ALoDLM: Adaptively Looped Diffusion Language Models

Anonymous authors
Paper under double-blind review

Abstract

Diffusion language models (DLMs) enable fast generation by predicting multiple tokens in parallel, yet their practical adoption remains hindered by a persistent quality gap relative to comparably sized autoregressive models. We attribute this gap to a *computation–difficulty mismatch*: within a partially observed sequence, some unknown tokens are readily predictable, while others require substantially more computation to resolve. Existing DLMs, however, apply uniform computational depth to every unknown position at each denoising step. We introduce **ALoDLM**, which replaces this uniform computation with token-adaptive latent recurrence. At each denoising step, ALoDLM iteratively refines representations in latent space, allocating computation based on token difficulty. Tokens ready to commit are fed back as discrete context, while unresolved tokens retain and refine their latent states through additional recurrent passes. To learn both token prediction and computation allocation end-to-end, we formulate token-wise computation schedules as latent variables and derive a conditional negative evidence lower bound (NELBO). We train ALoDLM at 1.7B and 8B parameter scales. Across eleven benchmarks, ALoDLM outperforms all evaluated DLMs and the corresponding autoregressive baselines in average benchmark score at both scales. Importantly, ALoDLM combines superior generation quality with fast parallel decoding, establishing a strong quality–efficiency trade-off among all evaluated autoregressive and diffusion models under optimized inference engines.

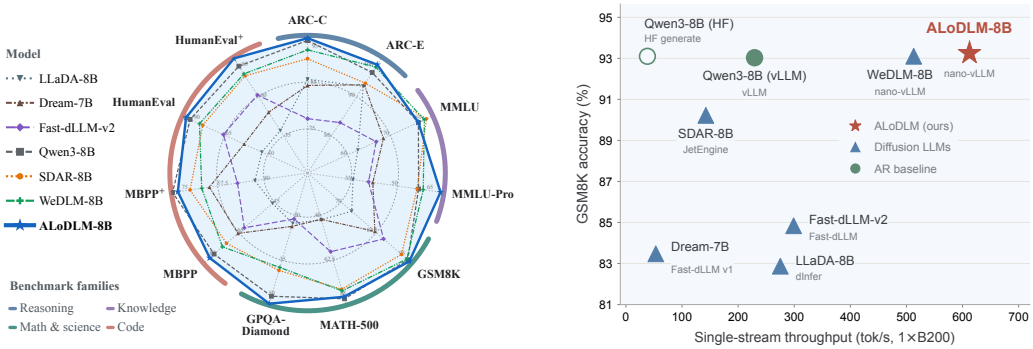


Figure 1: **ALoDLM combines strong performance with fast inference. Left:** ALoDLM-8B outperforms all baselines on average across 11 reasoning, coding, and knowledge benchmarks. **Right:** On GSM8K, ALoDLM-8B delivers $\sim 2.7\times$ the throughput of vLLM-served Qwen3-8B at comparable or higher accuracy, while outperforming the diffusion baselines shown in both accuracy and throughput.

1 Introduction

Autoregressive (AR) large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks (Radford et al., 2018; Raffel et al., 2020; Brown et al., 2020). However, their sequential, token-by-token decoding imposes a severe latency bottleneck on generation (Leviathan et al., 2023; Cai et al., 2024). Diffusion language models (DLMs) (Austin et al., 2021a; Lou et al., 2023; Shi et al., 2024; Ou et al., 2024; Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025; Song et al., 2025) offer a promising alternative: they iteratively denoise corrupted sequences, resolving multiple masked positions in parallel to enable substantially faster generation (Zhang et al., 2025).

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

Despite this potential, their practical adoption remains hindered by a persistent quality gap relative to comparably sized AR models. Existing approaches mitigate this gap either by reintroducing autoregressive structures via semi-AR designs like block diffusion (Arriola et al., 2025; Wu et al., 2025a; Cheng et al., 2025; Liu et al., 2025a; Fu et al., 2026), or by relegating DLMs to drafting candidates for subsequent AR verification (Liu et al., 2025b; Chen et al., 2026a; Yu et al., 2026; Sahoo et al., 2026). We instead ask a more fundamental question:

What is the root cause underlying the generation quality gap in DLMs?

We hypothesize that a key bottleneck in DLM's generation quality is a *computation–difficulty mismatch*. During a single denoising step, masked positions can vary significantly in prediction difficulty due to intrinsic uncertainty (Hüllermeier & Waegeman, 2021), multihop reasoning requirements (Huang et al., 2025), or dependencies on unresolved tokens (Wu et al., 2025b). Despite this variation, standard DLMs apply the same fixed-depth denoiser to all masked positions, training each to predict its target directly from a corrupted context. This uniform allocation wastes computation on easy predictions while starving difficult ones of necessary compute. AR generation largely avoids this within-step allocation problem by predicting a single next token conditioned on a fully resolved prefix. Confidence-based decoding (Nie et al., 2025) partially alleviates the problem by deferring low-confidence predictions to later denoising steps. However, the deferred tokens ultimately receive the same fixed compute since the latent computation performed for an unresolved token is discarded, and the token is recomputed from its mask embedding at the next step. This limitation motivates our central principle: DLMs should maintain and refine persistent latent states for challenging tokens within each denoising step, allowing computation to scale dynamically with prediction difficulty.

In this work, we propose **ALoDLM**, a novel class of DLMs designed to overcome this bottleneck by adaptively allocating computation across tokens during mask recovery. Our first contribution is a looped architecture with token-adaptive recurrent depth. In this design, easy tokens commit early to provide resolved context, while difficult tokens undergo additional recurrent passes to continually refine their persistent latent states. Building on this architecture, we formulate the token-wise computation schedule as a latent variable and derive a unified objective that jointly optimizes the base language model and the compute-allocation policy. Finally, we derive an unbiased stochastic gradient estimator equipped with variance reduction techniques for stable end-to-end training.

Experimental results demonstrate that ALoDLM achieves state-of-the-art generation quality while maintaining fast parallel decoding. We train ALoDLM at 1.7B and 8B scales through direct SFT of Qwen3 backbones on a 5B-token corpus. In quality evaluations across eleven benchmarks spanning general reasoning, mathematics, and coding, ALoDLM achieves average scores of 65.5 and 80.3, surpassing all evaluated DLMs and the corresponding AR baselines. Furthermore, with parallel decoding on GSM8K, ALoDLM-8B delivers approximately 2.7× the speed of vLLM-served Qwen3-8B at comparable or higher accuracy, demonstrating a strong quality–efficiency trade-off. Finally, our analyses show that recurrent depth provides an effective test-time scaling axis and that ALoDLM learns token-dependent refinement preferences without explicit difficulty labels. Our main contributions are:

- **Token-adaptive looped architecture.** We introduce a looped architecture with dynamic recurrent depth. Easy tokens commit early to act as resolved context, while difficult tokens continue to refine their accumulated latent states through additional recurrent passes.
- **Principled training objective.** We formulate token-wise exit schedules as latent variables and derive a conditional ELBO to jointly optimize token prediction and computation allocation. To stabilize training, we develop an unbiased single-trajectory gradient estimator coupled with effective variance reduction techniques.
- **Scaling and performance.** We scale ALoDLM to 8B parameters, making it to the best of our knowledge, the largest looped DLM to date. It achieves state-of-the-art average benchmark performance among all evaluated DLMs and outperforms its Qwen3-based AR counterparts, all while maintaining fast parallel generation.

2 Related Work

Looped Transformers. Looped (or recurrent-depth) transformers repeatedly apply layers or blocks of layers, effectively increasing network depth without expanding the parameter count (Dehghani et al., 2019; Xue et al., 2022; Zhu et al., 2025b; Bae et al., 2025; Nanbeige Lab et al., 2026; Wang et al., 2026;

Gao et al., 2026; Jeddi et al., 2026; Movahedi et al., 2026; Sun et al., 2026b). This looping mechanism serves as a new scaling axis, yielding performance gains on both algorithmic (Giannou et al., 2023; Yang et al., 2024) and reasoning tasks (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025b; Huang et al., 2026). A related line of research leverages this recurrence for *adaptive computation*, where the primary challenge is learning a policy to determine when to halt. ACT (Graves, 2016) and Universal Transformers (Dehghani et al., 2019) learn halting through continuous weighting of intermediate states or outputs. PonderNet (Banino et al., 2021) and Ouro (Zhu et al., 2025a) instead optimize the expected prediction loss by explicitly summing over candidate exit depths, weighted by learned halting probabilities. In ALoDLM, token commitments change the context of unresolved positions, making exact summation over joint exit schedules intractable.

Looped Diffusion Language Models. Diffusion language models (DLMs) generate text through parallel token denoising and are typically trained with an ELBO-derived cross-entropy objective over masked positions (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024). Recent work has integrated looping mechanisms into DLMs to improve training efficiency (Lee et al., 2026), parameter efficiency (Carballo-Castro et al., 2026), and step-wise adaptive computation (Miele et al., 2026). Crucially, they apply a uniform recurrent depth across all token positions within each denoising step, lacking the flexibility to let individual tokens exit latent refinement adaptively. While FReDA (Sun et al., 2026a) supports token-level adaptive commitment, it iteratively revises an explicit token draft rather than propagating an internal latent state across loops. In contrast, ALoDLM enables token-wise adaptive latent depth: easy tokens commit at earlier inner passes to provide their embeddings as context, while harder positions retain their evolving hidden states for continued refinement. For other related topics such as *self-conditioning*, we refer the reader to Appendix A for an extended discussion.

3 METHOD

We introduce ALoDLM, a family of DLMs with token-adaptive recurrent computation. We first describe its inference procedure in §3.1, then derive the training objective and present techniques for gradient estimation and variance reduction in §3.2.

3.1 INFERENCE

Overview. At each denoising step (the *outer loop*), a conventional fixed-depth DLM predicts masked tokens in parallel using the same denoiser depth. ALoDLM augments each step with an *inner loop* that allows tokens to commit at different recurrent depths. Committed tokens provide discrete context for subsequent predictions, while unresolved positions preserve their latent states for further refinement. Predictive confidence controls token commitment, and a learned halting policy determines when to end the inner loop. Algorithm 1 and Figure 2 summarize the procedure.

Architecture. We adopt a standard Transformer architecture, such as Qwen3 (Yang et al., 2025), and partition the model architecture into three components: a Prelude consisting of the token embedding layer and optional prefix Transformer blocks; a Recurrent Core comprising the intermediate Transformer blocks; and a Coda comprising any remaining suffix Transformer blocks.

Adaptive recurrent depth. At the beginning of each denoising step, let $\mathbf{x}_t := (x_1, \ldots, x_N)$ denote the corrupted input and $\mathcal{U} := \{i : x_i = [\texttt{MASK}]\}$ its initial set of unresolved positions. The Prelude produces the initial recurrent state $h^{(0)} = \mathsf{Prelude}(\mathbf{x}_t) \in \mathbb{R}^{N \times d}$, where d is the hidden dimension. At recurrent pass s, the Recurrent Core updates the current state $h^{(s-1)}$ to produce an intermediate state $\widetilde{h}^{(s)}$, which the Coda maps to a readout state:

$$\widetilde{h}^{(s)} = \mathsf{Recurrent\ Core}(h^{(s-1)}), \qquad r^{(s)} = \mathsf{Coda}(\widetilde{h}^{(s)}), \qquad s = 1, \ldots, K, \tag{1}$$

where K is the maximum recurrent depth. Finally, each readout state $r_i^{(s)}$ is processed by two heads in parallel. The unembedding head $\mathsf{LMHead} : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{V}|}$ produces the vocabulary distribution, while an additional head $\mathsf{ExitGate} : \mathbb{R}^d \to \mathbb{R}$ produces a scalar logit whose sigmoid defines the per-pass halting probability, *i.e.*, the probability of halting at pass s, conditioned on not having halted earlier:

$$p_i^{(s)} = \mathrm{softmax}\Big(\mathsf{LMHead}(r_i^{(s)})\Big), \qquad \lambda_i^{(s)} = \begin{cases} \sigma\Big(\mathsf{ExitGate}(\mathrm{sg}[r_i^{(s)}])\Big), & s < K, \\ 1, & s = K. \end{cases} \tag{2}$$

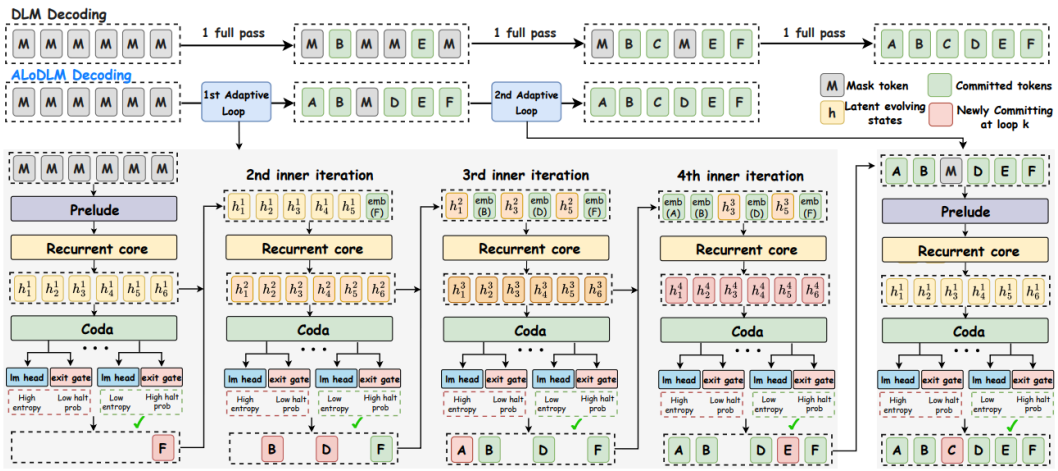


Figure 2: **Overview of ALoDLM.** Standard DLMs apply a fixed-depth denoiser at each denoising step. ALoDLM introduces token-adaptive recurrence within each step: confident tokens commit early and provide discrete context, while unresolved tokens retain and refine their latent states through additional recurrent passes.

Here, σ denotes the sigmoid function, and sg stops gradients from the exit gate into the backbone. We set $\lambda_i^{(K)} = 1$ to force all remaining tokens to exit at depth K, ensuring that the exit-depth probabilities sum to one. The two heads serve complementary roles: we use predictive confidence from the LMHead to determine which tokens are ready to commit, while the ExitGate controls whether to continue refining unresolved positions. We present the detailed procedure below.

Token commitment and latent feedback. At pass s, let $\mathcal{B}^{(s)} \subseteq \mathcal{U}$ contain the unresolved positions whose predictive entropy under $p_i^{(s)}$ is at most τ. For each $i \in \mathcal{B}^{(s)}$, we sample $\widehat{y}_i \sim p_i^{(s)}$ and remove the position from $\mathcal{U}$ (i.e. committed). Let $\mathcal{C}^{(s)} := \bigcup_{s'=1}^{s} \mathcal{B}^{(s')}$ denote all positions committed within the current outer step. The next recurrent input combines their token embeddings with the core outputs at the remaining positions:

$$h_i^{(s)} = \begin{cases} \mathrm{Emb}(\widehat{y}_i), & i \in \mathcal{C}^{(s)}, \\ \widetilde{h}_i^{(s)}, & \text{otherwise}, \end{cases} \tag{3}$$

where Emb is the token embedding function. Thus, committed tokens supply discrete context at subsequent passes, while unresolved positions build on their accumulated latent states.

Adaptive stopping. The exit gate controls whether to continue refining the remaining unresolved positions. Starting from $a_i^{(0)} = 0$, we accumulate the halting probabilities as $a_i^{(s)} = a_i^{(s-1)} + (1 - a_i^{(s-1)})\lambda_i^{(s)}$. After removing newly committed positions, the inner loop terminates if $\mathcal{U}$ is empty or if the mean cumulative halt probability over $\mathcal{U}$ reaches the threshold q. Otherwise, recurrence continues up to depth K.

Algorithm 1 ALoDLM decoding

Input: masked sequence $\mathbf{x}$, maximum depth K, entropy threshold τ, cumulative halt threshold q

while $\mathbf{x}$ contains [MASK] **do** ▷ **Outer loop**
  $\mathcal{U} \leftarrow \{i : x_i = \texttt{[MASK]}\}$
  $h \leftarrow \mathsf{Prelude}(\mathbf{x})$; $a \leftarrow 0$; $\mathcal{C} \leftarrow \varnothing$
  for $s = 1, \ldots, K$ **do** ▷ **Inner loop**
    $h \leftarrow \mathsf{Recurrent\ Core}(h)$
    $r \leftarrow \mathsf{Coda}(h)$
    Compute p_i, λ_i for $i \in \mathcal{U}$ by Eq. (2)
    $a_i \leftarrow a_i + (1 - a_i)\lambda_i, i \in \mathcal{U}$
    $\mathcal{B} \leftarrow \{i \in \mathcal{U} : \mathsf{Entropy}(p_i) \le \tau\}$
    $\widehat{y}_i \leftarrow \mathsf{Sample}(p_i), i \in \mathcal{B}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{B}$; $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{B}$
    $h_i \leftarrow \mathrm{Emb}(\widehat{y}_i), i \in \mathcal{C}$
      ▷ Committed tokens become context
    if $\mathcal{U} = \varnothing$ **or** $\mathrm{mean}_{i \in \mathcal{U}}\, a_i \ge q$ **then break**
  if $\mathcal{C} = \varnothing$ **then** ▷ Ensure progress
    $j \leftarrow \arg\min_{i \in \mathcal{U}} \mathsf{Entropy}(p_i)$
    $\widehat{y}_j \leftarrow \mathsf{Sample}(p_j)$
    $\mathcal{C} \leftarrow \{j\}$
  $x_i \leftarrow \widehat{y}_i, i \in \mathcal{C}$
return $\mathbf{x}$

At the end of the inner loop, committed tokens are written into the sequence. If no token has committed, we select the unresolved position with the lowest predictive entropy and commit its prediction to ensure progress. The next outer step reinitializes the recurrent state from the updated sequence, and decoding continues until no masked positions remain.

3.2 TRAINING

Training ALoDLM involves jointly learning a denoiser and a halting policy that dictates when to commit predictions. This section first introduces the joint exit schedule (Definition 3.1) and its effect on token predictions. We then derive a variational objective to optimize both components (Eq. (5)). Finally, we present an unbiased gradient estimator to optimize this objective (Proposition 1) and introduce practical regularization and variance reduction techniques.

Variational objective. To learn token-wise computation allocation, we treat exit depths as latent variables. As each commitment changes the context for unresolved positions, we model these depths jointly through an exit schedule.

Definition 3.1 (Exit schedule)**.** For a corrupted input $\mathbf{x}_t$, let $\mathcal{M}_t := \{i : x_i = [\texttt{MASK}]\}$ denote the masked positions. An *exit schedule* is a discrete random vector $\mathbf{z} = (z_i)_{i\in\mathcal{M}_t} \in \{1, \dots, K\}^{|\mathcal{M}_t|}$, where $z_i = s$ indicates that token i commits after recurrent pass s.

A noisy sequence $\mathbf{x}_t$ and a fixed exit schedule $\mathbf{z}$ fully specify one outer iteration which comprises up to K inner loops. During this process, early-committed tokens become context for the subsequent inner loops, while the remaining uncommitted tokens continue to refine their latent states. Formally, denote the clean sequence as $\mathbf{y} \in \mathcal{V}^N$. The base model gives us the likelihood of the clean data $\mathbf{y}_{\mathcal{M}_t}$, conditioned on the noisy sequence $\mathbf{x}_t$ and the exit schedule $\mathbf{z}$. An intuitive idea is to marginalize the exit schedules and seek to maximize the following conditional likelihood:

$$p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t) = \sum_{\mathbf{z}} \pi(\mathbf{z})\, p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t, \mathbf{z}). \tag{4}$$

However, this summation ranges over exponentially many schedules, each dictating a separate sequence of model fowards[1]: because committed tokens feed their embeddings back into the recurrent state (Eq. (3)), altering a single token's exit depth modifies the context for the remaining tokens, thereby shifting the entire subsequent trajectory.

We instead introduce a variational distribution $q_\phi(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t})$, parameterized by the exit gate, and derive a negative evidence lower bound (NELBO) with prior distribution $\pi(\mathbf{z})$ over exit schedules:

$$J(\theta, \phi; \mathbf{x}_t, \mathbf{y}) := \mathbb{E}_{\mathbf{z}\sim q_\phi}\Bigg[\underbrace{-\sum_{i\in\mathcal{M}_t} \log p_{\theta,i}^{(z_i)}(y_i \mid \mathbf{x}_t; \mathbf{z})}_{\mathcal{L}_{\text{traj}}(\mathbf{z})}\Bigg] + D_{\text{KL}}(q_\phi \| \pi) \geq -\log p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t). \tag{5}$$

The derivation is provided in Appendix C.1. Here, $p_{\theta,i}^{(z_i)}(y_i \mid \mathbf{x}_t; \mathbf{z})$ denotes the predicted likelihood of token y_i at exit depth z_i, conditioned on the noisy input $\mathbf{x}_t$ and the joint exit schedule $\mathbf{z}$. Summing this loss over all masked positions $i \in \mathcal{M}_t$ yields the overall trajectory loss $\mathcal{L}_{\text{traj}}$ for a given schedule. To encourage shallow exits, we follow Banino et al. (2021) and formulate the prior distribution $\pi(\mathbf{z})$ as a product of geometric distributions truncated to $\{1, \dots, K\}$.

The NELBO above is evaluated on a single corrupted example $\mathbf{x}_t$. The full variational training objective is given by:

$$\boxed{\mathcal{L}_{\text{train}}(\theta, \phi) := \mathbb{E}_{\mathbf{y}\sim p_{\text{data}},\, t\sim\text{Unif}[0,1],\, \mathbf{x}_t\sim q_t(\cdot|\mathbf{y})}\left[\frac{1}{t}\, J(\theta, \phi; \mathbf{x}_t, \mathbf{y})\right]}, \tag{6}$$

where p_{data} denotes the data distribution, and $q_t(\cdot \mid \mathbf{y})$ denotes the *forward diffusion process* that independently mask each position with probability t, and leave it unchanged with probability $1 - t$.

Gradient estimation. To estimate the gradients of Eq. (6) with respect to θ and ϕ, we sample an exit schedule $\mathbf{z} \sim q_\phi$ and evaluate $\mathcal{L}_{\text{traj}}(\mathbf{z})$ along the sampled trajectory. While the denoiser parameters θ can be updated via standard backpropagation, we cannot directly backpropagate through the exit gate ϕ because the discrete $\mathbf{z}$ precludes ordinary pathwise differentiation. To address this, we employ a *score-function estimator*[2] (Williams, 1992), yielding the following unbiased gradient estimator:

[1] For a nonempty masked set, evaluating a schedule $\mathbf{z}$ requires $\max_{i\in\mathcal{M}_t} z_i$ recurrent passes.

[2] Continuous relaxations, such as Gumbel-Softmax (Jang et al., 2017), provide a viable alternative. However, they would require extending Eq. (3) to accommodate partially committed tokens, potentially introducing additional optimization challenges during training.

Table 1: **Main results.** We compare ALoDLM against its corresponding AR baseline and recent diffusion language models (DLMs). Within each model scale, the overall best results are in **bold**, and the best non-AR results are underlined. Columns corresponding to our model are shaded in blue.

Benchmark	AR Baseline *Instruct Model*	DLM Baseline *Instruct Model*	Ours *Instruct Model*	AR Baseline *Instruct Model*	DLM Baseline *Instruct Model*					Ours *Instruct Model*
	Qwen3 1.7B	SDAR 1.7B	**ALoDLM 1.7B**	Qwen3 8B	LLaDA 8B	Dream 7B	Fast-dLLM-v2 7B	SDAR 8B	WeDLM 8B	**ALoDLM 8B**
General Reasoning										
ARC-C	**82.4**	74.9	77.7	93.9	85.6	84.3	77.2	90.0	91.9	**94.4**
ARC-E	**90.0**	86.5	88.7	96.1	92.6	93.0	83.4	93.4	97.5	**98.1**
MMLU	60.3	**63.4**	59.4	76.6	62.4	68.4	66.7	**78.5**	78.0	76.6
MMLU-Pro	39.0	37.0	**39.6**	56.8	35.6	42.0	40.6	56.3	58.3	**63.9**
Math & Science										
GSM8K	83.2	80.1	**85.5**	93.6	73.8	82.0	85.1	91.4	93.3	**94.2**
MATH-500	**72.0**	62.4	61.8	**81.8**	42.2	42.0	58.2	77.0	77.8	80.8
GPQA-Diamond	31.3	32.3	**41.4**	47.0	22.7	23.7	21.2	38.4	37.4	**49.5**
Code Generation										
MBPP	61.9	60.3	**65.5**	79.0	46.0	65.5	61.9	72.0	74.3	**81.5**
MBPP$^+$	58.5	59.4	**60.1**	**74.5**	43.4	60.6	50.0	67.9	63.5	72.5
HumanEval	62.8	61.6	**72.6**	85.4	43.3	53.7	65.9	78.0	79.9	**87.8**
HumanEval$^+$	60.4	53.0	**67.7**	79.3	38.4	50.0	61.0	73.2	74.4	**84.2**
Average	63.8	61.0	**65.5**	78.5	53.3	60.5	61.0	74.2	75.1	**80.3**

Proposition 1 (Unbiased gradient estimator)**.** *Sample $\mathbf{y}, t, \mathbf{x}_t$ as in Eq. (6), and sample $\mathbf{z} \sim q_\phi$ sequentially during a single recurrent rollout. With the sampled exit decisions held fixed during backpropagation, define*

$$\widehat{\mathcal{L}}_{\text{sur}}(\mathbf{z}) = \frac{1}{t}\left\{ \mathcal{L}_{\text{traj}}(\mathbf{z}) + \text{sg}\left[\mathcal{L}_{\text{traj}}(\mathbf{z}) + \log \frac{q_\phi(\mathbf{z})}{\pi(\mathbf{z})}\right] \log q_\phi(\mathbf{z}) \right\}. \tag{7}$$

Under the stop-gradient convention in Eq. (2), with the policy features held fixed when differentiating with respect to θ, this surrogate yields unbiased gradients:

$$\mathbb{E}_{\mathbf{y},t,\mathbf{x}_t,\mathbf{z}}\left[\nabla_{\theta,\phi}\widehat{\mathcal{L}}_{\text{sur}}(\mathbf{z})\right] = \nabla_{\theta,\phi}\mathcal{L}_{\text{train}}(\theta,\phi). \tag{8}$$

The proof is provided in Appendix C.2. In Eq. (7), the first term $\mathcal{L}_{\text{traj}}$ trains the denoiser to predict target tokens. The second term utilizes the score-function estimator to train the halting policy. The detached cost $\mathcal{L}_{\text{traj}}(\mathbf{z}) + \log \frac{q_\phi(\mathbf{z})}{\pi(\mathbf{z})}$ provides an outcome signal for the sampled schedule, giving each halting decision credit for its effect on the sequence as a whole. This encourages the policy to favor schedules that achieve high prediction accuracy while remaining close to the prior distribution.

Practical regularization. To maintain token-wise adaptivity, we relax the joint KL penalty term $\log \frac{q_\phi(\mathbf{z})}{\pi(\mathbf{z})}$. Specifically, we regularize the average depth distribution across the minibatch toward the geometric prior, while applying a weaker penalty to the variations among individual token depths. We defer the complete formulation to Appendix C.3.

Variance reduction. Eq. (7) directly supervises each token at a single sampled depth, although the same rollout produces predictions at every preceding pass. We reuse these predictions to give earlier passes a direct learning signal, even when a token exits later. For token i, we average the per-depth cross-entropy losses as $\sum_{s=1}^{z_i} \text{sg}[\omega_i^{(s)}]\ell_i^{(s)}$, where $\omega_i^{(s)}$ is the predicted exit-depth probability after assigning all mass beyond the sampled exit z_i to z_i, so that $\sum_{s=1}^{z_i} \omega_i^{(s)} = 1$. This requires no additional recurrent passes and empirically reduces denoiser gradient variance (Figure 5, right). Furthermore, we subtract the detached first-pass prediction cost $\ell_i^{(1)}$ as a *control variate* (Williams, 1992). The full formulation of the weighting scheme and the baseline derivation is given in Appendix C.4, and the complete training surrogate is Eq. (34).

4 Experiments

We first describe the experimental setup (§4.1), then evaluate generation quality (§4.2), inference speed and computational cost (§4.3), followed by analyses and ablations (§4.4).

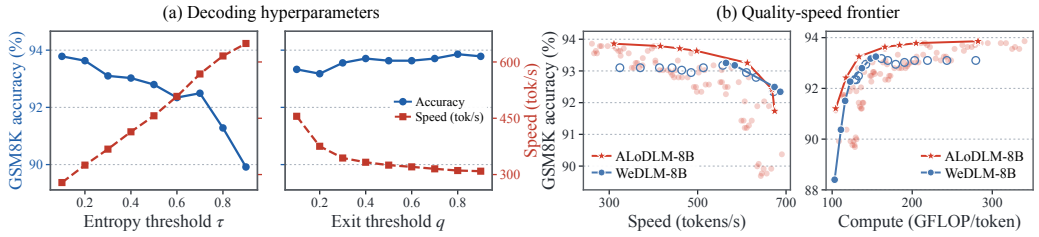


Figure 3: **(a) Complementary roles of the entropy and halting gates.** Increasing τ relaxes the commitment criterion and accelerates generation, with sharper accuracy losses at high thresholds (left). Increasing q permits deeper latent refinement, also yielding accuracy gains at lower throughput (right). These gates collectively balance parallel token commitment with adaptive refinement of unresolved tokens. **(b) Adaptive recurrence improves the quality–efficiency frontier.** Across most of the evaluated range, ALoDLM-8B achieves higher accuracy than WeDLM-8B at comparable throughput (left) or compute per generated token (right). Solid lines connect frontier configurations.

4.1 Evaluation Setup

Training details. We convert Qwen3-1.7B and Qwen3-8B (Yang et al., 2025) into ALoDLM-1.7B and ALoDLM-8B, respectively, using WeDLM's streaming block-diffusion framework (Liu et al., 2025a), which enables full-sequence conditioning with a standard causal attention mask. Both models use a maximum recurrent depth of $K = 4$. The recurrent core comprises all 28 Transformer layers at 1.7B and the middle 16 layers at 8B. Unlike prior conversion pipelines combining continued pretraining (CPT) and supervised fine-tuning (SFT) (Tian et al., 2025; Liu et al., 2025a; Cheng et al., 2025), we skip CPT and directly perform SFT on a 5B-token corpus using AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-5} and weight decay 0.01. We defer the full details to Appendix D.

Benchmarks. We evaluate general knowledge and scientific reasoning on ARC-Easy and ARC-Challenge (Clark et al., 2018), MMLU (Hendrycks et al., 2021a), MMLU-Pro (Wang et al., 2024), and GPQA-Diamond (Rein et al., 2023); mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021b); and code generation on HumanEval (Chen et al., 2021), sanitized MBPP (Austin et al., 2021b), and their EvalPlus variants (Liu et al., 2023). We report accuracy or exact match for non-code tasks, and execution-based pass@1 for coding benchmarks. Following SDAR (Cheng et al., 2025), we adopt the OpenCompass evaluation protocol (OpenCompass Contributors, 2023), utilizing each model's native chat template, greedy decoding, and a 4,096-token generation limit across all tasks. To ensure a fair comparison in Table 1, we constrain all methods to generate exactly one token per step according to their respective default decoding rules (*e.g.,* selecting the token with the lowest entropy). Full details are provided in Appendix D.2.

Baselines. We compare ALoDLM against both AR baselines and recent diffusion language models. For block diffusion baselines, we mainly focus on models converted from Qwen family of models to ensure fair comparison. Our main comparison includes Qwen3 (Yang et al., 2025), Dream (Ye et al., 2025), LLaDA (Nie et al., 2025), Fast-dLLM-v2 (Wu et al., 2025a), SDAR (Cheng et al., 2025), and WeDLM (Liu et al., 2025a) at the corresponding parameter scales. To ensure fair comparison, each model uses its optimized high-speed inference framework: LLaDA uses dInfer (Ma et al., 2025), SDAR uses JetEngine, and the WeDLM and Qwen models use vLLM (Kwon et al., 2023). Our ALoDLM is also served via vLLM using depth-aware KV caching (Elbayad et al., 2020; Geiping et al., 2025; Zhu et al., 2025a): for each prefix token, we use KV states at the same recurrent depth when available, and otherwise use its deepest available states (see Appendix B.1 for details).

4.2 Performance Evaluation

Table 1 compares ALoDLM against prior methods across eleven benchmarks. The results demonstrate that ALoDLM consistently surpasses all evaluated DLMs, even outperforming the strong AR baselines. At both the 1.7B and 8B parameter scales, ALoDLM establishes new state-of-the-art averages of 65.5 and 80.3, respectively. ALoDLM-8B outperforms the strong DLM baseline WeDLM-8B by an average of 5.2 points, leading on ten of the eleven tasks. Notably, ALoDLM achieves this performance through direct supervised fine-tuning (SFT) on a modest 5B-token corpus, bypassing the expensive continued-pretraining phase required by SDAR and WeDLM. Compared to the strong Qwen3 AR

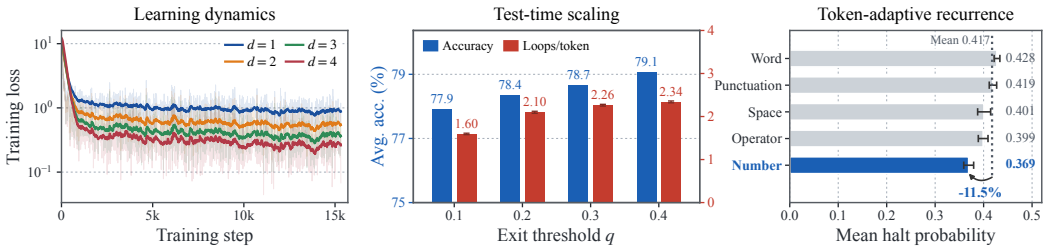


Figure 4: **Left: Learning dynamics.** After the initial training phase, deeper recurrent readouts exhibit lower training loss on their active tokens. **Middle: Test-time scaling with q.** Increasing q raises loops-per-token and the average score across eleven benchmarks. **Right: Token-adaptive recurrence.** Mean first-pass halting probabilities differ across token categories, with numerical tokens showing the lowest mean. These preferences are learned without explicit token-difficulty labels or predefined halting targets.

models, ALoDLM presents consistent improvements on both 1.7B (63.8 vs. 65.5) and 8B scales (78.5 vs. 80.3). The model shows particularly strong gains in code generation, sweeping all four coding benchmarks over all baselines (except slightly behind the AR models on MBPP+). This advantage may be attributed to its looped architecture, aligning with the findings of Wang et al. (2026).

4.3 Quality–Efficiency Trade-off

Complementary decoding controls. Figure 3(a) illustrates how the entropy threshold τ and exit threshold q balance parallel token commitment and latent refinement in ALoDLM-8B on GSM8K. At fixed $q = 0.5$, increasing τ from 0.1 to 0.6 relaxes the commitment criterion and raises throughput from 278.7 to 508.3 tokens/s, while accuracy decreases from 93.8% to 92.3%. More aggressive commitment incurs a larger quality penalty, with accuracy falling to 89.9% at $\tau = 0.9$. The exit threshold q controls how long unresolved tokens continue to refine their latent states. At fixed $\tau = 0.2$, increasing q from 0.1 to 0.9 raises accuracy from 93.3% to 93.8%, while throughput decreases from 455.3 to 309.0 tokens/s. Together, these controls allow confident predictions to become discrete context promptly while reserving further recurrent computation for unresolved tokens.

Quality–efficiency frontier. Figure 3(b) compares ALoDLM-8B with the strong WeDLM-8B baseline under single-stream inference on one NVIDIA B200. The evaluation covers 94 (q, τ) settings for ALoDLM and 21 entropy-threshold settings for WeDLM. The accuracy–throughput frontiers cross near 650 tokens/s: ALoDLM offers better trade-offs in the higher-accuracy regime, while WeDLM leads at the fastest displayed operating points. At a matched accuracy of 93.25%, optimized ALoDLM-8B reaches 612.4 tokens/s versus 564.2 for WeDLM, an 8.5% throughput gain.

Throughput depends on both arithmetic complexity and execution efficiency (e.g., kernel performance, memory traffic, and runtime overhead). To isolate and evaluate arithmetic efficiency, we compare the estimated GFLOPs per generated token (Figure 3(b), right). ALoDLM maintains higher accuracy across the entire evaluated compute range. At the 93.25% accuracy threshold, it requires only 133.5 GFLOPs per token compared to 154.6 for WeDLM, a 13.6% reduction. These results confirm that adaptive latent recurrence improves generation quality per unit of computation while maintaining competitive parallel decoding speeds.

4.4 Analysis and Ablations

Test-time scaling. Figure 4 (left) shows that, after the initial training phase, deeper recurrent readouts in ALoDLM-8B exhibit lower training loss on their active tokens, consistent with progressive refinement. At inference time, increasing the exit threshold q raises the average number of loops per token from 1.6 to 2.34 and the average score across eleven benchmarks from 77.9% to 79.1% (Figure 4, middle). These results validate q as a control for average recurrent depth and demonstrate the effectiveness of latent recurrence as an axis for test-time scaling.

Token-adaptive recurrence. Figure 4 (right) compares mean first-pass halt probabilities across token types in ALoDLM-8B, averaged equally over GSM8K, MATH-500, MBPP-sanitized, and HumanEval. Numerical tokens have the lowest mean halt probability (0.369), approximately 11.5% below the

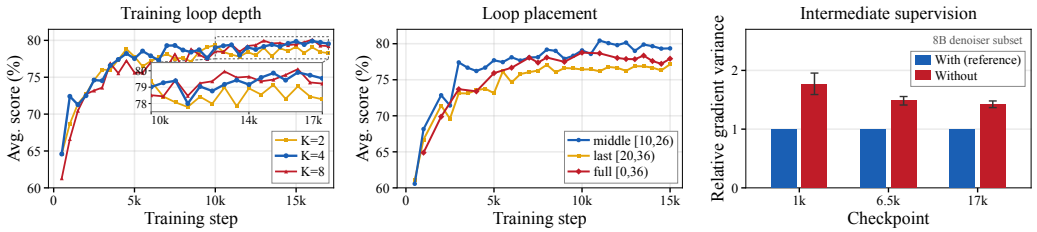


Figure 5: **Left:** At 8B, $K = 4$ outperforms $K = 8$ early in training and achieves comparable performance later, while $K = 2$ plateaus at a lower score. The inset enlarges steps 10,000–17,000. **Middle:** Middle-layer recurrence outperforms last-layer recurrence later in training. The full-stack curve is a historical reference evaluated on a different benchmark subset. **Right:** Intermediate supervision reduces conditional gradient variance in ALoDLM-8B. Variance across sampled exit trajectories is normalized to the value with intermediate supervision at each checkpoint. Error bars show paired 95% bootstrap confidence intervals across eight fixed input packs.

cross-dataset reference mean (0.417), while word tokens have the highest (0.428). This lower halting propensity indicates a stronger learned preference for further latent refinement at numerical positions. Such a preference is consistent with the importance of precise numerical predictions in mathematical reasoning and code generation, where small errors in values can alter the final result. These observations support our approach to mitigating the *computation–difficulty mismatch*: learned token-dependent halting preferences can guide additional computation toward predictions requiring further refinement. This adaptive behavior emerges from jointly learning token prediction and the halting policy, without explicit difficulty labels or predefined halting targets for different token categories.

Maximum recurrent depth. Figure 5 (left) compares ALoDLM-8B trained with $K \in \{2, 4, 8\}$ under matched settings, reporting the mean score across six benchmarks. At matched training steps, $K = 2$ and $K = 4$ improve faster initially than $K = 8$, but $K = 2$ plateaus at a lower score. With half the maximum recurrent depth, $K = 4$ achieves comparable performance to $K = 8$ later in training, motivating our choice of $K = 4$.

Loop placement. Prior looped models vary in the extent and placement of recurrence (Geiping et al., 2025; Zhu et al., 2025a; Lee et al., 2026). Figure 5 (middle) compares recurrence over the middle layers $[10, 26)$ and last layers $[20, 36)$ of Qwen3-8B, with the full-stack trajectory $[0, 36)$ included as a historical reference. Middle-layer recurrence outperforms last-layer recurrence later in training, although both cores contain 16 layers. This result suggests that effective latent refinement depends on where recurrence is applied, motivating the middle-layer core of ALoDLM-8B.

Variance reduction. Figure 5 (right) evaluates the effect of intermediate denoiser supervision on gradient noise in ALoDLM-8B. Using fixed corrupted inputs and paired exit trajectories, we compare weighted supervision over pre-commitment readouts with supervision at the sampled exit alone. For the measured subset of 4,096 denoiser normalization parameters, removing intermediate supervision increases conditional gradient variance to $1.76\times$, $1.49\times$, and $1.42\times$ the full-method value at training steps 1,000, 6,500, and 17,000, respectively. These results justify reusing intermediate predictions to reduce noise from exit-trajectory sampling, providing a less noisy denoiser training signal for latent refinement without additional recurrent forward passes.

5 Conclusion

We introduced ALoDLM, a family of adaptively looped diffusion language models that address the *computation–difficulty mismatch* through token-adaptive latent recurrence. Within each denoising step, confident tokens commit early to provide discrete context, while unresolved tokens retain and refine their latent states across recurrent passes. We derived a conditional ELBO over latent token-wise exit schedules to guide computation allocation through sequence-level credit assignment, and incorporated intermediate supervision to support denoiser training. Across eleven benchmarks, ALoDLM outperforms all evaluated diffusion models and the corresponding Qwen3 baselines in average score at both 1.7B and 8B scales. With optimized inference, ALoDLM-8B delivers approximately $2.7\times$ the throughput of vLLM-served Qwen3-8B at comparable accuracy on GSM8K.

AI USE STATEMENT

We used generative AI tools to assist with developing and checking mathematical derivations and proofs, implementing methods, revising the manuscript, and creating and modifying scientific figures. We reviewed all AI-assisted material, verified the mathematical arguments and implementation, and take responsibility for the final text, claims, and artifacts.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2503.09573.

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

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

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-Recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025. URL https://arxiv.org/abs/2507.10524.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021. URL https://arxiv.org/abs/2107.05407.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Alba Carballo-Castro, Julianna Piskorz, Paulius Rauba, Mihaela van der Schaar, and Pascal Frossard. Recursive scaling in masked diffusion models. *arXiv preprint arXiv:2606.18022*, 2026. URL https://arxiv.org/abs/2606.18022.

Michael Cardei, Huu Binh Ta, and Ferdinando Fioretto. Simple self-conditioning adaptation for masked diffusion models. *arXiv preprint arXiv:2604.26985*, 2026. URL https://arxiv.org/abs/2604.26985.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. *ArXiv*, abs/2602.06036, 2026a. URL https://api.semanticscholar.org/CorpusID:285304343.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. DMax: Aggressive parallel decoding for dLLMs. *arXiv preprint arXiv:2604.08302*, 2026b. URL https://arxiv.org/abs/2604.08302.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1910.10073.

Yonggan Fu, Lexington Allen Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukic, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. 2026. URL https://api.semanticscholar.org/CorpusID:289333310.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies! *arXiv preprint arXiv:2607.16051*, 2026. URL https://arxiv.org/abs/2607.16051.

Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. Set block decoding is a language model inference accelerator. *ArXiv*, abs/2509.04185, 2025. URL https://api.semanticscholar.org/CorpusID:281103543.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025. URL https://arxiv.org/abs/2502.05171.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442, 2023. URL https://arxiv.org/abs/2301.13196.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. URL https://arxiv.org/abs/1603.08983.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021b.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.17206.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. *arXiv preprint arXiv:2605.21488*, 2026.

Yu Huang, Zixin Wen, Aarti Singh, Yuejie Chi, and Yuxin Chen. Transformers provably learn chain-of-thought reasoning with length generalization. *ArXiv*, abs/2511.07378, 2025. URL https://api.semanticscholar.org/CorpusID:282912810.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01144.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026.

Mingyu Jo, Jaesik Yoon, Justin Deschenaux, Caglar Gulcehre, and Sungjin Ahn. Loopholing discrete diffusion: Deterministic bypass of the sampling wall. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.19304.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

Sanghyun Lee, Chunsan Hong, Seungryong Kim, Jonghyun Lee, Jongho Park, and Dongmin Park. Looped diffusion language models. *arXiv preprint arXiv:2605.26106*, 2026. URL https://arxiv.org/abs/2605.26106.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. WeDLM: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025a. URL https://arxiv.org/abs/2512.22737.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. *ArXiv*, abs/2511.08923, 2025b. URL https://api.semanticscholar.org/CorpusID:282940200.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023. URL https://arxiv.org/abs/2310.16834.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, Zhenzhong Lan, Junbo Zhao, Jianguo Li, and Da Zheng. dInfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv:2510.08666*, 2025. URL https://arxiv.org/abs/2510.08666.

Andrea Miele, Yiming Qin, Alba Carballo-Castro, Justin Deschenaux, and Pascal Frossard. Fixed-point masked generative modeling. *arXiv preprint arXiv:2605.31215*, 2026. URL https://arxiv.org/abs/2605.31215.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. *arXiv preprint arXiv:2606.18206*, 2026. URL https://arxiv.org/abs/2606.18206.

Nanbeige Lab, Chen Yang, Chengrui Huang, Fufeng Lan, Hanhui Chen, Hao Zhou, Huatong Song, Jiaqi Cao, Jiaying Zhu, Jinlin Niu, et al. Nanbeige4.2-3B: Unlocking agentic capabilities in a compact model. *arXiv preprint arXiv:2607.22083*, 2026. URL https://arxiv.org/abs/2607.22083.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

OpenCompass Contributors. OpenCompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024. URL https://arxiv.org/abs/2406.03736.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.07524.

Subham Sekhar Sahoo, Lingjie Chen, Khiem Pham, Jonathan Geuter, Chaitanya Dwivedi, Varad Pimpalkhute, Yash Akhauri, Alexander Moreno, Mikhail Yurochkin, Zhenting Wang, Mostafa Elhoushi, Nolan Dey, Shane Bergsma, Joel Hestness, John Thickstun, Eric Xing, and Zhengzhong Liu. Unlocking lossless speedups in LLMs via discrete diffusion. *arXiv preprint arXiv:2609.04010*, 2026. URL https://arxiv.org/abs/2609.04010.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.17416.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.04329.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025. URL https://arxiv.org/abs/2508.02193.

Haotian Sun, Rushi Qiang, Yuqian Zheng, and Bo Dai. Forward-free diffusion language models with BPTT-free looped refinement. *arXiv preprint arXiv:2606.08357*, 2026a. URL https://arxiv.org/abs/2606.08357.

Yutao Sun, Li Dong, Tianzhu Ye, Shaohan Huang, Jianyong Wang, and Furu Wei. Universal YOCO for efficient depth scaling. *arXiv preprint arXiv:2604.01220*, 2026b. URL https://arxiv.org/abs/2604.01220.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, et al. From next-token to next-block: A principled adaptation path for diffusion LLMs. *arXiv preprint arXiv:2512.06776*, 2025. URL https://arxiv.org/abs/2512.06776.

Shaowen Wang, Ge Zhang, Kairong Luo, Yuhao Wu, Shaofan Liu, Jiaheng Liu, Wenhao Huang, Shen Yan, and Jian Li. SMELT: Scaling laws for compute-matched MoE looped transformers. *arXiv preprint arXiv:2609.01343*, 2026. URL https://arxiv.org/abs/2609.01343.

Yubo Wang, Xueguang Ma, Ge Zhang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025b. URL https://arxiv.org/abs/2505.22618.

Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. Go wider instead of deeper. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8779–8787, 2022. URL https://arxiv.org/abs/2107.11817.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2311.12424.

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

Yifan Yu, Yuqing Jian, Junxiong Wang, Zhongzhu Zhou, Donglin Zhuang, Xinyu Fang, Sri Yanamandra, Xiaoxia Wu, Qingyang Wu, Shuaiwen Leon Song, Tri Dao, Ben Athiwaratkun, James Zou, Fan Lai, and Chenfeng Xu. Introspective diffusion language models. *arXiv preprint arXiv:2604.11035*, 2026. URL https://arxiv.org/abs/2604.11035.

Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S. Yu, and Aiwei Liu. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*, 2025. URL https://arxiv.org/abs/2508.08712.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason K. Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025a. URL https://arxiv.org/abs/2510.25741.

Ruike Zhu, Hanwen Zhang, Kevin Li, Tianyu Shi, Yiqun Duan, Chi Wang, Tianyi Zhou, Arindam Banerjee, and Zengyi Qin. Beyond parameters: Exploring virtual logic depth for scaling laws. *arXiv preprint arXiv:2506.18233*, 2025b. URL https://arxiv.org/abs/2506.18233.

APPENDIX CONTENTS

A Additional Related Work **17**

B Additional Method Details **17**

B.1 Depth-aware KV caching 17

B.2 Accounting for recurrent computation 18

C Training Objective and Gradient Derivations **18**

C.1 Schedule-conditioned likelihood and variational bound 18

C.2 Single-trajectory gradient estimator 19

C.3 Geometric prior and depth regularization 20

C.4 Variance reduction and the training surrogate 21

D Experimental Details **22**

D.1 Models and training 22

D.2 Generation quality and inference efficiency 23

E Case Studies **25**

A Additional Related Work

Discrete Diffusion Language Models generate text by reversing a categorical corruption process. Masked DLMs use an absorbing mask state, yielding weighted clean-token cross-entropy objectives and parallel token recovery (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024). LLaDA and Dream extend this formulation to large language models (Nie et al., 2025; Ye et al., 2025). Block Diffusion and SDAR combine left-to-right block generation with parallel denoising within each block, supporting flexible-length generation and prefix reuse (Arriola et al., 2025; Cheng et al., 2025). WeDLM enables parallel mask recovery with causal attention and streaming prefix commitment (Liu et al., 2025a). Fast-dLLM pairs approximate KV caching with confidence-aware parallel decoding to reduce redundant computation and limit errors from simultaneous predictions (Wu et al., 2025b). Diffusion drafting can also support autoregressive verification: TiDAR shares one model between both roles, whereas DFlash uses a separate diffusion drafter (Liu et al., 2025b; Chen et al., 2026a). These approaches organize token updates and contextual computation; `ALoDLM` additionally varies recurrent refinement before individual predictions become context.

Looped Transformers reuse shared blocks to increase effective depth without increasing parameter count. Huginn trains with variable recurrent depth (Geiping et al., 2025); Ouro and Mixture-of-Recursions learn token-wise depth allocation (Zhu et al., 2025a; Bae et al., 2025). Probabilistic halting and variational depth objectives have precedents in PonderNet and Ouro (Banino et al., 2021; Zhu et al., 2025a). Within masked diffusion, LoopMDM and R-MDM refine continuous hidden states through shared recurrent passes before an outer token update (Lee et al., 2026; Carballo-Castro et al., 2026). LoopMDM also supports adaptive inference by monitoring sequence-level hidden-state change. FReDA instead refines explicit token drafts, including confidence and commitment information, and backpropagates only through the final sampled refinement pass (Sun et al., 2026a). It supports early stopping and best-of-N refinement. `ALoDLM` interleaves latent refinement with token commitments inside each outer step. Since an earlier commitment changes subsequent predictions at unresolved positions, training treats commitment depths as a joint latent schedule. At inference, predictive entropy selects individual commitments, while the mean cumulative halt probability over unresolved positions controls termination of the inner loop.

Self-conditioning retains predictions or representations between denoising steps. Simple Self-Conditioning feeds back a detached clean-token distribution (Cardei et al., 2026); DiffusionGemma maps predicted probabilities to continuous features within multinomial diffusion (Team et al., 2026). Soft-Masked DLMs blend mask embeddings with confidence-weighted top-k token embeddings (Hersche et al., 2026), while DMax combines mixtures of prediction and mask embeddings with on-policy training (Chen et al., 2026b). Loopholing carries hidden states alongside the sampled sequence (Jo et al., 2026). CoFRe combines a fixed-point denoiser with token-aware state reuse, initializing the next solve differently for unchanged visible, still-masked, and newly revealed positions (Miele et al., 2026). `ALoDLM` shares the motivation of preserving useful information, but retains unresolved latent states within an outer step while committed positions supply token embeddings. Its outer-step initialization resets this inner state from the updated sequence, distinguishing its feedback mechanism from cross-step state reuse.

B Additional Method Details

This section describes depth-aware KV caching and the computational cost of recurrent inference.

B.1 Depth-aware KV caching

Weight sharing does not imply that attention keys and values are identical across recurrent passes. The same physical layer receives different hidden states at different depths. We therefore distinguish a layer index ℓ from its recurrent invocation d and maintain depth-specific logical buffers for the recurrent core:

$$\{K_\ell^{(d)}, V_\ell^{(d)}\}, \qquad \ell \in [L_{\text{start}}, L_{\text{end}}), \quad d \in \{1, \dots, K\}. \tag{9}$$

Prefill supplies prompt context at the required depths. During decoding, a depth-d attention invocation uses prefix entries computed at depth d when available, using the original sequence positions and the base model's attention mask.

Available depth and commitment depth. Let $D_{\ell,i}$ be the deepest computed cache depth for prefix position i at layer ℓ. When available depths form the prefix $1, \ldots, D_{\ell,i}$, the matching-depth lookup with deepest-available fallback reads

$$\big(K_{\ell,i}^{(\min(d,D_{\ell,i}))}, V_{\ell,i}^{(\min(d,D_{\ell,i}))}\big). \tag{10}$$

The available cache depth need not equal the token's commitment depth. In a dense recurrent pass, an already committed position can still be processed as embedding context for unresolved positions; the depth-specific entries produced by that computation are valid and should be retained. Reusing an earlier entry is necessary only when the matching entry is absent. In particular, embedding feedback should not be replaced by freezing every committed position's pre-commitment latent cache.

Coda readouts. For middle-layer recurrence, every pass branches the recurrent carry through the coda to obtain its prediction and gate readout. The next recurrent pass continues from the pre-coda carry. The serving implementation allocates separate depth buffers to recurrent layers, while prelude and coda layers use one buffer per physical layer. Coda entries for the active window are refreshed by each readout branch, so cached coda context uses the latest evaluated branch. This storage convention is distinct from allocating a separate cache for every coda invocation.

Memory and interpretation. For L total layers and $m = L_{\text{end}} - L_{\text{start}}$ recurrent layers, this layout has $L + (K-1)m$ logical KV-buffer pairs. At 8B, $L = 36$, $m = 16$, and $K = 4$ give 84 pairs, of which 64 belong to the recurrent core. The fallback and shared coda buffers are serving approximations: their outputs need not match those obtained by recomputing every prefix token at every depth. Related early-exit and recurrent models face the same need to provide attention context when computation depths differ (Elbayad et al., 2020; Geiping et al., 2025; Zhu et al., 2025a).

B.2 Accounting for recurrent computation

Layer evaluations clarify the cost of a middle-layer recurrent core. With S executed passes, the prelude runs once, whereas both the core and the coda readout run at every pass:

$$C_{\text{layers}}(S) = L_{\text{start}} + S(L_{\text{end}} - L_{\text{start}}) + S(L - L_{\text{end}}). \tag{11}$$

For the 8B partition $[10, 26)$, the full four-pass path evaluates $10 + 4 \cdot 16 + 4 \cdot 10 = 114$ Transformer layers, compared with $4 \cdot 36 = 144$ for four full-stack passes. The 1.7B configuration repeats all 28 layers, giving $28S$ evaluations.

These are per-position layer counts for a dense execution path, not FLOP counts per generated token. A window with unresolved positions can be processed several times, and a pass can commit several tokens. Moreover, positions already committed within the window can still incur computation as context. With batching, the shared execution can run to the largest stopping depth among the active sequences. We therefore measure serving cost from executed operations and generated-token counts, rather than inferring speed from the distribution of token commitment depths alone.

C Training Objective and Gradient Derivations

We give the variational-bound and gradient-estimator proofs, followed by the relaxed depth regularizer and variance-reduction details deferred from Section 3.2.

C.1 Schedule-conditioned likelihood and variational bound

Fix a corrupted example $\mathbf{x}_t$ with $t > 0$ and masked positions $\mathcal{M}_t$. For a schedule $\mathbf{z}$, let $B_s(\mathbf{z}) = \{i \in \mathcal{M}_t : z_i = s\}$ and $G_{<s}(\mathbf{z}) = \bigcup_{r<s} B_r(\mathbf{z})$. The readouts at pass s are computed before inserting the targets of tokens in B_s. They may depend on earlier committed targets $\mathbf{y}_{G_{<s}}$, which are fed back during likelihood evaluation, but not on the targets of tokens that remain active. Thus

$$\begin{aligned} p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t, \mathbf{z}) &= \prod_{s=1}^{K} \prod_{i \in B_s(\mathbf{z})} p_{\theta,i}^{(s)}\big(y_i \mid \mathbf{x}_t, \mathbf{y}_{G_{<s}(\mathbf{z})}; \mathbf{z}\big) \\ &= \exp[-\mathcal{L}_{\text{traj}}(\mathbf{z})]. \end{aligned} \tag{12}$$

Each readout is a normalized categorical distribution, and tokens within an exit group are conditionally independent given earlier groups under the parallel readout model. Summing over the last group's targets gives one; repeating in reverse exit order proves that Eq. (12) is normalized. Using observed earlier targets is teacher forcing for this exit order; generation from the same conditional model instead feeds back samples.

Let $\pi(\mathbf{z})$ be a normalized prior independent of the unknown masked targets. It may depend on the known mask set. Then Eq. (4) defines a normalized conditional likelihood. The variational distribution $q_\phi(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t})$ may depend on the observed targets; this does not change the generative model.

Proposition 2 (Conditional negative ELBO)**.** *For any q_ϕ supported on schedules with positive prior probability,*

$$-\log p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t) \le J := \mathbb{E}_{q_\phi}[\mathcal{L}_{\mathrm{traj}}(\mathbf{z})] + D_{\mathrm{KL}}(q_\phi \| \pi). \tag{13}$$

When these quantities are finite, the gap is

$$J + \log p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t) = D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t})). \tag{14}$$

The bound is tight exactly when q_ϕ equals the schedule posterior.

Proof. Write S_q for the support of q_ϕ and suppress conditioning arguments. Restricting the marginal sum to S_q and applying Jensen's inequality gives

$$\begin{aligned}\log p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t) &\ge \log \sum_{\mathbf{z} \in S_q} q_\phi(\mathbf{z}) \frac{\pi(\mathbf{z}) p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t, \mathbf{z})}{q_\phi(\mathbf{z})} \\ &\ge \mathbb{E}_{q_\phi}[-\mathcal{L}_{\mathrm{traj}}(\mathbf{z}) + \log \pi(\mathbf{z}) - \log q_\phi(\mathbf{z})] = -J. \end{aligned} \tag{15}$$

For the gap, Bayes' rule yields

$$\log \frac{q_\phi(\mathbf{z})}{p_\theta(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t})} = \log \frac{q_\phi(\mathbf{z})}{\pi(\mathbf{z})} + \mathcal{L}_{\mathrm{traj}}(\mathbf{z}) + \log p_\theta(\mathbf{y}_{\mathcal{M}_t} \mid \mathbf{x}_t).$$

Taking the q_ϕ-expectation proves Eq. (14); nonnegativity of KL gives the equality condition. □

This result concerns the joint KL in Eq. (5). Appendix C.3 describes its relaxation and the coefficients that recover the exact KL.

C.2 Single-trajectory gradient estimator

Schedule probability. Let $\mathcal{F}_s$ denote the history immediately before the exit decisions at pass s, including preceding commitments and current readouts. Active tokens exit independently conditional on $\mathcal{F}_s$, with probabilities $\lambda_i^{(s)}$; any remaining tokens exit deterministically at K. Define token i's accumulated decision log-probability by

$$\begin{aligned} u_i(\mathbf{z}) &= \sum_{s < z_i} \log(1 - \lambda_i^{(s)}) + \mathbf{1}\{z_i < K\} \log \lambda_i^{(z_i)}, \\ \log q_\phi(\mathbf{z} \mid \mathbf{x}_t, \mathbf{y}_{\mathcal{M}_t}) &= \sum_{i \in \mathcal{M}_t} u_i(\mathbf{z}). \end{aligned} \tag{16}$$

The hazards are evaluated along the realized history. Earlier exits can change later hazards for other tokens, so this chain-rule identity does not assume independent exit depths or identify u_i with a marginal log-probability. Normalized decisions and forced final exit make q_ϕ a normalized distribution over the finite decision tree. We assume differentiable probabilities on a fixed support, as obtained with finite sigmoid logits.

Derivative convention. To formalize the detached gate inputs in Eq. (2), introduce a reference backbone $\bar{\theta}$ for the policy and write

$$J_{\bar{\theta}}(\theta, \phi) = \mathbb{E}_{q_{\bar{\theta},\phi}}\left[\mathcal{L}_{\mathrm{traj},\theta}(\mathbf{z}) + \log \frac{q_{\bar{\theta},\phi}(\mathbf{z})}{\pi(\mathbf{z})}\right]. \tag{17}$$

The forward pass uses $\bar{\theta} = \theta$, but the denoiser derivative holds $\bar{\theta}$ fixed. Thus $\nabla_\theta J$ below denotes $\partial_\theta J_{\bar{\theta}}|_{\bar{\theta}=\theta}$, rather than a total derivative through the policy features. This notation requires no second backbone. At fixed decisions, the trajectory loss depends on θ but not on the separate gate parameters ϕ; the prior is fixed with respect to both parameter groups.

Proof of Proposition 1. Fix $(\mathbf{y}, t, \mathbf{x}_t)$ and suppress the reference backbone and conditioning arguments. Under the stated derivative convention,

$$\nabla_\theta J = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \nabla_\theta \mathcal{L}_{\text{traj}}(\mathbf{z}) = \mathbb{E}_{q_\phi}[\nabla_\theta \mathcal{L}_{\text{traj}}]. \tag{18}$$

For the gate parameters, differentiation gives

$$\begin{aligned} \nabla_\phi J &= \sum_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}) \left[\mathcal{L}_{\text{traj}}(\mathbf{z}) + \log \frac{q_\phi(\mathbf{z})}{\pi(\mathbf{z})} \right] + \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}) \\ &= \mathbb{E}_{q_\phi} \left[\left(\mathcal{L}_{\text{traj}} + \log \frac{q_\phi}{\pi} \right) \nabla_\phi \log q_\phi \right], \end{aligned} \tag{19}$$

where the last sum vanishes because

$$\mathbb{E}_{q_\phi}[\nabla_\phi \log q_\phi] = \sum_{\mathbf{z}} \nabla_\phi q_\phi(\mathbf{z}) = \nabla_\phi 1 = 0. \tag{20}$$

With the sampled decisions fixed and the cost detached, Eq. (7) has derivatives

$$\begin{aligned} \nabla_\theta \widehat{\mathcal{L}}_{\text{sur}} &= \frac{1}{t} \nabla_\theta \mathcal{L}_{\text{traj}}, \\ \nabla_\phi \widehat{\mathcal{L}}_{\text{sur}} &= \frac{1}{t} \left(\mathcal{L}_{\text{traj}} + \log \frac{q_\phi}{\pi} \right) \nabla_\phi \log q_\phi. \end{aligned} \tag{21}$$

Their schedule expectations are $(1/t)\nabla_{\theta,\phi} J$. Let μ denote the parameter-independent law of the clean data, corruption level, and corrupted input in Eq. (6). Assuming the $1/t$-weighted gradients are integrable and differentiation may pass through the outer expectation, the tower property gives

$$\mathbb{E}_{\mu, q_\phi}[\nabla_{\theta,\phi} \widehat{\mathcal{L}}_{\text{sur}}] = \mathbb{E}_\mu \left[\frac{1}{t} \nabla_{\theta,\phi} J \right] = \nabla_{\theta,\phi} \mathcal{L}_{\text{train}}. \tag{22}$$

The denoiser derivative throughout retains the fixed-policy meaning defined above. □

C.3 Geometric prior and depth regularization

The geometric prior in Eq. (5) is truncated and renormalized over $\{1, \dots, K\}$:

$$\pi_{0,c}(d) = \frac{e^{-cd}}{\sum_{r=1}^{K} e^{-cr}}, \qquad \pi_c(\mathbf{z}) = \prod_{i \in \mathcal{M}_t} \pi_{0,c}(z_i). \tag{23}$$

Here, $c > 0$ controls the preference for shallow exits. We use $K = 4$ and $c = 0.4$, giving $\pi_{0,c} \approx (0.4131, 0.2769, 0.1856, 0.1244)$ and a prior mean depth $\mathbb{E}_{D \sim \pi_{0,c}}[D] \approx 2.02$. Larger c favors shallower depths; the learned mean depth also depends on the prediction objective.

Separating individual and aggregate regularization. For $M > 0$ supervised masked positions with fixed positive depth distributions r_i, write $\bar{r} = M^{-1} \sum_i r_i$. Adding and subtracting $\log \bar{r}$ gives

$$\begin{aligned} R_{\alpha,\beta} &= \frac{\alpha}{M} \sum_i D_{\text{KL}}(r_i \| \pi_{0,c}) + (\beta - \alpha) D_{\text{KL}}(\bar{r} \| \pi_{0,c}) \\ &= \frac{\alpha}{M} \sum_i D_{\text{KL}}(r_i \| \bar{r}) + \beta D_{\text{KL}}(\bar{r} \| \pi_{0,c}). \end{aligned} \tag{24}$$

The first term in the second line penalizes differences among token profiles; the second matches their average to the prior. We use $(\alpha, \beta) = (0.1, 1)$ to allow tokens to favor different depths while retaining stronger aggregate prior matching. For samples with marginals $z_i \sim r_i$, this objective equals $\mathbb{E}[M^{-1} \sum_i \widetilde{C}_i]$, where

$$\widetilde{C}_i = \alpha \log \frac{r_i(z_i)}{\bar{r}(z_i)} + \beta \log \frac{\bar{r}(z_i)}{\pi_{0,c}(z_i)}. \tag{25}$$

The sampled regularization cost. Let $\mathbf{Z} = (\mathbf{z}_b)_b$ collect the sampled schedules in a local minibatch, and let M count its supervised masked positions. Using gate outputs along these rollouts, form

$$\widehat{r}_i(d) = \begin{cases} \lambda_i^{(d)} \prod_{s<d} (1 - \lambda_i^{(s)}), & d < K, \\ \prod_{s<K} (1 - \lambda_i^{(s)}), & d = K, \end{cases} \qquad b_{\mathbf{Z}}(d) = \frac{1}{M} \sum_i \widehat{r}_i(d). \tag{26}$$

These full-depth profiles require gate readouts at all nonterminal depths $s < K$, including after commitment. They summarize the realized rollouts rather than exact marginal probabilities over alternative histories. Replacing $\log r_i(z_i)$ by the realized decision score u_i in Eq. (16) and $\bar{r}$ by $b_{\mathbf{Z}}$ gives

$$C_i^{\alpha,\beta}(\mathbf{Z}) = \alpha[u_i - \log b_{\mathbf{Z}}(z_i)] + \beta[\log b_{\mathbf{Z}}(z_i) - \log \pi_{0,c}(z_i)]. \tag{27}$$

The cost is detached inside the sequence advantage. Because the profiles depend on the sampled history, its expectation need not equal $R_{\alpha,\beta}$; it is a rollout-based relaxation of that design.

Theorem 3 (Unit-weight equivalence)**.** *Fix a corrupted minibatch. For any sequence b with supervised positions $\mathcal{M}_b$, $\alpha = \beta = 1$, and any positive aggregate profile,*

$$\mathbb{E}_{\mathbf{Z}}\left[\sum_{i\in\mathcal{M}_b} C_i^{1,1}(\mathbf{Z})\right] = D_{\mathrm{KL}}(q_\phi(\mathbf{z}_b)\|\pi_c(\mathbf{z}_b))\,. \tag{28}$$

Independence of the exit depths within the sequence is not required.

Proof. The aggregate terms cancel for every realized minibatch. By Eq. (16) and the product prior,

$$\sum_{i\in\mathcal{M}_b} C_i^{1,1}(\mathbf{Z}) = \sum_{i\in\mathcal{M}_b} [u_i - \log \pi_{0,c}(z_i)] = \log \frac{q_\phi(\mathbf{z}_b)}{\pi_c(\mathbf{z}_b)}.$$

Taking the expectation proves the claim. □

Thus unit coefficients recover the original joint KL in expectation, whereas $(0.1, 1)$ gives a rollout-based relaxation with weaker regularization of individual variation. In the first line of Eq. (24), these settings correspond to KL weights $(1, 0)$ and $(0.1, 0.9)$, respectively.

C.4 Variance reduction and the training surrogate

First-pass control variate. For each independently attended sequence b, let $\mathcal{M}_b$ contain its supervised masked positions and let $M = \sum_b |\mathcal{M}_b| > 0$. Let $\ell_i^{(s)}$ denote the cross-entropy loss for token i at pass s. Define the sequence cost and baseline as

$$R_b = \sum_{i\in\mathcal{M}_b} [\ell_i^{(z_i)} + C_i^{\alpha,\beta}(\mathbf{Z})], \qquad B_b = \sum_{i\in\mathcal{M}_b} \ell_i^{(1)}. \tag{29}$$

The first-pass readout precedes every sampled commitment. Conditional on the corrupted inputs and current parameters, B_b is therefore independent of the sampled schedule, and

$$\mathbb{E}_{q_\phi}[B_b \nabla_\phi \log q_\phi(\mathbf{z}_b)] = B_b \nabla_\phi \sum_{\mathbf{z}_b} q_\phi(\mathbf{z}_b) = 0. \tag{30}$$

Subtracting this detached baseline leaves the expected gate update unchanged. The resulting gate surrogate (actor) is

$$\widehat{\mathcal{L}}_{\mathrm{gate}} = \frac{1}{M}\sum_b \mathrm{sg}[R_b - B_b] \log q_\phi(\mathbf{z}_b). \tag{31}$$

Returns and decision scores are summed within each independently attended sequence, including separate segments of packed examples. There is no current-minibatch advantage centering. The baseline supplies an input-dependent reference without an extra forward pass; its variance benefit depends on how well it tracks the sampled sequence cost.

Intermediate denoiser supervision. For a token exiting at z_i, let $v_i^{(s)} = \prod_{r<s}(1 - \lambda_i^{(r)})$ and define

$$\omega_i^{(s)} = \begin{cases} v_i^{(s)}\lambda_i^{(s)}, & s < z_i, \\ v_i^{(s)}, & s = z_i, \\ 0, & s > z_i. \end{cases} \qquad \sum_{s\le z_i} \omega_i^{(s)} = 1. \tag{32}$$

These weights assign the remaining probability mass to the sampled exit and exclude readouts after target feedback. For the single-t corruption used in the theoretical objective, detaching these weights gives the per-example denoiser loss

$$\widehat{\mathcal{L}}_{\text{den}} = \frac{1}{t} \sum_{i \in \mathcal{M}_t} \sum_{s \le z_i} \text{sg}[\omega_i^{(s)}] \ell_i^{(s)}. \tag{33}$$

Under blockwise corruption, each token instead uses its block's masking probability. Minibatch normalization follows the implementation described in Appendix D.1. This reuses pre-commitment predictions to train the denoiser; the gate cost in Eq. (29) still uses each token's sampled-exit loss. The weighted denoiser objective need not have the same expected gradient as sampled-exit supervision. We assess its variance empirically: Figure 5 (right) compares the trace of the gradient covariance for the measured parameter subset, using fixed corrupted inputs and paired sampled trajectories. This measures variation from trajectory sampling rather than from sampling training examples.

Complete training surrogate. After aggregating and normalizing the denoiser loss as described above, we combine it with the gate surrogate and an auxiliary AR loss $\widehat{\mathcal{L}}_{\text{AR}}$. This auxiliary loss is computed on the same sequences using standard next-token prediction, which has been shown to be beneficial for AR-to-DLM conversion (Liu et al., 2025a; Fu et al., 2026; Tian et al., 2025; Gat et al., 2025). With equal diffusion/AR weighting, the complete surrogate is:

$$\widehat{\mathcal{L}}_{\text{impl}} = \widehat{\mathcal{L}}_{\text{den}} + \widehat{\mathcal{L}}_{\text{gate}} + \widehat{\mathcal{L}}_{\text{AR}}. \tag{34}$$

The actor uses sequence-summed costs with a single $1/M$ normalization; inverse-mask-probability weighting applies only to the denoiser. This surrogate combines the sequence-level credit structure of Proposition 1 with relaxed depth regularization and intermediate supervision. The proposition's unbiasedness result concerns the original joint-KL objective.

D Experimental Details

This appendix describes the model configurations, optimization settings, and evaluation protocols used in Section 4. We distinguish the single-token decoding protocol used to compare generation quality from the parallel decoding protocol used to measure the quality–efficiency tradeoff.

D.1 Models and training

Architecture. We initialize ALoDLM-1.7B and ALoDLM-8B from the corresponding Qwen3 backbones (Yang et al., 2025). The 1.7B model applies its complete 28-layer Transformer stack recurrently. The 8B model partitions its 36 layers into a 10-layer prelude, a shared 16-layer recurrent core, and a 10-layer coda. Layer ranges below use zero-based, half-open indexing. Both configurations use a maximum recurrent depth of four and untied input and output embeddings. The coda and output head produce a readout at each recurrent pass; the coda is therefore re-evaluated when the core recurs.

Data and corruption. Both models are converted by supervised fine-tuning, without a separate continued-pretraining stage, using the WeDLM block-diffusion framework (Liu et al., 2025a) on the 5B-token corpus described in the main text. Training examples are formatted as conversations, packed into 4,096-token sequences, and supervised on the completion tokens; prompt tokens provide context. Each 32-token block receives an independently sampled masking probability in $[0, 1]$, and the denoiser losses use inverse-mask-probability weights normalized over the local minibatch's supervised masked positions. The outcome actor uses sequence-summed costs with one masked-token-count normalization, as specified in Appendix C.4. The training recipe also includes a next-token prediction auxiliary loss with the same relative weight as the diffusion loss.

Optimization. Table 2 lists the settings shared across the two scales. Both use AdamW (Loshchilov & Hutter, 2019) and DeepSpeed ZeRO-2. The 8B schedule runs for eight epochs (34,512 optimizer steps) and uses activation checkpointing on every second recurrent-layer invocation; the 1.7B configuration does not require it. For latent-depth training, each rollout samples commitments from the learned gate and feeds the committed target embeddings into later passes. The outcome actor, first-pass control variate, and complete training surrogate are given in Appendix C.4.

Table 2: Model configurations and common optimization settings. A batch contains 256 packed examples across all workers and accumulation steps.

Setting	ALoDLM-1.7B	ALoDLM-8B
Transformer layers	28	36
Prelude layers	None	$[0, 10)$
Recurrent core	$[0, 28)$	$[10, 26)$
Coda layers	None	$[26, 36)$
Maximum recurrent depth	4	4
Packed sequence length	4,096	4,096
Training block size	32	32
Global batch size	256	256
Optimizer	AdamW	AdamW
Peak learning rate	10^{-5}	10^{-5}
Learning-rate schedule	Cosine	Cosine
Warmup fraction	0.03	0.03
Weight decay	0.01	0.01
Gradient clipping norm	1.0	1.0
Arithmetic	bfloat16	bfloat16

D.2 Generation quality and inference efficiency

Generation quality. The benchmark comparison in Table 1 follows the OpenCompass protocol (OpenCompass Contributors, 2023) adopted by SDAR (Cheng et al., 2025). We apply each model's native chat template, use greedy token selection, and cap generation at 4,096 tokens. ARC-Easy, ARC-Challenge, MMLU, MMLU-Pro, and GPQA-Diamond measure general and scientific reasoning; GSM8K and MATH-500 measure mathematical reasoning; HumanEval, sanitized MBPP, and their EvalPlus variants measure code generation. We report accuracy or exact match for non-code benchmarks and execution-based pass@1 for code. The table's average is the unweighted mean of its eleven benchmark scores.

For this quality comparison, each model pass commits one token using each method's default decoding rule. ALoDLM uses a window size of 16 and evaluates the mean cumulative halting criterion on the remaining positions, with $q = 0.4$ at 1.7B and $q = 0.5$ at 8B. This preset disables entropy-based selection, so τ is inactive; a single denoising step can still commit multiple tokens over successive recurrent passes. Diffusion baselines use their single-token presets, including uncertainty-based position selection where supported.

Inference efficiency. Figures 1 and 3 use parallel commitment and single-stream generation on one NVIDIA B200. ALoDLM and WeDLM use their native CUDA-graph engines; Qwen3 uses vLLM (Kwon et al., 2023). SDAR uses JetEngine, LLaDA uses dInfer (Ma et al., 2025), Fast-dLLM-v2 uses its native engine, and Dream uses a compiled Fast-dLLM-v1 implementation with prefix caching (Wu et al., 2025b). Generation timing includes prefill and decoding, with device synchronization, and excludes input tokenization and answer scoring. Dataset throughput is the total generated-token count divided by the summed generation time, rather than the mean of per-example rates. The q, τ sweeps use a 16-token window and position coefficient $\gamma = 0.02$; the quality preset above and this entropy-based parallel decoder are separate operating modes.

Arithmetic cost. We estimate floating-point operations from execution records and model dimensions. For each request, let N_{layer} count token–Transformer-layer evaluations, N_{head} count vocabulary readouts, and N_{gen} count generated tokens. The estimated cost per generated token is

$$\text{FLOPs/token} = \frac{2P_{\text{layer}}N_{\text{layer}} + 2P_{\text{head}}N_{\text{head}} + 4Hd_{\text{head}}\bar{L}N_{\text{layer}}}{N_{\text{gen}}}, \tag{35}$$

where P_{layer} counts dense projection parameters in one Transformer layer, P_{head} counts output-head parameters, H is the number of query heads, d_{head} is the head dimension, and $\bar{L}$ is approximated by the prompt length plus half the generated length. Dataset-level cost is total estimated FLOPs divided by total generated tokens. Both ALoDLM-8B and WeDLM-8B have hidden dimension 4,096,

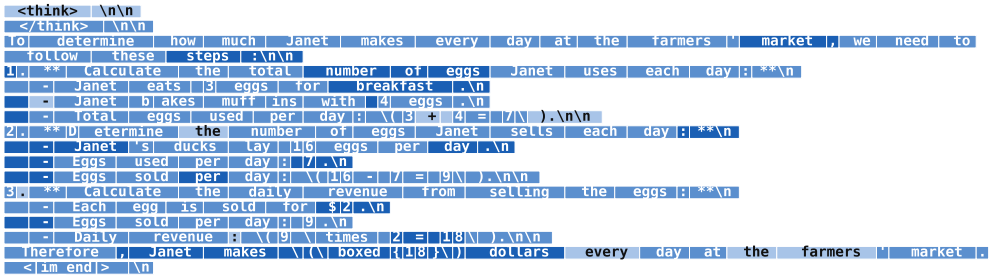
feed-forward dimension 12,288, 32 query heads, eight KV heads, head dimension 128, and vocabulary size 151,936, giving approximately 192.9M dense parameters per layer and 622.3M output-head parameters. A multiply and an add count as two floating-point operations.

The estimate includes prefill, evaluated window positions, CUDA-graph padding, and executed vocabulary readouts. For WeDLM, whole-forward graph replay bypasses host-side layer-counter updates, so executed work is reconstructed from recorded prefill lengths, decoding-step counts, and graph shapes. The attention term uses the context-length approximation above; minor elementwise operations are omitted. For reference, a complete K-pass 8B denoising invocation evaluates $10 + K(16 + 10)$ Transformer layers per position: 62, 114, and 218 at $K = 2, 4, 8$, respectively.

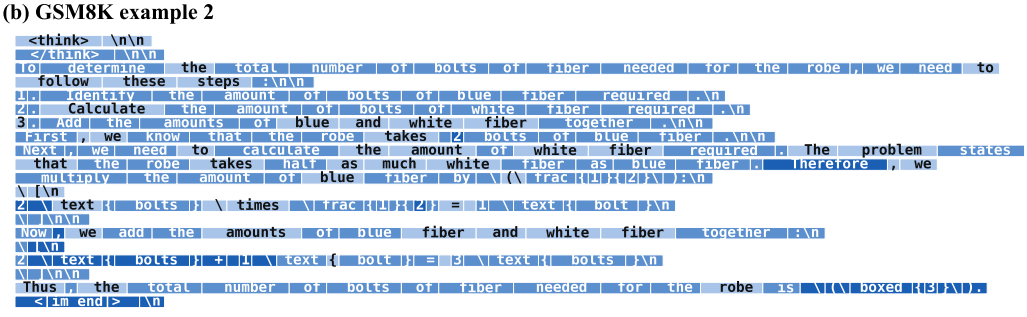
E CASE STUDIES

Aggregate depth statistics summarize computation across many responses. Figure 6 gives a complementary view of three individual GSM8K generations from our decoder. Tokens committed at different recurrent passes are interleaved within each response, illustrating how commitment depth varies across positions. The examples visualize these local depth assignments; the aggregate allocation analysis is reported in Figure 4.

(a) GSM8K example 1



(b) GSM8K example 2



(c) GSM8K example 3

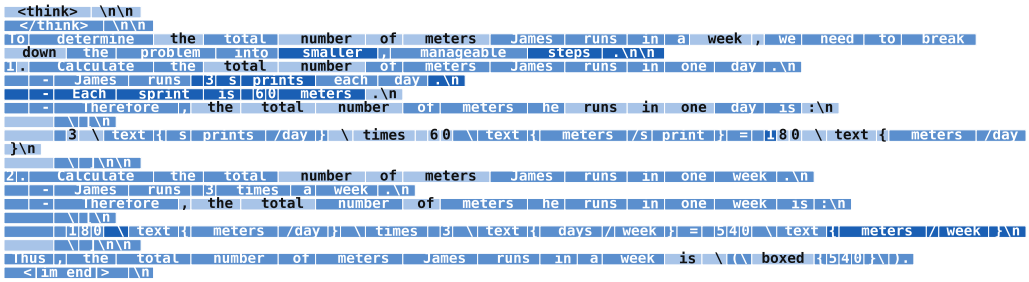




Figure 6: Token-wise recurrent depth in three GSM8K responses generated by ALoDLM. Each token is shaded by its recorded commitment pass; darker shades indicate later commitment within the corresponding denoising step. These examples use $K = 4$, $q = 0.5$, $\tau = 0.4$, and a 16-token decoding window.