Diffuse Thinking: Exploring Diffusion Language Models as Efficient Thought Proposers for Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated strong capabilities in complex reasoning tasks, yet the multi-step reasoning processes often lead to expensive computation cost. The recent advance of diffusion language models (DLMs) adopts a parallel, non-autoregressive generation mechanism, which enables the efficient production of multiple outputs. In this paper, we explore a collaborative reasoning framework that combines diffusion-based generation with autoregressive evaluation. Specifically, we leverage DLMs to efficiently propose stepwise reasoning thoughts, and employ LLMs as evaluators to assess and select candidates based on their plausibility and correctness. By decoupling proposal generation from evaluation, our framework exploits the strengths of both models: efficient exploration from diffusion models and causally grounded assessment from autoregressive models, which naturally aligns with *the divergent–convergent thinking framework* in cognitive psychology. Experiments across various mathematical and logical reasoning benchmarks demonstrate that, our framework improves inference efficiency while maintaining competitive or superior reasoning accuracy, laying the groundwork for building efficient reasoning architectures. Our code is open-source at https://anonymous.4open.science/r/Diffuse-Thinking-EC60 .

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, achieving strong performance across mathematical, logical, and multi-step decision problems (OpenAI, 2024a; Meta, 2024; Liu et al., 2024; GLM et al., 2024). Despite these advances, most contemporary LLMs adopt an autoregressive generation paradigm, in which tokens are produced sequentially from left to right and each token necessitates a separate forward pass computation. This token-by-token decoding mechanism makes large-scale

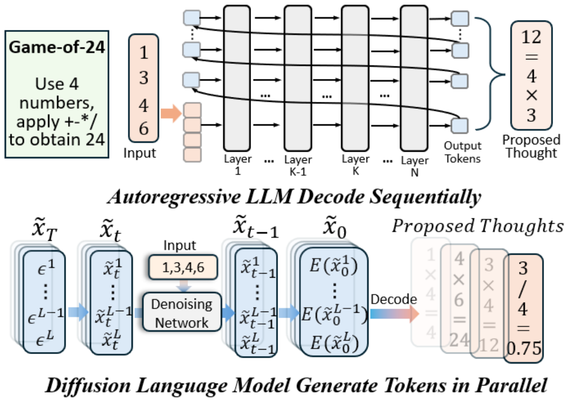

Figure 1: Comparison of generation in DLM and LLM: DLMs generate in parallel, producing multiple tokens simultaneously, while LLMs generate sequentially, one token at a time.

exploration of diverse reasoning paths computationally prohibitive. This bottleneck is particularly acute for advanced reasoning techniques, such as Chain-of-Thought (CoT) (Wei et al., 2022), Tree-of-Thought (ToT) (Yao et al., 2023), and large reasoning models like OpenAI-o3 (OpenAI, 2025), which explicitly rely on generating and evaluating numerous intermediate reasoning steps (OpenAI, 2024b; Xu et al., 2025; Kong et al., 2025).

The advancement of diffusion language models (DLMs) offers a promising direction for addressing the inefficiency of autoregressive reasoning. As illustrated in Figure 1, unlike autoregressive LLMs that generate tokens sequentially, DLMs adopt a parallel reverse denoising process that enables multiple tokens to be generated simultaneously. This non-autoregressive generation paradigm allows DLMs to efficiently produce diverse candidate outputs within a small number of forward passes, substantially reducing the cost of proposal generation. Moreover, while LLMs employ a uniform generation process for all tasks regardless of difficulty or quality requirements, DLMs can dynamically adjust their denoising steps, enabling a flexible trade-off between generation quality and speed tailored to the specific demands of the task.

These properties suggest that DLMs are particularly effective at exploring diverse candidate reasoning thoughts. In contrast, although LLMs incur high costs during output generation, they can process and store arbitrarily long input sequences within a single forward pass, making input-side computation relatively efficient. Moreover, due to their autoregressive modeling of conditional token dependencies, LLMs exhibit strong causal reasoning and logical consistency, which are essential for evaluating structured reasoning processes. As a result, LLMs are naturally suited to act as evaluators that assess the plausibility and correctness of candidate reasoning paths. This complementary division of roles enables a collaborative reasoning paradigm in which proposal generation and evaluation are decoupled and assigned to different model classes.

Notably, this design principle aligns with the *divergent–convergent thinking framework* (Guilford, 1967) in cognitive psychology, which distinguishes two complementary modes of problem solving. *Divergent thinking* emphasizes the broad generation of multiple, diverse candidate ideas without immediate commitment to their correctness, whereas *convergent thinking* focuses on systematically evaluating, refining, and selecting among these candidates based on logical consistency and task constraints. In our framework, DLMs naturally instantiate the divergent mode by efficiently proposing diverse candidate thoughts, while autoregressive LLMs realize the convergent mode by leveraging their causally grounded generation to perform logical evaluation and selection. This correspondence offers a concrete explanation, from a cognitive psychology perspective, for why diffusion-based generation and autoregressive evaluation complement each other in collaborative reasoning.

However, realizing effective collaboration between DLMs and LLMs in complex reasoning tasks presents several challenges. First, the collaboration mechanism between DLMs and LLMs remains underexplored, and the appropriate division of roles between the two paradigms is not yet well understood. Second, due to information loss inherent in the diffusion process, naively generated proposals may lack logical precision, limiting their usefulness for downstream reasoning. Third, the optimal granularity and scope of reasoning thoughts to be generated by DLMs remain unclear, making it nontrivial to determine how complex reasoning tasks should be decomposed for proposal generation.

To address these challenges, we propose Diffuse Thinking (DT), a novel diffusion–autoregressive collaborative reasoning framework. DT is built upon a clear division of labor: DLMs serve as proposers that generate diverse, single-step intermediate reasoning thoughts, while LLMs act as evaluators that assess and filter these proposals based on causal and logical consistency. The interaction between proposer and evaluator is orchestrated through a structured propose–evaluate&select loop, ensuring that only the most plausible reasoning steps are retained for subsequent expansion. Through this design, DT significantly improves inference efficiency while maintaining high reasoning accuracy. To further boost the quality of proposals in practice, we fine-tune DLMs on a curated dataset of single-step reasoning examples, further enhancing the reasoning capabilities of the framework.

We conduct extensive experiments on four mathematical and logical reasoning benchmarks. Across all benchmarks, DT consistently outperforms baselines, achieving higher reasoning accuracy while significantly improving inference throughput. On average, DT improves reasoning accuracy by 5% and throughput by 10% compared to the most accurate baselines. Our contributions are summarized as follows:

- We propose a novel collaborative reasoning framework that combines DLMs and LLMs, decoupling proposal generation from evaluation at the level of intermediate reasoning thoughts.

- We provide theoretical analyses from information-theoretic and computational complexity perspectives to explain the advantages of the proposed diffusion–autoregressive collaboration.

- We validate the effectiveness of our framework on diverse mathematical and logical reasoning benchmarks, demonstrating that it simultaneously improves reasoning accuracy and inference efficiency.

2 Related works

2.1 Chain-of-Thought reasoning in LLMs

Recent advances in LLMs have demonstrated that explicitly modeling intermediate reasoning steps is critical for solving complex mathematical and logical tasks. Early work on Chain-of-Thought (CoT) prompting (Wei et al., 2022) showed that encouraging models to generate step-by-step explanations can substantially improve reasoning accuracy. Building upon CoT, subsequent frameworks such as Tree-of-Thought (ToT) (Yao et al., 2023) and related search-based methods further extend this idea by decomposing problems into multiple reasoning branches, sampling diverse intermediate thoughts, and selecting promising trajectories via evaluation or search (Snell et al., 2024; Xie et al., 2024). In parallel, reinforcement learning and verifier-based training strategies have been introduced to strengthen reasoning behaviors, giving rise to so-called large reasoning models (Guo et al., 2025; OpenAI, 2024b; Shao et al., 2024; Xu et al., 2025). Despite their success, these approaches almost exclusively rely on autoregressive decoding to generate numerous intermediate thoughts (Shao et al., 2025a,b). As reasoning depth and the number of sampled thoughts increase, token-by-token generation becomes the dominant computational bottleneck, motivating the exploration of more efficient generation paradigms for complex reasoning.

2.2 Diffusion language models

Diffusion models have attracted increasing attention for text generation. To bridge the gap between continuous diffusion processes and discrete text tokens, two main categories of DLMs have been proposed: discrete DLMs, which corrupt tokens via Markov transitions (Zou et al., 2023; Hoogeboom et al., 2021; Zheng et al., 2023; Lou et al.), and continuous DLMs, which embed tokens and apply Gaussian noise. Both paradigms have demonstrated strong performance across a range of NLP tasks, including machine translation (Nachmani and Dovrat, 2021), controllable generation (Han et al., 2022; Li et al., 2022) and text summarization (Zhang et al., 2023). Beyond traditional NLP tasks, recent work has begun to explore DLMs for complex reasoning. Pretrained models such as Plaid (Gulrajani and Hashimoto, 2024) and SEDD (Lou et al., 2023) achieve generation quality comparable to autoregressive models of similar scale. Several efforts explicitly target reasoning: Diffusion-of-Thought (Ye et al., 2024b) incorporates CoT supervision into diffusion models, while (Ye et al., 2024a) improves training efficiency via token-level reweighting and achieves strong performance on structured tasks such as Sudoku, though it remains limited to numeric tokens and lacks general semantic reasoning capability.

More recently, scaling efforts such as LLADA (Nie et al., 2025) and DiffuLLaMA (Gong et al., 2024) extend DLMs to the 7–8B parameter range. In parallel, Mercury Coder (Khanna et al., 2025), a commercial-scale diffusion model for code generation, demonstrates generation speeds of up to 1000 tokens per second, substantially outperforming autoregressive models in throughput. These advances underscore the efficiency advantages of diffusion-based generation. Our work builds on these developments by proposing a collaborative framework that combines the efficient generation of DLMs with LLMs' strong causal evaluation.

3 Methodology

Inspired by the cognitive process of divergent and convergent thinking, we propose a collaborative reasoning framework. The reasoning task is decomposed into sub-tasks where the DLM extensively explores the solution space to generate diverse candidate proposals, while the LLM evaluates and selects the correct ones. This cycle of proposing and evaluating iterates until the answer is derived.

3.1 DLM as thought proposer

In our framework, the DLM assumes the role of **divergent thinking**, tasked with the broad and rapid exploration of the solution space. The following computational complexity analysis demonstrates why the DLM's parallel generation mechanism is ideally suited for this efficient proposing role.

Computational complexity derivation of DLM

For DLM, since the input and output lengths are fixed and equal, the computational complexity is primarily determined by the denoising transformer network layers and the number of denoising steps. For T denoising steps, the total FLOPs for the DLM can be expressed as:

$$\begin{aligned} F_{\text{DLM}} &= T \cdot \Bigg(2 \cdot N \cdot \left(12 \cdot L \cdot D^2 + 2 \cdot L^2 \cdot \frac{D}{H}\right) \\ &\quad + 2 \cdot L \cdot D \cdot E + 2 \cdot L \cdot (D + 2 \cdot E) \cdot V\Bigg) \\ &= \mathcal{O}\left(T \cdot L^2\right) \end{aligned} \tag{1}$$

For a more detailed derivation, please refer to Appendix. Considering the generation of K samples, the overall computational cost becomes:

$$F_{\text{DLM (K)}} = K \cdot F_{\text{DLM}} \tag{2}$$

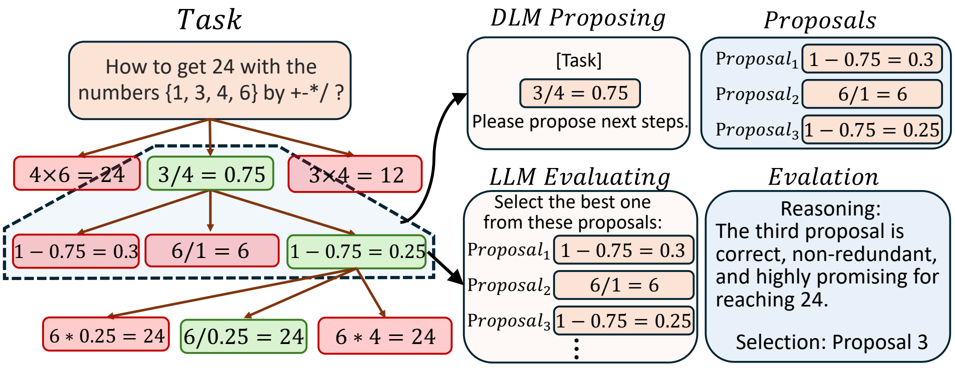

Figure 2: Overview of the proposed framework, exemplified by the Game of 24. The DLM efficiently generates diverse candidate thoughts, while the LLM evaluates these candidates to select the most promising proposal.

Parallel-time complexity. All matrix multiplications inside self-attention and MLP are parallelizable over the sequence dimension L, since the DLM processes the *entire sequence simultaneously* in forward passes. Assuming ideal hardware that provides $O(L)$ parallel lanes, the *wall-clock latency* per denoising step collapses to the longest dependency path, yielding

$$T_{\text{DLM}}^{\text{par}} = O\left(\frac{K_{DLM} \cdot L^2 \cdot T}{L}\right) = O(K_{DLM} \cdot L \cdot T). \tag{3}$$

Thus, under sufficient parallelism, the latency is *linear in* T and *linear in* L, while the total FLOPs still scale as $O(L^2T)$.

Computational complexity derivation of LLM

In contrast to DLM, the computational cost of LLMs is determined by both their backbone architecture and the variable length of output text. We denote the per-head dimension as: $d_h = \frac{D}{H}$. Considering KV-cache, the computational complexity of LLM can be expressed as:

$$\begin{aligned} F_{\text{total}} = \; & (2 \cdot D \cdot V + 24 \cdot N \cdot D^2) \cdot L_{out} \\ & + (2 + 24 \cdot N \cdot D) \cdot D \cdot L_{in} \\ & + 4 \cdot N \cdot \frac{D}{H} \cdot (L_{in}^2 + L_{out}^2 + L_{in} \cdot L_{out})). \end{aligned}$$

For a more detailed derivation, please refer to Appendix.

When generating K samples, the complexity scales by a factor of K and is further affected by the length of the longest generated sequence:

$$\begin{aligned} T_{\text{LLM}} &= \mathcal{O}\Big(K_{\text{LLM}}\big(L_{\text{out,max}}^2 + L_{\text{in}}^2 + L_{\text{in}}\,L_{\text{out}}\big)\Big) \\ &= \mathcal{O}\big(K_{\text{LLM}}\,L^2\big) \end{aligned} \tag{4}$$

Sequential-time complexity. Unlike DLMs, autoregressive language models generate tokens one by one; each new token depends on all previously generated tokens. Consequently, self-attention and MLP layers cannot be parallelized across the sequence dimension L_{out}, even under ideal hardware.

We also computationally demonstrate that K_{DLM} can reach 16 times as much as K_{LLM} for the same model architecture, computational resources, parameter sizes, and input/output lengths while best fitting our experimental setup. The above are all derived results considering KV-cache for LLM. If the KV-cache is not considered, the computational cost of the LLM increases rapidly as the sequence length increases, due to the cubic dependence on L_{out}, while the computational cost of the diffusion model remains low. For a more detailed derivation, please refer to Appendix.

The comparison of DLM and LLM highlights the significant advantage of DLM's parallel generation, which inspires us to fully leverage the parallelism of DLMs to replace LLMs for proposing a number of thoughts, thereby greatly improving inference efficiency. Assuming that the DLM generates M thoughts, then the denoising process can be defined as follows:

$$\mathcal{P} = \{\text{Denoised}_{(j)}(\tilde{x}_0) \mid j = 1, \ldots, M\} \tag{5}$$

"Denoised" represents the process in which DLM gradually denoises from noise and generates final outputs. The equation as a whole indicates that DLM generates M thoughts, which are placed in a set $\mathcal{P}$ for subsequent derivation.

3.2 LLM as thought evaluator

In our framework, the LLM assumes the role of **convergent thinking**, tasked with the rigorous evaluation and selection of candidate proposals. The underlying mechanisms of inference in LLMs and DLMs differ fundamentally. For LLMs, with the support of KV cache, the process of handling inputs and generating tokens is separated into two phases. In the first phase, known as prefill, LLM processes all input tokens in parallel to initialize the KV cache, performing a single forward pass to output the first token. The second phase is decoding, where the model generates subsequent tokens one by one, requiring a forward pass for each generated token. The separation of two phases explains why input tokens are cheaper than output tokens in current LLMs. However, for DLMs, handling inputs and generating outputs cannot be separated, and there is no advantage in processing inputs more efficiently than generating outputs. This indicates that LLMs exhibit high efficiency in processing input. Regardless of the input text length, an LLM only requires a single forward pass to encode the entire input into the initial state of the attention memory. This advantage is not present in DLMs, which inspires us to leverage LLMs as evaluators to assess and filter a large number of proposals. Moreover, LLMs inherently possess strong general semantic understanding capabilities, enabling them to identify the most promising and correct proposal from a set of candidates.

Specifically, once the DLM generates a set of candidate solutions, LLM can be leveraged to evaluate and select the most appropriate one, taking into account the context provided by the problem and the proposals. The evaluation process can be formulated as follows: The LLM receives a prompt that includes the problem q, the candidate solutions $\mathcal{P}$, and relevant guidance. The model then predicts both the most suitable solution index i^* and the corresponding reasoning r in a unified process by computing the joint probability distribution:

$$p_{\text{LLM}}(i^*, r|q, \mathcal{P}; \theta) \tag{6}$$

Here, i^* represents the chosen solution index, and r is the reasoning behind the choice. This formulation encapsulates both solution selection and reasoning generation in a single model evaluation. This formulation enables the LLM to provide both a solution and an explanation in one inference pass, significantly reducing computational overhead compared to evaluating each candidate solution individually.

3.3 DLM-LLM collaborative reasoning framework

Ultimately, we combine the rapid proposal generation of the DLM with the precise evaluation of the LLM. Through an iterative propose-evaluate mechanism, this design yields a highly efficient and capable reasoning system. The overall process proceeds as follows: First, the diffusion model generates a set of candidate solutions $\mathcal{P}$ by reversing the noise from the initial noisy states $\tilde{x}_T^{(j)}$ for each $j = 1, \ldots, M$. After generating the candidate solutions, the problem input q is concatenated with $\mathcal{P}$ to form a prompt for the LLM.

The LLM then evaluates all the candidates in one inference step, predicting the most suitable solution index i^* and providing the corresponding reasoning r. The selected solution is the one with the highest probability:

$$i^* = \arg\max_i p_{\text{LLM}}(i|q, \mathcal{P}; \theta) \tag{7}$$

Finally, the selected solution x^* is obtained as $x^* = x_{i^*}$, and the reasoning r provides an explanation for the selection.

We present a theoretical analysis that elucidates the fundamental limits of error in diffusion language models from an information-theoretic perspective. Specifically, the per-step information loss ΔI_t is defined as the difference between the model's estimated mutual information and the true mutual information:

$$\Delta I_t = I_{\text{indep}}(X_t|X_{t-1}) - I(X_t|X_{t-1}) \geq 0. \tag{8}$$

This loss quantifies the information about the dependencies between tokens x_i^t within step t (given X_{t-1}) that is omitted by the model.

Let $I_{\text{ideal}}(X^*; X_0)$ be the mutual information achievable by an ideal T-step diffusion process (without the independence assumption), and $I_{\text{indep}}(X^*; X_0)$ be the mutual information achieved by the model using the independence assumption.

The total accumulated information loss is:

$$\begin{aligned} &\text{TotalLoss}(T, L) = \\ &\quad I_{\text{ideal}}(X^*; X_0) - I_{\text{indep}}(X^*; X_0) \geq 0 \end{aligned} \tag{9}$$

This total loss reflects the cumulative impact of the per-step losses ΔI_t across all denoising steps. We derive the following inequality using Fano's inequality (Scarlett and Cevher, 2019):

$$\begin{aligned} &H_{\text{ideal}}(X^* \mid X_0) + \text{TotalLoss}(T, L) \\ &\leq H\big(E_{\text{Diff}}(T, L)\big) + E_{\text{Diff}}(T, L) \log\big(|\mathcal{X}| - 1\big). \end{aligned} \tag{10}$$

$\text{TotalLoss}(T, L)$ denotes the total accumulated information loss, while $H_{\text{ideal}}(X^*|X_0)$ represents the conditional entropy of the ideal denoising process. $E_{Diff}(T, L) = P(X_0 \neq X^*)$ denotes the final error probability, and $\mathcal{X}$ is a finite set of possible values for X^*. A detailed derivation of this inequality is provided in Appendix. This indicates that the lower bound of error in the diffusion language models is affected by the loss of information in the parallel prediction tokens, which is expected to increase as L increases. By integrating the diverse solution generation capabilities of diffusion models with the global reasoning capabilities of large language models (LLMs), the proposed framework systematically optimizes the solution for each step, narrowing the gap with the correct answer for that step. The LLM not only selects the optimal solution but also ensures transparency by providing a detailed explanation of its choice.

3.4 Learning to propose thoughts

To further elicit the intrinsic capability of DLMs for continuous and incremental problem-solving through training, we meticulously design the construction of the training data. Specifically, for each reasoning task, we perform task decomposition, breaking down the complete reasoning process into sub-tasks that are more manageable and relatively easier to solve. Each sub-task corresponds to a specific and practical reasoning thought, such as computation, concatenation, transformation, and so forth. Each ground truth thought is obtained through a *search*-based approach, where we explore and traverse the entire solution space. Ultimately, we filter and retain the most accurate and promising thoughts to be included in the dataset. Each thought constitutes a part of the solution, effectively advancing the problem-solving process. For each training instance, the input consists of the problem description of the reasoning task along with the preceding reasoning thoughts, while the output is the next reasoning thought.

This dataset construction approach encourages our DLM to spontaneously adopt step-by-step reasoning to tackle problems and propose the next-step reasoning thoughts. The DLM can perform denoising on a batch to effectively generate multiple next-step thoughts, forming a set of candidate proposals. These proposals are then evaluated and filtered, with one or several selected to proceed, enabling the iterative process of proposing and filtering until the problem is solved.

4 Experiments

We evaluate the comprehensive reasoning accuracy and efficiency of our framework across four challenging benchmarks against baselines. Furthermore, we conduct a detailed analysis of the quality of DLM-generated proposals and quantify the performance gains contributed by the LLM evaluator.

4.1 Experimental setup

Benchmarks: We select the Game-of-24, GPQA, ARC-C and Trip-planning problems as our evaluation benchmarks. Game-of-24 requires using four given numbers to compute the number 24 through addition, subtraction, multiplication, and division, assessing both fundamental arithmetic skills and the ability to maintain a global logical perspective throughout the reasoning process. GPQA presents a PhD-level task that demands identifying the single correct answer among four choices in biology, physics, or chemistry, foregrounding retrieval of rare factual knowledge, chaining of multi-step scientific reasoning, and resistance to subtle, expert-designed distractors. ARC-C tasks the solver with electing the single valid choice from four elementary-level science options, spotlighting common-sense induction, unseen domain transfer over distant concepts, and immunity to surface-level semantic mirages. Trip-planning entails composing a uniquely-valid multi-city itinerary under deterministic flight-connectivity and day-level duration constraints, foregrounding implicit constraint propagation, cross-domain temporal reasoning.

For the Game-of-24 , we exhaustively enumerated number combinations and employed a backtracking approach to solve it. We considered all non-repetitive, diverse solutions, resulting in a dataset of 168,046 24-point problems, with the maximum value for the four numbers limited to 30. For GPQA benchmark, we used the original dataset from (Rein et al., 2024). For ARC-C, we used the challenge set from the original dataset from (Clark et al., 2018). For Trip-planning, we used the 3-city subset of the original dataset from (Zheng et al., 2024). All the datasets used and created in this paper will be open-sourced at the project link.

4.1.1 Metrics

Acc Except for the Game-of-24, all benchmarks admit a unique solution. Reasoning accuracy is computed by comparing the generated outputs with

Solutions	Game-of-24		Trip-planning		GPQA		ARC-C	
(Proposer + Evaluator)	Acc	ThroughPut	Acc	ThroughPut	Acc	ThroughPut	Acc	ThroughPut
Llama3-8B + Llama3.3-70B	0.12	0.30	0.04	0.23	0.32	0.13	0.80	0.49
Mistral-7B + Llama3.3-70B	0.08	0.41	0.07	0.18	0.33	0.19	0.81	0.57
Deepseek-7B + Llama3.3-70B	0.12	0.36	0.11	0.25	0.33	0.38	0.82	0.53
DT(LLADA-8B + Llama3.3-70B)	0.12	**0.48**	0.16	0.35	0.33	**0.56**	0.82	**1.05**
DT(Dream-7B + Llama3.3-70B)	**0.13**	0.45	**0.29**	**0.40**	**0.37**	0.46	**0.83**	0.59

Table 1: Performance of our proposed framework(**DT**). In each benchmark, the highest reasoning accuracy and throughput is highlighted in **bold**, and the second-highest is underlined.

the ground truth. For the Game-of-24, beyond verifying whether the final result equals 24, we additionally evaluate the validity of the intermediate reasoning steps, including whether all four numbers are properly used and whether each arithmetic operation is correct.

Throughput Following a standard experimental setup, we measure *throughput*, the number of samples processed per minute (it/min) with a batch size of one during inference, to assess the framework's time efficiency in problem solving.

Time To provide an intuitive measure of efficiency, we measure *time* in single-step evaluations, reporting the average inference time per step (in seconds).

Pass@5 To examine the quality of the thoughts, we measure the probability that at least one of the five proposals is correct for a single sub-task.

Baselines Given that the parameter count of the DLM model falls within the range of 7 and 8 billion, to ensure a fair comparison, we restrict our consideration to LLM models with parameters close to 8 billion. Ultimately, we select the recently released and widely recognized models: Llama3-8B (Meta, 2024), Mistral-7B (Jiang et al., 2023) and Deepseek-7B (Bi et al., 2024) as proposers. We configure the evaluation model, Llama3-70B (Meta, 2024), to assess the thoughts generated by each proposer.

Implementation details In inference, the number of sampling steps T is dynamic. This demonstrates the flexibility of Diffusion LM and improves the efficiency of proposal generation. We set $T = 8$ for Game-of-24, $T = 64$ for GPQA ,Trip-planning and ARC-C. For the Main Results 4.2 and Quality of Proposals 4.3, we directly use two open-source discrete DLMs, namely Dream (Ye et al., 2025) and LLADA (Nie et al., 2025). In the finetuning experiment, we train our model on a single NVIDIA A100-80G with a batch size of 2, and perform sampling on the same device. The DLM is finetuned from the pretrained discrete diffusion language model LLADA (Nie et al., 2025). During finetuning, we set the maximum sequence length to 4096 and use a learning rate of 1e-5.

Models	Game-of-24		Trip-planning		GPQA		ARC-C	
	pass@5	Time	pass@5	Time	pass@5	Time	pass@5	Time
Llama3-8B	0.15	6.43	0.33	42.12	0.55	78.28	0.88	28.32
Mistral-7B	0.065	4.76	0.33	40.62	0.25	49.94	0.83	18.88
Deepseek-7B	0.125	18.06	0.36	38.94	0.33	55.71	0.85	15.21
LLaDA-8B	0.165	**2.16**	**0.60**	**23.86**	**0.57**	**10.19**	0.87	**13.25**
Dream-7B	**0.185**	3.97	0.47	29.91	0.56	28.48	**0.89**	18.31

Table 2: Quality of reasoning thoughts. For pass@5, higher values (↑) are better; for Time, lower values (↓) are better.

4.2 Main results

Table 1 presents the performance comparison between our proposed framework and the baselines. In terms of **efficiency**, our framework significantly outperforms autoregressive models. Specifically, DT (LLADA-8B) achieves the highest throughput across benchmarks; for instance, on ARC-C, it reaches 1.05 it/min, approximately doubling that of Deepseek-7B (0.53 it/min).

Regarding **accuracy**, DT (Dream-7B) demonstrates remarkable robustness. On the challenging Trip-planning benchmark, it attains an accuracy of 0.40, vastly exceeding Llama3-8B (0.04) and Mistral-7B (0.18). This indicates that the diverse, parallel proposals generated by DLMs effectively cover the solution space in complex tasks, allowing the LLM evaluator to identify superior reasoning paths that greedy autoregressive decoding misses. Furthermore, on the expert-level GPQA benchmark, our framework maintains superior performance (0.37), proving its versatility across varying difficulty levels. Overall, these results validate that Diffuse Thinking establishes a better trade-off between inference speed and reasoning quality compared to traditional autoregressive pipelines.

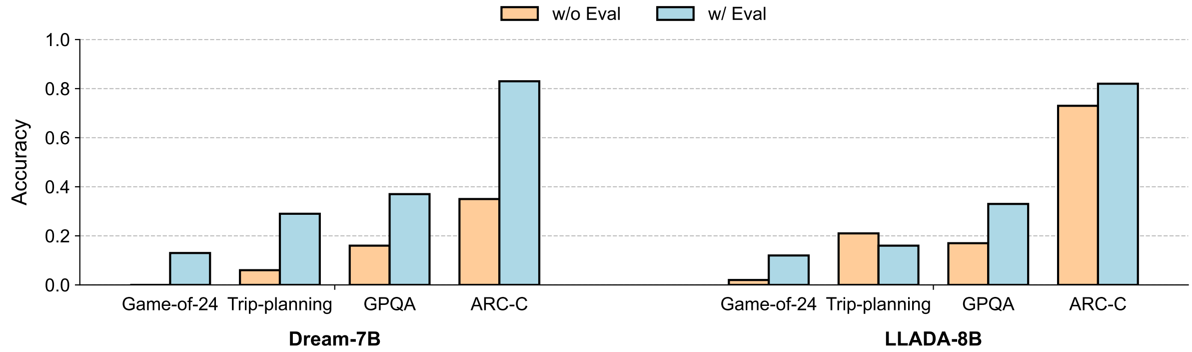

Figure 3: Ablation study on the LLM evaluator.

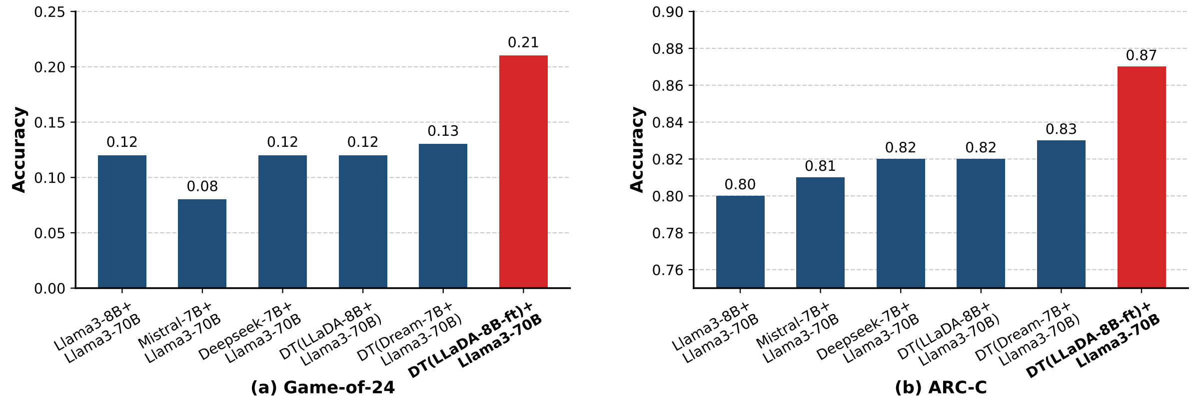

Figure 4: Performance improvements on different benchmarks after fine-tuning.

4.3 Quality of proposals

Furthermore, to examine the quality of the thoughts proposed by the DLM without finetuning, we omitted the LLM evaluator and independently assessed the quality of the thoughts generated by several proposers. The results in Table 2 clearly demonstrate that on different benchmarks, the quality of reasoning generated by DLM is better than that of the baseline models or at least comparable to the baseline models. The average accuracy advantage exceeds 10%, and the generation efficiency is also notably better, with an average throughput advantage surpassing 15%. This fully corroborates our previous complexity analysis of the DLM and LLM models.

4.4 Ablation Study

We conduct ablation studies to evaluate the necessity of the LLM-based evaluation phase. We examine the reasoning capability of DLMs in isolation.

As summarized in Figure 3, the role of the LLM evaluator is pivotal. The performance of DLMs (Dream and LLADA) degrades significantly when performing multi-step reasoning without an evaluation phase. The sharp decline, particularly in *Game-of-24*, underscores the severity of error accumulation in vanilla DLMs and validates our collaboration framework.

4.5 Learning to propose thoughts

In this subsection, we demonstrate the significant performance improvements achieved by fine-tuning our model on two benchmarks: ARC-C and Game-of-24. Figures 4 illustrates the performance gains on the ARC-C and Game-of-24 datasets, respectively. As shown in the figures, the model's performance on the ARC-C dataset increased from 13% to 21% after fine-tuning, while on the Game-of-24 dataset, the performance improved from 83% to 87%. These results underscore the effectiveness of fine-tuning DLM as an efficient proposer in enhancing our framework's reasoning and problem-solving capabilities. For detailed information on the training data, please refer to Appendix E.2.

5 Conclusion

In this paper, we propose Diffuse Thinking, a novel collaborative framework that synergizes DLMs and LLMs to structurally embody the *divergent–convergent thinking theory* in efficient reasoning. By distinguishing the roles of broad exploration and logical evaluation, our work offers a fresh paradigm for complex problem-solving. Ultimately, Diffuse Thinking lays the groundwork for the exploration of future efficient reasoning architectures, paving the way for developing more scalable and cognitively grounded AI systems.

References

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, and 1 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

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*.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *arXiv preprint arXiv:2501.15602*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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*.

Joy Paul Guilford. 1967. The nature of human intelligence.

Ishaan Gulrajani and Tatsunori B Hashimoto. 2024. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36.

Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. 2025. Mercury: Ultra-fast language models based on diffusion. https://drive.google.com/file/d/1xrqTqF88OZblf0NgMjr1REU4doYlkNXf/view.

Deyang Kong, Qi Guo, Xiangyu Xi, Wei Wang, Jingang Wang, Xunliang Cai, Shikun Zhang, and Wei Ye. 2025. Rethinking the sampling criteria in reinforcement learning for llm reasoning: A competence-difficulty alignment perspective. *arXiv preprint arXiv:2505.17652*.

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.

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

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2023. Discrete diffusion language modeling by estimating the ratios of the data distribution.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Eliya Nachmani and Shaked Dovrat. 2021. Zero-shot translation using diffusion models. *arXiv preprint arXiv:2111.01471*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*.

OpenAI. 2024a. Hello gpt-4o.

OpenAI. 2024b. Learning to reason with llms.

OpenAI. 2025. Introducing openai o3 and o4-mini. https://openai.com/zh-Hans-CN/index/introducing-o3-and-o4-mini/. Accessed: December 13, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Jonathan Scarlett and Volkan Cevher. 2019. An introductory guide to fano's inequality with applications in statistical estimation. *Preprint*, arXiv:1901.00555.

Chenyang Shao, Tianxing Li, Chenhao Pu, Fengli Xu, and Yong Li. 2025a. Agentstealth: Reinforcing large language model for anonymizing user-generated text. *ArXiv*, abs/2506.22508.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. 2025b. Route-and-reason: Scaling large language model reasoning with reinforced model router. *ArXiv*, abs/2506.05901.

Chenyang Shao, Fengli Xu, Bingbing Fan, Jingtao Ding, Yuan Yuan, Meng Wang, and Yong Li. 2024. Beyond imitation: Generating human mobility from context-aware reasoning with large language models. *ArXiv*, abs/2402.09836.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Fengli Xu, Qianyue Hao, Chenyang Shao, Zefang Zong, Yu Li, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Jie Feng, Chen Gao, and Yong Li. 2025. Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns*, 6.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

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

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and 1 others. 2024b. Diffusion of thoughts: Chain-of-thought reasoning in diffusion language models. *arXiv preprint arXiv:2402.07754*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b. Accessed: 2025-08-02.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. Diffusum: Generation enhanced extractive summarization with diffusion. *arXiv preprint arXiv:2305.01735*.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, and 1 others. 2024. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. 2023. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*.

Hao Zou, Zae Myung Kim, and Dongyeop Kang. 2023. A survey of diffusion models in natural language processing. *arXiv preprint arXiv:2305.14671*.

A Limitations

While Diffuse Thinking establishes a robust framework for efficient collaborative reasoning, we identify the scope of the selection strategy as a primary avenue for future enhancement. Currently, our framework operates on a step-wise greedy paradigm, where the evaluator selects the optimal proposal based on the immediate likelihood of the next thought. Although this approach yields competitive performance across various benchmarks, it prioritizes local optimality at each reasoning step. This modular architecture, however, presents a significant opportunity: it naturally accommodates the integration of global planning algorithms. Future work can extend this framework by incorporating look-ahead search mechanisms, such as Monte Carlo Tree Search (MCTS) or backtracking strategies, to further enhance performance on tasks requiring long-horizon strategic foresight.

B Detailed FLOPs calculation for diffusion language model

In this section, we will provide a detailed derivation and computation of the forward inference computational cost (in FLOPs) for diffusion models that use a transformer decoder as their backbone.

B.1 Symbol definitions and assumptions

The following symbols are used: sequence length $L = 4096$, model dimension $D = 4096$, embedding dimension $E = 4096$, number of attention heads $H = 32$, number of Transformer blocks $N = 32$, vocabulary size $V = 126,464$, and number of denoising steps T (variable). Moreover, F_{SA} represents FLOPs for a single self-attention operation. F_{MLP} represents FLOPs for a single feedforward network. F_{others} represents FLOPs for embedding and other operations.

B.2 FLOPs for self-attention and feedforward

The self-attention (SA) mechanism includes the following components: 1. Generating queries, keys, and values: $F_{\text{QKV}} = 2 \cdot 3 \cdot L \cdot D^2$. 2. Computing attention scores and normalization: $F_{\text{attention\_scores}} = 2 \cdot L^2 \cdot D/H$. 3. Weighted sum and output projection: $F_{\text{weighted\_sum}} + F_{\text{out\_projection}} = 2 \cdot L^2 \cdot D + 2 \cdot L \cdot D^2$.

Thus, the total FLOPs for SA is:

$$F_{\text{SA}} = 8 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot D/H \tag{11}$$

The feedforward network (MLP) consists of up-projection and down-projection, each contributing $8 \cdot L \cdot D^2$, resulting in:

$$F_{\text{MLP}} = 16 \cdot L \cdot D^2 \tag{12}$$

For one Transformer block, the total FLOPs is:

$$F_{\text{block}} = F_{\text{SA}} + F_{\text{MLP}} = 24 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot D/H \tag{13}$$

B.3 Full model FLOPs per step

Considering N Transformer blocks, the total FLOPs for the blocks is:

$$\begin{aligned} F_{\text{blocks}} &= N \cdot F_{\text{block}} \\ &= N \cdot \left(24 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot D/H\right). \end{aligned} \tag{14}$$

In addition, the embedding and output layers contribute:

$$F_{\text{others}} = 2 \cdot L \cdot D \cdot E + 2 \cdot L \cdot (D + 2 \cdot E) \cdot V \tag{15}$$

The total FLOPs per step is then:

$$F_{\text{step}} = F_{\text{blocks}} + F_{\text{others}}. \tag{16}$$

B.4 Total FLOPs for T steps

For T denoising steps, the total FLOPs is:

$$\begin{aligned} F_{\text{DLM}} =& T \cdot F_{\text{step}} \\ =& T \cdot \Big(N \cdot \left(24 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot D/H\right) \\ & +2 \cdot L \cdot D \cdot E + 2 \cdot L \cdot (D + 4 \cdot E) \cdot V\Big) \end{aligned} \tag{17}$$

Obviously, the square term of L occupies an absolute dominant position in the amount of calculation. So the time complexity can be written as:

$$T_{\text{DLM}} = \mathcal{O}\left(L^2 \cdot T\right) \tag{18}$$

It can be seen that the final time complexity is proportional to the number of denoising steps T and proportional to the square of the sequence length L. For DLMs, T is a hyperparameter that can be less than the number of tokens generated. By adjusting T, multiple tokens can be generated simultaneously, balancing quality and efficiency.

Parallel-time complexity. All matrix multiplications inside self-attention and MLP are parallelizable over the sequence dimension L, since the DLM processes the *entire sequence simultaneously* in forward passes. Assuming ideal hardware that provides $O(L)$ parallel lanes, the *wall-clock latency* per denoising step collapses to the longest dependency path, yielding

$$T_{\text{DLM}}^{\text{step}} = O\left(\frac{LD^2}{L} + \frac{L^2 D/H}{L}\right) = O(D^2 + LD/H). \tag{19}$$

Multiplying by the number of denoising steps T, the *end-to-end parallel-time complexity* is therefore

$$T_{\text{DLM}}^{\text{par}} = O(T \cdot L). \tag{20}$$

Thus, under sufficient parallelism, the latency is *linear in* T and *linear in* L, while the total FLOPs still scale as $O(L^2 T)$.

C Detailed FLOPs calculation for autoregressive LLM

In this section, we provide a detailed derivation of the forward inference computational cost (in FLOPs) for decoder-only autoregressive large language models. The analysis includes key components such as attention, feedforward layers, and token embeddings.

C.1 Definitions and notations

We define the following symbols:

- L: Sequence length.
- D: Model dimension.
- H: Number of attention heads.
- N: Number of Transformer blocks.
- V: Vocabulary size.
- E: Embedding dimension (equal to D).

C.2 FLOPs of each module

Embedding Layer

- Token embedding: $F_{\text{emb}} = 2 \cdot L \cdot D$.
- Output projection: $F_{\text{out}} = 2 \cdot L \cdot D \cdot V$.

Thus, the total FLOPs for the embedding layer is:

$$F_{\text{embedding}} = 2 \cdot L \cdot D + 2 \cdot L \cdot D \cdot V. \tag{21}$$

Multi-Head Self-Attention Mechanism (MHSA) For a single Transformer block:

- Linear projections for Q, K, and V: $6 \cdot L \cdot D^2$.(Consider KV-cache: $6 \cdot D^2$)
- Attention score computation: $F_{\text{attn\_scores}} = 4 \cdot L^2 \cdot \frac{D}{H}$.(Consider KV-cache: $4 \cdot L \cdot \frac{D}{H}$)
- Output projection: $F_{\text{attn\_out}} = 2 \cdot L \cdot D^2$.(Consider KV-cache: $2 \cdot D^2$)

The total FLOPs for self-attention is:

$$F_{\text{SA}} = 8 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot \frac{D}{H} \quad \text{(Consider KV-cache: } 8 \cdot D^2 + 4 \cdot L \cdot \tfrac{D}{H}\text{).} \tag{22}$$

Feedforward Network (FFN) The FFN consists of two linear transformations:

- Up projection: $F_{\text{ffn\_up}} = 8 \cdot L \cdot D^2$.(Consider KV-cache: $8 \cdot D^2$).
- Down projection: $F_{\text{ffn\_down}} = 8 \cdot L \cdot D^2$.(Consider KV-cache: $8 \cdot D^2$).

Thus, the total FLOPs for the FFN is:

$$F_{\text{FFN}} = 16 \cdot L \cdot D^2.\text{(Consider KV-cache: } 16 \cdot D^2\text{)} \tag{23}$$

Transformer Block The total FLOPs for a single Transformer block is:

$$F_{\text{block}} = F_{\text{SA}} + F_{\text{FFN}} = 24 \cdot L \cdot D^2 + 4 \cdot L^2 \cdot \frac{D}{H}. \tag{24}$$

$$\text{(Consider KV-cache: } 24 \cdot D^2 + 4 \cdot L \cdot \tfrac{D}{H}\text{)} \tag{25}$$

So for N Transformer blocks, the total FLOPs are:

$$F_{\text{blocks}} = N\big(24 \cdot L \cdot D^2\big) + N\big(4 \cdot L^2 \cdot \tfrac{D}{H}\big), \quad \text{(Consider KV-cache: } F'_{\text{blocks}} = N\big(24 \cdot D^2\big) + N\big(4 \cdot L \cdot \tfrac{D}{H}\big).\text{)} \tag{26}$$

C.3 Total generation FLOPs

Since each token in the autoregressive language model is generated sequentially, each step requires recomputing self-attention and feedforward layers. The sequence length at each step increases from L_{in} to $L_{\text{in}} + L_{\text{out}}$. Thus, we approximate the total FLOPs for generation as:

$$F_{\text{total}} = F_{\text{embedding}} + L_{\text{out}} \cdot F_{\text{blocks}}\Big|_{L \to L_{\text{in}}+L_{\text{out}}}$$
$$= 2 \cdot L_{\text{in}} \cdot D + 2 \cdot L_{\text{out}} \cdot D \cdot V + 24 \cdot N \cdot L_{\text{out}} \cdot (L_{\text{in}} + L_{\text{out}}) \cdot D^2 + 4 \cdot N \cdot L_{\text{out}} \cdot (L_{\text{in}} + L_{\text{out}})^2 \cdot \frac{D}{H}. \quad (27)$$

(Consider KV-cache:

$$F_{\text{total}} = F_{\text{embedding}} + L_{\text{out}} \cdot F'_{\text{blocks}}\Big|_{L \to L_{\text{in}}+L_{\text{out}}} + F_{\text{blocks}}\Big|_{L \to L_{\text{in}}}$$
$$= 2 \cdot L_{\text{in}} \cdot D + 2 \cdot L_{\text{out}} \cdot D \cdot V + 24 \cdot N \cdot L_{\text{out}} \cdot D^2 + 4 \cdot N \cdot L_{\text{out}} \cdot (L_{\text{in}} + L_{\text{out}}) \cdot \frac{D}{H} + 24 \cdot N \cdot L_{\text{in}} \cdot D^2 + 4 \cdot N \cdot L_{\text{in}}^2 \cdot \frac{D}{H}.) \quad (28)$$

Expanding terms:

$$F_{\text{total}} = 2 \cdot L_{\text{in}} \cdot D + 2 \cdot L_{\text{out}} \cdot D \cdot V + 24 \cdot N \cdot L_{\text{out}} \cdot L_{\text{in}} \cdot D^2 + 24 \cdot N \cdot L_{\text{out}}^2 \cdot D^2 + 4 \cdot N \cdot L_{\text{out}} \cdot L_{\text{in}}^2 \cdot \frac{D}{H} + 8 \cdot N \cdot L_{\text{out}} \cdot L_{\text{in}} \cdot L_{\text{out}} \cdot \frac{D}{H} + 4 \cdot N \cdot L_{\text{out}}^3 \cdot \frac{D}{H}. \quad (29)$$

(Consider KV-cache:

$$F_{\text{total}} = \left(2 \cdot D \cdot V + 24 \cdot N \cdot D^2\right) \cdot L_{\text{out}} + \left(2 + 24 \cdot N \cdot D\right) \cdot D \cdot L_{\text{in}} + 4 \cdot N \cdot \frac{D}{H} \cdot L_{\text{in}}^2 + 4 \cdot N \cdot \frac{D}{H} \cdot L_{\text{out}}^2 + 4 \cdot N \cdot \frac{D}{H} \cdot L_{\text{in}} \cdot L_{\text{out}}.) \quad (30)$$

Asymptotic Complexity:

$$F_{\text{total}} = \mathcal{O}\left(L_{\text{out}}^3\right) \quad (31)$$
$$(\text{Consider KV-cache: } F_{\text{total}} = \mathcal{O}\left(L_{\text{out}}^2\right)). \quad (32)$$

Sequential-time complexity. Unlike diffusion language models, autoregressive language models generate tokens one by one; each new token depends on all previously generated tokens. Consequently, self-attention and MLP layers cannot be parallelized across the sequence dimension L_{out}, even under ideal hardware.

C.4 Batch generation comparison

Assuming the same model architecture, computational resource, parameter sizes, and input/output lengths($L_{in} + L_{out} = L$), the difference in GPU memory occupied by DLM and LLM in the inference process lies in the activation values and KV-cache. For LLM, the number of parameters in the KV-cache can be expressed as:

$$Memory_{KV_{cache}} = 2 \cdot N \cdot K_{LLM} \cdot D \cdot (L_{in} + L_{out}). \quad (33)$$

As for activation values, only the most numerous parts need to be counted:

- MHSA: $Memory_{MHSA} = K_{LLM} \cdot L_{in}^2 \cdot H$.
- FFN: $Memory_{FFN} = 4 \cdot K_{LLM} \cdot L_{in} \cdot D$.

For DLM, there is no KV-cache, only the activation values need to be considered:

- MHSA: $Memory_{MHSA} = K_{DLM} \cdot L^2 \cdot H$.
- FFN: $Memory_{FFN} = 4 \cdot K_{DLM} \cdot L \cdot D$.

If $4 \cdot D \geq L \cdot H$ (typically, and in line with our experimental setups):

$$K_{DLM}/K_{LLM} = \frac{2L_{\text{in}} + N(L_{\text{in}} + L_{\text{out}})}{2(L_{\text{in}} + L_{\text{out}})}. \quad (34)$$

Substituting $N = 32$:

$$K_{DLM}/K_{LLM} > 16. \quad (35)$$

If $L \cdot H > 4 \cdot D \geq L_{in} \cdot H$:

$$K_{DLM}/K_{LLM} = \frac{4L_{\text{in}} \cdot D + 2 \cdot D \cdot N \cdot (L_{\text{in}} + L_{\text{out}})}{H \cdot (L_{\text{in}} + L_{\text{out}})^2}. \quad (36)$$

Substituting $H = 32$,$N = 32$:

$$K_{DLM}/K_{LLM} > 2 \cdot D/L. \quad (37)$$

If $L_{in} \cdot H > 4 \cdot D$:

$$K_{DLM}/K_{LLM} = \frac{L_{\text{in}}^2 \cdot H + 2 \cdot D \cdot N \cdot (L_{\text{in}} + L_{\text{out}})}{H \cdot (L_{\text{in}} + L_{\text{out}})^2}. \quad (38)$$

Substituting $H = 32$,$N = 32$:

$$K_{DLM}/K_{LLM} > 2 \cdot D/L. \quad (39)$$

D Error analysis in diffusion language models

We extend the information-theoretic approach in Gan et al. (Gan et al., 2025) to analyze diffusion language models. Diffusion models generate a sequence X_0 of length L by iteratively denoising an initial noise sequence X_T over T steps: $X_T \to X_{T-1} \to \cdots \to X_1 \to X_0$. A common practice is to predict tokens in parallel at each step for efficiency.

D.1 Information loss from independence assumption

Let $X_t = (x_1^t, \ldots, x_L^t)$ be the sequence at denoising step t. Consider the transition from X_{t-1} to X_t. The true conditional probability is $p(X_t|X_{t-1})$, and the associated conditional entropy is $H(X_t|X_{t-1})$.

Many diffusion models approximate the reverse process by assuming conditional independence of tokens given the previous state:

$$p_\theta(X_t|X_{t-1}) = \prod_{i=1}^{L} p_\theta(x_i^t|X_{t-1}) \quad (40)$$

Under this assumption, the conditional entropy calculated by the model is:

$$H_{\text{indep}}(X_t|X_{t-1}) = \sum_{i=1}^{L} H(x_i^t|X_{t-1}) \quad (41)$$

where the entropy $H(x_i^t|X_{t-1})$ is computed based on the marginal $p_\theta(x_i^t|X_{t-1})$.

As derived from the chain rule of entropy and the property that conditioning reduces entropy:

$$\begin{aligned} H(X_t|X_{t-1}) &= \sum_{i=1}^{L} H(x_i^t|x_1^t, \ldots, x_{i-1}^t, X_{t-1}) \\ &\leq \sum_{i=1}^{L} H(x_i^t|X_{t-1}) \\ &= H_{\text{indep}}(X_t|X_{t-1}) \end{aligned} \quad (42)$$

Thus, the independence assumption leads to an overestimation of the conditional entropy:

$$H(X_t|X_{t-1}) \leq H_{\text{indep}}(X_t|X_{t-1}) \quad (43)$$

This implies an underestimation of the mutual information between consecutive steps:

$$I(X_t; X_{t-1}) = H(X_t) - H(X_t|X_{t-1}) \quad (44)$$

$$I_{\text{indep}}(X_t; X_{t-1}) = H(X_t) - H_{\text{indep}}(X_t|X_{t-1}) \quad (45)$$

$$\implies I_{\text{indep}}(X_t; X_{t-1}) \leq I(X_t; X_{t-1}) \quad (46)$$

Definition 1 (Per-Step information loss in diffusion) *The information loss at denoising step t due to the independence assumption is the difference between the model's estimated conditional entropy and the true conditional entropy:*

$$\Delta H_t = H_{\textit{indep}}(X_t|X_{t-1}) - H(X_t|X_{t-1}) \geq 0 \quad (47)$$

Alternatively, it's the difference in mutual information:

$$\Delta I_t = I(X_t; X_{t-1}) - I_{\textit{indep}}(X_t; X_{t-1}) \geq 0 \quad (48)$$

This loss ΔI_t quantifies the information about the dependencies between tokens x_i^t within step t (given X_{t-1}) that is discarded by the model. This loss is expected to increase with sequence length L.

D.2 Cumulative loss and final error probability

Let X^* be the ground truth sequence the model aims to generate. The difference in mutual information hinders the model's ability to maximize the mutual information between the final output X_0 and the target X^*, denoted $I(X^*; X_0)$.

Let $I_{\text{ideal}}(X^*; X_0)$ be the mutual information achievable by an ideal T-step diffusion process (without the independence assumption), and $I_{\text{indep}}(X^*; X_0)$ be the mutual information achieved by the model using the independence assumption. The total accumulated information loss is:

$$\begin{aligned} \text{TotalLoss}(T, L) = I_{\text{ideal}}(X^*; X_0) - I_{\text{indep}}(X^*; X_0) \\ \geq 0 \end{aligned} \quad (49)$$

This loss reflects the cumulative impact of ΔI_t for $t = T, \ldots, 1$.

Let $E_{Diff}(T, L) = P(X_0 \neq X^*)$ be the final error probability. Using Fano's inequality on the pair (X^*, X_0):

$$\begin{aligned} H(X^*|X_0) \leq H(E_{Diff}(T, L)) \\ + E_{Diff}(T, L) \log(|\mathcal{X}| - 1) \end{aligned} \quad (50)$$

where $\mathcal{X}$ is the space of possible sequences.

The conditional entropy for the actual process is:

$$H_{\text{indep}}(X^*|X_0) = H(X^*) - I_{\text{indep}}(X^*; X_0) \quad (51)$$

Substituting the definition of TotalLoss:

$$H_{\text{indep}}(X^*|X_0) = H(X^*) - (I_{\text{ideal}}(X^*; X_0) \\ - \text{TotalLoss}(T, L)) \tag{52}$$

$$H_{\text{indep}}(X^*|X_0) = (H(X^*) - I_{\text{ideal}}(X^*; X_0)) \\ + \text{TotalLoss}(T, L) \tag{53}$$

$$H_{\text{indep}}(X^*|X_0) = H_{\text{ideal}}(X^*|X_0) \\ + \text{TotalLoss}(T, L) \tag{54}$$

where $H_{\text{ideal}}(X^*|X_0)$ is the conditional entropy of the ideal process.

Plugging this back into the Fano inequality bound:

$$H_{\text{ideal}}(X^*|X_0) + \text{TotalLoss}(T, L) \leq \\ H(E_{Diff}(T, L)) + E_{Diff}(T, L) \log(|\mathcal{X}| - 1) \tag{55}$$

Implication: This inequality shows that the final error probability $E_{Diff}(T, L)$ is lower-bounded by a term that grows with the total accumulated information loss $\text{TotalLoss}(T, L)$ resulting from the independence assumption across the T steps.

D.3 Dependence on L and T

Sequence length (L): Since the per-step loss ΔI_t likely increases with L (more dependencies to ignore), the cumulative loss $\text{TotalLoss}(T, L)$ also increases with L. Therefore, the final error $E_{Diff}(T, L)$ is expected to increase with L.

Number of steps (T): The dependence is complex. Increasing T allows more denoising (information recovery about X^*, reducing $H_{\text{ideal}}(X^*|X_0)$). Empirically, $E_{Diff}(T, L)$ decreases with T up to a point, indicating the denoising benefit usually dominates. However, the minimum error is limited by $\text{TotalLoss}(T, L)$.

E Discussions

E.1 Implementation details

Here we provide detailed experimental settings to facilitate the reproducibility of our results in Table 3. All experiments were run three times except for the scaling of proposals for one time.

Table 3: **Detailed Experimental Settings**

Module	Element	Detail
System	Operating System	Ubuntu 20.04.6 LTS
	CUDA Version	12.4
	Python Version	3.10.16
	PyTorch	2.6.0
	DeepSpeed	0.16.8
	Accelerate	1.4.0
	PEFT	0.15.1
	Compute Device	1 × NVIDIA A100 (80G)
Model	Base Model	Llama3-70B
	Inference Engine	vLLM (0.9.2)
	Tensor Parallel	2
	Data Type	FP16
	Context Window	8,192
Training	Fine-tuning Mode	LoRA
	Batch Size	2
	Training Epochs	1
	Max Token Length	4,096
	LoRA Rank (r)	128
	Optimizer	AdamW
	Learning Rate	1×10^{-5}

E.2 Code of ethics

In this paper, we use open-source models, which involve no problem regarding privacy and copyright. We use open-source datasets and self-constructed datasets, which involve no problem regarding privacy and copyright. Except for Game-of-24, we all use open-source datasets. For the Game-of-24 , we exhaustively enumerated number combinations and employed a backtracking approach to solve it. We considered all non-repetitive, diverse solutions, resulting in a dataset of 168,046 24-point problems, with the maximum value for the four numbers limited to 30 to make it more challenging. We have cited all open-source resources. Our project code has also been released and is available through the following anonymous link: `https://anonymous.4open.science/r/Diffuse-Thinking-EC60`. The self-constructed data used in this paper is also released at the provided anonymous link.

E.3 Further Comparison with Prior Works

Table 4 contrasts our hybrid framework with DiffuLLaMA (Gong et al., 2024) and DeepSeek-R1-8B. Our work is a hybrid framework rather than pure DLM solutions like DoT(Ye et al., 2024b) and DiffuLLaMA. We add DiffuLLaMA as a baseline. Its performance is lower than ours, because a pure DLM solution cannot correct wrong thoughts with AR LLM. We also compare our framework with a SOTA reasoning model: DeepSeek R1 8B (Guo et al., 2025)(we use small scale version for fairness). Its performance is significantly lower than ours, highlighting the advantage of collaborative framework.

Table 4: Extra baselines on four reasoning benchmarks (accuracy).

Model	**Game of 24**	**Trip planning**	**GPQA**	**ARC-c**
DiffuLLaMA	0.00	0.00	0.03	0.01
DeepSeek-R1-8B	0.02	0.00	0.04	0.00
Ours	**0.13**	**0.29**	**0.37**	**0.83**

F Diversity analysis

Game of 24 is rule-simple and solution-diverse, making it the cleanest test-bed for studying generation diversity. As shown in Figure 5, the cumulative count of unique solutions keeps rising even after 50 proposals, yet the slope differs dramatically across models. Dream leads the pack with the steepest slope, ending near 30 distinct expressions, while Deepseek-7B trails at 25; both still climb, yet Dream ascends faster. Mistral-7B and Llama3-8B, in comparison, level off quickly around 15 and 10 solutions, revealing a tendency to recycle once common templates are found.

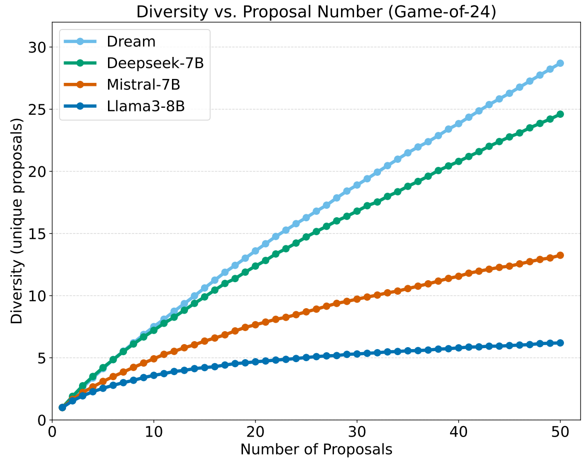

Figure 5: Diversity vs. Proposal Number (Game-of-24).

G Case study

To better understand the mechanism of our framework, we examined a representative case study on the game of 24.

We have four numbers: 1, 14, 16, and 25. We provide DLM with prompt:

Prompt

```
Here is a task for you: use these
    numbers <<1,14,16,25>>  to
    obtain 24 through the basic
    operation of (+- */). Each
    number can only be used once
    and must be used.
Please output the next possible
    operation directly for only one
     line, in the format of:
    Equation (remaining numbers)
```

DLM generates the following text:

Response

```
'16-1=15 (14,25,15)', '16-1=15
    (14,25,15)', '1*25=25
    (15,16,25)', '16*6=15
    (14,25,15)', '16-1=15
    (14,25,15)', '1*15=15
    (25,14,15)', '1*25=25
```

```
    (25,14,25)', '14+1=15
    (16,25,15)', '14+1=15
    (16,25,15)', '1+14=15
    (16,25,15)', '1*11=15
    (14,25,15)', '1+25=16
    (25,16,25)', '14+1=15
    (25,16,15)', '16+6=17
    (14,25,17)', '20-1=19
    (14,16,19)', '16-1=15
    (25,16,17)', '1*15=15
    (25,14,16)', '16-1=15
    (14,25,15)', '16+1=17
    (14,25,17)', '1*25=16
    (26,14,16)'
```

Then we label them and provide LLM with prompt:

Prompt

```
You must consider whether the
    expression calculation in the
    next thought proposal<<>>is
    correct,
whether the number on the left side
     of the equation is in the
    remaining<<>>,
whether the number on the right
    side of the equation is in the
    left<<>>, and
whether the number in the left<<>>
    is only '24' left or more
    likely to achieve 24 through
    basic arithmetic operations (+-
     */).
Here are some candidate solutions
    for the next step.Their serial
    numbers are in [].[1]16-1=15
    (14,25,15),[2]16-1=15
    (14,25,15), [3]1*25=25
    (15,16,25), [4]16*6=15
    (14,25,15), [5]16-1=15
    (14,25,15), [6]1*15=15
    (25,14,15), [7]1*25=25
    (25,14,25), [8]14+1=15
    (16,25,15), [9]14+1=15
    (16,25,15), [10]1+14=15
    (16,25,15), [11]1*11=15
    (14,25,15), [12]1+25=16
    (25,16,25), [13]14+1=15
    (25,16,15), [14]16+6=17
    (14,25,17), [15]20-1=19
    (14,16,19), [16]16-1=15
    (25,16,17), [17]1*15=15
    (25,14,16), [18]16-1=15
    (14,25,15), [19]16+1=17
    (14,25,17), [20]1*25=16
    (26,14,16)
Please choose the best one and tell
     me the serial number you have
    chosen.OUTPUT FORMAT:'Reasons
    :....  [serial  number]'.
```

LLM responses:

Response

```
After analyzing the given options,
    I found that most of the
    correct calculations are not
    achieving the target of getting
     closer to 24 or ensuring that
    the numbers involved are or can
     lead to 24 through basic
    arithmetic operations.

The correct calculations that lead
    to correct results but do not
    necessarily help achieve 24 or
    are not the focus based on the
    provided context are numerous.
    However, to identify the best
    candidate that fits the
    criteria of correct calculation
     and the potential to
    contribute to reaching 24 or
    being part of a sequence that
    could, with further operations,
     result in 24, I must consider
    the arithmetic operations'
    correctness and the numbers
    involved.

Given the constraints and looking
    at the patterns and potential
    for leading to 24, I notice
    that many of these equations
    simply demonstrate arithmetic
    operations without a clear
    progression towards achieving
    24. To select the best option,
    I would look for operations
    that either directly involve
    numbers close to 24 or
    demonstrate a path that could,
    with additional operations,
    lead to 24.

However, since most provided
    equations do not directly aim
    at achieving 24 or do not
    specify the next steps in
    calculations that would lead to
     24, the choice seems to be
    more about identifying correct
    arithmetic rather than
    progression towards a specific
    goal.

Given this context and looking for
    an option that seems plausible
    or directly correct in its
    arithmetic without the
    additional context of how it
    leads to 24 (since such context
     is not provided in most
    options), I would choose an
    equation that is both correct
    and involves numbers that could
     potentially lead to
    interesting further
    calculations.
```

```
Considering the arithmetic
    correctness and the involvement
     of numbers that could be part
    of further calculations leading
     to 24 (though none directly
    achieve this), I pick an option
     that demonstrates a
    straightforward and correct
    arithmetic operation.

Reasons: The equation is simple,
    the arithmetic is correct, and
    it involves basic addition
    which could be a starting point
     for further operations.
    However, given the constraint
    and looking for the best
    representation of arithmetic
    that could lead to further
    calculations, [8] stands out
    for its simplicity and
    correctness: 14+1=15, which is
    a straightforward and correct
    operation, even though it doesn
    't directly aim for 24.
[8]
```

We obtain [8] 14+1=15 (16, 25, 15), and then this process will continue to iterate until the problem is solved.