Diffusion Language Model Inference with Monte Carlo Tree Search

Zheng Huang[1,2*] **Kiran Ramnath**[1,†] **Yueyan Chen**[1,†] **Aosong Feng**[1]
Sangmin Woo[1] **Balasubramaniam Srinivasan**[1] **Zhichao Xu**[1]
Kang Zhou[1] **Shuai Wang**[1] **Haibo Ding**[1] **Lin Lee Cheong**[1]
[1]AWS AI Labs, [2]Dartmouth College

Abstract

Diffusion language models (DLMs) have recently emerged as a compelling alternative to autoregressive generation, offering parallel generation and improved global coherence. During inference, DLMs generate text by iteratively denoising masked sequences in parallel; however, determining which positions to unmask and which tokens to commit forms a large combinatorial search problem. Existing inference methods approximate this search using heuristics, which often yield suboptimal decoding paths; other approaches instead rely on additional training to guide token selection. To introduce a principled search mechanism for DLMs inference, we introduce **MEDAL**, an inference-time scaling framework that integrates **M**onte Carlo Tree S**E**arch initialization for **D**iffusion L**A**nguage Mode**l** inference. We employ Monte Carlo Tree Search at the initialization stage to explore promising unmasking trajectories, providing a robust starting point for subsequent refinement. This design enables efficient inference-time scaling, allowing generation quality to improve as the search budget increases, without additional training. Across multiple benchmarks, **MEDAL** achieves up to 22.0% improvement over existing inference strategies, establishing a new paradigm for search-based inference in DLMs.

1 Introduction

In recent years, diffusion language models (DLMs) have emerged as a powerful alternative for generative modeling over discrete sequences (Zhu et al., 2025; Ye et al., 2025). Unlike autoregressive (AR) models (Achiam et al., 2023; Minaee et al., 2024), which rely on a strictly left-to-right factorization, DLMs learn to invert a stochastic corruption process that independently masks tokens (Lou et al., 2023; Shi et al., 2024; Nie et al., 2025). This formulation enables parallel refinement, improves global coherence, and offers flexible quality-latency trade-offs, thus challenging the dominance of AR paradigms (Li et al., 2025).

The inference process of DLMs can be naturally formulated as a search problem: starting from a corrupted sequence, the model iteratively decides which positions to unmask and which tokens to assign, navigating an exponential space of possible trajectories. Existing inference-time methods approximate this search using confidence-driven heuristics, such as greedily unmasking the highest-confidence tokens (Kim et al., 2025; Ben-Hamu et al., 2025; Luxembourg et al., 2025). Although effective in reducing short-term uncertainty, these strategies are inherently myopic: once high-confidence tokens are fixed, subsequent steps are forced to adapt around them, often leading to suboptimal trajectories. Another line of work adjusts masking schedules dynamically (Peng et al., 2025; Zhao et al., 2024), but such approaches typically require training auxiliary samplers to determine token updates at each step, introducing additional complexity and limiting general applicability.

These limitations highlight the need for a principled search mechanism that can explore alternative unmasking trajectories without additional training overhead. To this end, we propose **MEDAL**, a framework that integrates **M**onte Carlo Tree S**E**arch initialization for **D**iffusion L**A**nguage Mode**l** inference. Unlike heuristic or schedule-based methods, which either commit to tokens greedily or depend on auxiliary samplers, **MEDAL** adopts a principled search-based approach, using Monte Carlo Tree Search (MCTS) (Yoon et al., 2025; Browne et al., 2012) to balance exploitation of high-confidence tokens with exploration of alternative unmasking trajectories relying on signals obtained from the model's own distribution. We propose two key innovations that enable MCTS to be applied effectively to DLM inference. First,

[*]Work done while interning at AWS AI Labs.[†]Correspondence: zheng.huang.gr@dartmouth.edu, raxkiran@amazon.com, yyanc@amazon.com.

Findings of the Association for Computational Linguistics: EACL 2026, pages 3493–3512
March 24-29, 2026

we introduce a confidence-guided filtering mechanism that focuses inference on the most promising tokens and positions. Second, we design an information-gain reward that guides MCTS by favoring token choices that not only resolve the current position but also increase the model's confidence in predicting the remaining tokens. Together, these innovations make it possible to enable the four stages of MCTS, Selection, Expansion, Simulation, and Backpropagation, within DLMs. Rather than applying MCTS exhaustively, we employ it strategically in the early stages of inference to construct a robust initialization, after which the process continues with efficient heuristics. To further address complex prompts that induce high uncertainty, we incorporate a task-decomposition module that automatically splits the input into smaller subtasks, thereby reducing ambiguity and providing structured guidance for subsequent unmasking decisions. Our contributions are summarized as follows:

- **Novel Formulation**: We frame DLM inference as a search problem and introduce **MEDAL**, the first framework to integrate MCTS into DLM inference, enabling principled exploration beyond greedy heuristics or schedule-based methods
- **Novel Design**: We design a new DLM inference approach that combines an MCTS-guided initialization module with a task-decomposition module, enabling both efficient search and improved handling of complex tasks.
- **Extensive Experiments**: We conduct evaluations on various benchmarks, demonstrating that our method outperforms existing inference strategies for DLMs by up to 22.0% when restricting MCTS to initialization. We show that generation quality continues to improve even further with diminishing gains as the MCTS initialization budget increases, validating the effectiveness of our search-based approach.

2 Preliminary

2.1 Discrete Diffusion Language Models

DLMs adapt the diffusion paradigm from continuous domains (e.g., image generation) to discrete text sequences. Let $x_0 = (x_0^1, \ldots, x_0^L)$ denote a token sequence of length L sampled from the data distribution. The core idea is to define a forward noising process that progressively corrupts x_0 into increasingly noisy sequences $\{x_t\}_{t=1}^T$, and to train a neural model to learn the corresponding reverse denoising process that reconstructs x_0 from noise.

Forward process. In discrete DLMs, the forward process is typically defined by a time-dependent transition matrix Q_t over the vocabulary (Li et al., 2025). At each time t, the probability of a state x_t given an initial state x_0 is given by a categorical distribution:

$$q(x_t \mid x_0) = \text{Cat}(x_t; x_0 \overline{Q}_t), \quad \overline{Q}_t = \prod_{i=1}^{t} Q_i. \quad (1)$$

Reverse Process The reverse process learns to invert the corruption by predicting the original token distribution given a corrupted sequence:

$$p_\theta(x_{t-1} \mid x_t) = \prod_{i \in \mathcal{U}_t} p_\theta(x_{t-1}^i \mid x_t, t). \quad (2)$$

Here, p_θ is parameterized by a transformer, trained to minimize a cross-entropy objective over masked positions, and $\mathcal{U}_t$ denotes the set of positions unmasked at timestep t.

Generation. During inference, generation begins from a fully masked sequence:

$$x_T = (\texttt{[MASK]}, \ldots, \texttt{[MASK]}). \quad (3)$$

At each denoising step t, the model outputs a distribution over the vocabulary for every position. A subset of tokens with the highest confidence is selected, unmasked, and fixed, while the remaining positions stay masked. The process then advances to step $t-1$, where the model re-predicts distributions conditioned on both the fixed tokens and the still-masked positions (Zhu et al., 2025; Nie et al., 2025). This iterative refinement continues until all positions are resolved, yielding the final sequence x_0. The generation can be viewed as a sequence of partially completed states, where the model progressively transitions from a fully masked input to a coherent, fully unmasked output.

2.2 Monte Carlo Tree Search

MCTS is a general algorithm for decision-making in large combinatorial search spaces, where exhaustive enumeration of all trajectories is infeasible (Browne et al., 2012). The goal of MCTS

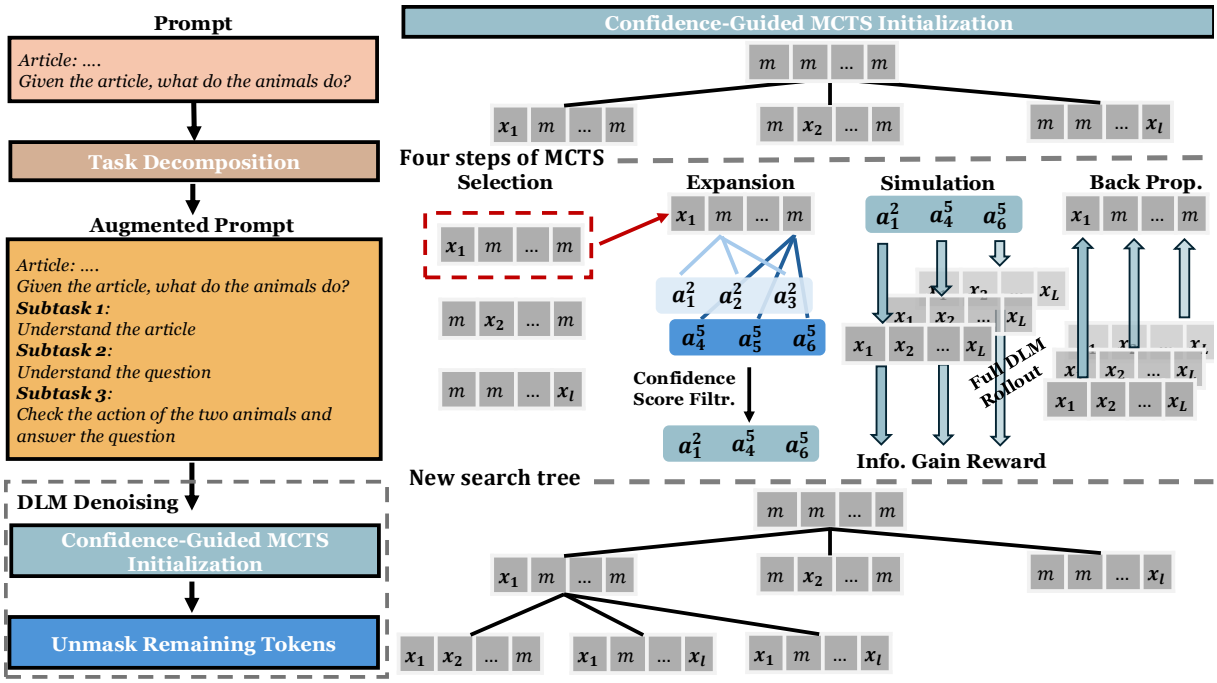

Figure 1: Overview of the **MEDAL** framework. It consists of an MCTS-guided initialization module and a task-decomposition module, enabling both efficient search and improved handling of complex tasks. The root node represents the initial fully masked input, and leaf nodes correspond to partially unmasked sequences. The MCTS involves committing to a specific generation decision, expressed as an action $a_v^i = (i, v)$, where i identifies the masked position to reveal and v denotes the token selected for that position.

is to evaluate possible decision paths efficiently, balancing the exploration of new options with the refinement of promising ones. The search tree begins at the root node, representing the initial state of the problem, and grows toward leaf nodes, which correspond to unexplored frontier states. Each iteration of MCTS consists of four steps: (i) Selection: choose the node from the root to a leaf according to a selection policy, such as the upper confidence bound (UCB); (ii) Expansion: add one or more child nodes at the leaf; (iii) Simulation: evaluate the newly expanded node by performing rollouts or applying heuristic approximations; (iv) Back propagation: propagate the result upward to update statistics of visited nodes.

3 Method

In this section, we present our framework **MEDAL** for enhancing DLM inference (Figure 1). We start with the notation used throughout the section. Second, we introduce our key contributions: confidence score filtering (Section 3.1.1) and information-gain reward (Section 3.1.2) that enable MCTS to be effectively applied to DLMs. Building on these foundations, we then introduce our MCTS-guided initialization strategy to efficiently explore promising candidates during the early stages of generation. Finally, we describe how task decomposition via prompt guidance can further reduce uncertainty and improve generation quality.

Notation. We use $\mathcal{V}$ for the vocabulary (size $|\mathcal{V}|$); L for target sequence length; $x = (x^1, \ldots, x^L)$ for a (partially) formed sequence; x^i the token at position i; $x_{\setminus i}$ the sequence with position i masked; $t \in \{0, \ldots, T\}$ the reverse denoising step index with T total steps; $\mathcal{M}_t$ the set of masked positions at step t.

3.1 MCTS for DLMs Inference

To bridge MCTS with DLMs, we construct the search tree over unmasked sequences, where the root corresponds to the initial masked input, and leaves as partially unmasked sequences.

3.1.1 Confidence Score Filtering

The classical MCTS aims to balance exploitation and exploration, but in DLMs the search space spans the full vocabulary at every masked position, making naïve search intractable. To address this, we introduce a confidence filtering strategy that restricts the search space of masked positions and tokens to a far smaller action set. We define the

action as a position–token pair $a_v^i = (i, v)$, where i is a masked position and v is the token-candidate chosen, and it represents a specific unmasking decision of the search precess. Next, we describe how to build the action set.

Given the model's logits $\ell^i \in \mathbb{R}^{|\mathcal{V}|}$ of an input sequence at a masked position i, we first convert ℓ^i into probabilities

$$p^i(v) = \frac{\exp(\ell_v^i)}{\sum_{w \in \mathcal{V}} \exp(\ell_w^i)}, \qquad v^i \in \mathcal{V}. \quad (4)$$

For a candidate token v at position i, we then define the confidence-adjusted score as:

$$s_v^i = p^i(v) \cdot \phi_{\text{ent}}^i \cdot \phi_{\text{mar}}^i, \quad (5)$$

where ϕ_{ent}^i and ϕ_{mar}^i are respectively the entropy penalty and top-2 margin, defined below:

- **Entropy Penalty.** We use $H^i = -\sum_{v \in \mathcal{V}} p^i(v) \log(p^i(v) + \varepsilon)$, with $\phi_{\text{ent}}^i = \exp(-H^i) \in (0, 1]$, where ε is a small constant ensuring numerical stability. Positions with higher entropy (low confidence) are down-weighted.
- **Top-2 Margin.** Let $p_{(1)}^i \geq p_{(2)}^i$ be the top-two probabilities. The margin $\Delta^i = p_{(1)}^i - p_{(2)}^i$ is mapped to $\phi_{\text{mar}}^i = \sigma(\gamma \Delta^i)$ with γ is a hyperparameter and σ is a sigmoid function.

For each masked position $i \in \mathcal{M}_t$, we retain the top-K_1 candidate tokens ranked by the confidence-adjusted score s_v^i, forming an action candidate set:

$$\mathcal{A}_t^i = \{a_v^i : v \in \text{Top-}K_1(s_v^i)\} \quad (6)$$

$$\mathcal{A}_t = \bigcup_{i \in \mathcal{M}_t} \mathcal{A}_t^i. \quad (7)$$

From the action candidate set $\mathcal{A}_t$, we then select top-K_2 actions with the highest score:

$$\mathcal{A}_t' = arg\ \text{Top-}K_2\{s_v^i : a_v^i \in \mathcal{A}_t\} \quad (8)$$

The best K_2-scoring actions are then applied to the simulation of MCTS (Section 3.1.3).

3.1.2 Information-Gain Reward

Given the selected action set $\mathcal{A}_t'$, the next step is to evaluate the impact of each candidate action. An effective action should not only specify a token at one masked position, but also provide contextual information that improves the model's confidence over the remaining masked positions. In other words, a good choice at position i should make the subsequent predictions for $\mathcal{M}_t \setminus \{i\}$ more confident.

Our objective is therefore to design a reward function that quantifies how much an action improves the model's confidence in the unresolved positions. Formally, for an action $a_v^i \in \mathcal{A}_t'$, selecting token v to fill position i updates the masked set from $\mathcal{M}_t$ to $\mathcal{M}_t' = \mathcal{M}_t \setminus \{i\}$. We define the reward as the information-gain reward, which measures the entropy reduction across the remaining positions:

$$r_{\text{IG}}(a_v^i) = \frac{\sum_{j \in \mathcal{M}_t} H_\theta^{\text{before}}(j) - \sum_{j \in \mathcal{M}_t'} H_\theta^{\text{after}}(j)}{\sum_{j \in \mathcal{M}_t} H_\theta^{\text{before}}(j)}, \quad (9)$$

where $H_\theta(j)$ denotes the predictive entropy at position j. Position i is excluded from the second summation: once it has been filled, its entropy is always zero regardless of the token chosen, and

Algorithm 1 Confidence-Guided MCTS for DLMs Inference (CGMCTS)

Require: Prompt tokens $x_{1:|P|}$, target length L, MCTS init length $L_c < L$, action candidate token number K_1, the highest-scoring number K_2, candidate size C, margin scale γ.

Initialize root$\leftarrow [x_{1:|P|}] \oplus [\texttt{[MASK]}]^L$
while (CollectCandidates(root, C, L_c)) **do** ▷ Stop when collect C candidates with length L_c
  $x \leftarrow$ NodeSelection($root$) ▷ UCB selection
  Identify masked positions $\mathcal{M}_t$ in x
  for all $i \in \mathcal{M}_t$ **do**
    Compute distribution $p^i(v) = \text{softmax}(\ell^i)$
    Compute confidence factors: $\phi_{\text{ent}}^i = \exp(-H^i)$, $\phi_{\text{mar}}^i = \sigma(\gamma \Delta^i)$
    Score tokens: $s_v^i = p^i(v) \cdot \phi_{\text{ent}}^i \cdot \phi_{\text{mar}}^i$
    Select candidates $\mathcal{A}_t^i \leftarrow \text{Top}-K_1(s_v^i)$
  end for
  Aggregate $\mathcal{A}_t = \bigcup_i \mathcal{A}_t^i$
  Build $\mathcal{A}_t' = arg\ \text{Top-}K_2\{s_v^i : a_v^i \in \mathcal{A}_t\}$
  for all $a \in \mathcal{A}_t'$ **do**
    $x_a \leftarrow$ Apply action a to x
    Node$\leftarrow$ ExpandNode(X_a)
    Rollout to obtain $r_{IG}(a)$
    Backpropagate $r_{IG}(a)$ ▷ Update search tree
  end for
end while
return C collected candidates with length L_c

including it would contribute only a constant term.

Finally, each action in $\mathcal{A}'_t$ is rolled out and assigned a reward value, which serves as the feedback signal to guide tree selection. This design ensures that the search prioritizes actions that not only fill a mask but also help improve model confidence for future predictions.

3.1.3 MCTS Steps

Building on the description above, we detail how the four traditional steps of MCTS—Selection, Expansion, Simulation, and Backpropagation—are adapted. The algorithm overview is in Algorithm 1, and an example is shown in Appendix A.3.

Selection. In this phase, we traverse the tree from the root to a leaf using UCB (Yoon et al., 2025; Kocsis and Szepesvári, 2006), selecting the child node that maximizes $Q(x,a) + c\sqrt{\frac{\ln N(x)}{N(x,a)}}$, where x represents the current partially masked sequence state, $Q(x,a)$ is the mean reward of action a, $N(x)$ is the visit count of node x, $N(x,a)$ is the visit count of the child node, and c is the exploration constant.

Expansion We build the action candidate set $\mathcal{A}'_t$ using confidence score filtering (Section 3.1.1). We then pass each action $a_v^i \in \mathcal{A}'_t$ to the simulation step to evaluate its reward.

Simulation. The simulation step evaluates the impact of each action $a_v^i \in \mathcal{A}'_t$ by rolling out the remaining masked positions by DLMs. Specifically, after applying action a_v^i to fill position i with token v, we let the DLM fill the remaining masked positions $\mathcal{M}'_t = \mathcal{M}_t \setminus \{i\}$ by sampling from the model's predicted distribution. This yields a completed sequence. The information-gain reward $r_{IG}(a_v^i)$ is then computed based on the entropy reduction across the remaining masked positions, as described in Section 3.1.2.

Backpropagation. After the simulation step, the reward obtained from evaluating the complete sequence is backpropagated through the tree to update the value estimates of all parent nodes along the path to the root.

3.2 Confidence-Guided MCTS Initialization

While conceptually appealing, applying MCTS throughout the entire generation is computationally prohibitive: the branching factor scales with vocabulary size $|\mathcal{V}|$ and sequence length L, and repeated rollouts over T denoising steps incur exponential cost. This leads to full-sequence MCTS being impractical for large-scale DLM inference. To balance search quality with efficiency, we restrict MCTS to the initialization phase. Specifically, we terminate the MCTS once a candidate set of C partially unmasked sequences, each of length L_c, has been formed. Each candidate is then fully rolled out by the DLM to compute its information-gain reward, and the sequence with the highest reward is selected as the final resolved output. Once a partially resolved sequence is obtained, the remaining tokens are filled without further tree search: at each step, we directly apply the confidence-adjusted score to select high-confidence tokens until no masks remain. This strategy enables test-time scaling during critical early decision

Algorithm 2 Overview of **MEDAL**

Require: Prompt tokens $x_{1:|P|}$, target length L, total steps T, MCTS init length $L_c{<}L$, action candidate token number K_1, the highest-scoring number K_2, candidate size C, margin scale γ.

Ensure: Completed sequence x

(A) Task decomposition

$\hat{P} \leftarrow \textsc{TaskDecompose}(x_{1:|P|})$ ▷ augment prompt with self-generated subtasks

$x \leftarrow [\hat{P}] \oplus [\texttt{[MASK]}]^L$

(B) MCTS-based initialization

Candidates$\leftarrow$ CGMCTS$(x, L, L_c, K_1, K_2, \gamma)$

$x \leftarrow$ SelectCandidate(Candidates) ▷ Select the one that has the highest info. gain reward

(C) Confidence-guided unmasking for remaining steps

for $t = T$ to 1 **do**

  Identify masked positions $\mathcal{M}_t$ in x; **if** $\mathcal{M}_t{=}\varnothing$ **then break**

  for all $i \in \mathcal{M}_t$ **do**

    Compute distribution $p^i(v) = \mathrm{softmax}(\ell^i)$

    Compute confidence factors $\phi^i_{\text{ent}} = \exp(-H^i)$, $\phi^i_{\text{mar}} = \sigma(\gamma\Delta^i)$

    Score tokens $s^i_v = p^i(v) \cdot \phi^i_{\text{ent}} \cdot \phi^i_{\text{mar}}$

  end for

  Sample action $a^i_v \propto \mathrm{Softmax}(s^i_v)$

  Apply action: $x[i] \leftarrow v$

end for

return x

stages through structured search over generation trajectories, while maintaining tractable inference via confidence-guided decoding in later stages.

3.3 Task Decomposition via Prompt Guidance

In this section, we present our method for decomposing complex tasks into simpler sub-tasks, improving confidence, and providing guidance for the DLM during reasoning.

Given an input prompt P, instead of asking the model to directly generate an answer, we guide the model to break down the problem into a series of manageable steps. Specifically, we provide the model with an augmented prompt $\hat{P}$ including illustrative two-shot examples showing how a complex question can be divided into a sequence of subtasks. Each subtask is framed with a distinct goal—such as understanding the input, identifying relevant information, or synthesizing the final response—so that guide the model to solve the problem in a structured manner.

At inference time, the model is encouraged to produce its own decomposition for the given input, guided by the example provided in the prompt $\hat{P}$. It then solves the subtasks sequentially, with intermediate outputs serving as context for subsequent steps. In summary, the full algorithm of **MEDAL** is shown in Algorithm 2.

3.4 Theoretical Guarantee

In this section, we provide a theoretical guarantee for **MEDAL**. At each decoding step i, when revealing a set z_i of tokens independently, the total error decomposes into a model term and a joint-dependence term $\mathrm{KL}(q(x_{z_i} \mid x_{z<i}) \,\|\, \prod_{\ell \in z_i} q(x_\ell \mid x_{z<i}))$. Following Ben-Hamu et al. (2025), this dependence error is upper-bounded by the entropy-gap surrogate $B(z_i) = \sum_{\ell \in z_i} H(p_\theta(x_\ell \mid x_{z<i})) - \max_{\ell \in z_i} H(p_\theta(x_\ell \mid x_{z<i}))$. Summing over K initialization steps yields $\sum_{i=1}^{K} \mathrm{DepErr}_i \leq \sum_{i=1}^{K} B(z_i)$. Our MCTS initialization explicitly minimizes the surrogate cost $J(z_{1:K}) = \sum_{i=1}^{K} B(z_i)$ across candidate schedules, thus selecting a prefix that achieves the smallest available upper bound on cumulative dependence error among explored options. This establishes that the proposed initialization is theoretically grounded: although we cannot minimize the true dependence error directly, MCTS optimizes a computable surrogate that provably controls it. A full derivation is in Appendix A.4.

4 Experiments

In this section, we conduct extensive experiments to evaluate **MEDAL**, guided by the following questions: **RQ1**: How does our proposed method perform on various datasets compared to state-of-the-art baselines? **RQ2**: What's the contribution of each component in our framework to the overall performance? **RQ3**: How does the model's performance vary with different hyperparameter settings?

4.1 Experimental Setup

We conduct experiments on six widely-used datasets: GSM8K (Cobbe et al., 2021), ARC-C (Clark et al., 2018), HumanEval (Chen et al., 2021), MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019) and Countdown (Pan et al., 2025). Experiments are conducted on models with a similar scale (7B-8B parameters) to ensure a fair comparison. We evaluate our method on three backbone DLMs, LLaDA-8B-Instruct (Nie et al., 2025), LLaDA1.5-8B (Zhu et al., 2025), and Dream-7B (Ye et al., 2025), and compare their performance against (i) the original models, (ii) the original models with a Best-of-5 decoding strategy (generating five samples and selecting the majority answer, bst5), and (iii) the original models with our method. We additionally report results for Llama-3 8B (Touvron et al., 2023) as an autoregressive LLM baseline. We use accuracy for GSM8K, ARC-C, and MMLU; Exact Match for Countdown; pass@1 for HumanEval; and F1 for DROP. For MCTS initialization on all datasets, we set the MCTS initialization candidate length L_c to 20, $K_1 = 3$, $K_2 = 5$, and the stability constant $\varepsilon = 10^{-8}$. The generation length for DLMs is 256. For task-decomposition prompting, we set the number of subtasks to 3. We set the candidate size $C = 3$. All the experiments are conducted on 2 NVIDIA A100 GPUs with 40GB memory, and the random seed is set to 1. Detailed settings are provided in Appendix A.1, and computational costs in Appendix A.2.

4.2 Effectiveness of MEDAL

We evaluate our method on the five datasets using three different backbone DLMs. The results are shown in Table 1. We observe that our method consistently improves the performance of all backbone models across all datasets, with up to 18.2% average and 8.1 absolute improvement, indicating the effectiveness and generality of our approach.

Model	GSM8K	ARC-C	HumanEval	MMLU	DROP	Countdown	Ave. Imprv.
LLaDA	58.3	72.2	40.2	36.0	58.2	15.6	—
+ Ours	66.7 Δ+8.4 (14.4% ↑)	82.1 Δ+9.9 (13.7% ↑)	47.5 Δ+7.3 (18.2% ↑)	44.0 Δ+7.0 (19.4% ↑)	71.0 Δ+12.8 (22.0% ↑)	18.9 Δ+3.3 (21.2% ↑)	Δ+8.1 (18.2% ↑)
+ Bst5	64.2 Δ+5.9 (10.1% ↑)	77.2 Δ+5 (6.9% ↑)	43.4 Δ+3.2 (8.0% ↑)	39.0 Δ+3.0 (8.3% ↑)	64.3 Δ+6.1 (10.4% ↑)	16.9 Δ+1.3 (8.3% ↑)	Δ+4.1 (8.7% ↑)
LLaDA1.5	62.3	74.9	45.0	37.3	60.5	16.8	—
+ Ours	69.0 Δ+6.7 (10.6% ↑)	82.8 Δ+7.9 (10.5% ↑)	51.0 Δ+6.0 (13.3% ↑)	44.5 Δ+7.2 (19.3% ↑)	71.1 Δ+10.6 (17.5% ↑)	19.2 Δ+2.4 (14.3% ↑)	Δ+6.8 (14.3% ↑)
+ Bst5	64.5 Δ+2.2 (3.5% ↑)	77.6 Δ+2.7 (3.6% ↑)	48.2 Δ+3.2 (7.1% ↑)	39.6 Δ+2.3 (6.2% ↑)	65.0 Δ+4.5 (7.4% ↑)	17.7 Δ+0.9 (5.4% ↑)	Δ+2.6 (5.6% ↑)
Dream	60.6	78.0	40.0	35.2	55.4	13.1	—
+ Ours	65.7 Δ+5.1 (8.4% ↑)	88.5 Δ+10.5 (13.5% ↑)	47.7 Δ+7.7 (19.3% ↑)	43.3 Δ+8.1 (23.0% ↑)	65.4 Δ+10.0 (18.1% ↑)	15.6 Δ+2.5 (19.1% ↑)	Δ+7.3 (16.9% ↑)
+ Bst5	62.9 Δ+2.3 (3.8% ↑)	80.0 Δ+2.0 (2.6% ↑)	43.1 Δ+3.1 (7.8% ↑)	38.1 Δ+2.9 (8.2% ↑)	57.7 Δ+2.3 (4.2% ↑)	15.4 Δ+1.3 (9.1% ↑)	Δ+2.3 (6.1% ↑)
Llama	59.2	70.5	45.7	44.5	60.0	3.2	—

Table 1: Results on using different backbone models across various benchmarks. For each setting, we report absolute improvements (Δ) and percentage gains relative to the corresponding backbone. The best improvement is highlighted in Dark Green. The average improvement is shown in the last column.

Method	ARC-C	HumanEval	DROP
Ours	82.1	47.5	71.0
W/o MCTS	81.0	44.3	64.9
W/o T. Dcp.	77.3	43.4	65.7
W/o Cf.+t2	75.0	40.9	60.1
Llada	72.2	40.2	58.2

Table 2: Ablation study results on ARC-C, HumanEval, and DROP using LLaDA as the backbone model.

Notably, even a model that underperforms Llama (i.e., Llada, which lags behind Llama on 4 out of 5 baselines in Table 1) achieves comparable or superior results on most datasets when equipped with our method. These results highlight the potential of DLMs when guided with appropriate strategies. The standard deviation is reported in Table 6.

4.3 Ablation Study

To understand the contribution of each component in our framework, we conduct ablation studies on ARC-C, HumanEval and DROP on backbone model LLaDA. To evaluate the impact of MCTS, we remove it from our framework and let the model generate answers directly based on the decomposed tasks (**W/o MCTS**). To assess the importance of task decomposition, we eliminate this step and have the model generate answers directly from the original prompt using MCTS (**W/o T. Dcp.**). Additionally, we remove the confidence-adjusted score and using only use top-2 margin (**W/o Cf.+t2**) to select the candidate tokens during MCTS.

The results are presented in Table 2. Overall, removing any component leads to a performance drop arcoss all datasets. Specifically, removing or replacing the confidence-adjusted score results in the largest performance drop, highlighting its critical role in selecting tokens to unmask. Moreover, eliminating MCTS also causes a performance drop, demonstrating its effectiveness in exploring multiple generation paths and selecting the most promising ones during initialization.

4.4 Hyperparameter Analysis

Tasks No.	ARC-C	HumanEval	DROP
1	79.8	44.0	69.2
3	82.1	47.5	71.0
5	76.5	42.9	62.8
10	75.5	41.9	58.0

Table 3: Performance with different numbers of decomposed tasks on ARC-C, HumanEval, and DROP using LLaDA as the backbone model.

In this section, we analyze the impact of key hyperparameters in our framework, including the number of decomposed tasks, the number of candidate tokens selected K_2 during MCTS, and MCTS initialization candidate length L_c. We vary the number of decomposed tasks and evaluate the performance on ARC-C, HumanEval, and DROP using LLaDA as the backbone model. The results are shown in Table 3. We observe that decomposing the task into 3 subtasks yields the best performance. Decomposing into too few (1) or too many (5 or 10) subtasks leads to a performance drop, indicating

MCTS Cand.	ARC-C	HumanEval	DROP
1	79.2	44.4	65.5
3	81.5	46.4	70.1
5	82.1	47.5	71.0
10	82.9	46.6	70.5

Table 4: Performance with different numbers of candidate tokens K_2 during MCTS on ARC-C, HumanEval, and DROP using LLaDA as the backbone model.

Method	DROP	MMLU
Ours + ADAS	73.0	46.5
LLada + ADAS	71.2	41.0
Llama + ADAS	65.2	45.2
LLada	58.2	36.0
Llama	60.0	44.5

Table 5: Results on DROP and MMLU using different methods in an agentic setting.

that an optimal level of decomposition is crucial for balancing complexity and guidance.

Then we vary the number of candidate tokens selected during MCTS and evaluate the performance on the same datasets. The results are shown in Table 4. We find that selecting 5 candidate tokens during MCTS achieves the best performance across all datasets. Selecting too few (1 or 3) candidate tokens would made the exploration space too small, limiting the potential of finding better generation paths. On the other hand, selecting too many (10) tokens would introduce too much noise, making it harder for the model to focus on the most promising paths. Therefore, we find that selecting 5 candidate tokens works well which is essential for effective exploration and exploitation during MCTS.

Finally, we vary the number of MCTS initialization candidate length L_c and evaluate the performance on data, including ARC-C, HumanEval, DROP and Countdown, for generating sentences of length 256. The results are shown in Figure 2. We observe that increasing L_c can improve the performance, and the performance gain tends to saturate after 20 steps. It indicates that using MCTS for initialization is sufficient, and it is a feasible approach for effective generation. This result suggests that the most critical generation trajectory decisions occur early, meaning that extended search beyond this point yields diminishing returns relative to the computational cost.

4.4.1 Case Study: Agentic Workflows

In this section, we present a case study exploring the potential of using DLMs in an agentic setting. We integrate our method with the ADAS (Hu et al., 2024), which utilizes an LLM to automatically invent building blocks and design powerful agentic systems to solve given tasks. We replace the LLMs in ADAS with our DLM (LLaDA with our method) and compare the performance with the original ADAS using LLaDA and Llama as backbones. The results on DROP and MMLU are shown in Table 5. We observe that integrating our method with ADAS leads to further performance improvement compared to using LLaDA and LLama, demonstrating our method's ability to enhance the reasoning and planning capabilities of DLMs in complex agentic settings. These findings suggest that DLMs, when equipped with effective strategies like ours, can be powerful tools for building intelligent agents capable of solving challenging tasks.

5 Related works

Diffusion Language Models. Diffusion models are a powerful class of generative models (Podell et al., 2023; Rombach et al., 2022; Li et al., 2025; Zhang et al., 2023). Building on their success, DLMs have emerged as promising alternatives for text generation. One line of work adapts continuous diffusion to discrete text via continuous relaxations (Li et al., 2022; Strudel et al., 2022), while another operates directly in the discrete token space, corrupting text by masking or token replacement (He et al., 2022; Austin et al., 2021). Leveraging mature scaling techniques, large-scale DLMs have been developed (Nie et al., 2025; Gong et al., 2024; Nie et al., 2024), achieving performance competitive with AR models (Touvron et al., 2023). Despite strong generative ability, DLMs often struggle with controllability and reasoning in deciding the token order to unmask tokens during inference. To address this, we propose a principled search-based framework with MCTS to enhance DLM generation capabilities.

Enhancing Reasoning Capabilities of Diffusion Language Models. To improve the reasoning performance of DLMs Kim et al. (2025) propose incorporating model uncertainty into the diffusion process to enhance generation quality. Entropy-based planning methods have been introduced (Ben-Hamu et al., 2025; Ye et al., 2025) to bet-

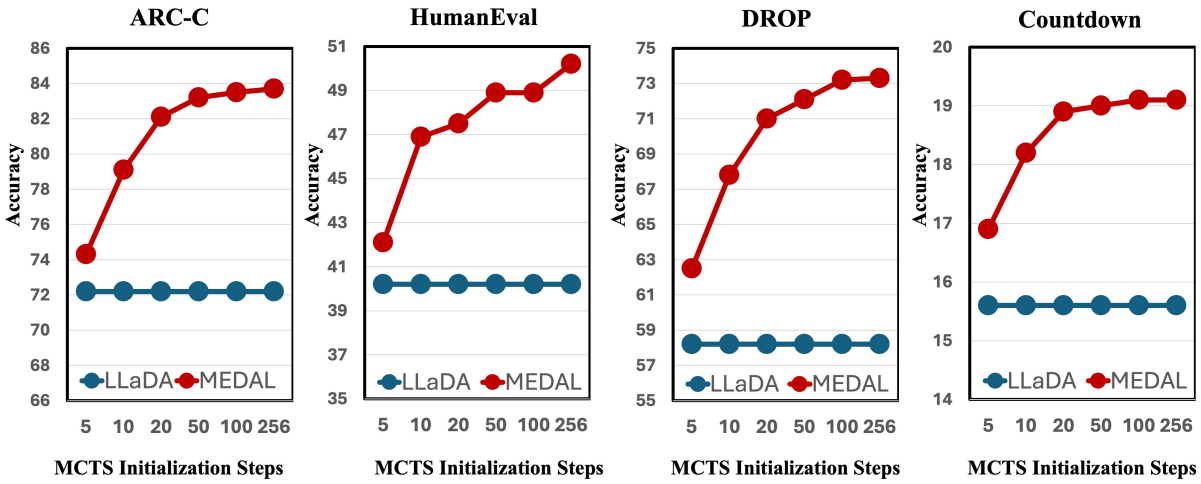

Figure 2: Performance of **MEDAL** (in red) with varying MCTS init length L_c on ARC-C, HumanEval, DROP and Countdown using LLaDA as the backbone model. The blue line denotes the performance of LLaDA without our method. The x-axis indicates the number of MCTS initialization steps, and the y-axis shows the accuracy.

ter capture the model's confidence. Planner-guided generation is studied in (Peng et al., 2025), enabling token-level refinement during sampling. More advanced reinforcement learning (RL) approaches further boost performance (Zekri and Boullé, 2025; Zhu et al., 2025; Gong et al., 2025). Particularly, Diffu-GRPO (Zhao et al., 2025) applies policy-gradient RL to DLMs via a mean-field approximation and prompt masking, while TraDo (Wang et al., 2025) aligns DLM inference trajectories with training objectives by optimizing the sampling process. Additionally, wd1 (Tang et al., 2025) reformulates the RL objective as a weighted likelihood to avoid biased policy ratios, improving reasoning accuracy. While effective, these methods rely on additional training that involves substantial computational cost and engineering overhead. In contrast, we enhance DLM reasoning purely at inference time, making our approach more efficient and practical for real-world applications.

6 Conclusion

In this work, we introduced **MEDAL**, a framework that casts diffusion language model inference as a structured search problem and integrates it with MCTS. By applying MCTS during initialization, **MEDAL** explores promising unmasking trajectories before refinement. Its confidence-guided filtering and information-gain reward enable efficient, targeted search and improve global certainty. Across multiple benchmarks, **MEDAL** consistently outperforms existing inference strategies, achieving gains of up to 22.0%.

7 Limitation

While our study demonstrates the effectiveness of confidence-guided MCTS initialization for diffusion language models, several limitations remain. First, we primarily evaluate unimodal text-based DLMs; extending our framework to multimodal DLMs (e.g., vision-language or audio-language models) would provide a more comprehensive assessment of its generality. Second, our current experiments focus on standalone inference; integrating the method into agentic settings, where reasoning and decision-making unfold over multiple steps and interactions, poses both challenges and opportunities for future exploration.

8 Potential Risks

This work does not introduce additional ethical or societal risks beyond those associated with existing large language models. While our method utilizes test-time scaling to enhance generation quality, it operates by allocating compute to resolve uncertainty within the fixed model distribution. Consequently, it does not alter the model's fundamental objectives, training data, or safety alignment.

9 Acknowledgements

We used generative AI tools to assist in language polishing and writing refinement. All conceptual, methodological, and experimental contributions are solely those of the authors.

References

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

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

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. 2025. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

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*.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. 2025. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. 2025. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

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

Omer Luxembourg, Haim Permuter, and Eliya Nachmani. 2025. Plan for speed–dilated scheduling for masked diffusion language models. *arXiv preprint arXiv:2506.19037*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*.

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*.

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

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. 2025. Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

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.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and 1 others. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. 2025. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*.

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*.

Jaesik Yoon, Hyeonseo Cho, Doojin Baek, Yoshua Bengio, and Sungjin Ahn. 2025. Monte carlo tree diffusion for system 2 planning. *arXiv preprint arXiv:2502.07202*.

Oussama Zekri and Nicolas Boullé. 2025. Fine-tuning discrete diffusion models with policy gradient methods. *arXiv preprint arXiv:2502.01384*.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. 2024. Informed correctors for discrete diffusion models. *arXiv preprint arXiv:2407.21243*.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

A Appendix

A.1 Experiment Setting

In this section, we provide more details about the experimental settings. We use three different backbone models: LLaDA-8B-Instruct (Nie et al., 2025), LLaDA1.5-8B (Zhu et al., 2025), and Dream-7B (Ye et al., 2025), and compare the results with the original models without our method. We also include Llama3-8B (Touvron et al., 2023) as an AR LLM baseline for comparison. The details of the models are as follows:

- **LLaDA-8B-Instruct.** LLaDA is a large-scale diffusion language model (DLM) that replaces the traditional autoregressive next-token prediction paradigm with a masked diffusion process. Instead of generating text sequentially from left to right, LLaDA learns to reconstruct corrupted sequences by iteratively predicting masked tokens. The model defines a forward process that randomly masks tokens and a reverse process—parameterized by a transformer without causal masking—that predicts the original tokens, allowing bidirectional context modeling. Trained under a principled likelihood bound, LLaDA achieves scalable probabilistic inference comparable to large autoregressive models such as Llama3. It supports parallel token refinement, enabling faster generation and better global coherence. After pre-training on 2.3T tokens and fine-tuning on 4.5M instruction pairs, LLaDA demonstrates strong performance across reasoning, coding, and multilingual tasks, rivaling or surpassing autoregressive LLMs of similar scale while also addressing their limitations in efficiency, reversibility, and bidirectional reasoning.

- **LLaDA1.5-8B.** LLaDA1.5 is an advanced version of LLaDA, specifically designed to be better aligned with human preferences through reinforcement learning. The key innovation is a framework called Variance-Reduced Preference Optimization (VRPO), which addresses the primary challenge of applying direct preference optimization to diffusion models: the high variance in their likelihood estimations. VRPO introduces theoretically-grounded, unbiased techniques, such as optimal Monte Carlo budget allocation and antithetic sampling, to significantly reduce this variance, leading to more stable and effective training.

- **Dream-7B.** Dream-7B is a diffusion large language model that generates text by iteratively refining sequences in parallel rather than sequentially like autoregressive models. It is trained using two key techniques: initializing its weights from a pre-trained AR model and employing a context-adaptive token-level noise rescheduling mechanism. While achieving performance competitive with leading AR models like Qwen2.5 7B on general, mathematical, and coding benchmarks, Dream 7B demonstrates substantial abilities on complex planning tasks such as Sudoku and trip planning.

The data we used for evaluation includes six widely used datasets: GSM8K (Cobbe et al., 2021), ARC-C (Clark et al., 2018), HumanEval (Chen et al., 2021), MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019) and Countdown (Pan et al., 2025). The details of the datasets are as follows:

- **GSM8K.** A dataset to evaluate the mathematical and scientific reasoning capabilities of large language models. The dataset consists of high-quality, grade-school-level math word problems that require multiple steps to solve. These problems are designed to test a model's ability to perform

- **ARC-C.** The ARC Challenge (ARC-C) is a subset of the AI2 Reasoning Challenge (ARC) dataset, which is designed to evaluate the reasoning capabilities of AI systems. The ARC-C consists of multiple-choice questions, requiring deeper reasoning and understanding of scientific concepts. The questions cover a wide range of topics in science and are intended to test a model's ability to apply knowledge rather than just recall facts.

- **HumanEval.** A benchmark dataset for evaluating the programming capabilities of models by presenting them with a series of coding challenges. These challenges require the model to generate functional code from natural language descriptions, often in a zero-shot setting where no examples are provided.

- **MMLU.** The Massive Multitask Language Understanding (MMLU) benchmark is de-

signed to measure the general knowledge, reasoning, and commonsense abilities of large language models. It consists of multiple-choice questions, including humanities, social sciences, STEM fields, and more. The benchmark tests a model's ability to understand and generate text in various contexts, requiring both factual knowledge and reasoning skills.

- **DROP.** A reading comprehension benchmark that requires discrete reasoning over paragraphs of text. The dataset consists of questions that necessitate operations such as addition, counting, and sorting to arrive at the correct answer.

- **Countdown.** A benchmark that contains arithmetic reasoning problems where the model is given three or four numbers and must construct an expression that uses each number exactly once with basic operations (+, –, ×, ÷) to reach a target integer. Each example includes a list of numbers and a target.

The prompt we used for each dataset is shown in Figure 4, 5, 6 , 7, and 8.

A.2 Additional Experiments

In this section, we report the standard deviation and the computing overhead.

Standard Deviation. The standard deviation of all the methods across various datasets. The results are shown in Table 6. Overall, the results show that incorporating our method generally stabilizes model performance by reducing variance across benchmarks, particularly on reasoning-intensive tasks such as HumanEval, MMLU, and DROP.

Computational Costs. We evaluate computational overhead by comparing a standard LLaDA run with LLaDA using Best-of-15 (Bst15) decoding on GSM8K. The results are reported in Table 7. As shown, MEDAL achieves comparable runtime to Bst15 while delivering better accuracy, indicating that MEDAL allocates inference-time compute more effectively under constrained resources.

A.3 MCTS for DLMs Inference Example

We provide a single-step illustrative example of applying MCTS to DLM inference in Figure 3. (1) Selection: among three partially unmasked sequence nodes, we pick the one with the highest UCB score (Node 1). (2) Expansion: given the model logits at Node 1, we compute confidence-adjusted scores for all masked positions and construct the candidate action set $\mathcal{A}'_t$. For each candidate $a \in \mathcal{A}'_t$ (e.g., inserting the token "quick" at position 1), we apply the edit and create a new child node. (3) Simulation: from each expanded node, we unmask the remaining tokens using the DLM to obtain a completed sequence. (4) Backpropagation: we compute the information-gain reward (Equation (9)) for each simulation outcome and propagate it up the tree, updating value estimates for all parent nodes along the path to the root.

A.4 Formal Derivation of the MCTS Initialization Guarantee

Setup and notation. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a sequence of discrete tokens. We denote by q the (unknown) true data distribution and by p_θ the model distribution. Decoding proceeds in steps $i = 1, 2, \ldots$; at step i we reveal a (possibly multi-token) index set $z_i \subseteq \{1, \ldots, n\}$, and write $x_{z<i}$ for all tokens revealed before step i. Let U denote a set of indices of masked tokens. For such a set U and context $x_{z<i}$, we define the model conditional entropy as $H_\theta(x_\ell \,|\, x_{z<i}) := H(p_\theta(x_\ell \,|\, x_{z<i}))$, and the entropy-gap surrogate

$$B(U \mid x_{z<i}) := \sum_{\ell \in U} H_\theta(x_\ell \mid x_{z<i}) - \max_{\ell \in U} H_\theta(x_\ell \mid x_{z<i}). \tag{10}$$

The term $B(U \mid x_{z<i})$ measures the total uncertainty of the tokens in U, penalized by subtracting the largest single-token entropy. Intuitively, it is small when one token in U dominates the uncertainty and large when multiple tokens are simultaneously high-entropy, making it a useful surrogate for the dependence error incurred by unmasking them together. To simplify the notation, we write $B(z_i) := B(z_i \mid x_{z<i})$ when the context is clear.

Per-step decomposition When sampling x_{z_i} independently given $x_{z<i}$, the per-step KL divergence decomposes into a token-wise model error and a joint-dependence error(Ben-Hamu et al., 2025):

Model	GSM8K	ARC-C	HumanEval	MMLU	DROP
LLaDA	±1.36	±1.48	±2.51	±4.34	±3.17
+ ours	±2.37	±2.03	±2.98	±3.67	±3.25
LLaDA1.5	±3.05	±2.00	±3.22	±4.00	±2.12
+ ours	±3.27	±3.65	±2.44	±3.77	±4.44
Dream	±3.32	±2.09	±3.04	±2.27	±3.38
+ ours	±2.93	±3.32	±3.85	±2.41	±4.00
Llama	±3.05	±3.31	±4.22	±3.02	±2.03

Table 6: Results on standard deviation using different backbone models across various benchmarks.

	Running time	Accuracy
LLaDA	c=9.64s	58.3
+bst15	15c	65.3
+ours	12.3c	66.7

Table 7: Computational costs and accuracy on GSM8K using different backbone models, with accuracy reported in the leftmost column.

$$\underbrace{\sum_{\ell \in z_i} \mathrm{KL}\big(q(x_\ell \mid x_{z<i}) \,\|\, p_\theta(x_\ell \mid x_{z<i})\big)}_{\text{model error}} + \underbrace{\mathrm{KL}\Big(q(x_{z_i} \mid x_{z<i}) \,\Big\|\, \textstyle\prod_{\ell \in z_i} q(x_\ell \mid x_{z<i})\Big)}_{\text{dependence error} := \mathrm{DepErr}_i}, \tag{11}$$

where the first term, model error, sums the KL divergence for each $l \in z_i$, quantifying the per-token discrepancy between the model's conditional distribution and the true conditional given the current context, and the joint-dependence error quantifies the penalty from sampling all tokens in z_i independently, measuring how far the true joint conditional distribution $q(x_{z_i} \mid x_{z<i})$ is from the product of its marginals. It captures the correlations among tokens in z_i that are ignored when they are sampled independently, rather than jointly conditioned on one another. Our analysis focuses on the dependence term DepErr_i.

Standing assumption.

Assumption 1 (Entropy-gap upper bound)**.** *For every step i and context $x_{z<i}$,*

$$\mathrm{DepErr}_i = \mathrm{KL}\Big(q(x_{z_i} \mid x_{z<i}) \,\Big\|\, \textstyle\prod_{\ell \in z_i} q(x_\ell \mid x_{z<i})\Big) \le B(z_i). \tag{12}$$

Our contribution is to show how the MCTS initializer is designed to minimize the RHS (a computable surrogate), thereby controlling the cumulative dependence error.

Cumulative bound.

Lemma 1 (Prefix dependence is bounded by cumulative entropy gaps)**.** *For any $K \ge 1$,*

$$\sum_{i=1}^{K} \mathrm{DepErr}_i \;\le\; \sum_{i=1}^{K} B(z_i).$$

Proof. Apply Assumption 1 to each step and sum over $i = 1, \ldots, K$. □

Search space and surrogate objective. Fix $K \ge 1$ and let $\mathcal{S}_K$ be the set of feasible K-step schedules $z_{1:K} = (z_1, \ldots, z_K)$ (e.g., obeying any architectural or mask constraints). Define the *surrogate cost* of a schedule by

$$J(z_{1:K}) \;:=\; \sum_{i=1}^{K} B(z_i).$$

(Optionally, one may add a tokenwise uncertainty term; see Remark 1 below.)

MCTS estimator and selection rule. Given a budget of N simulations, MCTS constructs an empirical estimate $\widehat{J}_N(z_{1:K})$ for candidate schedules and returns

$$z_{1:K}^{(N)} \;\in\; \arg \min_{z_{1:K} \in \mathcal{S}_K \text{ explored}} \widehat{J}_N(z_{1:K}).$$

We adopt the UCT tree policy (Kocsis and Szepesvári, 2006), which is asymptotically consistent: under standard assumptions (bounded costs and unbiased rollout estimates), the empirical estimates $\widehat{J}_N(z_{1:K})$ converge to the true surrogate

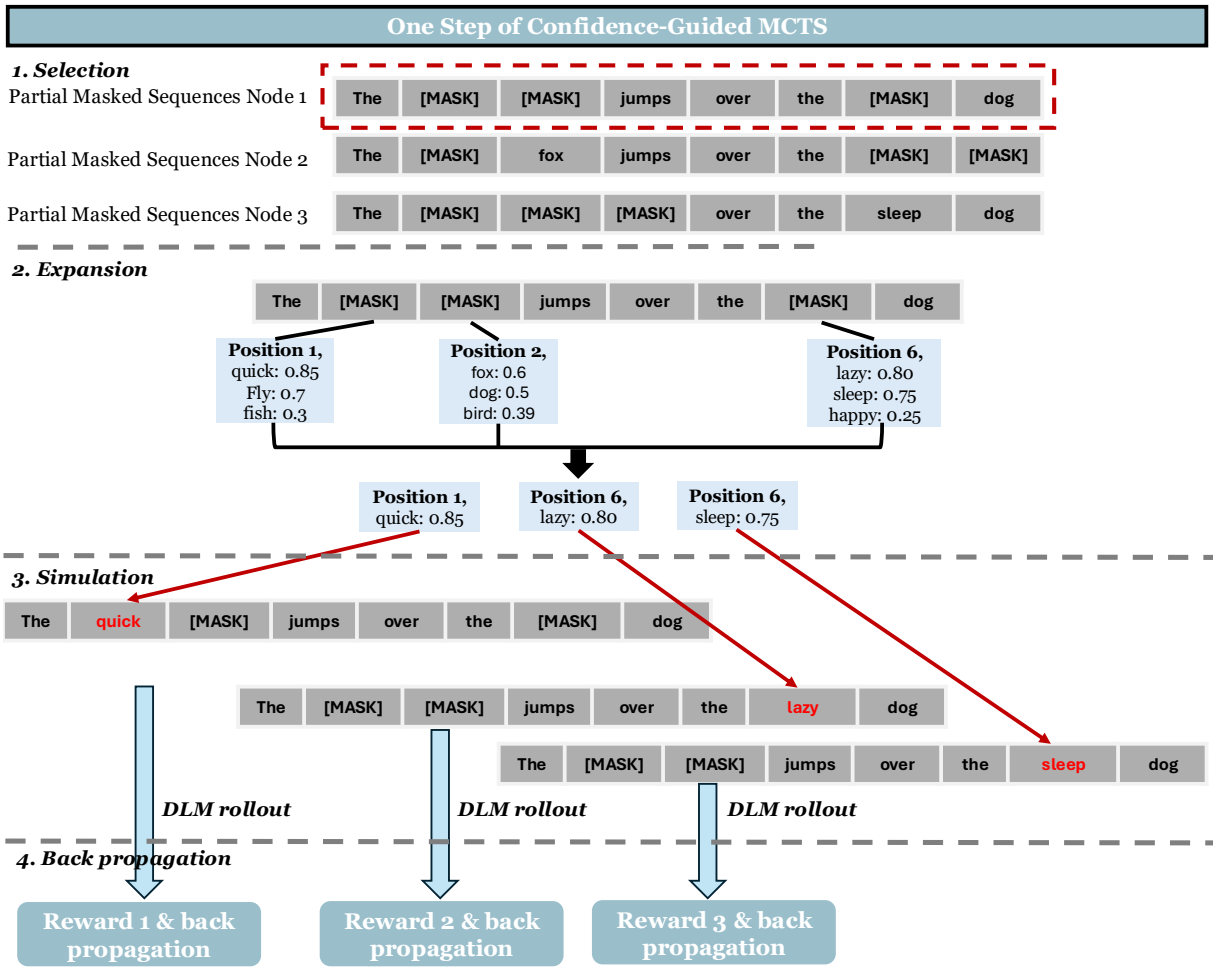

Figure 3: A single-step illustrative example of applying MCTS to DLM inference

cost $J(z_{1:K})$ as $N \to \infty$. Consequently, the initialization schedule returned by MCTS converges almost surely to the minimizer of $J(z_{1:K})$ within the feasible search space.

Theorem 1 (Bounded dependence minimized by MCTS initialization)**.** *Under Assumption 1 and the consistency conditions above, every limit point* $z^\star_{1:K}$ *of* $\{z^{(N)}_{1:K}\}_{N\geq 1}$ *satisfies*

$$z^\star_{1:K} \in \arg\min_{z_{1:K}\in\mathcal{S}_K} J(z_{1:K}), \text{ and hence}$$
$$\sum_{i=1}^{K} \mathrm{DepErr}_i \;\leq\; J(z^\star_{1:K}) \;=\; \min_{z_{1:K}\in\mathcal{S}_K} J(z_{1:K}). \tag{13}$$

In particular, among all feasible K-step prefixes, the selected initialization attains the smallest achievable upper bound *on cumulative dependence error.*

Proof. By Lemma 1, any schedule $z_{1:K}$ satisfies $\sum_{i=1}^{K} \mathrm{DepErr}_i \leq J(z_{1:K})$. By the assumed consistency of MCTS, $\widehat{J}_N \to J$ pointwise on explored nodes and the selection rule is asymptotically optimal: every accumulation point $z^\star_{1:K}$ minimizes J over $\mathcal{S}_K$. Evaluating the bound at such a minimizer yields $\sum_{i=1}^{K} \mathrm{DepErr}_i \leq J(z^\star_{1:K}) = \min_{z_{1:K}\in\mathcal{S}_K} J(z_{1:K})$, which is the tightest bound attainable within $\mathcal{S}_K$. □

Corollary 1 (Comparison to any baseline prefix)**.** *Let $\widetilde{z}_{1:K} \in \mathcal{S}_K$ be any baseline schedule (e.g., random or greedy). Then, under the conditions of Theorem 1, for large enough N,*

$$\sum_{i=1}^{K} \mathrm{DepErr}_i \;\leq\; J(z^{(N)}_{1:K}) \;\leq\; J(\widetilde{z}_{1:K}).$$

Thus the MCTS-chosen prefix achieves a (weakly) smaller upper bound on cumulative dependence than the baseline.

Remarks and extensions.

Remark 1 (Adding model-error proxies)**.** *If one augments the surrogate with a token-wise model-error proxy* prox_ℓ *(e.g., entropy,* $1-$*confidence, top-2 margin),*

$$J_\lambda(z_{1:K}) = \sum_{i=1}^{K} \Big[\lambda_{\mathrm{dep}} B(z_i) + \lambda_{\mathrm{mod}} \sum_{\ell \in z_i} \mathrm{prox}_\ell\Big], \tag{14}$$

where $\lambda_{\mathrm{dep}}, \lambda_{\mathrm{mod}} > 0$*, then minimization of* J_λ *with* $\lambda_{\mathrm{dep}} > 0$ *still minimizes* $\sum_i B(z_i)$ *subject to the chosen trade-off. The dependence part of Theorem 1 carries through unchanged, yielding the same bound on* $\sum_i \mathrm{DepErr}_i$ *with* J *replaced by the dependence component of* J_λ*.*

Remark 2 (On computability and calibration)**.** *All terms in* $B(z_i)$ *are computed from* $p_\theta(\cdot \mid x_{z<i})$ *(model logits), so* J *is observable at inference time. The tightness of the bound depends on how well the model entropies reflect the coupling structure under* q*; improving calibration may further tighten the practical gap between* J *and* $\sum_i \mathrm{DepErr}_i$*.*

Remark 3 (Feasibility constraints)**.** *The set* $\mathcal{S}_K$ *can encode architectural or policy constraints (e.g., block sizes, mask connectivity). Theorem 1 is* relative *to* $\mathcal{S}_K$*: MCTS finds the tightest upper bound* within *the feasible class.*

Prompt for GSM8K

Decompose the question into different steps and solve the math problem follow the examples. Show your final answer after ####.

Example:
Problem: Janet's ducks lay 16 eggs per day. She eats 3 for breakfast every morning and bakes 4 into muffins for her friends every day. She sells the remainder at the farmers' market for $2 per fresh duck egg. How much money does she make every day at the farmers' market?

1. Understand how many eggs Janet's ducks lay.
2. Calculate how many eggs Janet eat, bake, and how many to sell.
3. Calculate the money she make.

18

Problem: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts of fiber does it take?

1. Identify how many bolts of blue fiber are needed.
2. Calculate how many bolts of white fiber are needed.
3. Calculate the total number of bolts needed.

3

Figure 4: Prompt used for GSM8K

Prompt for ARC-C

Answer the following multiple-choice questions. For each question, decompose it following the example and select the most appropriate answer. Provide your final answer after ####.

Question: What is the main difference between an animal cell and a plant cell?
(A) Animal cells have a cell wall, while plant cells do not.
(B) Plant cells have chloroplasts, while animal cells do not.
(C) Animal cells have mitochondria, while plant cells do not.
(D) Plant cells have a nucleus, while animal cells do not.

1. Identify the main difference between animal and plant cells.
2. Recall relevant facts: Animal cells do not have cell walls, but plant cells do have rigid cell walls made of cellulose.
Plant cells have chloroplasts which contain chlorophyll for photosynthesis, while animal cells do not have chloroplasts.
Both animal and plant cells have mitochondria for cellular respiration.
Both animal and plant cells have a nucleus containing genetic material.
3. Select the option.
B

Question: Which of the following is NOT a renewable energy source?
(A) Solar power
(B) Wind power
(C) Natural gas
(D) Hydroelectric power

1. Identify which energy source is NOT renewable.
2. Recall relevant facts: Solar power comes from the sun, which is essentially unlimited, so it's renewable.
Wind power harnesses the energy of moving air, which is constantly replenished, so it's renewable.
Natural gas is a fossil fuel formed over millions of years and exists in finite quantities, so it's not renewable.
Hydroelectric power uses the energy of flowing water, which is replenished through the water cycle, so it's renewable.
3. Select the option.
C

Now answer this question:
Question: {question}

Figure 5: Prompt used for ARC-C

Prompt for HumanEval

Decompose the given function following the examples and solve it based on the docstring requirements.

Example 1:
1. Understand the description of the function.
2. Understand the goal and what the function will return.
3. Write valid Python code

```
def add(a, b):
    """Add two numbers a and b.
    >>> add(1, 2)
    3
    >>> add(5, 7)
    12
    """
    return a + b
```

Example 2:
1. Understand the description of the function.
2. Understand the goal and what the function will return.
3. Write valid Python code

```
def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """ Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
```

Now complete the following function:

Figure 6: Prompt used for HumanEval

Prompt for MMLU

Split the question into different steps and solve the multiple-choice problem. Show your final answer letter after ####.

Example:
Question: What is the capital of France?
(A) London
(B) Berlin
(C) Paris
(D) Madrid

1. Understand what we're looking for: the capital city of France.
2. Recall relevant facts about each option: London→UK, Berlin→Germany, Paris→France, Madrid→Spain.
3. Evaluate each option and select the option.

C

Question: Which of the following is a noble gas?
(A) Oxygen
(B) Chlorine
(C) Nitrogen
(D) Helium

1. Understand what we're looking for: identify a noble gas (Group 18).
2. Recall group memberships: Oxygen→Group 16, Chlorine→Group 17 (halogen), Nitrogen→Group 15, Helium→Group 18 (noble gas).
3. Eliminate options that are not in Group 18 and select the option.

D

Figure 7: Prompt used for MMLU

Prompt for DROP

Decompose and answer the question based on the given passage, following the examples.

Passage: The Vikings were a seafaring people from Scandinavia who raided and traded across wide areas of Europe from the late 8th to the late 11th century. The Vikings employed wooden longships with wide, shallow-draft hulls, allowing navigation in rough seas or in shallow river waters.

Question: How long did the Viking Age last?
1. Understand the passage.
2. Recall relevant contents in the passage about Viking Age.
3. Provide the answer.

Answer: 3 centuries

Passage: The Battle of Gettysburg was fought July 1–3, 1863, in and around the town of Gettysburg, Pennsylvania, by Union and Confederate forces during the American Civil War. Between 46,000 and 51,000 soldiers from both armies were casualties in the three-day battle, the most costly in US history.

Question: How many days did the Battle of Gettysburg last?
1. Understand the passage.
2. Recall relevant contents in the passage about the Battle of Gettysburg .
3. Provide the answer

Answer: 3 days

Now answer this question:

Passage: {passage}

Question: {question}
Answer:

Figure 8: Prompt used for DROP