update,*

$$|\mathcal{M}^{(s+1)} \cap \mathcal{B}_j| \;\leq\; |\mathcal{M}^{(s)} \cap \mathcal{B}_j| - 1.$$

Moreover, if $|\mathcal{E}^{(s)}| \geq k_{\min}$, then

$$|\mathcal{M}^{(s+1)} \cap \mathcal{B}_j| \;\leq\; |\mathcal{M}^{(s)} \cap \mathcal{B}_j| - k_{\min}.$$

Proof. By construction, $\mathcal{S}^{(s)} \subseteq \mathcal{E}^{(s)} = \mathcal{M}^{(s)} \cap \mathcal{B}_j$ and each $i \in \mathcal{S}^{(s)}$ is committed from <mask> to a concrete token, hence removed from $\mathcal{M}^{(s)}$. If $\mathcal{E}^{(s)} \neq \emptyset$, the clamp in (23) ensures $k^{(s)} \geq 1$, yielding the first inequality. When $|\mathcal{E}^{(s)}| \geq k_{\min}$, we have $k^{(s)} \geq k_{\min}$, yielding the second inequality. □

Corollary 1 (Per-block and total NFE bound)**.** *For block size b, each block $\mathcal{B}_j$ is resolved within at most $\lceil |\mathcal{B}_j|/k_{\min} \rceil$ NFE. Therefore, the total NFE satisfies*

$$S \;\leq\; \sum_{j=1}^{J} \left\lceil \frac{|\mathcal{B}_j|}{k_{\min}} \right\rceil \;\leq\; \left\lceil \frac{L_{gen}}{k_{\min}} \right\rceil + J \;\leq\; \left\lceil \frac{L_{gen}}{k_{\min}} \right\rceil + \left\lceil \frac{L_{gen}}{b} \right\rceil,$$

and is additionally capped by the implementation limit $S_{\max}$.

D.5 Special Cases and Relation to Fixed-step Decoding

Proposition 2 (Pure diffusion as a special case)**.** *Setting $b = L_{gen}$ (i.e., $J = 1$) reduces block diffusion with CAD to pure diffusion with CAD, because $\mathcal{E}^{(s)} = \mathcal{M}^{(s)}$ for all s.*

Proposition 3 (Reduction to fixed-step block diffusion)**.** *If $\gamma_{conf} = -\infty$ and $k_{\min} = k_{\max} = k$ (a constant), then CAD commits exactly k positions per NFE within the active block (until fewer than k masks remain), matching a fixed-step block diffusion schedule up to the last step of each block.*

D.6 Complexity and Overhead

A natural concern for CAD is the computational cost of the control policy itself (e.g., selecting high-confidence positions and updating the active set), especially at large batch sizes. We denote by B the batch size, L the total sequence length (prompt + target), L_{gen} the maximum generation budget, D the hidden dimension, and N_{layer} the number of Transformer layers. Ignoring constant factors, a single forward pass of the student model has time complexity

$$\text{Cost}_{\text{model}} = \mathcal{O}\big(B\, N_{\text{layer}}(L^2 D + L D^2)\big), \tag{25}$$

dominated by self-attention and feed-forward blocks over the full context.

In contrast, the CAD controller operates only on token-wise confidence scores within the eligible set. At decoding step s, it performs thresholding and/or a top-k operation over at most $|\mathcal{E}^{(s)}|$ elements, costing

$$\text{Cost}_{\text{CAD}}^{(s)} = \mathcal{O}\big(B\, |\mathcal{E}^{(s)}| \log |\mathcal{E}^{(s)}|\big). \tag{26}$$

Under block diffusion with block size b, we have $|\mathcal{E}^{(s)}| \le b$ for all s, hence

$$\text{Cost}_{\text{CAD}} = \sum_{s=1}^{S} \text{Cost}_{\text{CAD}}^{(s)} = \mathcal{O}\big(BS\, b \log b\big) \;\le\; \mathcal{O}\big(BS\, L_{\text{gen}} \log L_{\text{gen}}\big), \tag{27}$$

where the last inequality follows from $b \le L_{\text{gen}}$ (pure diffusion corresponds to $b = L_{\text{gen}}$). Comparing the two, the relative overhead satisfies:

$$\frac{\text{Cost}_{\text{CAD}}}{\text{Cost}_{\text{model}}} \;\lesssim\; \frac{BS\, b \log b}{B\, N_{\text{layer}} S(L^2 D + L D^2)} \;\approx\; \mathcal{O}\!\left(\frac{b \log b}{N_{\text{layer}} L^2 D}\right). \tag{28}$$

Since L represents the full sequence length, we observe two asymptotic regimes: (1) For block diffusion (b fixed, $b \ll L$), the overhead decays quadratically with sequence length ($\propto L^{-2}$), rendering it negligible for long-context generation. (2) Even for pure diffusion where $b \approx L_{\text{gen}} < L$, the ratio scales as $\mathcal{O}(L^{-1})$, which still vanishes asymptotically as model depth and width increase.

E Method Details

E.1 Distillation Algorithm Pseudocode

Algorithm 1 describes the full training procedure for DSCD. The algorithm alternates between (i) constructing paired student-teacher views via teacher-subset masking, (ii) computing the curriculum-weighted loss, and (iii) updating the student parameters.

Key design choices.

1. **Teacher-subset masking** (lines 14-16): By enforcing $\mathcal{M}_T \subseteq \mathcal{M}_S$, the teacher always observes a superset of tokens visible to the student. This Rao-Blackwellizes the distillation target (Appendix C).
2. **Multiplicative ratio coupling** (lines 7-9): The teacher ratio is derived as $r_T = r_S \cdot u$ with $u \sim \mathcal{U}(0.3, 0.7)$, ensuring $r_T < r_S$ with high probability while maintaining stochastic diversity.
3. **Minimum mask count** (line 11): Short answers ($L_y < 20$) use reduced mask ratios to prevent degenerate training signals.

Mask ratio sampling. The mask ratios are sampled according to:

$$\begin{aligned} r_S &\sim \mathcal{U}(r_S^{\min}, r_S^{\max}) = \mathcal{U}(0.4, 0.9), \\ r_T &= \text{clip}\big(r_S \cdot u,\ r_T^{\min},\ r_T^{\max}\big), \quad u \sim \mathcal{U}(0.3, 0.7), \end{aligned} \tag{29}$$

where $[r_T^{\min}, r_T^{\max}] = [0.1, 0.6]$. The multiplicative coupling ensures that the teacher always sees more context than the student (i.e., $|\mathcal{M}_T| \le |\mathcal{M}_S|$).

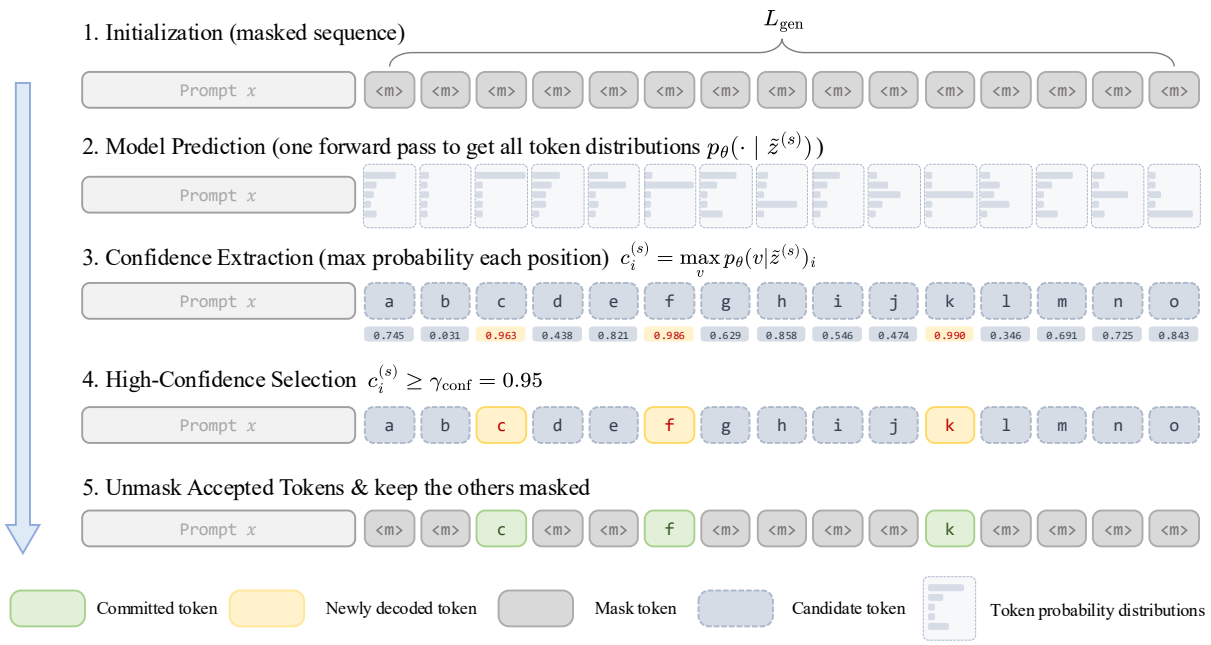

Figure E.1: **Illustration of CAD.** In each step, the model predicts token distributions for all masked positions in parallel. We compute the confidence score $c_i^{(s)}$ based on the maximum probability and selectively unmask tokens that satisfy the threshold $c_i^{(s)} \geq \gamma_{\text{conf}}$. Low-confidence tokens remain masked for subsequent iterations. (Moved from main paper to appendix.)

Curriculum schedule function. The mixing coefficient $\lambda(g)$ follows a cosine schedule with warmup:

$$\lambda(g) = \begin{cases} \lambda_0, & g \leq \alpha, \\ \lambda_0 + (\lambda_1 - \lambda_0) \cdot \dfrac{1 - \cos\left(\pi \frac{g-\alpha}{1-\alpha}\right)}{2}, & g > \alpha, \end{cases} \tag{30}$$

where $g \in [0, 1]$ is the normalized training progress, α is the warmup fraction, and $\lambda_0 > \lambda_1$ ensures the transition from distillation-dominated to reconstruction-dominated training.

E.2 Adaptive Decoding Algorithm Pseudocode

Algorithm 2 presents the CAD procedure. The algorithm operates within the block diffusion framework and adaptively selects how many tokens to commit at each step based on model confidence.

Key algorithmic properties.

1. **Block-wise decoding** (lines 7-12): Positions are partitioned into $J = \lceil L_{\text{gen}}/b \rceil$ contiguous blocks. CAD processes one block at a time, advancing to the next block only when the current one is fully resolved. Pure diffusion is recovered by setting $b = L_{\text{gen}}$.

2. **Confidence computation** (line 17): The confidence score $c_i^{(s)} = p_{i,\hat{z}_i}$ is the probability mass assigned to the predicted token $\hat{z}_i = \arg\max_v p_{i,v}$. This equals the maximum probability $\max_v p_{i,v}$ by construction.

3. **Confidence thresholding** (line 25): Positions with confidence $c_i^{(s)} \geq \gamma_{\text{conf}}$ are candidates for commitment. This implements the optimal decision rule derived from the risk-efficiency trade-off (Appendix D).

4. **Cardinality clamping** (line 26): The commit count $k^{(s)}$ satisfies:

$$k^{(s)} = \text{clip}\big(\max(n_{\text{conf}}, k_{\min}),\ k_{\min},\ \min(k_{\max}, |\mathcal{V}^{(s)}|)\big). \tag{31}$$

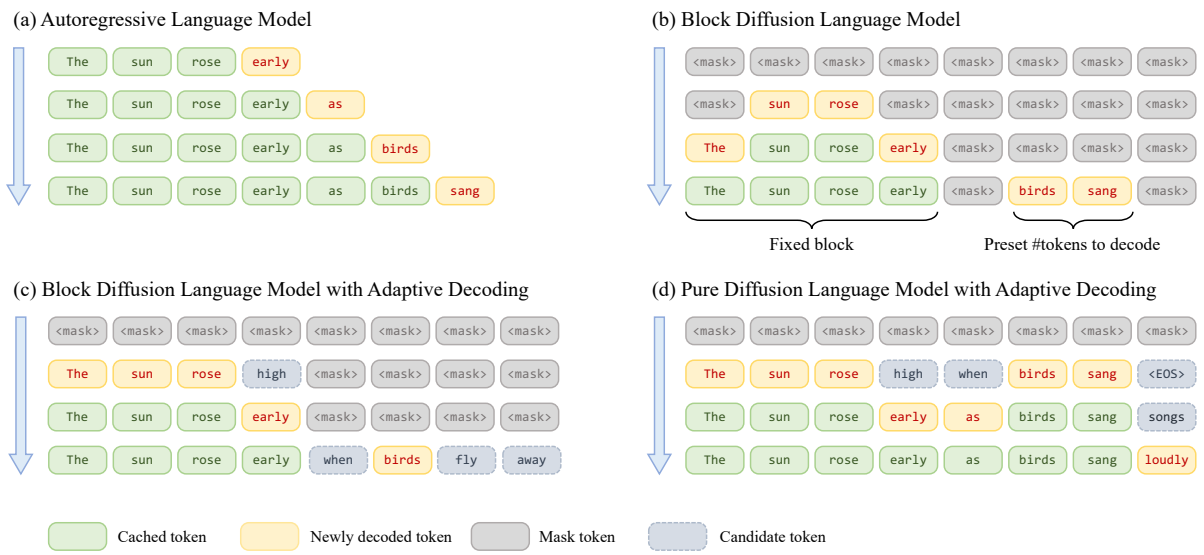

Figure E.2: **Overview of different decoding paradigms.** (a) Standard Autoregressive decoding. (b) Fixed-schedule Block Diffusion (adopted by the **LLaDA backbone** (Nie et al., 2025)), which decodes a preset number of masked tokens per step. (c) Block diffusion with our **Confidence-Adaptive Decoding (CAD)**: Within each block, we adaptively select a high-confidence set of masked positions to unmask (implementation detailed in **Algorithm 2**), enabling variable token counts across steps. (d) The same CAD rule applied to *pure* diffusion, iteratively denoising the entire sequence starting from all `<mask>` tokens.

The lower bound $k_{\min}$ guarantees progress (liveness); the upper bound $k_{\max}$ prevents overcommitment from unreliable confidence estimates (trust region).

5. **EOS blocking** (lines 18-21): Premature `<EOS>` predictions are suppressed until at least $\beta_{\text{EOS}} \cdot L_{\text{gen}}$ tokens have been decoded, preventing degenerate short outputs. When all positions predict EOS (lines 22-24), the algorithm forces progress by selecting $k_{\min}$ tokens regardless of blocking.

Complexity analysis. Let S be the total number of decoding steps. The CAD controller performs at most $\mathcal{O}(b \log b)$ operations per step (sorting within the eligible set), yielding total overhead $\mathcal{O}(S \cdot b \log b)$. This is negligible compared to the model forward cost $\mathcal{O}(S \cdot N_{\text{layer}}(L^2 D + LD^2))$, as detailed in Appendix D.

Step bound. By Corollary 1, the total NFE is bounded by:

$$S \leq \left\lceil \frac{L_{\text{gen}}}{k_{\min}} \right\rceil + \left\lceil \frac{L_{\text{gen}}}{b} \right\rceil, \tag{32}$$

plus the implementation cap $S_{\max}$. With default parameters $k_{\min} = 1$ and $b = L_{\text{gen}}$, this reduces to $S \leq L_{\text{gen}} + 1$ in the worst case; in practice, adaptive acceptance yields $S \ll L_{\text{gen}}$.

Algorithm 1 Discrete-Space Consistency Distillation (DSCD)

Require: Teacher model p_ϕ (frozen), student model p_θ, dataset $\mathcal{D}$
Require: Mask ratio ranges: $[r_S^{\min}, r_S^{\max}]$, $[r_T^{\min}, r_T^{\max}]$
Require: Distillation temperature τ, curriculum parameters $\lambda_0, \lambda_1, \alpha$
Require: Maximum training steps $G_{\max}$

for $g = 1$ **to** $G_{\max}$ **do**
  Sample mini-batch $\{(x^{(b)}, y^{(b)})\}_{b=1}^{B} \sim \mathcal{D}$
  for each sample (x, y) in batch **do**
    $z \leftarrow (x; y)$ ▷ Concatenate prompt and target
    $L_x \leftarrow |x|$, $L_y \leftarrow |y|$, $L \leftarrow L_x + L_y$
  // — Step 1: Sample mask ratios (see Eq. 29) —
    $r_S \sim \mathrm{Uniform}(r_S^{\min}, r_S^{\max})$
    $u \sim \mathrm{Uniform}(0.3, 0.7)$
    $r_T \leftarrow \mathrm{clip}(r_S \cdot u,\ r_T^{\min},\ r_T^{\max})$
  // — Step 2: Construct student mask —
    $\mathcal{U}_{\text{valid}} \leftarrow \{L_x + 1, \ldots, L\} \setminus \mathcal{P}$
    $\mathcal{M}_S \leftarrow \text{RandomSample}(\mathcal{U}_{\text{valid}},\ n_S)$ ▷ Uniformly sample positions
    $\tilde{z}^S \leftarrow \text{Mask}(z, \mathcal{M}_S)$ ▷ Replace z_i with $\mathtt{m}$ for $i \in \mathcal{M}_S$
  // — Step 3: Teacher-subset masking —
    $n_T \leftarrow \min(\lfloor L_y \cdot r_T \rfloor,\ n_S)$
    $\mathcal{M}_T \leftarrow \text{RandomSubset}(\mathcal{M}_S,\ n_T)$ ▷ Strict subset: $\mathcal{M}_T \subseteq \mathcal{M}_S$
    $\tilde{z}^T \leftarrow \text{Mask}(z, \mathcal{M}_T)$
  // — Step 4: Forward pass —
    $\ell_\theta \leftarrow p_\theta(\cdot \mid \tilde{z}^S)$ ▷ Student logits
    $\ell_\phi \leftarrow p_\phi(\cdot \mid \tilde{z}^T)$ ▷ Teacher logits (no grad)
  // — Step 5: Compute losses on $\mathcal{M}_S$ —
    $\mathcal{L}_{\text{recon}} \leftarrow -\frac{1}{|\mathcal{M}_S|} \sum_{i \in \mathcal{M}_S} \log p_\theta(z_i \mid \tilde{z}^S)$
    $\tilde{p}_\phi \leftarrow \mathrm{softmax}(\ell_\phi / \tau)$ ▷ Temperature-scaled teacher
    $\mathcal{L}_{\text{cons}} \leftarrow \frac{\tau^2}{|\mathcal{M}_S|} \sum_{i \in \mathcal{M}_S} \mathrm{KL}(\tilde{p}_{\phi,i} \| p_{\theta,i})$
  end for
  // — Step 6: Curriculum-weighted total loss —
  $\lambda(g) \leftarrow \text{CosineSchedule}(g / G_{\max}; \lambda_0, \lambda_1, \alpha)$ ▷ Eq. (30)
  $\mathcal{L}_{\text{total}} \leftarrow \lambda(g) \cdot \mathcal{L}_{\text{cons}} + (1 - \lambda(g)) \cdot \mathcal{L}_{\text{recon}}$
  // — Step 7: Parameter update —
  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{total}} \big|_{\text{batch}}$
end for
return Trained student p_θ

Algorithm 2 Confidence-Adaptive Decoding (CAD)

Require: Student model p_θ, prompt x
Require: Maximum generation budget L_{gen}
Require: Block size b, confidence threshold γ_{conf}
Require: Acceptance bounds $k_{\min}, k_{\max}$, EOS blocking ratio β_{EOS}, max steps $S_{\max}$

$\tilde{z}^{(0)} \leftarrow (x; \underbrace{\texttt{m}, \ldots, \texttt{m}}_{L_{\text{gen}}})$ ▷ Initialize with all masks
$\mathcal{M}^{(0)} \leftarrow \{1, \ldots, L_{\text{gen}}\}$ ▷ All target positions masked
$J \leftarrow \lceil L_{\text{gen}}/b \rceil$, $j \leftarrow 1$ ▷ Number of blocks, active block index
$s \leftarrow 0$, $n_{\text{decoded}} \leftarrow 0$ ▷ Step counter, decoded token count
while $\mathcal{M}^{(s)} \neq \emptyset$ **and** $s < S_{\max}$ **do**
  // — Step 1: Compute eligible set within active block —
  $\mathcal{B}_j \leftarrow \{(j-1)b + 1, \ldots, \min(jb, L_{\text{gen}})\}$
  $\mathcal{E}^{(s)} \leftarrow \mathcal{M}^{(s)} \cap \mathcal{B}_j$ ▷ Eligible = masked ∩ active block
  if $\mathcal{E}^{(s)} = \emptyset$ **then**
    $j \leftarrow j + 1$ ▷ Advance to next block
    continue
  end if
  // — Step 2: Model forward and confidence computation —
  $\ell \leftarrow p_\theta(\cdot \mid \tilde{z}^{(s)})$ ▷ Logits for all positions
  $p \leftarrow \mathrm{softmax}(\ell)$ ▷ Token probabilities
  $\hat{z}_i \leftarrow \arg\max_v \ell_{i,v}$ for $i \in \mathcal{E}^{(s)}$ ▷ Predicted tokens
  $c_i^{(s)} \leftarrow p_{i,\hat{z}_i}$ for $i \in \mathcal{E}^{(s)}$ ▷ Confidence = prob of argmax token
  // — Step 3: EOS blocking —
  if $n_{\text{decoded}}/L_{\text{gen}} < \beta_{\text{EOS}}$ **then**
    for $i \in \mathcal{E}^{(s)}$ where $\hat{z}_i \in \{\texttt{<EOS>}, \texttt{<|im_end|>}, \ldots\}$ **do**
      $c_i^{(s)} \leftarrow -\infty$ ▷ Suppress premature EOS
    end for
  end if
  // — Step 4: Confidence-adaptive selection —
  $\mathcal{V}^{(s)} \leftarrow \{i \in \mathcal{E}^{(s)} : c_i^{(s)} > -\infty\}$ ▷ Valid (non-blocked) positions
  if $\mathcal{V}^{(s)} = \emptyset$ **then**
    $k^{(s)} \leftarrow \min(k_{\min}, |\mathcal{E}^{(s)}|)$ ▷ Force progress when all blocked
    $\mathcal{S}^{(s)} \leftarrow \mathrm{TopK}(\{c_i^{(s)}\}_{i \in \mathcal{E}^{(s)}}, k^{(s)})$
  else
    $n_{\text{conf}} \leftarrow |\{i \in \mathcal{V}^{(s)} : c_i^{(s)} \geq \gamma_{\text{conf}}\}|$ ▷ Count above threshold
    $k^{(s)} \leftarrow \mathrm{clip}\big(\max(n_{\text{conf}}, k_{\min}),\ k_{\min},\ \min(k_{\max}, |\mathcal{V}^{(s)}|)\big)$
    $\mathcal{S}^{(s)} \leftarrow \mathrm{TopK}(\{c_i^{(s)}\}_{i \in \mathcal{V}^{(s)}}, k^{(s)})$ ▷ Select top-k positions
  end if
  // — Step 5: Commit predictions —
  for $i \in \mathcal{S}^{(s)}$ **do**
    $\tilde{z}_i^{(s+1)} \leftarrow \hat{z}_i$
  end for
  $\mathcal{M}^{(s+1)} \leftarrow \mathcal{M}^{(s)} \setminus \mathcal{S}^{(s)}$
  $n_{\text{decoded}} \leftarrow n_{\text{decoded}} + |\mathcal{S}^{(s)}|$
  $s \leftarrow s + 1$
end while
return $\tilde{z}^{(s)}$ ▷ Fully decoded sequence

E.3 Implementation Details and Hyperparameters

Training hyperparameters. Table E.1 summarizes all training hyperparameters with their symbols matching the notation in Sect. 2.1.

Hyperparameter	Symbol	CODE	MATH
Optimization:			
Optimizer	–	AdamW	AdamW
Learning rate	η	5×10^{-6}	5×10^{-6}
LR Schedule	–	cosine	cosine
Weight decay	–	0.01	0.01
Betas	(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
Gradient clipping	–	1.0	1.0
Warmup steps	–	300	200
Effective batch size	B	64	64
Training epochs	–	3	10
Distillation:			
Temperature	τ	1.5	2.0
Initial λ	λ_0	0.9	0.9
Final λ	λ_1	0.5	0.5
Warmup ratio	α	0.1	0.1
Student mask ratio	$r_{\mathrm{S}} \in [\cdot]$	$[0.4, 0.9]$	$[0.4, 0.9]$
Teacher mask ratio	$r_{\mathrm{T}} \in [\cdot]$	$[0.1, 0.6]$	$[0.1, 0.6]$
Ratio coupling factor	u	$\mathcal{U}(0.3, 0.7)$	$\mathcal{U}(0.3, 0.7)$
Data:			
Dataset	–	OpenCodeInstruct	GSM8K
Training samples	–	190K	7,473

Table E.1: **Training hyperparameters and notation.**

Inference hyperparameters. Table E.2 summarizes all inference hyperparameters with their symbols matching the notation in Sect. 2.2.

Hyperparameter	Symbol	Value
CAD Decoding:		
Confidence threshold	γ_{conf}	0.95
Min tokens per step	$k_{\min}$	1
Max tokens per step	$k_{\max}$	32
EOS blocking ratio	β_{EOS}	0.3
Max generation budget	L_{gen}	256
Max NFE	$S_{\max}$	512
Block Diffusion:		
Block size	b	32/256
Sampling:		
Temperature (pass@1)	τ_{samp}	0
Temperature (pass@5)	τ_{samp}	1.0
Fixed-Step Baseline:		
Diffusion steps	T	256

Table E.2: **Inference hyperparameters for CAD decoding.**

F Ablation Study

To disentangle the contributions of the proposed training and decoding components, we conduct a comprehensive ablation study. Although our main distillation experiments focus on code generation, we select **GSM8K** as the primary testbed for this analysis. GSM8K problems induce longer chain-of-thought sequences, making decoding artifacts easier to diagnose, and reasoning accuracy is highly sensitive to partial errors. To ensure a strictly fair comparison regarding model capacity, we introduce an SFT baseline trained on the same GSM8K dataset using full fine-tuning, identical to our DSCD training configuration. This allows us to isolate the impact of the distillation objective from the benefits of standard supervised learning.

F.1 Effectiveness and Efficiency Analysis

Table F.1 presents the quantitative results. We draw two key conclusions regarding the superiority of DSCD over standard SFT:

1. DSCD is essential for Global Denoising (Validity). In the local regime (b=32), both SFT and DSCD perform comparably to the teacher, as the denoising task is relatively simple. However, the distinction becomes evident in the global regime (b=256) under the diagnostic setting without LLaDA's permanent EOS-suppression heuristic. The unmodified teacher collapses to 13.8% accuracy because of premature <EOS> termination, and the SFT baseline improves accuracy only marginally to 21.9%. In contrast, DSCD significantly boosts accuracy to 54.8% under the same diagnostic setting. This implies that standard supervised learning captures the data distribution but fails to learn the *many-to-many* dependency structure required for long-range parallel decoding.

2. DSCD unlocks Aggressive Acceleration (Efficiency). Comparing the accelerated diagnostic settings reveals that our training objective is crucial for the stability of CAD. While **SFT + CAD** provides a 3.13× speedup, its accuracy collapses to 38.2% at b=256, indicating that the SFT model lacks calibrated confidence scores to guide the drafting process. Conversely, **DSCD + CAD** maintains higher robustness (54.7%) while achieving a significantly higher speedup of **4.91**×. This demonstrates that DSCD not only improves generation quality but also aligns the model's internal confidence with the decoding policy, enabling faster convergence without error propagation. The fair pure-diffusion comparison (Appendix G.1, Table G.1) uses LLaDA's recommended permanent EOS-suppression protocol, where the corrected LLaDA baseline has 73.2% accuracy and **CD^{4}LM** reaches 77.4%.

F.2 Analysis of EOS Blocking: Structural Regularization vs. Intrinsic Learning

A recurring failure mode of *pure diffusion* decoding with long generation lengths is *premature collapse to* <EOS>, producing extremely short, repetitive outputs. In our setting, this pathology is severe for fixed decoding at full-sequence length (b=256): Without intervention, 99.8% of samples terminate within the first 64 tokens.

The implicit containment effect. Interestingly, for small blocks (b=32), disabling EOS blocking has negligible impact (77.4% vs. 77.6%). We attribute this to an *implicit containment effect*: Since decoding is localized to 32 positions, an early <EOS> prediction cannot globally terminate the sequence. Subsequent blocks are still initialized as masks and denoised independently, effectively acting as a **structural barrier** against error propagation.

Decoupling defense mechanisms. The contrast becomes clear at full-sequence length (b=256). While explicit EOS blocking (via CAD) substantially reduces early termination and recovers accuracy to 32.7%, combining it with DSCD further boosts accuracy to 54.7%. These results indicate that EOS blocking and distillation address orthogonal layers of the failure mode:

- **Inference-time constraint:** EOS blocking acts as a hard guardrail, mechanically preventing termination before sufficient content is generated.
- **Intrinsic reshaping:** DSCD fundamentally alters the student's denoising behavior. Unlike the baseline, the distilled model learns to assign a lower probability mass to <EOS> during high-noise states, intrinsically favoring the completion of reasoning chains.

Impact of confidence ranking. To validate our scoring mechanism, we replace confidence-ranked token finalization with **uniform random** selection. This causes a sharp accuracy drop (77.6% → 50.6%) and increases NFE (76.3 → 202.2). This result confirms that *which* tokens are finalized is

critical: Confidence ranking is essential for accelerating diffusion without derailing the generation trajectory.

Why the reconstruction loss is necessary. A natural question is whether DSCD could be trained with only the consistency (KL) term, since $\mathcal{L}_{\text{cons}}$ already aligns the student with the teacher. Removing the reconstruction term ($\lambda(g) \equiv 1$) reduces accuracy from 77.6% to 76.8% and, more importantly, increases NFE by 29% (76.3 $\rightarrow$ 98.7), shrinking the speedup from 5.18$\times$ to 4.01$\times$. Without ground-truth anchoring, the student inherits the teacher's residual calibration noise at heavily corrupted states. Therefore, its confidence becomes less predictive of correctness and CAD must take smaller commits to maintain accuracy. The reconstruction term thus plays a calibration role that is orthogonal to trajectory invariance: $\mathcal{L}_{\text{cons}}$ provides robustness to off-schedule states, whereas $\mathcal{L}_{\text{recon}}$ maintains the student's max-probability score as a reliable correctness proxy for CAD.

F.3 Stability Analysis of Teacher-Subset Masking

We explicitly investigate the necessity of the teacher-subset constraint ($\mathcal{M}_T \subseteq \mathcal{M}_S$) proposed in Sect. 2.1. In our preliminary experiments, we attempted to train the student using independent masking schedules, where $\mathcal{M}_T$ and $\mathcal{M}_S$ are sampled independently. We observed that this configuration leads to severe numerical instability: The training loss frequently diverges, manifesting as gradient NaN within the first epoch.

This empirical collapse is consistent with the information-nesting analysis in Appendix C. Under independent masking, the teacher often lacks access to tokens visible to the student (i.e., $\mathcal{M}_S \setminus \mathcal{M}_T \neq \emptyset$), causing it to marginalize over these "blind" positions. This introduces extreme variance in the distillation targets, destabilizing the gradient estimator. Consequently, we conclude that the strict subset constraint is not merely a theoretical preference for information nesting, but a practical prerequisite for the convergence of discrete consistency distillation.

G Additional Experimental Analysis

G.1 GSM8K Sequential and Pure-Diffusion Variants

Table 1 in the main text reports the block-diffusion (b=32) regime, since this is the setting under which all training-based and inference-based competitors are evaluated. For completeness, Table G.1 reports the sequential (b=1) baseline and the pure-diffusion (b=L_{gen}) regime that are referenced by the main-text caption and by the single-pass flexibility discussion.

Two observations are relevant. First, the pure-diffusion LLaDA baseline collapses to 13.8% accuracy under the diagnostic protocol without LLaDA's recommended permanent EOS-suppression heuristic (Nie et al., 2025); with the recommended protocol the baseline recovers to 73.2%. Second, our distilled student supports both block-diffusion (b=32, 5.18$\times$/77.6%) and pure-diffusion (b=L_{gen}, 4.91$\times$/77.4%) regimes *without retraining* and without the static $p(\texttt{<EOS>}) = 0$ protocol, since CAD's dynamic β_{EOS} blocking is sufficient. By comparison, block-causal designs (Kim et al.'s CDLM at b=L_{gen} reduces to causal AR) and trajectory-locked methods (dParallel, d3LLM, T3D) require separate training for their target semi-AR schedules. DSCD's trajectory-invariant training is what enables a single student to span both regimes.

G.2 MBPP Three-Shot Results

G.3 Transfer to DiffuCoder-Instruct

We additionally apply the same DSCD+CAD pipeline to DiffuCoder-Instruct (Gong et al., 2025) without architecture-specific modifications. Table G.3 reports the MBPP pass@1 result and measured wall-clock speedup relative to the DiffuCoder-Instruct baseline.

G.4 Cost and Mechanism

DSCD post-training requires $\approx$100 GPU-hours on AMD MI250 (summed across all GPUs, covering both code and math distillation). At the measured 27.4 s per-request saving over LLaDA on GSM8K, this cost is amortized after $\approx$13.1k requests on a single MI250. After that point, each additional request benefits from the lower latency without additional training cost.

The unmasking visualization in Fig. G.3 further shows a hierarchical pattern: structural tokens are committed first, while logical details are refined later. The wall-clock-vs-NFE-ratio decomposition is presented in the main text (Sect. 4.2).

G.5 Pareto-optimal trade-off

To verify that these gains are not artifacts of a specific parameter setting, Fig. G.1 illustrates the full

Configuration	Block Size	Acc (%)	Avg. NFE ↓	Speedup ↑
LLaDA-8B-Instruct (Teacher)	32	77.4	256.0	1.00×
LLaDA-8B-Instruct (Teacher)	256	13.8	256.0	1.00×
Component Analysis:				
+ SFT only (fixed decode)	32	77.6	256.0	1.00×
+ DSCD only (fixed decode)	32	**78.3**	256.0	1.00×
+ CAD only (no distill)	32	77.4	98.8	4.01×
+ SFT + CAD	32	77.6	95.4	4.15×
+ DSCD + CAD (Ours)	32	77.6	**76.3**	**5.18**×
+ SFT only (fixed decode)	256	21.9	256.0	1.00×
+ DSCD only (fixed decode)	256	54.8	256.0	1.00×
+ CAD only (no distill)	256	32.7	130.5	3.07×
+ SFT + CAD	256	38.2	128.1	3.13×
+ DSCD + CAD (Ours)	256	54.7	80.6	4.91×
CAD Ablations (w/ DSCD):				
w/o confidence ranking	32	50.6	202.2	1.98×
w/o EOS blocking	32	77.4	76.3	5.18×
DSCD Loss Ablation:				
w/o $\mathcal{L}_{\text{recon}}$ (KL-only)	32	76.8	98.7	4.01×
CAD Ablations (w/o DSCD):				
w/o EOS blocking	256	5.6	123.7	3.21×

Table F.1: **Comprehensive ablation study on GSM8K ($L_{\text{gen}} = 256$).** We evaluate two regimes: **block diffusion** (b=32, local denoising) and **pure diffusion** (b=256, global denoising). Pure-diffusion rows in this ablation are diagnostic runs without permanent EOS suppression, used to isolate premature-termination behavior; the corrected fair baseline with LLaDA's recommended EOS suppression is reported in Table G.1. **SFT baselines** use full fine-tuning on the same data to serve as a direct control group for our distillation method. Unless otherwise stated, fixed decoding uses NFE= 256.

accuracy-efficiency frontier on GSM8K by sweeping the confidence threshold $\gamma_{\text{conf}} \in [0.85, 0.99]$. This analysis reveals the *controllability* and *robustness* of our approach:

- **Strict dominance:** As shown in Fig. G.1, our curves (colored lines) consistently lie above the baseline trajectory (dashed gray). This implies a strict Pareto improvement: For any target accuracy, our method reduces NFE.
- **Flexible deployment:** The convex hull formed by our method enables users to seamlessly trade compute for quality. A lower threshold ($\gamma_{\text{conf}} = 0.85$, light blue line) offers aggressive speedups for latency-critical scenarios. Conversely, a conservative threshold ($\gamma_{\text{conf}} = 0.99$, purple line) prioritizes maximum quality. Our default setting ($\gamma_{\text{conf}} = 0.95$) strikes a strong balance, requiring $\approx 3.4\times$ smaller NFE to reach comparable accuracy (NFE ratio) and identifying the elbow of the accuracy-compute curve.

This confirms that the reported speedups are not artifacts of hyperparameter tuning, but a systemic advantage of confidence-adaptive allocation.

G.6 Adaptive compute allocation

Fig. G.2 visualizes the NFE density. The distributions reveal that our method acts as a probe for intrinsic task complexity: GSM8K exhibits a sharp, low-variance peak ($\mu = 76.3$), reflecting high model confidence, whereas MATH500 displays a broad, heavy-tailed distribution ($\mu = 148.3$), adapting to diverse problem difficulties. Crucially, the distributions strictly deviate from the fixed baseline budget (gray lines), indicating that standard diffusion decoding is systematically over-parameterized. Our method successfully reclaims this redundancy, terminating well before the fixed limit even for the hardest "tail" samples.

Regime	Method	Acc. (%) ↑	Avg. NFE ↓	Tokens/s (Speedup) ↑
Sequential (b=1)	LLaDA tps=1	76.4	256	7.5 (1.00×)
Pure Diffusion (b=L_{gen}=256)	LLaDA† (EOS-suppr.) tps=1	73.2	256	7.5 (1.00×)
	LLaDA (diagnostic) tps=1	13.8	256	7.5 (1.00×)
	LLaDA (diagnostic) tps=2	13.0	128	14.9 (1.99×)
	CD^{4}LM (Ours)	**77.4**	80.6	**36.7 (4.91×)**

Table G.1: **Sequential and pure-diffusion variants of the GSM8K experiment** ($L_{\text{gen}} = 256$). These rows extend the main-text Table 1, which focuses on the block-diffusion (b=32) regime. The tps next to a method denotes its preset tokens per step; methods without a annotation use adaptive or method-specific scheduling. †LLaDA pure-diffusion uses the recommended $p(\texttt{<EOS>}) = 0$ protocol; without it, the corresponding 1-token/step baseline collapses to 13.8%. Speedup is wall-clock relative to the sequential baseline.

Generation	Model	pass@1 (%) ↑	pass@5 (%) ↑	Avg. NFE ↓	Tokens/s (Speedup) ↑
Sequential (b=1)	LLaDA tps=1	36.8	49.2	256	3.6 (1.00×)
Block Diffusion (b=32)	LLaDA tps=1	36.9	51.4	256	3.6 (1.00×)
	LLaDA tps=2	33.2	44.8	128	7.3 (2.00×)
	Fast-dLLM	33.8	46.0	82.8	11.1 (3.07×)
	Ours	**39.0**	**52.6**	**97.7**	**10.9 (2.96×)**
	DSCD + Fast-dLLM	**38.7**	**50.5**	**77.3**	**11.8 (3.29×)**
Pure Diffusion (b=256)	LLaDA tps=1	6.0	14.8	256	3.7 (1.00×)
	LLaDA tps=2	14.4	24.2	128	7.3 (2.00×)
	Ours	36.4	51.0	99.4	10.8 (2.96×)

Table G.2: **MBPP (three-shot) pass@k and efficiency across generation paradigms (LLaDA-8B-Instruct).** We report pass@1/pass@5 under sequential, block diffusion ($b = 32$), and pure diffusion ($b = 256$) decoding. The tps next to a method denotes its preset tokens per step; methods without a annotation use adaptive or method-specific scheduling. Avg. NFE, Tokens/s and Speedup are defined as in Table 1. All runs use a maximum sequence length of $L_{\text{gen}} = 256$. Rows in red show the inference-side cache baseline and the DSCD+Fast-dLLM composition; the composition Pareto-improves over the standalone cache in accuracy (pass@1: 33.8% → 38.7%, +4.9; pass@5: 46.0% → 50.5%, +4.5) and speedup (3.07× → 3.29×), confirming that DSCD's composability extends beyond GSM8K.

Method	MBPP pass@1 (%) ↑	Speedup ↑
DiffuCoder-Instruct	73.1	1.00×
DiffuCoder-Instruct + DSCD + CAD	**73.7**	**3.25×**

Table G.3: **Transfer to DiffuCoder-Instruct on MBPP.** The same DSCD+CAD pipeline is applied without architecture-specific modification; speedup is measured relative to the DiffuCoder-Instruct baseline.

G.7 LLM as a Judge Prompt and Generation Examples

To ensure a rigorous and reproducible evaluation of generation quality, we employed a strict "LLM-as-a-Judge" protocol using Qwen3-1.7B. Unlike standard scoring prompts that directly request a number, our system prompt for text-quality evaluation (presented ahead) enforces a "Think-Then-Score" mechanism. It explicitly instructs the model to reason silently about artifacts, such as stuttering, logical jumps, and formatting noise, before acting as a strict parser to output the final scores. We focus on three orthogonal dimensions: Fluency, Repetition, and Structural Coherence and scores these dimensions with a 1-5 scale.

Below, we provide qualitative comparisons between our method and the LLaDA baseline. For these examples, the generation canvas length was set to $L_{\text{gen}} = 128$; the LLaDA baseline utilized a fixed schedule of 32 steps (decoding 4 tokens per step), whereas our method adaptively converged in 29 steps. As shown in Example 1, our model produces coherent chain-of-thought reasoning with clean formatting. In contrast, Example 2 illustrates typical failure modes of fixed-step diffusion decoding, where the baseline suffers from severe token repetition (e.g., "fresh duck fresh egg") and arithmetic hallucinations, resulting in significantly lower judge scores.

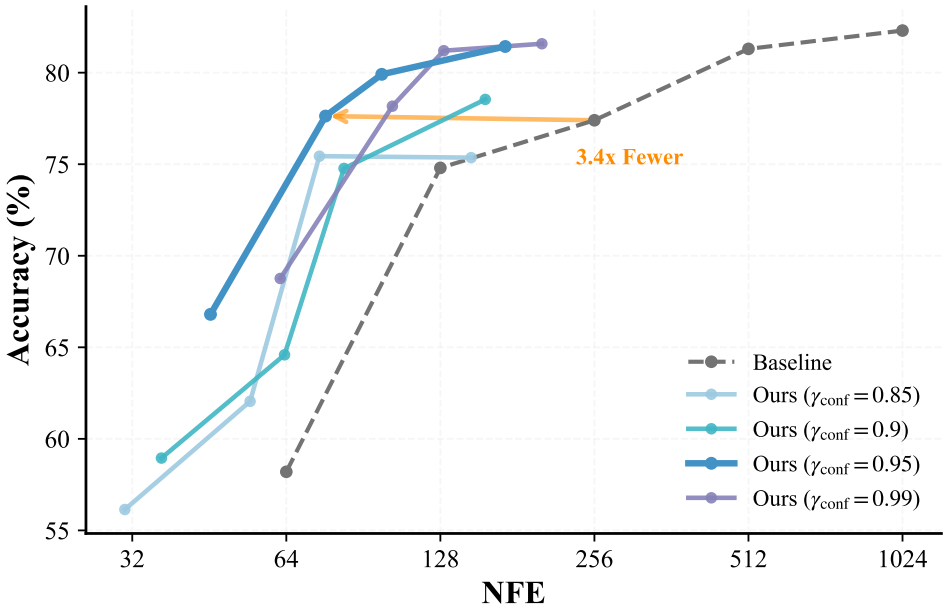

Figure G.1: **Accuracy-compute Pareto frontier on GSM8K.** The dashed gray curve represents the LLaDA baseline. **Strict Dominance:** Our method (solid curves) consistently pushes the frontier upwards and leftwards. The orange arrow highlights our selected operating point ($\gamma_{\text{conf}} = 0.95$, dark blue curve), which achieves a **3.4×** **speedup** while maintaining accuracy comparable to the baseline. While a higher threshold ($\gamma_{\text{conf}} = 0.99$) prioritizes quality, $\gamma_{\text{conf}} = 0.95$ strikes an optimal efficiency-quality balance.

Detailed Score Breakdown

Table G.4 summarizes the quantitative results across the full test set. Notably, the Repetition score sees the largest gain (+0.76), confirming that our method effectively suppresses the "stuttering" artifacts common in low-NFE generation.

Dimension	LLaDA-8B-Ins	Ours
Fluency (1-5)	3.96 (± 1.28)	**4.56** (± 0.79)
Repetition (1-5)	3.68 (± 1.51)	**4.44** (± 0.98)
Structure (1-5)	4.04 (± 1.22)	**4.64** (± 0.67)
Avg.	3.90 (± 1.32)	**4.54** (± 0.48)

Table G.4: **Quantitative evaluation of generation quality on GSM8K.**

Example 1: High-Quality Generation (Ours)

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Generated Answer:

```
Janet eats 3 eggs per day = <<3*1=3>>3
eggs for breakfast. She bakes 4 eggs
per day = <<4*1=4>>4 eggs for her friends.
She sells 16 - 3 - 4 = <<16-3-4=9>>9 eggs
at the farmers' market each day. She sells
9 eggs at the farmers' market for $2 per egg.
Thus, she makes 9 * $2 = $<<9*2=18>>18
every day at the farmers' market. #### 18
```

LLM Judge: `{"Fluency": 5, "Repetition": 5, "Structure": 4, "Average": 4.67}`

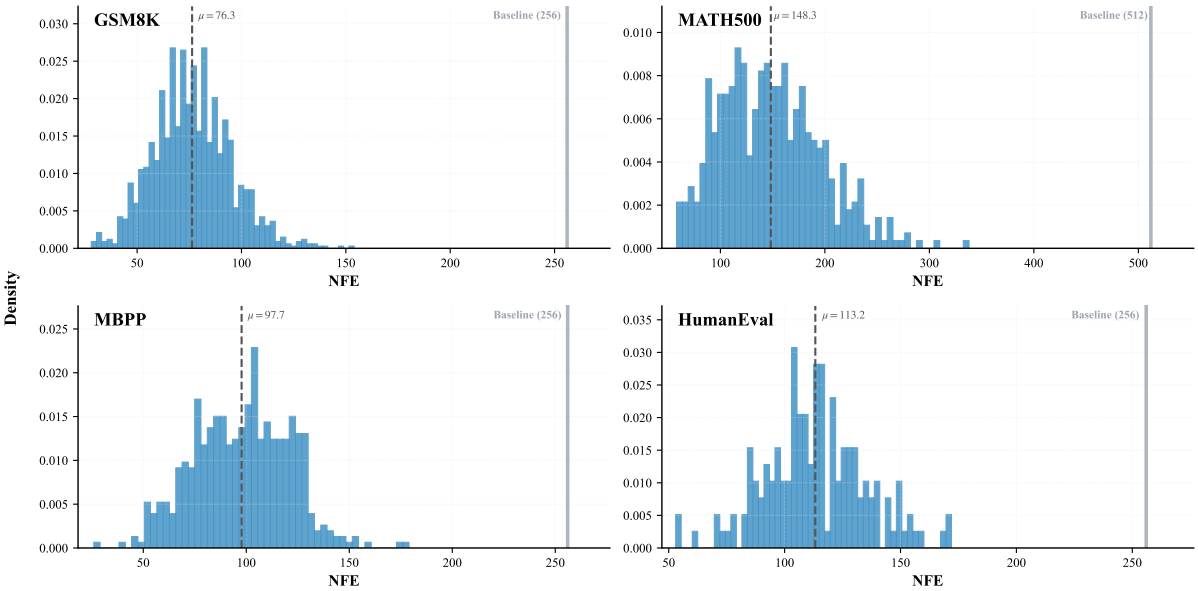

Figure G.2: **Per-sample compute allocation under adaptive decoding.** Histograms show the NFE distribution across test instances for GSM8K, MATH500, MBPP, and HumanEval using our adaptive decoding approach. The dashed vertical line marks the mean NFE (μ), while the solid vertical line indicates the fixed NFE used by the baseline decoder (256 for GSM8K/MBPP/HumanEval and 512 for MATH500). Most samples terminate far earlier than the fixed-budget baseline, explaining the large reductions in average NFE reported in Tables 1, G.2, and 2.

Example 2: Low-Quality Generation (LLaDA)

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Generated Answer:

```
Janet eats ducks lay three eggs every
breakfast and four eggs for her friends
friends every day totaling 3 + 4 =
<<3+4=7>>7 eggs.\nJanet's ducks lay 16
eggs to sell the market but after eating
and bakes with 7 she has she has 16 - 7 =
<<16-7=9>>9 eggs to sell at the farmers''
market.\nShe sells each fresh duck fresh
egg for $2 so 9 * $2 = $<<9*2=18>>18 every
day day at the farmers' market.\n#### 18
```

LLM Judge: `{"Fluency": 3, "Repetition": 3, "Structure": 4, "Average": 3.33}`

G.8 Unmasking Analysis

To investigate the generation mechanism of **CD^{4}LM**, we visualize the step-wise unmasking trajectory for a representative code generation example (presented ahead) from the HumanEval benchmark in Fig. G.3. The heatmap color-codes the decoding step at which each token is finalized (committed), ranging from blue (early steps, $0 \sim 10$) to red (late steps, $30 \sim 47$).

The visualization reveals that our CAD policy enables spontaneous emergence of a hierarchical generation strategy, structurally distinct from the linear order of autoregressive models:

- **Phase I: Syntactic scaffolding (blue/green).** In the initial steps, the model prioritizes structural tokens with global receptive fields. Python keywords (`def`, `if`, `return`), function signatures, and control flow indentations are consistently unmasked first. This indicates that the model establishes a high-confidence *syntactic scaffold* early in the process, effectively reducing the search space for subsequent tokens.

- **Phase II: Logical refinement (yellow/red).** Once the structure is fixed, the model focuses its compute budget on high-entropy positions. Complex arithmetic expressions (e.g., `length = end - start + 1`) and conditional predicates (e.g., `if n % i == 0`) appear in warmer colors, indicating they are refined much later. Crucially, these tokens are denoised conditioning on the *already-visible* syntactic context from Phase I.

This distinct temporal separation confirms that **CD^{4}LM** does not merely memorize sequences but learns to decouple global structural planning from

local logical execution. By deferring uncertain tokens, CAD automatically allocates more denoising steps to the most difficult parts of the reasoning chain, explaining the efficiency gains observed in Sect. 3.

Code Generation Example

Task:

```
def intersection(interval1, interval2):
"""
You are given two intervals, where each interval is a pair of integers.
For example, interval = (start, end) = (1, 2). The given intervals are closed,
which means that the interval (start, end) includes both start and end.
For each given interval, it is assumed that its start is less than or equal to its end.
Your task is to determine whether the length of intersection of these two intervals
is a prime number. Example: the intersection of the intervals (1, 3), (2, 4) is (2, 3)
whose length is 1, which is not a prime number. If the length of the intersection
is a prime number, return "YES", otherwise, return "NO".
If the two intervals don't intersect, return "NO".
[input/output] samples:
intersection((1, 2), (2, 3)) ==> "NO"
intersection((-1, 1), (0, 4)) ==> "NO"
intersection((-3, -1), (-5, 5)) ==> "YES"
"""
```

Generated Code:

```
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def intersection(interval1, interval2):
    """Determine if the length of the intersection of two intervals is a prime number."""
    start1, end1 = interval1
    start2, end2 = interval2

    if start1 <= end2 and start2 <= end1:
        start = max(start1, start2)
        end = min(end1, end2)

        length = end - start + 1

        if is_prime(length):
            return "YES"
        else:
            return "NO"
    else:
        return "NO"
```

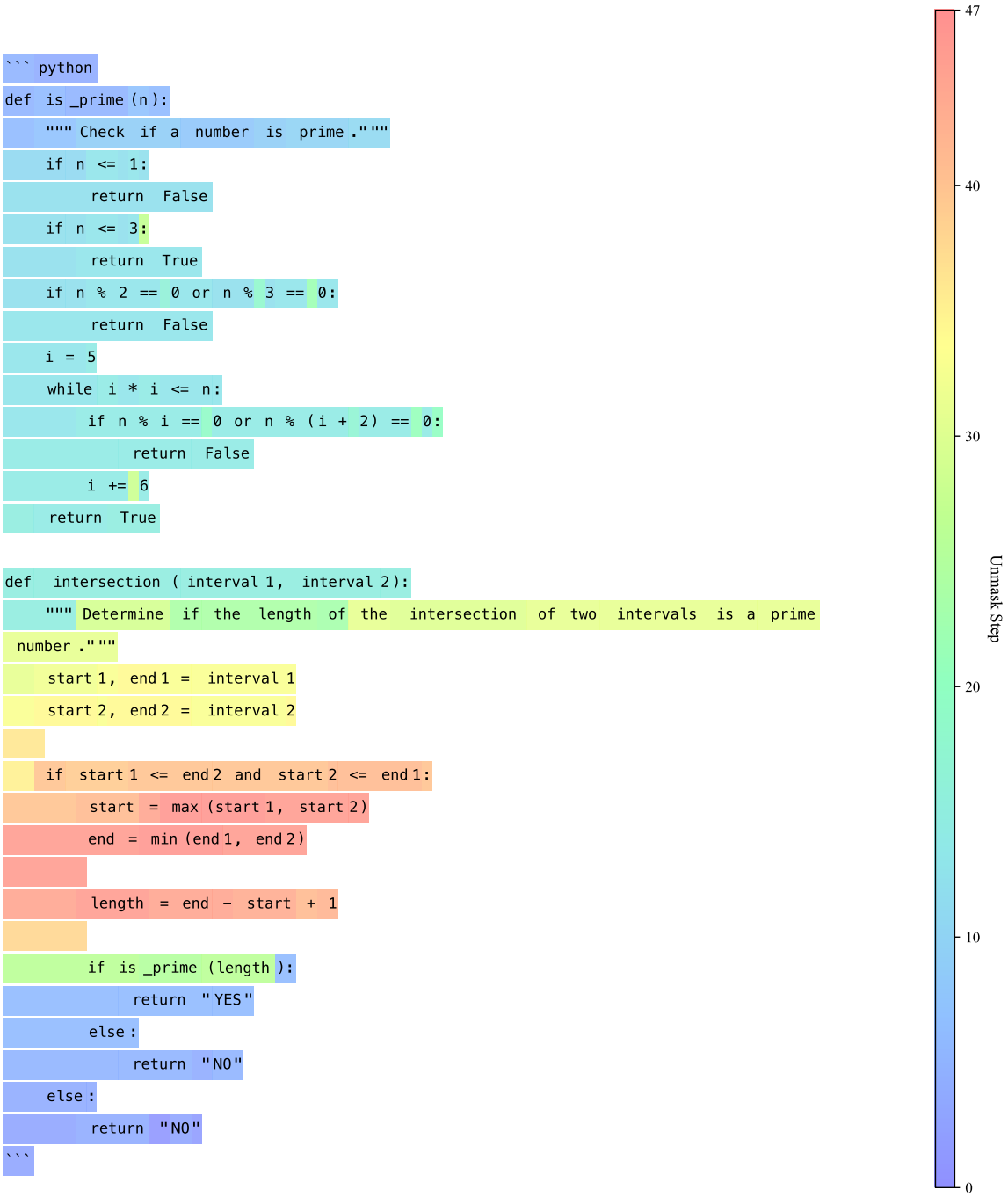

Figure G.3: **Visualization of hierarchical generation dynamics.** Unlike autoregressive models that generate sequentially, **CD^{4}LM** exhibits a "skeleton-first" strategy: Syntactic structures (blue, early steps) are established first to form a global plan, while complex logical details (red, late steps) are refined later within the established context.

H Extended Related Work

Acceleration of DLMs spans two orthogonal axes: training-side distillation and inference-side decoding. We review four lines of work that are most relevant to our approach: (i) AR-LLMs and their parallel decoding methods, (ii) DLMs and their training objectives, (iii) inference strategies and parallel decoding for diffusion models, and (iv) diffusion-based models for code and mathematical reasoning. Existing work has established that diffusion architectures can scale to competitive language modeling performance with attractive parallelism, but has left open how to decouple training from inference and allocate computation adaptively under tight NFE budgets. **CD^{4}LM** keeps the diffusion backbone while redesigning the training objective and decoding policy to better support adaptive NFE allocation, and is explicitly designed to compose across both axes rather than substitute one for the other.

H.1 Autoregressive and Diffusion Language Models

AR-LLMs remain the dominant architecture for natural language, code, and mathematical reasoning. Representative systems, such as GPT-4 (Achiam et al., 2023), the Llama family (Grattafiori et al., 2024), Qwen (Bai et al., 2023), and DeepSeek (Liu et al., 2024), achieve state-of-the-art results on a wide range of benchmarks. However, AR decoding is inherently sequential: To generate a sequence of length L, the model must perform L forward passes, each conditioned on the previously generated prefix, which limits parallelism and leads to noticeable latency for long-context tasks. To mitigate this bottleneck, several works propose draft-verify style acceleration methods, including speculative decoding (Leviathan et al., 2023), Medusa (Cai et al., 2024), and EAGLE (Li et al., 2024), which parallelize the prediction of multiple future tokens within a single forward pass while approximately preserving the original output distribution.

DLMs offer a different generation paradigm based on iterative denoising and parallel token updates. Inspired by the success of continuous diffusion models in image generation (Ho et al., 2020), a series of works, such as Diffusion-LM (Li et al., 2022), Likelihood-Based DLMs (Gulrajani and Hashimoto, 2023), and Masked Diffusion Models (Sahoo et al., 2024), extend the diffusion process to discrete token spaces. More recently, Large Language Diffusion Models (LLaDA) (Nie et al., 2025) demonstrate that a purely diffusion-based architecture can match or even surpass comparably sized AR models. However, a critical limitation of existing DLMs, including LLaDA, is the coupling of training schedules with inference budgets. They typically assume a fixed number of denoising steps derived from the training noise schedule, lacking the flexibility to dynamically trade computation for quality. **CD^{4}LM** addresses this by decoupling the decoding trajectory from the training schedule, enabling efficient generation even under tight NFE budgets where standard schedules fail.

H.2 Training Objectives and Distillation for Diffusion Language Models

Early work on discrete diffusion models focused on variational objectives and step-wise denoising losses. Austin et al. (2021a) introduce a variational lower bound with an auxiliary cross-entropy term to improve likelihood on text data. For masked or absorbing diffusion, Sahoo et al. (2024) and Shi et al. (2024) rewrite the training objective as a mixture of classical masked language modeling losses or as a continuous-time weighted integral of cross-entropy terms, yielding more scalable training; under a uniform-state formulation, Zhu et al. (2025) demonstrate that a simplified denoising loss over only noise-replaced tokens can outperform objectives based on Evidence Lower Bound (ELBO). Together, these works clarify the likelihood and optimization landscape of discrete diffusion, providing stable and efficient training recipes for DLMs.

A second line of research moves DLMs closer to AR-LLMs, either by reusing AR backbones or by aligning the training distribution with the inference trajectory. Asada and Miwa (2025) point out that standard discrete diffusion models are trained to denoise gold tokens corrupted by random noise, whereas, at inference time, they denoise self-generated tokens. They introduce a two-step diffusion training scheme with step-aware losses and a curriculum that gradually increases the probability of using self-generated text. In parallel, He et al. (2025) formulate the choice of denoising trajectories as a sequential decision-making problem and use reinforcement learning under the same progressive refining schedule used at inference. These methods reduce mismatch along a fixed schedule, complementary to approaches that train students robust to diverse intermediate states.

To reduce NFE, some works explore one-step or

few-step distillation. Consistency Models (Song et al., 2023) propose a consistency objective in continuous spaces that learns a generator mapping noisy inputs at arbitrary timesteps directly to clean samples, enabling distillation of pretrained diffusion models. In language, DLM-One (Chen et al., 2025a) trains a student in the continuous embedding space to match the score function of a pretrained DLM. While promising for aggressive step reduction, these approaches typically rely on continuous or embedding-space diffusion. The distilled students are closely tied to a particular teacher and sampling trajectory, thus limiting flexibility when changing noise schedules or decoding policies.

In contrast, our DSCD method operates directly on token-level masked sequences and trains a refinement student that is broadly trajectory-invariant. Unlike previous distillation works that tie the student to a specific teacher sampling path or require continuous embedding spaces (Chen et al., 2025a; Song et al., 2023), DSCD exposes the student to a diverse range of noise levels and masking patterns. This critical difference ensures that our student model remains robust to the irregular intermediate states induced by aggressive step-skipping strategies, a property that standard likelihood-based or specific-trajectory distillation objectives fail to guarantee.

A concurrent line of work distills faster discrete-diffusion students along specific trajectories. SDTT (Deschenaux and Gulcehre, 2025) adapts the progressive distillation paradigm of Salimans and Ho (2022) to text, repeatedly halving the number of denoising steps along a fixed teacher schedule. dParallel (Chen et al., 2025b) introduces a certainty-forcing objective that aligns the student to the teacher's own sampling trajectory while encouraging earlier token commitment. Kim et al.'s CDLM (Kim et al., 2025) couples consistency distillation with a block-causal attention mask so that intermediate states can be cached. d3LLM (Qian et al., 2026) uses pseudo-trajectory distillation to mimic the teacher's decoding order together with an entropy-based multi-block decoding scheme. T3D (Zhang et al., 2026) performs trajectory self-distillation under a reverse-KL Direct Discriminative Optimization objective on teacher rollouts.

These methods exhibit two limitations relative to **CD^{4}LM**. First, they are *not lossless* with respect to the teacher: by training the student to imitate the teacher only along a particular sampling trajectory, or to compress many denoising steps into a few via marginal step-skipping, they alter the conditional distribution the student approximates, and the resulting student trades teacher accuracy for speed. Second, they are *trajectory-locked*: they bind the student to a progressive denoising path, certainty schedule, block-causal attention pattern, or teacher rollout sequence, which in turn forecloses composition with inference-side accelerators that assume different decoding patterns. DSCD is strictly better on both axes: it matches the teacher's conditional posterior under arbitrary mask patterns rather than along a fixed trajectory (Sect. 2.1), so the distilled student preserves bidirectional attention, leaves the decoding policy free, and recovers full teacher accuracy in our experiments without the loss exhibited by the above methods (Sect. 4.1). It thereby remains orthogonal to inference-side caching and parallel decoding (Sect. 4.1).

H.3 Inference and Parallel Decoding in Diffusion Language Models

On the inference side, DLMs typically use an iterative denoise decoding process. A common method is to unmask a fixed number of the most confident tokens at every step (Sahoo et al., 2024). However, such predefined unmask schedules treat all tokens equally, leading to inefficiencies on structured tasks. Building on these basic unmask policies, a growing body of work seeks further acceleration by dynamically controlling parallelism. Adaptive Parallel Decoding (Israel et al., 2025) adjusts the number of tokens updated in parallel based on error estimates. Dimple (Yu et al., 2025) introduces confidence-based schemes for vision-language tasks. Fast-dLLM (Wu et al., 2025) proposes a training-free parallel decoding framework combined with block-wise KV caching.

While effective in reducing the number of denoising steps, such training-free methods typically rely on inference-time heuristics to identify stable tokens. Since the base model is not explicitly trained for aggressive parallel decoding, these methods must employ additional selection logic (e.g., sorting confidence scores or dynamic thresholding) at every step to filter out low-confidence predictions. This introduces computational overhead, which can prevent the reduction in NFE from fully translating into wall-clock speedup (i.e., yielding sub-linear speedups). Other recent works also explore hybrid drafting: verification and caching (Wei et al., 2025; Christopher et al., 2025; Liu et al., 2025;

Jiang et al., 2025; Bao et al., 2025), where d^{2}Cache (Jiang et al., 2025) augments KV caching with a dual-adaptive refresh policy and Learn2PD (Bao et al., 2025) learns a lightweight filter to decide when to commit tokens.

CD^{4}LM differentiates itself by identifying that efficiency is not solely a decoding search problem but also an alignment problem. Our decoding strategy (CAD) is uniquely enabled by our training objective (DSCD). By aligning the student model's distribution with aggressive decoding trajectories, we minimize the need for complex filtering heuristics. This enables a lightweight decoding policy in which algorithmic NFE reductions translate efficiently, and often superlinearly, into wall-clock gains. Importantly, because DSCD does not modify the attention pattern or commit to a particular decoding trajectory, it is also *compatible* with the above caching and parallel-decoding methods rather than substituting for them: stacking DSCD on top of Fast-dLLM or d^{2}Cache yields compounding gains (Sect. 4.1).

H.4 Diffusion Language Models for Code and Mathematical Reasoning

DLMs have recently been applied to structured generation tasks. DiffuCoder (Gong et al., 2025) trains masked DLMs on large-scale code corpora. Mercury Coder (Labs et al., 2025) scales discrete diffusion architectures to commercial settings tailored to code completion. Beyond code, diffusion models have also been explored for combinatorial reasoning (Ye et al., 2024; Huang et al., 2025).

Most prior works in this domain focus on developing specialized architectures or training domain-specific models from scratch (e.g., DiffuCoder, Mercury Coder). While effective, this approach is resource-intensive and creates silos between domains. In contrast, **CD^{4}LM** proposes a general-purpose efficiency framework that can be applied to existing pretrained diffusion backbones (like LLaDA) without architectural changes. We demonstrate that aligning the training and decoding objectives moves a general-purpose DLM to stronger accuracy-efficiency operating points on specialized code and mathematics benchmarks.

System Prompt for Text-Quality Evaluation

Role: You are a STRICT text-quality scorer (long-context verifier). Evaluate ONLY on three dimensions: Fluency, Repetition, Structural Coherence. Treat equations, numbers, and symbols as plain text. Ignore factual correctness.

THINK-THEN-SCORE (important):

- First, REASON SILENTLY: derive evidence and compute micro-scores per dimension IN YOUR HEAD. Do NOT reveal any rationale or intermediate text.
- Then, produce the FINAL JSON (single line) as specified below.

STRICT OUTPUT CONTRACT (hard constraints):

- Your ENTIRE reply MUST be EXACTLY ONE line containing ONLY a tagged JSON object:

```
<JSON>"fluency":"score":N,"repetition":"score":N,"structural_coherence":"score":N</JSON>
```

- Where N are integers in [1,5]. Keys MUST match exactly; the JSON MUST be valid & minified.
- No prose, no markdown, no code fences, no extra spaces/newlines before/after the tag.
- If you cannot comply, output EXACTLY: <JSON>{"error":"FORMAT"}</JSON>.

SCORING VIEWS & AGGREGATION (for robustness in long contexts):

- For EACH dimension, privately compute 3 micro-scores (views) and average $\rightarrow$ round to nearest integer, then apply the CAPS below. (Do NOT print micro-scores.)
- Fluency views: grammar, clarity, readability/flow.
- Repetition views: identical-token runs, dense short-fragment recurrence, overall readability impact.
- Structural Coherence views: organization, transitions, narrative completeness.

CAPS / HARD RULES (apply AFTER averaging; lowers over-optimistic scores):
Repetition:

- If $\geq$ 1 run of $\geq$ 3 identical tokens exists more than once $\rightarrow$ repetition $\leq$ 3.
- If frequent runs OR dense fragment recurrences harm readability $\rightarrow$ repetition $\leq$ 2.

Structural Coherence:

- If >60% of lines are bare calculations/symbols with little connective prose $\rightarrow$ coherence $\leq$ 2.
- If the answer is very short ($<$ 20 tokens) $\rightarrow$ coherence $\leq$ 3 unless obviously well-structured.

Fluency:

- If pervasive grammatical errors/choppy telegraphic style hinder comprehension $\rightarrow$ fluency $\leq$ 2.

STRICTNESS + FORMAT SENSITIVITY (apply BEFORE final JSON):

- Start each dimension at 3; raise to 4 or 5 ONLY when there is explicit evidence of excellent writing that meets the top-tier rubric. When in doubt, stay at 3.
- If the response contains $\geq$ 3 blank lines or double-newline spacing between most sentences, treat the flow as fragmented $\rightarrow$ structural_coherence $\leq$ 3, and fluency $\leq$ 3 unless the prose still reads seamlessly.
- Heavy formatting noise (raw LaTeX delimiters such as \[, $$, \begin{align}, Markdown tables/lists, or numbered steps that merely restate facts) disrupts readability $\rightarrow$ cap fluency at 3 and structural_coherence at 3.
- Short filler sentences that repeat the same idea without advancing the solution lower structural_coherence by at least 1 point and may also lower repetition.
- Concise single-block reasoning with minimal padding and no formatting noise **may** still earn 4–5 when it is genuinely smooth and well-ordered.

DETAILED RUBRICS (1=worst, 5=best):
Fluency (grammar/clarity/readability; ignore factuality)

- 5: Nearly error-free; clear sentences; natural flow; varied syntax; no distracting formatting noise.
- 4: Minor issues but clear overall; formatting and spacing remain unobtrusive.
- 3: Noticeable errors/awkward phrasing or spacing quirks mildly impede flow.
- 2: Frequent errors or choppy style hinder comprehension.
- 1: Very poor grammar/word salad; hard to understand.

Repetition (penalize meaningless identical-word/span loops; DO NOT penalize necessary reuse of terms/digits/operators)

- Definitions:
  - run = $\geq$ 3 identical tokens in a row (e.g., "eggs eggs eggs").
  - dense fragment recurrence = the same 2–5 word fragment appears many times within a short span.
- 5: No runs; only natural reuse; zero filler restatements.
- 4: One short run OR a few mild recurrences; readability intact with minimal filler.
- 3: Multiple short runs OR several recurrences; still readable but noticeably repetitive.
- 2: Frequent runs OR dense recurrences that harm readability.
- 1: Heavy looping/spam; large stretches of repeated tokens/fragments.

Structural Coherence (organization/flow; NOT correctness of content/math)

- Heuristics (apply flexibly, not mechanically):
  - Enumerated steps or clear transitions raise coherence.
  - Mostly raw calculations/symbols lower coherence.
  - Very short answers rarely justify 5 unless clearly structured.
- 5: Well-organized narrative/steps with clear transitions; tight paragraphing without stray blank lines.
- 4: Generally organized; minor jumps but understandable; spacing supports the flow.
- 3: Mixed; some organization but noticeable gaps/abrupt jumps or distracting spacing.
- 2: Mostly disorganized or just raw calculations; hard to follow.
- 1: No discernible structure; fragments/out-of-order snippets.

FINAL SELF-CHECK (internal; do NOT print):

- Apply CAPS after averaging the three views per dimension.
- Check for $\geq$ 3-in-a-row runs or dense recurrences before setting repetition.
- If >60% lines are bare calculations, enforce coherence $\leq$ 2.
- Ensure the three scores are mutually consistent with rubrics.

Few-shot Examples (STRICT single-line JSON only):
Example 1 (high quality, no repetition)
Student Answer:
Janet sells 16 - 3 - 4 = «16-3-4=9»9 duck eggs a day. She makes 9 * 2 = $«9*2=18»18 every day at the farmer's market. #### 18.
Expected Output:
<JSON>"fluency":"score":5,"repetition":"score":5,"structural_coherence":"score":5</JSON>
Example 2 (moderate repetition)
Student Answer:
Janet's ducks lay eggs eggs per day for which.6 eats «3 eggs +4 eggs = «3 eggs=16»16 eggs... friends friends friends... She sells sells sells the remainder... daily day... 16 - 7 = «16-7=9»9... She sells... fresh fresh fresh... makes makes makes...
Expected Output:
<JSON>"fluency":"score":3,"repetition":"score":2,"structural_coherence":"score":3</JSON>
Example 3 (extreme repetition)
Student Answer:
Janet eats eggs eggs eggs breakfast breakfast breakfast br...eakfast eats eats eats breakfast breakfast breakfast ... #### 18
Expected Output:
<JSON>"fluency":"score":1,"repetition":"score":1,"structural_coherence":"score":1</JSON>