

CD⁴LM: Consistency Distillation and aDaptive Decoding for Diffusion Language Models

Yihao Liang^{1,2} Ze Wang² Ximeng Sun² Jiang Liu² Jialian Wu²
Hao Chen² Xiaodong Yu² Emad Barsoum² Zicheng Liu² Niraj K. Jha¹

¹ Princeton University ² AMD
{yhliang, jha}@princeton.edu

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 **CD⁴LM**, 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*, while keeping the trained student compatible with external inference-side accelerators. On GSM8K, **CD⁴LM** preserves accuracy while achieving a $5.18\times$ speedup; stacking **CD⁴LM** with per-step cache accelerators further pushes speedup to $9.73\times$ 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 et al., 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., 2026; 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., 2025), 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 **CD⁴LM**, a composable framework for accurate DLM acceleration. Its training component, Discrete-Space Consistency Distillation (DSCD), trains a student across diverse masked states via teacher-subset masking without fixing a decoding trajectory. 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 distribution 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, **CD⁴LM** reaches 77.6% on GSM8K at a $5.18\times$ wall-clock speedup (Table 1). Stacking the same student on top of per-step cache accelerators further pushes the frontier: **CD⁴LM+dLLM-Cache** (Liu et al., 2025) reaches higher speed and **CD⁴LM+d²Cache** (Jiang et al., 2025) reaches higher accuracy, with both composed configurations Pareto-dominating the standalone accelerators reproduced in the same harness in Table 1. A single **CD⁴LM** 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 decoding-pattern-agnostic training and adaptive decoding framework.* We propose **CD⁴LM** = 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 an idealized conditional-projection view of DSCD, together with a posterior-risk analysis under a Bayes-teacher assumption.
3. *Strong accuracy-latency trade-offs and empirical composability.* **CD⁴LM** reaches 77.6% on GSM8K at $5.18\times$ speedup. Stacking with reproduced per-step cache accelerators further pushes speedup to $9.73\times$ (**CD⁴LM+dLLM-Cache**, 77.6%) and accuracy to **78.0%** (**CD⁴LM+d²Cache**, $8.96\times$), with both composed configurations Pareto-dominating the standalone baselines reproduced in the same harness 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.

2 Method

CD⁴LM 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, 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).

1. Input & Masking Process

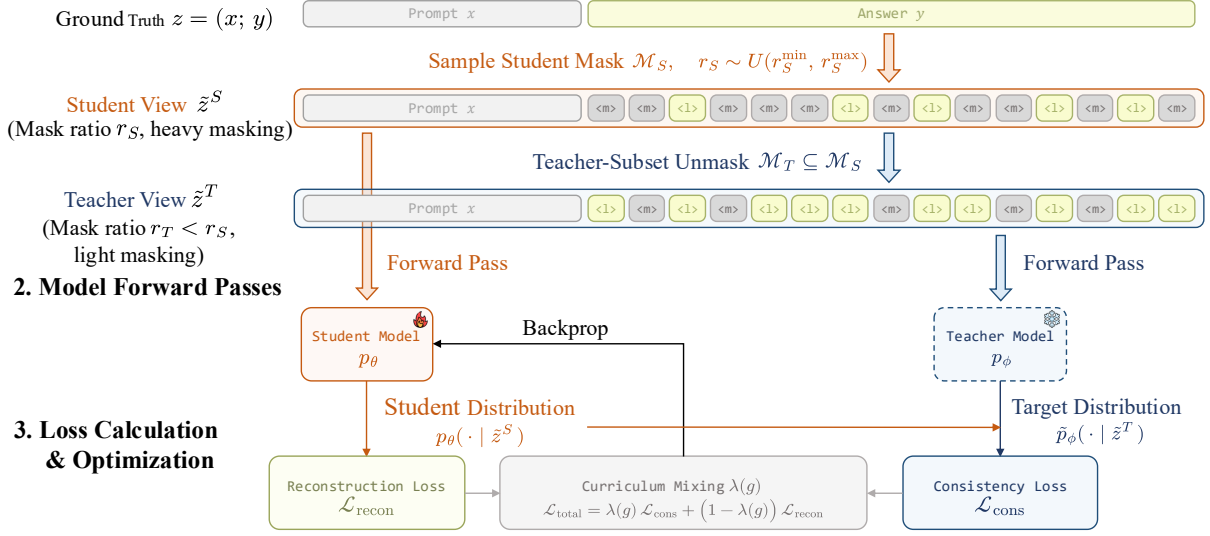


Figure 1: **Overview of the DSCD pipeline.** Teacher-subset masking ($\mathcal{M}_T \subseteq \mathcal{M}_S$) lets the teacher condition on a superset of the student’s context, which is typically strict before integer rounding. The student is trained jointly under $\mathcal{L}_{\text{recon}}$ (ground-truth 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.

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

$$p_\theta(z | \tilde{z}_t) \approx p_\theta(z | \tilde{z}_{t'}) \approx p_\phi(z | \tilde{z}_t, t) \quad (1)$$

for all $t, t' \in [0, 1]$.

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 $r_T = \text{clip}(r_S \cdot u, r_T^{\min}, r_T^{\max})$, where $u \sim \mathcal{U}(u_{\min}, u_{\max})$ with $u_{\max} < 1$. We convert each ratio to its diffusion timestep using the same noise schedule, yielding t_S and t_T . Let $n_S = \max(1, \lfloor L_y r_S \rfloor)$ and $n_T = \min(\lfloor L_y r_T \rfloor, n_S)$. We form the student masked set $\mathcal{M}_S$ by uniformly sampling (without replacement) n_S target positions from $\{L_x + 1, \dots, L\}$. 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 at least as rich as the student; the inclusion is typically strict but can become equality after integer rounding. 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 predictions under a nested 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 an information-nested distillation target (idealized posterior-risk analysis 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):

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

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

For $i \in \mathcal{M}_S \setminus \mathcal{M}_T$, the teacher observes z_i whereas the student does not; thus, the KL term combines soft distillation on $\mathcal{M}_T$ with information-enriched reconstruction targets on $\mathcal{M}_S \setminus \mathcal{M}_T$. This formulation encourages refinement across the sampled nested mask pairs (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., 2025) 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), aligning the student with teacher targets over sampled student information sets (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 $\mathcal{B}_j = \{(j-1)b + 1, \dots, \min(jb, L_{\text{gen}})\}$ 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 $\mathcal{M}^{(s)} = \{i \in [L_{\text{gen}}] : \tilde{z}_i^{(s)} = \mathfrak{m}\}$ be the masked set. CAD operates on the *eligible set* within the active block:

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

This reduces to $\mathcal{E}^{(s)} = \mathcal{M}^{(s)}$ when $b = L_{\text{gen}}$ (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 single forward pass (Step 2). We then compute a confidence score $c_i^{(s)}$ for each eligible position

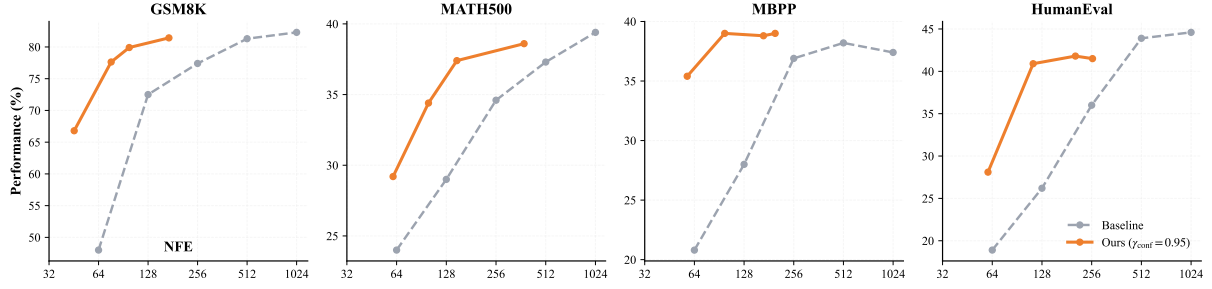


Figure 2: **Accuracy–compute trade-off under different decoding budgets.** We plot performance (%) versus the average NFE across benchmarks. The dashed gray curve denotes the LLaDA-8B-Instruct baseline evaluated under increasing NFE budgets, while the solid orange curve denotes our CAD with the given 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.

$i \in \mathcal{E}^{(s)}$ (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 $\hat{\mathcal{S}}^{(s)} = \{i \in \mathcal{E}^{(s)} : c_i^{(s)} \geq \gamma_{\text{conf}}\}$ 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

$$k^{(s)} = \min\{|\mathcal{E}^{(s)}|, k_{\text{max}}, \max(|\hat{\mathcal{S}}^{(s)}|, k_{\text{min}})\},$$

$$\mathcal{S}^{(s)} = \text{TopK}(\{c_i^{(s)}\}_{i \in \mathcal{E}^{(s)}}, k^{(s)}), \quad (5)$$

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 with the same DSCD objective and the task-specific hyperparameters listed in Table E.1.

3.2 Benchmarks

Code generation. We evaluate the model in a zero-shot setting on HumanEval (Chen et al., 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 $8 \times \text{AMD MI250}$ GPUs, AdamW at 5×10^{-6} , cosine decay, and 10% learning-rate warmup (300 code steps; 200 mathematics steps). Code and mathematics run for three and ten epochs with $\tau = 1.5$ and 2.0, respectively. Detailed hyperparameters and prompt templates are provided in Appendix E.3.

Method	Acc. (%) $\uparrow$	Avg. NFE $\downarrow$	Tokens/s (Speedup) $\uparrow$
<i>LLaDA-8B-Instruct baselines (fixed tokens-per-step)</i>			
LLaDA $\text{tps}=1$	77.4	256.0	7.5 (1.00 $\times$)
LLaDA $\text{tps}=2$	74.8	128.0	14.9 (1.99 $\times$)
LLaDA $\text{tps}=4$	43.6	64.0	29.7 (3.96 $\times$)
LLaDA $\text{tps}=8$	21.2	32.0	59.5 (7.93 $\times$)
<i>Training-based accelerators[†]</i>			
dParallel (Chen et al., 2025)	76.7	30.0	64.0 (8.53 $\times$)
CDLM (Kim et al., 2025)	73.9	57.7	33.3 (4.44 $\times$)
d ³ LLM (Qian et al., 2026)	74.2	36.8	52.2 (6.96 $\times$)
<i>Inference-based accelerators</i>			
Fast-dLLM (Wu et al., 2025)	75.8	84.1	32.4 (4.30 $\times$)
d ² Cache (Jiang et al., 2025)	77.4	256	29.4 (3.90 $\times$)
dLLM-Cache (Liu et al., 2025)	76.9	256	31.7 (4.25 $\times$)
<i>Ours (training + inference)</i>			
DSCD + CAD (Our CD⁴LM)	77.6	76.3	38.7 (5.18$\times$)
DSCD + Fast-dLLM	77.2	70.5	42.0 (5.60 $\times$)
Ours + d²Cache	78.0	76.5	67.0 (8.96 $\times$)
Ours + dLLM-Cache	77.6	76.6	72.7 (9.73$\times$)

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 tps annotation use adaptive or method-specific scheduling. Speedup is wall-clock relative to the LLaDA $\text{tps}=1$ 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), d²Cache (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 $\text{tps}=1$ to 74.8% at $\text{tps}=2$ and finally 21.2% at $\text{tps}=8$: 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 d³LLM 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 (d²Cache, 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. **CD⁴LM** 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, **CD⁴LM** stacks directly on cache-based per-step accelerators: **CD⁴LM**+d²Cache and **CD⁴LM**+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 the standalone entries reproduced in the same harness 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, **CD⁴LM** 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.

Uncertainty of accuracy and efficiency. Paired-bootstrap confidence intervals show that the headline GSM8K accuracy difference is +0.2 points with a 95% confidence interval (CI) of [−1.0, +1.3]. Across three training seeds, **CD⁴LM** obtains 77.5 ± 0.4 accuracy and 76.8 ± 1.9 NFE, while HumanEval pass@5 varies by ± 0.9 points across five decoding seeds. These results support accuracy preservation rather than a significant accuracy gain; paired NFE and latency differences are significant at $p < 0.001$.

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. Direct calibration measurements support this comparison: On held-out corrupted reference states, expected calibration error decreases from 0.064 for SFT to 0.042 for DSCD, and on-policy committed-token precision at $\gamma_{\text{conf}} = 0.95$ increases from 98.0% to 98.9%. Restricting the KL term to positions masked for both teacher and student preserves sim-

Configuration	Acc (%)	NFE	Speedup
<i>DSCD composes with inference accelerators</i>			
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×
<i>SFT-based training is constrained</i>			
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 (SFT) while holding the decoder fixed. Under both compared decoders, DSCD achieves a higher speedup than SFT at similar accuracy (CAD: 5.18× vs. 4.15×; Fast-dLLM: 5.60× vs. 4.73×).

ilar accuracy (77.3% vs. 77.6%) but increases NFE from 76.3 to 84.2 and reduces committed-token precision to 98.4%. Under the diagnostic global-denoising setting ($b=L_{\text{gen}}$) without permanent EOS suppression, SFT+CAD reaches 38.2% at 3.13×, whereas DSCD+CAD reaches 54.7% at 4.91× (Appendix F).

Component controls support the core design. Sampling teacher and student masks independently causes frequent loss divergence with gradient NaNs within the first epoch, whereas teacher-subset masking avoids this failure mode (Appendix F.3). Among convergent variants, removing the reconstruction loss retains 76.8% accuracy but raises NFE from 76.3 to 98.7; replacing confidence-ranked commitment with uniform random selection is more damaging, reducing accuracy from 77.6% to 50.6% and raising NFE to 202.2 (Appendix F). Together, these controls isolate the two parts of our story: Nested-mask supervision and ground-truth anchoring make the student trainable and efficient across irregular states, while confidence ranking converts that signal into safe parallel commitments.

EOS handling and fair pure-diffusion comparison. CAD suppresses positions whose top-ranked prediction is EOS until at least $\beta_{\text{EOS}} L_{\text{gen}}$ target positions have been committed; we use $\beta_{\text{EOS}} = 0.3$. If all eligible positions are blocked, the k_{min} fallback still commits tokens to guarantee progress. This dynamic guardrail is released later in decoding, whereas the permanent EOS-suppression protocol sets $p(<\text{EOS}>) = 0$ throughout generation.

Table 4: **GSM8K latency decomposition** (batch size 1; identical prompts, hardware, full canvas, and harness). Similar output lengths rule out EOS length as the main source of the gap between NFE reduction and end-to-end speedup.

Metric	LLaDA fixed	DSCD+CAD
Avg. NFE	256.0	76.3
Avg. output length to EOS	141	138
Processed target canvas	256	256
Model forward / step	128 ms	83 ms
Total latency / step	133 ms	87 ms
CAD controller / step	—	0.6 ms (< 1%)
End-to-end / request	34.1 s	6.6 s

For a fair pure-diffusion comparison, both LLaDA and the DSCD student use permanent EOS suppression: LLaDA reaches 73.2% and the same DSCD checkpoint reaches 77.4% with CAD without retraining from the $b = 32$ setting. Without permanent EOS suppression, the diagnostic LLaDA and DSCD+CAD results are 13.8% and 54.7%, respectively (Appendix F). All settings retain the same $L_{\text{gen}} = 256$ canvas; hence, shorter outputs do not explain the reported compute reduction.

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, **CD⁴LM** 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

Table 5: **One-factor CAD sweeps on GSM8K.** Lower confidence thresholds reach more aggressive budgets but expose an accuracy–NFE boundary; $k_{\max}$ is comparatively stable in accuracy.

Knob	Value	Acc. (%)	NFE
γ_{conf}	0.95	77.6	76.3
	0.80	61.3	31.2
	0.60	42.1	15.8
$k_{\max}$	8	77.7	117
	16	77.6	88
	32	77.6	76

Table 6: **Calibration and KL-support analysis on GSM8K.** Precision is measured on committed tokens at $\gamma_{\text{conf}} = 0.95$.

Target	ECE ↓	Prec. ↑	Acc.	NFE ↓
SFT + CAD	0.064	98.0	77.6	95.4
DSCD ($\mathcal{M}_T$)	–	98.4	77.3	84.2
DSCD ($\mathcal{M}_S$)	0.042	98.9	77.6	76.3

account for the measured latency gain; the observed throughput also reflects per-step execution cost. Table 4 shows that the fixed loop averages 133 ms per step and the CAD loop averages 87 ms per step, while both process the same full canvas and produce similar output lengths. The remaining factor is therefore a decode-loop implementation effect rather than a training-side speedup; CAD controller overhead is below 1% of per-step latency.

One student spans mechanically different decoders. All composed entries reuse the same DSCD checkpoint: CAD and Fast-dLLM change the token-commit policy, whereas d²Cache and dLLM-Cache reduce per-step compute; none requires retraining. The checkpoint also switches from block diffusion at $b = 32$ (77.6%, 5.18 $\times$) to pure diffusion at $b = L_{\text{gen}}$ under permanent EOS suppression (77.4%, 4.91 $\times$). Thus, the evidence for decoding-pattern-agnostic training is not tied to one favorable decoder: accuracy remains stable across multiple inference paths that induce different intermediate states and compute profiles.

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 $\times$ wall-clock speedup (Appendix G.3), providing evidence of transfer to a second absorbing-state DLM backbone.

Decoding sensitivity. Accuracy remains within one point across $b \in \{16, 32, 64\}$ (77.2%, 77.6%, and 76.8%). In contrast, increasing $k_{\min}$ from 1 to 2 reduces NFE by about 12% but costs about three accuracy points; larger values move toward the fixed-tps failure regime. We, therefore, use $k_{\min} = 1$. Lowering γ_{conf} enables 31.2 and 15.8 NFE at 61.3% and 42.1% accuracy, respectively, showing an empirical, task-dependent boundary rather than unlimited NFE compression.

5 Conclusion

We presented **CD⁴LM**, a training-side DLM accelerator that commits to no decoding pattern: DSCD trains a student over random teacher-subset mask pairs without fixing a decoding path, and CAD uses this flexibility for dynamic confidence-adaptive decoding. The result reaches teacher-level accuracy on GSM8K (77.6% at 5.18 $\times$ speedup) and transfers to DiffuCoder-Instruct without modification. A paired bootstrap confirms that the +0.2-point GSM8K accuracy difference is within statistical uncertainty (95% CI $[-1.0, +1.3]$), whereas the NFE and latency reductions are significant at $p < 0.001$. Direct calibration measurements further show that DSCD lowers expected calibration error (ECE) from 0.064 to 0.042 and raises committed-token precision from 98.0% to 98.9%, supporting more reliable confidence-guided commitment. Because DSCD’s supervision is decoupled from any specific inference policy, stacking it with per-step cache accelerators is orthogonal: **CD⁴LM+dLLM-Cache** and **CD⁴LM+d²Cache** reach the table’s highest speedup (**9.73 $\times$** at 77.6%) and highest accuracy (**78.0%** at 8.96 $\times$), respectively, Pareto-dominating the standalone accelerators reproduced in the same harness.

Outlook. Decoupling training from any specific decoding pattern is a productive design discipline: Stacking DSCD with future inference-side accelerators and pairing decoding-pattern-agnostic supervision with on-policy refinement are natural next steps, with **CD⁴LM** providing a stable training-side layer beneath the evolving inference-side ecosystem.

Limitations and Discussion

We close by acknowledging where **CD⁴LM** 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.

Decoding-pattern-agnostic supervision trades some standalone speed for composability. Among the training-side baselines reported in Table 1, dParallel (Chen et al., 2025) and d³LLM (Qian et al., 2026) have higher reported/extrapolated standalone speedups than **CD⁴LM** ($8.53\times$ and $6.96\times$ vs. $5.18\times$), but lower accuracy (76.7% and 74.2% vs. 77.6%). **CD⁴LM** reaches higher speedups only once composed with reproduced cache accelerators. Whether these composability gains generalize beyond the three evaluated inference-side accelerators (Fast-dLLM, d²Cache, 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, et al. 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*.
- 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, et al. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. 2025. 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, et al. 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, et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- 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*.
- 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. d²Cache: 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*.
- 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*.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, 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.
- 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.

- 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, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2026. Accelerating diffusion large language models with slowfast sampling: The three golden principles. In *International Conference on Learning Representations*, volume 2026, pages 4340–4354.
- 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, 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*.
- Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Chengzhi Mao, et al. 2026. Few-step diffusion language models via trajectory self-distillation. *arXiv preprint arXiv:2602.12262*.

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 $x = (x_1, \dots, x_{L_x})$ is the prompt (e.g., a coding problem or mathematics question) and $y = (y_1, \dots, y_{L_y})$ is the target sequence (e.g., the solution program or derivation). We concatenate them into a single sequence $z = (x; y) = (z_1, \dots, z_L)$, with $L = L_x + L_y$. The base vocabulary $\mathcal{V}$ is augmented with a special absorbing mask token $\mathfrak{m} \notin \mathcal{V}$, and an end-of-sequence token $\langle \text{EOS} \rangle$, giving the extended vocabulary: $\mathcal{V}^+ = \mathcal{V} \cup \{\mathfrak{m}, \langle \text{EOS} \rangle\}$. 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 $q_t(\tilde{z} | z)$ gradually destroys information in the target region while preserving the prompt:

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

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

where δ is the Kronecker delta. Once a token is replaced by $\mathfrak{m}$, it remains masked for all larger t ; hence, the forward chain is an absorbing Markov process. We denote the masked set by $\mathcal{M}(\tilde{z}) = \{i : \tilde{z}_i = \mathfrak{m}\}$. 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 $\tilde{z} = \text{Mask}(z, b)$, which matches $q_t(\tilde{z} | z)$ 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:

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

where each $p_\phi(z_i | \tilde{z}, t)$ 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:

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

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 $t_1 > \dots > t_T$, 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 Conditional Projection View of DSCD

This section gives an idealized conditional-projection view of DSCD. For a fixed student state and an unrestricted student family, minimizing $\mathcal{L}_{\text{cons}}$ yields the conditional average of teacher targets over compatible nested teacher views. This result motivates the pairwise surrogate but does not guarantee trajectory invariance across all timesteps or mask distributions.

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 $\mathbf{m}$.

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:

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

We write

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

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

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

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

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

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

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

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

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

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

$$\mathcal{L}_{\text{DSCD}}(\theta) = \mathbb{E}_{S,T}[D_{\text{KL}}(p_\phi(\cdot | T) \| p_\theta(\cdot | S))] \quad (12)$$

$$= \mathbb{E}_{S,T} \left[\sum_{z \in \mathcal{Z}} p_\phi(z | T) \log \frac{p_\phi(z | T)}{p_\theta(z | S)} \right] \quad (13)$$

$$= \underbrace{\mathbb{E}_{S,T} \left[\sum_{z \in \mathcal{Z}} p_\phi(z | T) \log p_\phi(z | T) \right]}_{\text{constant w.r.t. } \theta} - \mathbb{E}_{S,T} \left[\sum_{z \in \mathcal{Z}} p_\phi(z | T) \log p_\theta(z | S) \right]. \quad (14)$$

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

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

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

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

which is maximized when $p_\theta(\cdot | s) = Q(\cdot | s)$. Therefore, $p_\theta^*(\cdot | S) = Q(\cdot | S)$. $\square$

This proposition shows that, for each fixed student state under the stated model-class assumption, the optimum need not reproduce a particular teacher trajectory. Instead, it matches the teacher distribution averaged over compatible richer views, i.e., the projection onto the coarser information σ -algebra $\mathcal{F}_S$.

C An Idealized Posterior-Risk View of Teacher-Subset Masking

We analyze the *teacher-subset masking* scheme ($\mathcal{M}_T \subseteq \mathcal{M}_S$) as an information-nesting design. 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 superset of the student’s information, typically a strict superset before integer rounding. Under an idealized Bayes predictor, the corresponding conditional-expectation result states that posterior residual risk cannot increase as the information set is refined. Independent masking has no such nesting property, 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 $Y = p_\phi(\cdot | \tilde{z}^T)$, 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 can be a non-empty set of “blind” positions $\mathcal{B} = \mathcal{M}_T \cap (\{L_x + 1, \dots, L\} \setminus \mathcal{M}_S)$ 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*

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

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

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

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

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

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

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

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

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

Under this idealization, teacher-subset masking is analogous to Rao-Blackwellization 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 $\mathcal{M}^{(s)} = \{i \in [L_{\text{gen}}] : \tilde{z}_i^{(s)} = \mathbf{m}\}$. For block diffusion with block size b , define blocks

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

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

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

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:

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

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

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

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:

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

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* ($|\mathcal{S}| \leq k_{\text{max}}$) to prevent committing too many tokens based on local confidence estimates, and *liveness* ($|\mathcal{S}| \geq k_{\text{min}}$) to guarantee progress whenever possible. This yields the constrained problem

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

Sorting candidates by decreasing confidence produces an ordering $i_{(1)}, \dots, i_{(|\mathcal{V}^{(s)}|)}$ with $c_{i_{(1)}}^{(s)} \geq \dots \geq c_{i_{(|\mathcal{V}^{(s)}|)}}^{(s)}$. 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} ,

$$n_{\text{conf}} = \left| \left\{ j \leq \min(k_{\text{max}}, |\mathcal{V}^{(s)}|) : c_{i_{(j)}}^{(s)} \geq \gamma_{\text{conf}} \right\} \right|, \quad (22)$$

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

$$k^{(s)} = \min \left(\max(k_{\text{min}}, n_{\text{conf}}), \min(k_{\text{max}}, |\mathcal{V}^{(s)}|) \right), \quad \mathcal{S}^{(s)} = \{i_{(1)}, \dots, i_{(k^{(s)})}\}. \quad (23)$$

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., $\langle \text{mask} \rangle$ 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:

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

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 decoded-length ratio β_{EOS} . While $n_{\text{decoded}}/L_{\text{gen}} < \beta_{\text{EOS}}$, eligible positions whose top-ranked prediction is EOS are excluded from confidence-adaptive selection whenever at least one non-EOS position remains. If all eligible positions are blocked, the $k_{\min}$ fallback still commits positions to guarantee progress; after the decoded-length threshold is reached, EOS predictions are again eligible. Because every update still commits at least one position when masks remain, this guardrail 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 $\langle \text{mask} \rangle$ 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. $\square$

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*

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

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 $b = L_{\text{gen}}$ (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 $\gamma_{\text{conf}} = -\infty$ and $k_{\min} = k_{\text{max}} = k$ (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

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

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

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

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

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

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

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

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** (Step 3): 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**: The teacher ratio is derived from $r_S \cdot u$ with $u \sim \mathcal{U}(0.3, 0.7)$ and then clipped to its configured range. The resulting mask sets satisfy $|\mathcal{M}_T| \leq |\mathcal{M}_S|$ after integer rounding.
3. **Minimum mask count**: The student view masks at least one valid target position, preventing a zero-denominator reconstruction or consistency loss on short targets.

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

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

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., $|\mathcal{M}_T| \leq |\mathcal{M}_S|$).

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

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

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.

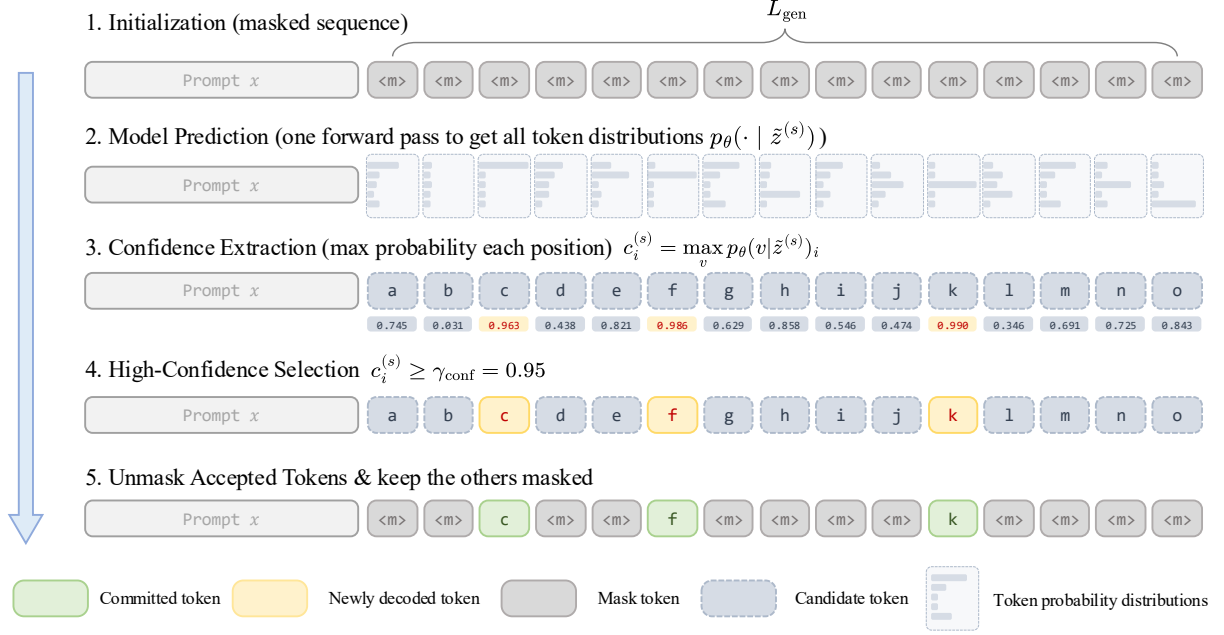


Figure E.1: **Illustration of CAD.** In each step, the model predicts token distributions for all masked positions in parallel. We compute the confidence score $c_i^{(s)}$ based on the maximum probability and selectively unmask tokens that satisfy the threshold $c_i^{(s)} \geq \gamma_{\text{conf}}$.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

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:

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

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 $\mathcal{O}(S \cdot N_{\text{layer}}(L^2 D + L D^2))$, as detailed in Appendix D.

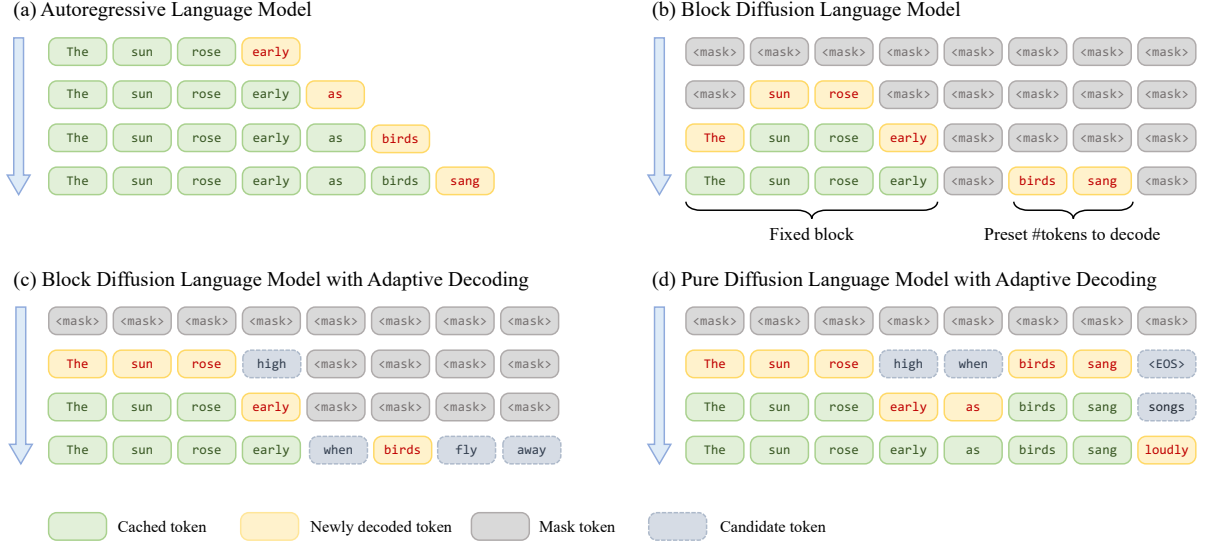


Figure E.2: **Overview of different decoding paradigms.** (a) Standard Autoregressive decoding. (b) Fixed-schedule Block Diffusion (adopted by the **LLaDA backbone** (Nie et al., 2025)), which decodes a preset number of masked tokens per step. (c) Block diffusion with our **Confidence-Adaptive Decoding (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.

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

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

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 $G_{\max}$

```
1: for  $g = 1$  to  $G_{\max}$  do
2:   Sample mini-batch  $\{(x^{(b)}, y^{(b)})\}_{b=1}^B \sim \mathcal{D}$ 
3:   for  $b = 1$  to  $B$  do
4:      $z \leftarrow (x^{(b)}; y^{(b)})$  ▷ Concatenate prompt and target
5:      $L_x \leftarrow |x^{(b)}|$ ,  $L_y \leftarrow |y^{(b)}|$ ,  $L \leftarrow L_x + L_y$ 
6:     // — Step 1: Sample mask ratios (see Eq. 29) —
7:      $r_S \sim \text{Uniform}(r_S^{\min}, r_S^{\max})$ 
8:      $u \sim \text{Uniform}(0.3, 0.7)$ 
9:      $r_T \leftarrow \text{clip}(r_S \cdot u, r_T^{\min}, r_T^{\max})$ 
10:    // — Step 2: Construct student mask —
11:     $\mathcal{U}_{\text{valid}} \leftarrow \{L_x + 1, \dots, L\}$ 
12:     $n_S \leftarrow \min(|\mathcal{U}_{\text{valid}}|, \max(1, \lfloor |\mathcal{U}_{\text{valid}}| \cdot r_S \rfloor))$ 
13:     $\mathcal{M}_S \leftarrow \text{RandomSample}(\mathcal{U}_{\text{valid}}, n_S)$  ▷ Uniformly sample positions
14:     $\tilde{z}^S \leftarrow \text{Mask}(z, \mathcal{M}_S)$  ▷ Replace  $z_i$  with m for  $i \in \mathcal{M}_S$ 
15:    // — Step 3: Teacher-subset masking —
16:     $n_T \leftarrow \min(\lfloor |\mathcal{U}_{\text{valid}}| \cdot r_T \rfloor, n_S)$ 
17:     $\mathcal{M}_T \leftarrow \text{RandomSubset}(\mathcal{M}_S, n_T)$  ▷ Nested subset:  $\mathcal{M}_T \subseteq \mathcal{M}_S$ 
18:     $\tilde{z}^T \leftarrow \text{Mask}(z, \mathcal{M}_T)$ 
19:    // — Step 4: Forward pass —
20:     $\ell_\theta \leftarrow p_\theta(\cdot \mid \tilde{z}^S)$  ▷ Student logits
21:     $\ell_\phi \leftarrow p_\phi(\cdot \mid \tilde{z}^T)$  ▷ Teacher logits (no grad)
22:    // — Step 5: Compute losses on  $\mathcal{M}_S$  —
23:     $\ell_{\text{recon}}^{(b)} \leftarrow -\frac{1}{|\mathcal{M}_S|} \sum_{i \in \mathcal{M}_S} \log p_\theta(z_i \mid \tilde{z}^S)$ 
24:     $\tilde{p}_\phi \leftarrow \text{softmax}(\ell_\phi / \tau)$  ▷ Temperature-scaled teacher
25:     $\ell_{\text{cons}}^{(b)} \leftarrow \frac{\tau^2}{|\mathcal{M}_S|} \sum_{i \in \mathcal{M}_S} \text{KL}(\tilde{p}_{\phi, i} \parallel p_{\theta, i})$ 
26:  end for
27:  // — Step 6: Curriculum-weighted total loss —
28:   $\mathcal{L}_{\text{recon}} \leftarrow \frac{1}{B} \sum_{b=1}^B \ell_{\text{recon}}^{(b)}$ ,  $\mathcal{L}_{\text{cons}} \leftarrow \frac{1}{B} \sum_{b=1}^B \ell_{\text{cons}}^{(b)}$ 
29:   $\lambda(g) \leftarrow \text{CosineSchedule}(g/G_{\max}; \lambda_0, \lambda_1, \alpha)$  ▷ Eq. (30)
30:   $\mathcal{L}_{\text{total}} \leftarrow \lambda(g) \cdot \mathcal{L}_{\text{cons}} + (1 - \lambda(g)) \cdot \mathcal{L}_{\text{recon}}$ 
31:  // — Step 7: Parameter update —
32:   $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{total}}$ 
33: end for
34: return Trained student  $p_\theta$ 
```

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 $S_{\max}$

```
1:  $\tilde{z}^{(0)} \leftarrow (x; \underbrace{\mathbf{m}, \dots, \mathbf{m}}_{L_{\text{gen}}})$  ▷ Initialize with all masks

2:  $\mathcal{M}^{(0)} \leftarrow \{1, \dots, L_{\text{gen}}\}$  ▷ All target positions masked
3:  $J \leftarrow \lceil L_{\text{gen}}/b \rceil, j \leftarrow 1$  ▷ Number of blocks, active block index
4:  $s \leftarrow 0, n_{\text{decoded}} \leftarrow 0$  ▷ Step counter, decoded token count
5: while  $\mathcal{M}^{(s)} \neq \emptyset$  and  $s < S_{\max}$  do
    // — Step 1: Compute eligible set within active block —
    6:  $\mathcal{B}_j \leftarrow \{(j-1)b + 1, \dots, \min(jb, L_{\text{gen}})\}$ 
    7:  $\mathcal{E}^{(s)} \leftarrow \mathcal{M}^{(s)} \cap \mathcal{B}_j$  ▷ Eligible = masked  $\cap$  active block
    8: if  $\mathcal{E}^{(s)} = \emptyset$  then
    9:      $j \leftarrow j + 1$  ▷ Advance to next block
    10:    continue
    11:  end if
    // — Step 2: Model forward and confidence computation —
    12:  $\ell \leftarrow p_\theta(\cdot \mid \tilde{z}^{(s)})$  ▷ Logits for all positions
    13:  $p \leftarrow \text{softmax}(\ell)$  ▷ Token probabilities
    14:  $\hat{z}_i \leftarrow \arg \max_v \ell_{i,v}$  for  $i \in \mathcal{E}^{(s)}$  ▷ Predicted tokens
    15:  $c_i^{(s)} \leftarrow p_{i, \hat{z}_i}$  for  $i \in \mathcal{E}^{(s)}$  ▷ Confidence = prob of argmax token
    // — Step 3: EOS blocking —
    16: if  $n_{\text{decoded}}/L_{\text{gen}} < \beta_{\text{EOS}}$  then
    17:   for  $i \in \mathcal{E}^{(s)}$  where  $\hat{z}_i \in \{\langle \text{EOS} \rangle, \langle \text{im\_end} \rangle, \dots\}$  do
    18:     $c_i^{(s)} \leftarrow -\infty$  ▷ Suppress premature EOS
    19:  end for
    20: end if
    // — Step 4: Confidence-adaptive selection —
    21:  $\mathcal{V}^{(s)} \leftarrow \{i \in \mathcal{E}^{(s)} : c_i^{(s)} > -\infty\}$  ▷ Valid (non-blocked) positions
    22: if  $\mathcal{V}^{(s)} = \emptyset$  then
    23:    $k^{(s)} \leftarrow \min(k_{\min}, |\mathcal{E}^{(s)}|)$  ▷ Force progress when all blocked
    24:    $\mathcal{S}^{(s)} \leftarrow \text{TopK}(\{c_i^{(s)}\}_{i \in \mathcal{E}^{(s)}}, k^{(s)})$ 
    25: else
    26:    $n_{\text{conf}} \leftarrow |\{i \in \mathcal{V}^{(s)} : c_i^{(s)} \geq \gamma_{\text{conf}}\}|$  ▷ Count above threshold
    27:    $k^{(s)} \leftarrow \min(\max(n_{\text{conf}}, k_{\min}), \min(k_{\max}, |\mathcal{V}^{(s)}|))$ 
    28:    $\mathcal{S}^{(s)} \leftarrow \text{TopK}(\{c_i^{(s)}\}_{i \in \mathcal{V}^{(s)}}, k^{(s)})$  ▷ Select top- $k$  positions
    29: end if
    // — Step 5: Commit predictions —
    30: for  $i \in \mathcal{S}^{(s)}$  do
    31:    $\tilde{z}_i^{(s+1)} \leftarrow \hat{z}_i$ 
    32: end for
    33:  $\mathcal{M}^{(s+1)} \leftarrow \mathcal{M}^{(s)} \setminus \mathcal{S}^{(s)}$ 
    34:  $n_{\text{decoded}} \leftarrow n_{\text{decoded}} + |\mathcal{S}^{(s)}|$ 
    35:  $s \leftarrow s + 1$ 
    36: end while
    37: 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 (10%)	–	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	$r_S \in [\cdot]$	[0.4, 0.9]	[0.4, 0.9]
Teacher mask ratio	$r_T \in [\cdot]$	[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 $\langle \text{EOS} \rangle$ 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 $3.13\times$ 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 $4.91\times$ without permanent EOS suppression. 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) applies the same permanent EOS-suppression protocol to both methods, under which LLaDA has 73.2% accuracy and **CD⁴LM** 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 $\langle \text{EOS} \rangle$* , 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 $\langle \text{EOS} \rangle$ 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 $\langle \text{EOS} \rangle$ 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% $\rightarrow$ 50.6%) and increases NFE (76.3 $\rightarrow$ 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, these runs indicate that the nested-subset constraint is not merely a theoretical preference for information nesting, but an important practical condition for stable discrete consistency distillation in our setting.

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 ($b=L_{\text{gen}}$) 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 permanent EOS suppression, it recovers to 73.2%. Second, the same distilled checkpoint supports block diffusion ($b=32$, $5.18\times/77.6\%$) and pure diffusion ($b=L_{\text{gen}}$, $4.91\times/77.4\%$) without retraining; the 77.4% pure-diffusion result uses the same permanent $p(\langle \text{EOS} \rangle) = 0$ protocol as the 73.2% LLaDA baseline. Without permanent suppression, the corresponding diagnostic DSCD+CAD result is 54.7% (Table F.1). By comparison, block-causal designs (Kim et al.’s CDLM at $b=L_{\text{gen}}$ 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 ≈ 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 $\approx 13.1\text{k}$ 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).

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 fair comparison applying permanent EOS suppression to both LLaDA and **CD⁴LM** 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.

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 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 max-

imum 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.

Regime	Method	Acc. (%) $\uparrow$	Avg. NFE $\downarrow$	Tokens/s (Speedup) $\uparrow$
Sequential ($b=1$)	LLaDA $\text{tps}=1$	76.4	256	7.5 (1.00 $\times$)
Pure Diffusion ($b=L_{\text{gen}}=256$)	LLaDA [†] (EOS-suppr.) $\text{tps}=1$	73.2	256	7.5 (1.00 $\times$)
	LLaDA (diagnostic) $\text{tps}=1$	13.8	256	7.5 (1.00 $\times$)
	LLaDA (diagnostic) $\text{tps}=2$	13.0	128	14.9 (1.99 $\times$)
	CD ⁴ LM [†] (EOS-suppr.)	77.4	80.6	36.7 (4.91$\times$)

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 tps annotation use adaptive or method-specific scheduling. [†] marks pure-diffusion results using permanent $p(\langle \text{EOS} \rangle) = 0$ suppression; without it, LLaDA and DSCD+CAD obtain 13.8% and 54.7% accuracy, respectively, as reported in Table F.1. Speedup is wall-clock relative to the sequential baseline.

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

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 tps 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% $\rightarrow$ 38.7%, +4.9; pass@5: 46.0% $\rightarrow$ 50.5%, +4.5) and speedup (3.07 $\times$ $\rightarrow$ 3.29 $\times$), confirming that DSCD’s composability extends beyond GSM8K.

Method	MBPP pass@1 (%) $\uparrow$	Speedup $\uparrow$
DiffuCoder-Instruct	73.1	1.00 $\times$
DiffuCoder-Instruct + CD⁴LM	73.7	3.25$\times$

Table G.3: **Transfer to DiffuCoder-Instruct on MBPP**. The same CD⁴LM pipeline is applied without architecture-specific modification; speedup is measured relative to the DiffuCoder-Instruct baseline.

Our method successfully reclaims this redundancy, terminating well before the fixed limit even for the hardest “tail” samples.

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 eval-

uation (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 to-

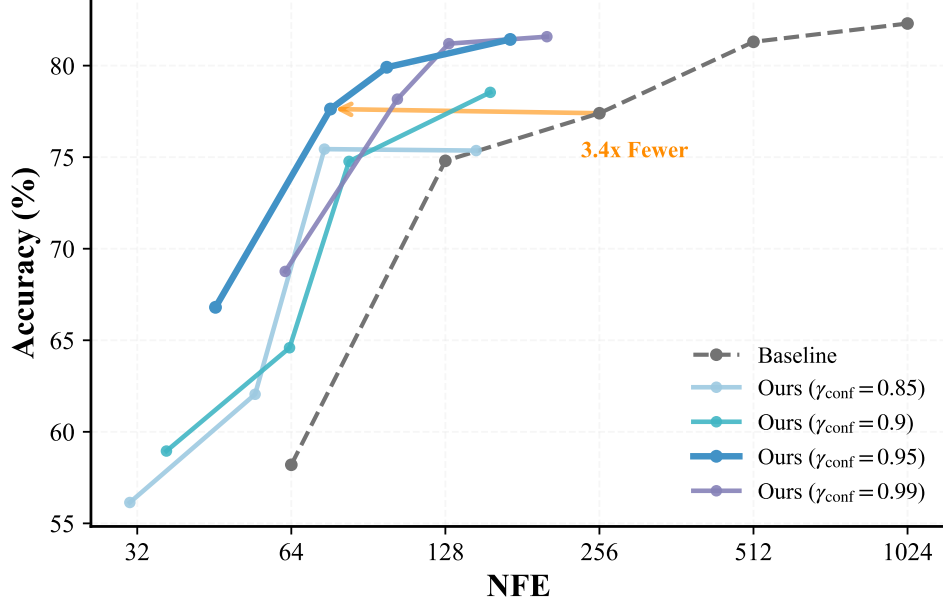


Figure G.1: **Accuracy-compute Pareto frontier on GSM8K.** The dashed gray curve represents the LLaDA baseline. **Strict Dominance:** Our method (solid curves) consistently pushes the frontier upwards and leftwards. The orange arrow highlights our selected operating point ($\gamma_{\text{conf}} = 0.95$, dark blue curve), which achieves a **3.4 $\times$ 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.

ken repetition (e.g., “fresh duck fresh egg”) and arithmetic hallucinations, resulting in significantly lower judge scores.

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 = $\ll 3 \times 1 = 3 \gg 3$ eggs for breakfast. She bakes 4 eggs per day = $\ll 4 \times 1 = 4 \gg 4$ eggs for her friends. She sells $16 - 3 - 4 = \ll 16 - 3 - 4 = 9 \gg 9$ eggs at the farmers' market each day. She sells 9 eggs at the farmers' market for \$2 per egg. Thus, she makes $9 \times \$2 = \$\ll 9 \times 2 = 18 \gg 18$ every day at the farmers' market. ##### 18

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

Example 2: Low-Quality Generation (LLaDA)

Generated Answer:

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

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

G.8 Unmasking Analysis

To investigate the generation mechanism of CD⁴LM, we visualize the step-wise unmasking

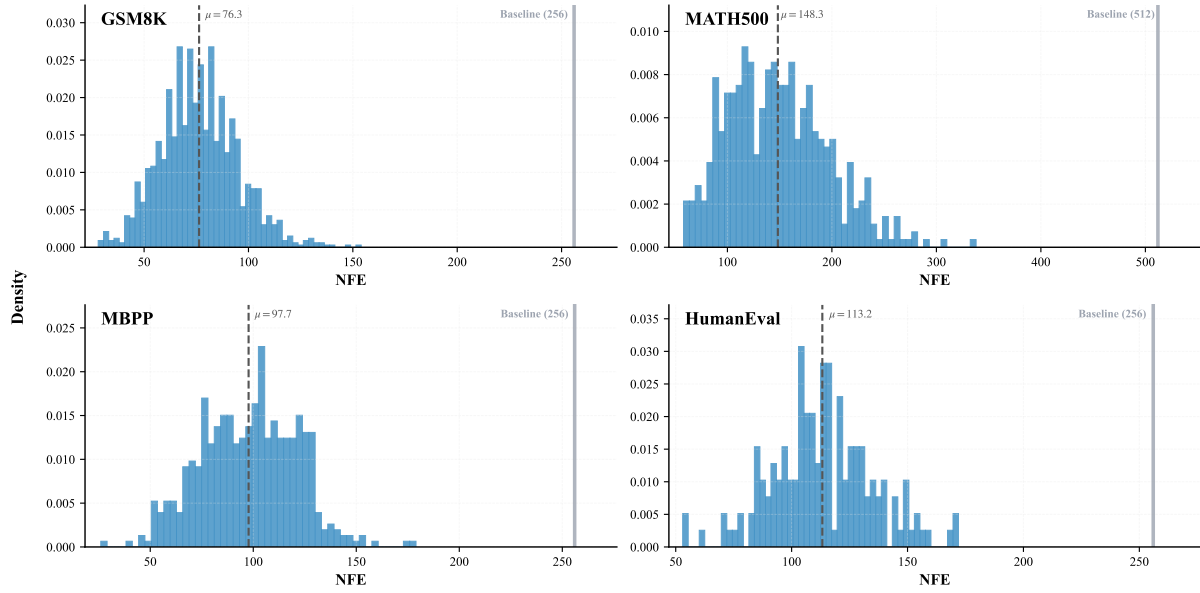


Figure G.2: **Per-sample compute allocation under adaptive decoding.** Histograms show the NFE distribution across test instances for GSM8K, MATH500, MBPP, and HumanEval using our adaptive decoding approach. The dashed vertical line marks the mean NFE (μ), while the solid vertical line indicates the 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.

trajectory for a representative HumanEval code generation example 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 condition-

ing on the *already-visible* syntactic context from Phase I.

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.

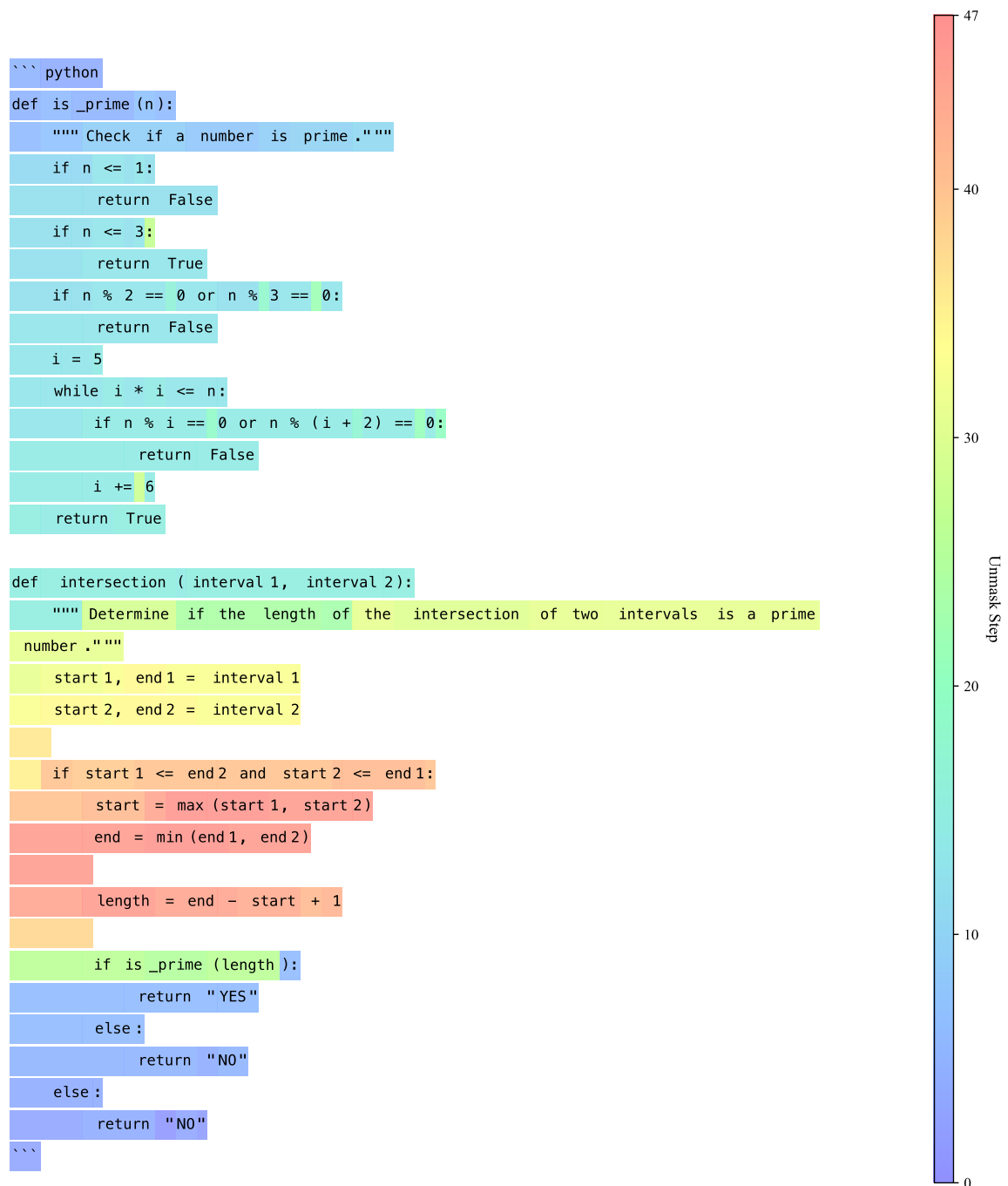


Figure G.3: **Visualization of hierarchical generation dynamics.** Unlike autoregressive models that generate sequentially, **CD⁴LM** exhibits a “skeleton-first” strategy: Syntactic structures (blue, early steps) are established first to form a global plan, while complex logical details (red, late steps) are refined later within the established context.

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 → 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 ≥ 1 run of ≥ 3 identical tokens exists more than once → repetition ≤ 3 .
- If frequent runs OR dense fragment recurrences harm readability → repetition ≤ 2 .

Structural Coherence:

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

Fluency:

- If pervasive grammatical errors/choppy telegraphic style hinder comprehension → fluency ≤ 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 ≥ 3 blank lines or double-newline spacing between most sentences, treat the flow as fragmented → structural_coherence ≤ 3 , and fluency ≤ 3 unless the prose still reads seamlessly.
- Heavy formatting noise (raw LaTeX delimiters such as $\backslash$ [, $\$$ %, $\backslash$ begin{align}, Markdown tables/lists, or numbered steps that merely restate facts) disrupts readability → 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 = ≥ 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 ≥ 3 -in-a-row runs or dense recurrences before setting repetition.
- If $>60\%$ lines are bare calculations, enforce coherence ≤ 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 $16 - 3 - 4 = \llbracket 16-3-4=9 \rrbracket 9$ duck eggs a day. She makes $9 * 2 = \llbracket 9*2=18 \rrbracket 18$ 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 $\llbracket 3 \rrbracket 3$ eggs +4 eggs = $\llbracket 3 \rrbracket 3$ eggs=16 $\rrbracket 16$ eggs... friends friends friends... She sells sells sells the remainder... daily day... $16 - 7 = \llbracket 16-7=9 \rrbracket 9$... 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>
```