

DIP: Dynamic In-Context Planner For Diffusion Language Models

Yang Li¹ Han Meng¹ Chenan Wang¹ Zhenyu Bi² Xuan Wang² Haipeng Chen¹

College of William & Mary¹, Virginia Tech²

{yli102, hmeng, cwang33, hchen23}@wm.edu¹, {zhenyub, xuanw}@vt.edu²

Abstract

Diffusion language models (DLMs) have shown strong potential for general natural language tasks with in-context examples. Existing In-Context Learning (ICL) approaches largely inherit the practice of autoregressive language models (ARLMs), incorporating all examples into a fixed prompt. However, applying this rigid, static-prompt paradigm to DLMs incurs substantial computational overhead, as the model must evaluate the maximum context length at every step. We address this inefficiency with a key discovery: the block-wise KV-cache mechanism inherent to DLM inference enables the *low-cost dynamic adjustment of the context*. Following this intuition, our core idea is to start generation with a minimal prompt and progressively insert additional examples on the fly only when the generated tokens are of low confidence. Through rigorous empirical evaluations, we observe that average verified token confidence correlates strongly with generation accuracy, making it a reliable and computationally efficient signal of token quality. Formally, we propose **Dynamic In-Context Planner (DIP)**, a context-optimization algorithm based on average verified confidence that dynamically ranks and inserts in-context examples during generation, rather than providing all examples up front. Experimental results on math and coding benchmarks with LLaDA-1.5 and LLaDA-8B-Instruct show that DIP achieves up to $1.59\times$ and $1.36\times$ speedups, respectively, while largely preserving the generation quality of the fixed-prompt baseline. Code: <https://github.com/wmd3i/DIP>

1 Introduction

Diffusion language models have emerged as a new paradigm for language modeling, demonstrating strong potential across general natural language tasks (Nie et al., 2025; Ye et al., 2025; Austin et al., 2021a; DeepMind, 2025; Arriola et al., 2025; Lou et al., 2024; Sahoo et al., 2024). DLMs’ decod-

ing mechanism differs fundamentally from that of ARLMs: By utilizing bidirectional attention and iterative refinement, DLMs remove the strict left-to-right constraint inherent to AR generation. This paradigm shift offers fine-grained control over the generation process, enabling flexible editing and global planning (Zhang et al., 2025a; Jin et al., 2025; Li et al., 2025; Chen et al., 2026).

Similar to ARLMs, DLMs also benefit from In-Context Learning (ICL) (Nie et al., 2025), in which providing in-context examples improves performance on many retrieval-augmented generation tasks (Gao et al., 2023). Existing ICL approaches for DLMs largely inherit the AR convention (Radford et al., 2018; Bi et al., 2025), which keeps the prompt fixed and uses all in-context examples as the prefix throughout generation (Radford et al., 2018; Dong et al., 2024). While this practice is reasonable in ARLMs, the higher computational cost of bidirectional attention in DLMs significantly limits inference speed as the context size grows.

To reduce computational cost in DLMs, recent works have explored KV caching (Liu et al., 2025; Song et al., 2026), parallel decoding (Jiang et al., 2026; Wu et al., 2026), and block-wise decoding (Wu et al., 2026). While these approaches successfully mitigate the overhead of processing static prompts, they overlook a critical source of computational waste: the rigid assumption that the model requires a maximum fixed context length throughout the generative process.

This work addresses this issue with a **key discovery**: unlike ARLMs, which require a fixed context, the block-wise KV-cache mechanism inherent in DLM inference enables the *dynamic adjustment of context* across decoding blocks at low computational cost. Exploiting this mechanism, our core idea is to start generation with a shorter prompt and progressively increase the context length on the fly, thereby significantly reducing the required attention computation without sacrificing quality. To

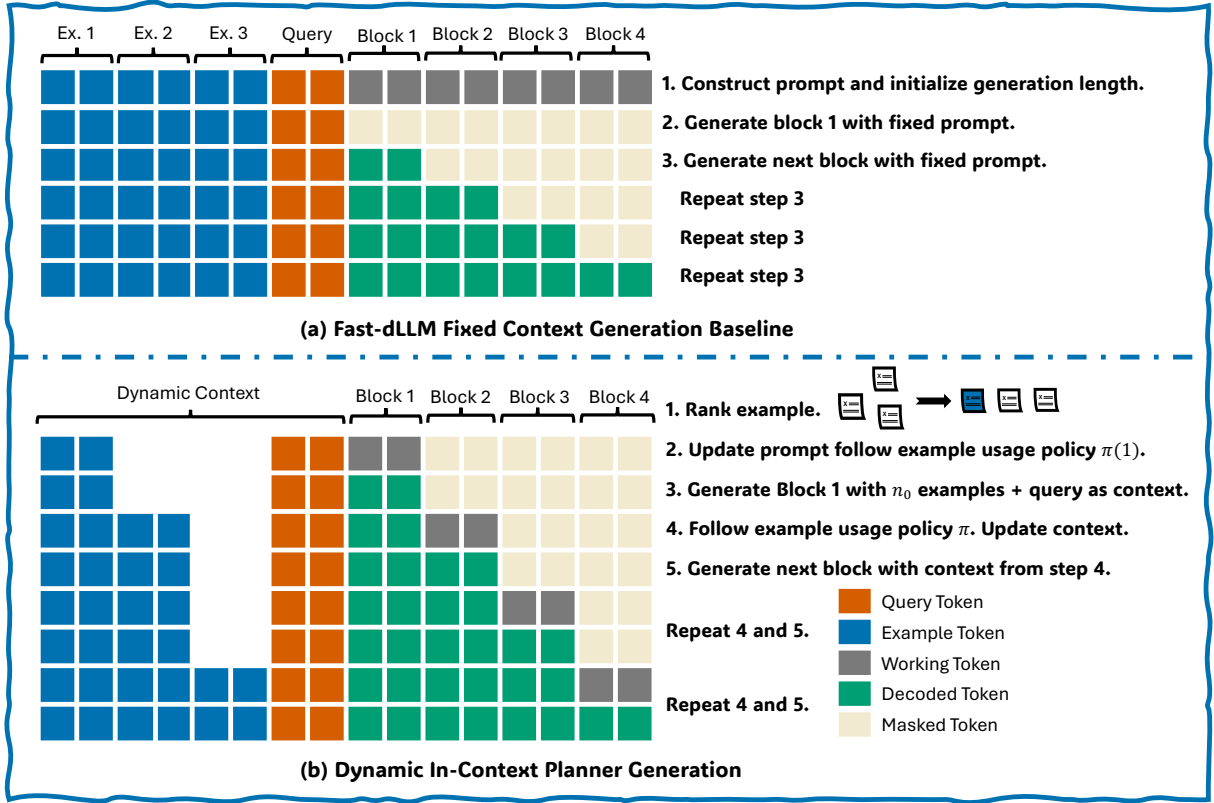


Figure 1: Inference process of fixed context generation baseline and dynamic in-context planner inference using Fast-dLLM (Wu et al., 2026) inference framework. (a) Illustration of the Fast-dLLM generation process under a fixed context setting, which is commonly used in 3-shot in-context learning scenarios. (b) Illustration of a dynamic context policy π based generation process. During decoding, each block may use a different number of in-context examples, depending on the policy. A more detailed example can be found in Appendix A.

make this concrete, consider a few-shot GSM8K task. In ARLMs, all examples must be rigidly loaded into the prompt before generation begins. By contrast, as illustrated in Fig. 1 part (b), the block-wise KV cache enhanced diffusion generation paradigm allows for generation with varying numbers of examples across different stages of the process (e.g., 1-shot at block 1, 2-shot at block 2 and 3, and 3-shot at block 4). By dynamically adapting the number of in-context examples to the evolving needs of the generation, we can significantly reduce computational cost in earlier decoding stages while leveraging the full set of in-context examples in later stages, without compromising generation quality.

A critical question naturally emerges: on what basis should we dynamically adjust the context? To fully exploit this architectural freedom, we need a reliable, real-time proxy to tell us when the model is struggling. Motivated by recent work on predictive metrics and test-time methods (Fu et al., 2026; Li et al., 2026; Geng et al., 2024), we investigated the interplay among predictive metrics, task

accuracy, and the number of in-context examples. Our analysis revealed clear empirical patterns of the average verified confidence for the generated tokens, making it an ideal signal for our selection policy: (1) correct and incorrect answers exhibit a distinct distributional difference in the average verified confidence of the generated tokens (Fig. 2); (2) scaling up the number of in-context examples reliably boosts confidence and accuracy (Fig. 3).

Building on these insights, we propose **Dynamic In-Context Planner (DIP)**, an average verified confidence-based context-optimization algorithm that dynamically selects in-context examples during generation. DIP consists of a sentence embedding-based example-ranking stage and an average verified confidence-aware example-insertion policy that makes insertion decisions during generation. DIP is training-free and can serve as a plug-in method that efficiently selects examples and updates the prompt using prominent DLM inference frameworks like Fast-dLLM (Wu et al., 2026).

Our main contributions include: (1) We demonstrate that DLMs’ generation paradigm allows *low-*

cost dynamic context adjustment, breaking the rigid static-prompt assumption of traditional ARLMs, opening a new avenue for efficient DLMs inference based on dynamic context optimization. (2) We investigate DLMs’ predictive metrics, establishing that average verified confidence could serve as an accurate, real-time signal for when to inject new examples. (3) We introduce DIP, a practical context optimization framework that dynamically scales in-context examples during generation based on verified confidence. (4) Experimental results on math and coding benchmarks with LLaDA-1.5 and LLaDA-8B-Instruct show that DIP achieves up to $1.59\times$ and $1.36\times$ speedups, respectively, while largely preserving the generation quality of the fixed-prompt baseline.

2 Related Work

In-Context Learning. ICL enables language models to perform tasks by conditioning on a few input-output demonstrations in the prompt without parameter updates (Brown et al., 2020; Dong et al., 2024). Prior work has shown that the choice of demonstrations significantly affects performance: Min et al. (2022) find that format and input distribution matter more than label correctness, while Liu et al. (2022) and Rubin et al. (2022) demonstrate that retrieving semantically similar examples substantially outperforms random selection. However, in ARLMs, all examples must be fixed before generation begins. Recent DLMs (Nie et al., 2025; Ye et al., 2025; Arriola et al., 2025) are free from this causal constraint, opening new possibilities for ICL. For instance, Jin et al. (2025) explore in-place condition injection. Yet, all existing approaches still fix the prompt before generation.

Efficient Inference in DLMs. Masked DLMs use bidirectional attention, which typically scales quadratically with the input and generation lengths. Following the success of KV cache reuse in ARLMs (Kwon et al., 2023; Zheng et al., 2024), recent works, like Fast-dLLM (Wu et al., 2026) and dKV-Cache (Ma et al., 2025), have achieved significant inference speedup with KV cache reuse (Jiang et al., 2026; Song et al., 2026; Liu et al., 2025; Ma et al., 2025). Other lines of work also explore parallel decoding (Wu et al., 2026; Israel et al., 2025; Zhang et al., 2025a) to sample multiple tokens per iteration, thereby improving inference speed. Despite these initial successes, these methods focus solely on the computational cost of a static prompt

and ignore the opportunity to dynamically adjust the context used for generation.

Token Confidence In Language Models. Token confidence or logits have gained increasing attention in language model research. They are not only useful for token generation but also reveal many insights into the generated tokens. In ARLMs, recent work has used reasoning-trace-level confidence to enable effective test-time scaling and trace selection (Fu et al., 2026; Kang et al., 2025). Token confidence in DLMs could also reveal insights, such as whether the generation length is sufficient based on the EOS token’s confidence (Li et al., 2026). Despite these initial successes, the correlation between DLM’s trace-level confidence and the number of in-context examples remains poorly understood.

We aim to fill these research gaps by exploring DLMs’ token confidence and dynamically adjusting in-context examples during diffusion generation, exploiting the unique flexibility of DLMs’ bidirectional attention.

3 Preliminary: Masked DLMs

Masked DLMs (absorbing state discrete diffusion models) (Sahoo et al., 2024; Zhu et al., 2026; Ye et al., 2025; Austin et al., 2021a) consist of a forward and reverse process, where an absorbing state is a mask token, denoted as [MASK], which represents missing or corrupted information (Lou et al., 2024; Nie et al., 2025).

In the *forward* process, q with an initial sequence y_0 of length N is organized into discrete *time steps*, where each time step corresponds to a single masking stage. At each time step, independent tokens are gradually replaced by [MASK] tokens with probability $t \in [0, 1]$, which is defined as:

$$q(y_t|y_0) = \prod_{i=0}^{N-1} q(y_t^i|y_0^i), \quad (1)$$

where $i \in [0, N]$ and y^i stands for the i -th token. Once a token transitions to [MASK], it remains in this state throughout the forward process. As t increases to 1, the original sequence will eventually become a fully masked sequence.

Conversely, in the *reverse* process p_θ , at each time step, masked DLMs aim to predict the probability distribution over all masked tokens simultaneously given y_t , which can be formulated as:

$$p_\theta(y_s|y_t) = \prod_{i=0}^{N-1} p_\theta(y_s^i|y_t^i). \quad (2)$$

Based on these predictions, the additional decoding method D decides which tokens to "unmask" at each step. As t decreases to 0, the masked sequence will eventually recover the original sequence.

4 Accelerating DLMs via Dynamic Context Adaptation

ARLMs via Causal Attention. ARLMs (Hu et al., 2024) employ causal attention (Vaswani et al., 2017) for sequential generation, restricting the contextualized representation h_i to left-to-right dependencies (Radford et al., 2018; Brown et al., 2020). This causal masking can be formally expressed by bounding key and value indices to $\leq i$:

$$h_i = \text{Attention}(x_i W_Q, x_{\leq i} W_K, x_{\leq i} W_V). \quad (3)$$

Because the structural context is rigidly bound to the prefix, the key and value are cached sequentially to avoid redundant computation.

As a result, ARLMs are fundamentally inefficient in dynamic context scenarios during inference. Any mid-generation modification to the context instantly forces a recomputation of the invalidated sequence. Rebuilding this cache from scratch imposes a computational cost of approximately:

$$O((L_{\text{new}} + L_{\text{gen}})^2), \quad (4)$$

where L_{new} is the length of the new context and L_{gen} is the length of generated tokens so far.

DLMs via Full Attention. In contrast, masked DLMs (Zhu et al., 2026; Ye et al., 2025) employ full attention (Devlin et al., 2019), where the i -th token attends to the entire sequence of length L to compute its representation h_i :

$$h_i = \text{Attention}(x_i W_Q, x_{1:L} W_K, x_{1:L} W_V). \quad (5)$$

During generation, DLMs iteratively refine a fully masked sequence by predicting and unmasking tokens across multiple denoising steps. Because this full attention mechanism inherently recomputes the entire sequence at each step, updating the context mid-generation does not trigger the cache-rebuild penalty observed in ARLMs. Yet, this full attention mechanism in DLMs renders long contexts computationally expensive.

Key Discovery: *Low-cost dynamic context adaptation in DLMs allows us to actively adjust the context mid-generation, achieving substantial computational savings.* In practice, recent KV cache inference frameworks such as Fast-dLLM (Wu et al.,

2026) adopt block-wise generation¹, which divides the output into N equal-length blocks and performs semi-AR decoding at the block level. Crucially, at each block boundary, a full forward pass is already required to recompute the attention states for the next block. This provides a natural, low-overhead opportunity to update the prompt (e.g., inserting an in-context example) without incurring penalties beyond basic tokenization and matrix concatenation.

Given a full static prompt containing all in-context examples with context length of L_F , and a reduced dynamic prompt with length L_{new} (where $L_{\text{new}} \leq L_F$), one can achieve a significant reduction in the per-denoising iteration cost, which can be roughly estimated as:

$$O((L_F + L_{\text{gen}})^2 - (L_{\text{new}} + L_{\text{gen}})^2), \quad (6)$$

where L_{gen} is the length of the generated tokens. Because DLM decoding requires evaluating bidirectional attention across multiple iterative denoising calls per block, maintaining a minimal L_{new} when the model is confident yields substantial, compounding reductions in overall computation.

5 Problem Statement

Following the above key finding, we define a new dynamic in-context planning task in DLMs: instead of using all examples as context throughout the DLM's generation process, we aim to determine when and which examples to insert during generation to accelerate inference while maintaining inference quality in an efficient inference setting.

Definition 1 (ICL Setting) *Given a prompt p with a pool of in-context examples $E_{\text{pool}} = \{(q_i, a_i)\}_{i=1}^K$, where q_i and a_i denote the query and answer of the i -th example, respectively.*

Definition 2 (Efficient Inference Setting) *Given a KV cache and block diffusion-based inference process divided into N non-KV cache steps (or blocks), indexed by $b \in \{1, \dots, N\}$.*

Goal: The objective is to find a training-free policy π that selects a subset of examples $E_b \subseteq \{E_{\text{pool}}, \emptyset\}$ for each block b , which minimizes the cumulative computational cost while preserving the baseline generation quality.

¹While we focus on block-wise DLM inference as it is the predominant inference framework, this discovery also holds for non-block-based DLM inference.

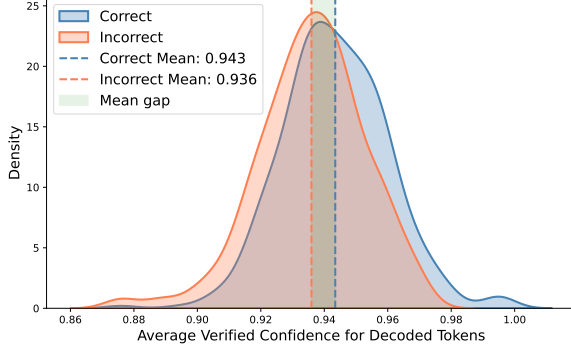


Figure 2: Average verified confidence distribution for correct vs. incorrect traces using LLaDA-8B-Instruct on GSM8K. There are 400 data points for each of the correct and incorrect categories.

6 Dynamic In-Context Planner

In this section, we formally introduce Dynamic In-Context Planner (DIP). As shown in Figure 1, DIP is a light weight framework that decouples selection complexity into: (1) an example ranking stage, which ranks candidate examples in E_{pool} based on their sentence embedding, and (2) an example insertion policy, which is a monotonic-increasing policy that progressively inserts these examples into the context at different generation blocks b based on generation confidence. Appendix B outlines the formal algorithm.

6.1 Example Ranking

The effectiveness of in-context learning relies heavily on the quality of the examples provided (Carbonell and Goldstein, 1998; Pickett et al., 2024; Nafee et al., 2025). As Fig. 3 highlights, injecting additional demonstrations can sometimes degrade rather than improve performance. To balance our dual goals of high-speed inference and sustained accuracy, DIP employs a training-free ranking stage to dynamically filter and select only the most beneficial examples during generation.

We tested two common sentence embedding-based (Reimers and Gurevych, 2019) approaches to rank examples: (1) cosine similarity (Zebaze et al., 2025; Reimers and Gurevych, 2019; Zhang et al., 2025b), and (2) maximal marginal relevance (MMR) (Carbonell and Goldstein, 1998; Pickett et al., 2024), which could adjust the relevance and diversity of examples based on the choice of λ .

$$\text{MMR}(q_i, \lambda) = \lambda \cdot \text{Sim}(q_p, q_i) - (1 - \lambda) \max_{j \in S} \text{Sim}(q_i, q_j), \quad (7)$$

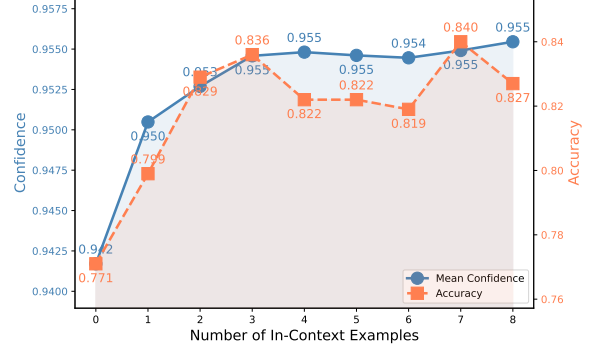


Figure 3: Impact of number of in-context examples vs. confidence and accuracy. Data are collected from the same 1000 sampled questions in the GSM8K training set with 0 to 8 in-context examples.

where $\text{Sim}(\cdot, \cdot)$ denotes the cosine similarity, q_p is the representation of the actual query the DLM aims to solve, S is the index set of already selected examples, and $\lambda \in [0, 1]$ controls the trade-off between relevance and redundancy. As $\lambda \rightarrow 1$, the ranking favors similarity; as $\lambda \rightarrow 0$, the ranking favors diversity. It is important to note that when $\lambda = 1$, MMR is effectively the same as cosine similarity.

6.2 Average Verified Confidence as a Proxy for Generation Quality

On what basis should we dynamically adjust the context? To fully exploit this architectural freedom, we need a reliable, real-time proxy to tell us when the model is struggling. Motivated by recent work on predictive metrics and test-time methods (Fu et al., 2026; Kang et al., 2025), we investigated 5 unique predictive metrics derived from DLMs' internal token distributions: (1) Average speculative entropy (ASPE), which is the average entropy at the unmasked indices over the block. (2) Average speculative confidence (ASPC), which is the confidence of speculative tokens at masked indices. (3) Average selected confidence (ASEC): the average confidence of the selected tokens, obtained when the token gets unmasked. (4) Average verified entropy (AVE): the average entropy of the last call over the block, obtained at the last call on the block level. (5) Average verified confidence: the average confidence of the last DLM's call over the selected tokens, obtained at the last call on the block level.

Through a rigorous empirical comparison of correlations between accuracy and the scaling of the number of examples, we identified **average verified confidence (AVC)** as the most suitable signal,

	Dataset	GSM8K (8 shot)		Math 500 (8 shot)	
Model	Method	Throughput	Acc	Throughput	Acc
LLaDA-8B Instruct	Fast-dLLM Fixed Prompt	66.58(1.0 $\times$)	77.3	63.69(1.0 $\times$)	38.2
	DIP($n_0=1$, RT=0.00)	87.80(1.32 $\times$)	76.2	76.34(1.20 $\times$)	32.4
	DIP($n_0=2$, RT=0.00)	84.08 (1.26 $\times$)	78.6	72.63 (1.14 $\times$)	36.2
	DIP($n_0=1$, RT=0.05)	90.24(1.36 $\times$)	76.4	78.13(1.23 $\times$)	33.2
	DIP($n_0=2$, RT=0.05)	86.49(1.30 $\times$)	77.4	74.19(1.16 $\times$)	35.6
LLaDA 1.5	Fast-dLLM Fixed Prompt	58.60(1.0 $\times$)	80.1	49.04(1.0 $\times$)	37.2
	DIP($n_0=1$, RT=0.00)	87.07(1.49 $\times$)	78.9	76.28 (1.56 $\times$)	37.2
	DIP($n_0=2$, RT=0.00)	82.92 (1.42 $\times$)	80.3	73.64(1.50 $\times$)	36.8
	DIP($n_0=1$, RT=0.05)	87.73(1.50 $\times$)	79.2	78.00(1.59 $\times$)	36.4
	DIP($n_0=2$, RT=0.05)	83.66(1.43 $\times$)	80.2	74.71(1.52 $\times$)	36.6

Table 1: DIP’s results on 8-shot GSM8K and 8-shot MATH500 compared against fixed prompt baseline using LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2026). **Blue**: relative speedup. We also **bold** the DIP configuration that achieves the highest speedup while largely preserving or yielding better generation quality.

exhibiting the strongest correlation ($r = 0.92$) with accuracy across varying numbers of in-context examples (Fig. 3). The AVC is defined as:

$$\bar{C} = \frac{1}{L_{gen}} \sum_{i=1}^{L_{gen}} C_i, \quad (8)$$

where L_{gen} denotes the length of the generated tokens, and C_i denotes the confidence of the DLM’s last call on the i -th generated token. Additional comparison results can be found in Appendix C.

6.3 Example Insertion Policy

Given a pool of ranked examples and our established quality signal (AVC), DIP must choose how many examples to use and update the prompt accordingly. An optimal insertion policy maximizes inference throughput by minimizing the context length, while strictly intervening to preserve generation quality when the model’s certainty drops. To address this, we propose an example-insertion policy governed by the dynamic fluctuations of the AVC throughout the generation process.

Confidence Change Ratio. We formally define the confidence-change ratio used by our proposed policy within the Fast-dLLM inference framework. Given an average newly generated token confidence for the current j -th block $\bar{C}_{cur}$, where the confidence is obtained by the last DLM call on that j -th block, we denote the average cached verified confidence for all previously generated tokens from block 1 to block $j - 1$ as $\bar{C}_{prev}$:

$$\bar{C}_{prev} = \frac{1}{j-1} \sum_{k=1}^{j-1} \bar{C}_k. \quad (9)$$

We further define the confidence change ratio against previously generated tokens R as:

$$R(\bar{C}_j) = 1 - \bar{C}_{cur} / \bar{C}_{prev}, \quad (10)$$

where $R < 0$ indicates that the newly generated block has higher verified confidence than the previous generation, and $R > 0$ indicates a degradation in verified confidence.

Final Policy. The final policy can be formally defined as follows:

$$\pi(b) = \begin{cases} n_0 & \text{if } b = 1 \\ n_0 & \text{if } b = 2 \ \& \ \bar{C}_1 \geq \text{CST} \\ n_0 + 1 & \text{if } b = 2 \ \& \ \bar{C}_1 < \text{CST} \\ n_{b-1} & \text{if } b \geq 3 \ \& \ R(\bar{C}_{b-1}) < \text{RT} \\ n_{b-1} + 1 & \text{if } b \geq 3 \ \& \ R(\bar{C}_{b-1}) \geq \text{RT} \end{cases} \quad (11)$$

where b denotes the current block index, n_0 denotes the initial number of examples for block 1, CST stands for cold start confidence threshold for block 2 decision, RT stands for the ratio threshold, and n_{b-1} stands for the number of examples used in the block $(b - 1)$.

7 Experiment

7.1 Experiment Setup

Dataset and Tasks. Following previous studies (Wu et al., 2026) and considering the availability of few-shot examples, we evaluate DIP under 5-shot and 8-shot settings on three generative tasks: (1) GSM8K (Cobbe et al., 2021) consists of math word problems, (2) MATH500 (Hendrycks et al., 2021; Lewkowycz et al., 2022), a high-quality subset of

	Dataset	GSM8K (5 shot)		Math 500 (5 shot)	
Model	Method	Throughput	Acc	Throughput	Acc
LLaDA-8B Instruct	Fast-dLLM Fixed Prompt	75.87(1.0 $\times$)	77.7	71.69(1.0 $\times$)	37.8
	DIP($n_0=1$, RT=0.00)	87.08 (1.15 $\times$)	77.5	75.50(1.05 $\times$)	35.4
	DIP($n_0=2$, RT=0.00)	83.85(1.11 $\times$)	77.1	74.29(1.04 $\times$)	35.4
	DIP($n_0=1$, RT=0.05)	89.42(1.18 $\times$)	76.8	f5(1.08 $\times$)	35.2
	DIP($n_0=2$, RT=0.05)	86.38(1.14 $\times$)	76.3	75.76 (1.06 $\times$)	37.2
LLaDA 1.5	Fast-dLLM Fixed Prompt	69.94(1.0 $\times$)	79.8	55.23(1.0 $\times$)	35.8
	DIP($n_0=1$, RT=0.00)	86.55(1.24 $\times$)	78.9	76.26(1.38 $\times$)	34.4
	DIP($n_0=2$, RT=0.00)	82.57(1.18 $\times$)	78.7	74.02(1.34 $\times$)	37.2
	DIP($n_0=1$, RT=0.05)	89.18(1.28 $\times$)	78.8	77.91(1.41 $\times$)	34.8
	DIP($n_0=2$, RT=0.05)	85.63 (1.22 $\times$)	79.5	75.69 (1.37 $\times$)	38.4

Table 2: DIP’s results on 5-shot GSM8K and 5-shot MATH500 compared against fixed prompt baseline using LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2026).

Ranking Method	Throughput	Acc
Random Order	84.06	77.9
MMR($\lambda = 0.2$)	83.88	78.4
MMR($\lambda = 0.4$)	84.12	78.2
MMR($\lambda = 0.6$)	83.93	77.2
MMR($\lambda = 0.8$)	83.90	77.9
Cosine Similarity	84.26	78.6

Table 3: Ablation study of DIP’s different ranking methods on 8-shot GSM8K with LLaDA-8B-Instruct (Nie et al., 2025).

math problems spanning 7 topics from high school math competitions, and (3) MBPP (Austin et al., 2021b), a coding benchmark consists of Python programming problems. To ensure reproducibility and the use of identical few-shot (in-context) examples under the n -shot setting, our experiment uses the *lm-eval* library (Gao et al., 2024).

Evaluation Metrics. We use two metrics: (1) inference throughput measured by the average number of output tokens generated over total runtime (including embedding, ranking, and prompt update latency), and (2) accuracy.

Baseline. We compare DIP against *Fast-dLLM Fixed Prompt*, a strong, state-of-the-art baseline representing standard KV cache-enhanced inference with a static context. We evaluate both approaches under 5- and 8-shot settings (Wu et al., 2026). We sample the in-context examples with the *lm-eval* package and present them in random order. All results were obtained using the official Fast-dLLM codebase with the default configurations: a generation length of 256 tokens, a block size of 32, and 8 blocks (unless otherwise noted).

Models and Implementation Details. Following prior work (Li et al., 2026; Jin et al., 2025), we experiment with the baseline and DIP on two representative masked DLMs: LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2026). To balance computational efficiency and sentence-embedding quality, we use Qwen3-Embedding-0.6B (Zhang et al., 2025b; Reimers and Gurevych, 2019) for obtaining the embeddings of questions. We run all experiments on NVIDIA RTX 5090 GPUs to ensure a fair comparison (unless otherwise noted).

7.2 Main Results

In Tables 1 and 2, we compare DIP’s performance against the fixed-prompt baseline, using cosine similarity and $CST=0.93$ (unless otherwise noted) in the 5-shot and 8-shot settings on two math reasoning tasks (MATH500 and GSM8K). In summary, the results show that DIP consistently improves inference throughput while largely preserving, and in some cases even improving, the generation quality of the fixed-prompt baseline.

As shown in Table 1 in the 8-shot setting, DIP notably achieves up to 1.50 $\times$ inference speedup on the GSM8K task and 1.59 $\times$ on MATH500 with LLaDA-1.5 while largely preserving the generation quality. Results also show improved inference throughput on LLaDA-8B-Instruct, with up to 1.36 $\times$ on GSM8K and 1.23 $\times$ on the MATH500 dataset. In the 5-shot setting (Table 2), we observed that DIP’s inference still improves inference speed, with throughput similar to that of 8 shots. This suggests that DIP can benefit from additional examples without incurring too much computational cost as the number of examples increases.

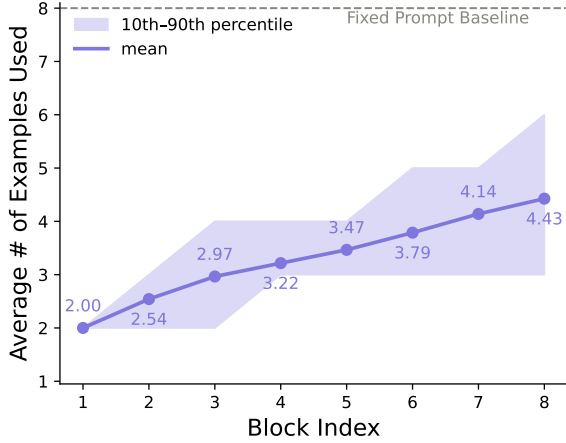


Figure 4: Example usage distribution along the block index on 8-shot GSM8K with LLaDA-8B-Instruct.

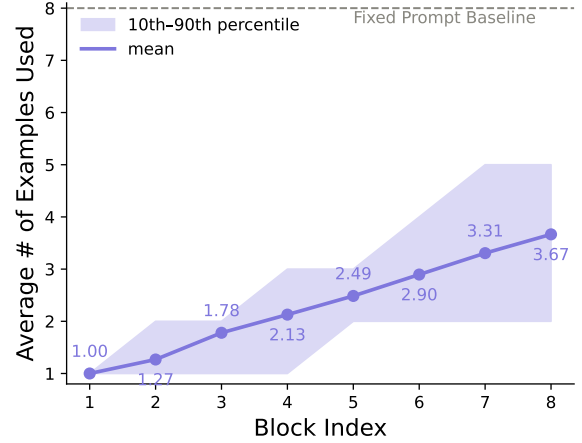


Figure 5: Example usage distribution along the block index on 8-shot MATH500 with LLaDA-1.5.

For the coding task, namely MBPP, experiments show a similar pattern in DIP’s effectiveness, yielding up to a $1.45\times$ speedup with the LLaDA-8B-Instruct Model. Due to limited space, MBPP results are included in the Appendix D. We also include additional stability analysis in the Appendix E.

7.3 Ablation Study and Additional Analysis

Ranking Methods. To further understand the example order’s influence on DIP, we conducted an ablation over the ranking methods: (1) cosine similarity and (2) MMR with different λ . As shown in Table 3, the order of example insertion does not influence the throughput, while improving accuracy with the right ordering. More results with the MMR example ranking are in Appendix F.

Example Usage By Block and Speedup Analysis. Fig. 4 presents the example usage distribution by block index on 8-shot GSM8K with the LLaDA-8B-Instruct model. We obtained this data using DIP with $n_0 = 2$, $RT = 0$, $CST = 0.93$, and cosine-similarity ranking, yielding a $1.26\times$ speedup over the fixed-prompt baseline. In Fig. 5, results show similar trends in 8-shot MATH500 with the LLaDA-1.5 model. This data was obtained using DIP with $n_0 = 1$, $RT = 0$, $CST = 0.93$, and cosine-similarity ranking, yielding a $1.56\times$ speedup over the fixed-prompt baseline. As shown, DIP’s example-insertion policy yields greater speedup in the early stages than in the later stages, contributing to higher throughput.

Hyperparameter Sensitivity. To further investigate DIP’s hyperparameter sensitivity, we conduct an ablation study to evaluate the impact of hyperparameter choices, using LLaDA-1.5 on the 8-shot

GSM8K. The results in Appendix G provide clear guidance for selecting hyperparameters: larger n_0 , larger RT , and larger CST are preferred for accuracy. Smaller n_0 , larger RT , and smaller CST are preferred to maximize inference throughput.

Performance Across 256 and 512 Generation Lengths. As noted in Eq. 6, the overall computation expense is based on two factors: (1) the length of the prompt and (2) generation lengths. Increasing generation length could reduce speedup. To further investigate performance differences across generation lengths, we compared LLaDA-8B-Instruct and LLaDA-1.5 in the 8-shot GSM8K setting with generation lengths of 256 and 512. As shown in Table 4, consistent with Eq. 6, the speedup decreases as the generation length increases from 256 to 512. Overall, DIP can still yield up to a $1.21\times$ speedup at 512-token generation length and $1.42\times$ speedup at 256-token generation length. Due to memory constraint, this experiment is performed using NVIDIA RTX PRO 6000 GPUs.

Prompt Change In KV-Cache Enhanced DLMs Inference Framework. Because DIP aims to optimize inference throughput in DLMs, it must be compatible with the KV cache-enhanced inference framework. DIP is based on Fast-dLLM (Wu et al., 2026), a KV cache-enhanced block-wise inference framework, which requires a necessary non-KV cache call at the start of discrete block decoding to update the KV cache (Appendix B).

In principle, DIP is a plug-in method that works with any masked DLM KV-cache-enhanced inference framework, and a prompt update can occur alongside any non-KV-cache call.

Model	Method	Gen Len	Throughput	Acc
LLaDA-8B-Instruct	Fast-dLLM Fixed Prompt	256	66.58(1.0 $\times$)	77.3
	DIP($n_0 = 2, RT = 0$)	256	84.08(1.26 $\times$)	78.6
	Fast-dLLM Fixed Prompt	512	43.18(1.0 $\times$)	74.7
	DIP($n_0 = 2, RT = 0$)	512	52.46(1.21 $\times$)	75.6
LLaDA 1.5	Fast-dLLM Fixed Prompt	256	58.60(1.0 $\times$)	80.1
	DIP($n_0 = 2, RT = 0$)	256	82.92(1.42 $\times$)	80.3
	Fast-dLLM Fixed Prompt	512	44.70(1.0 $\times$)	78.2
	DIP($n_0 = 2, RT = 0$)	512	53.10(1.19 $\times$)	79.2

Table 4: Performance comparison between 256 and 512 under the 8-shot GSM8K Setting.

8 Conclusion

In this paper, we make a key discovery: DLMs enable *low-cost dynamic context adjustment*. We also empirically evaluate correlations between generation accuracy and token confidence for DLMs and identify average verified confidence as the key signal for guiding context adjustment. Building on these insights, we propose DIP, a practical context optimization framework that dynamically scales in-context examples during generation based on verified confidence. Experimental results on math and coding benchmarks with LLaDA-1.5 and LLaDA-8B-Instruct show that DIP achieves up to 1.59 $\times$ and 1.36 $\times$ speedups, respectively, while largely preserving the generation quality of the fixed-prompt baseline. Our key findings open a new avenue for efficient DLM inference through dynamic context optimization.

Acknowledgments

This research was supported by the National Science Foundation under IIS-2348405, 2442253, 2607580, NIH 1R21AG091260-01, Commonwealth Cyber Initiative, and generous gifts from Nvidia, Cisco, and the Amazon-Virginia Tech Initiative. The authors acknowledge the National Artificial Intelligence Research Resource (NAIRR) Pilot (through allocation NAIRR250123 and NAIRR240202), William & Mary Research Computing, UIUC Delta, and Purdue Anvil supercomputers for contributing to this research result.

Limitations

Our proposed approach has a few limitations. First, the training-free nature of DIP introduces a trade-off: the example-usage policy depends on hyperparameters that must be empirically tuned to accommodate varying task difficulties. Second, we

constrain our exploration of DLMs’ architectural freedom strictly to example usage in in-context learning. While dynamic context in DLMs holds significant potential for applications beyond ICL, we leave these broader structural investigations to future work.

References

- Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Zhenyu Bi, Meng Lu, Yang Li, Swastik Roy, Weijie Guan, Morteza Ziyadi, and Xuan Wang. 2025. Optagent: Optimizing multi-agent llm interactions through verbal reinforcement learning for enhanced reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1713–1728.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.
- Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings*

- of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pages 335–336.
- Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. 2026. dparallel: Learnable parallel decoding for dllms. In *International Conference on Learning Representations*, volume 2026, pages 53193–53211.
- 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*.
- DeepMind. 2025. Gemini Diffusion — deepmind.google. [Accessed 01-01-2026].
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128.
- Yichao Fu, Xuwei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. 2026. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.
- Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepl, Preslav Nakov, and Iryna Gurevych. 2024. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Bokai Hu, Sai Ashish Somayajula, Xin Pan, and Pengtao Xie. 2024. Improving the language understanding capabilities of large language models using reinforcement learning. *arXiv preprint arXiv:2410.11020*.
- Daniel Mingyi Israel, Guy Van den Broeck, and Aditya Grover. 2025. Accelerating diffusion LLMs via adaptive parallel decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. 2026. d² cache: Accelerating diffusion-based llms via dual adaptive caching. In *International Conference on Learning Representations*, volume 2026, pages 137511–137531.
- Xiangqi Jin, Yuxuan Wang, Yifeng Gao, Zichen Wen, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2025. Thinking inside the mask: In-place prompting in diffusion llms. *arXiv preprint arXiv:2508.10736*.
- Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. Scalable best-of-n selection for large language models via self-certainty. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.
- Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. 2026. Beyond fixed: Training-free variable-length denoising for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*.
- Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. 2025. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In

- Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. 2025. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. dKV-cache: The cache for diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 11048–11064.
- Mahmud Wasif Nafee, Maiqi Jiang, Haipeng Chen, and Yanfu Zhang. 2025. Dynamic retriever for in-context knowledge editing via policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16744–16757.
- 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Marc Pickett, Jeremy Hartman, Ayan Kumar Bhowmick, Raquib-ul Alam, and Aditya Vempaty. 2024. Better rag using relevant information gain. *arXiv preprint arXiv:2407.12101*.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2655–2671.
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2026. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33038–33046.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2026. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pages 57027–57051.
- 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*.
- Armel Randy Zebaze, Benoît Sagot, and Rachel Bawden. 2025. In-context example selection via similarity search improves low-resource machine translation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1222–1252. Association for Computational Linguistics.
- Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S Yu, and Aiwei Liu. 2025a. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.
- Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 11425–
11460.

A Appendix: An Example Context Change Through Different Blocks Using MATH500

Context for block 1 with $n_0 = 1$, i.e., one example:

Problem:
Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.

Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$.

Final Answer: The final answer is $[2,5)$. I hope it is correct.

Problem:
Define

$$p = \sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$q = \sum_{k=1}^{\infty} \frac{1}{k^3}$$
Find a way to write

$$\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$$
in terms of p and q .

Solution:

Context for block 2 and block 3 with two examples:

Problem:
Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.

Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$.

Final Answer: The final answer is $[2,5)$. I hope it is correct.

Problem:
If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det (\mathbf{A} \mathbf{B})$.

Solution: We have that $\det (\mathbf{A} \mathbf{B}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$.

Final Answer: The final answer is 24 . I hope it is correct.

Problem:
Define

$$p = \sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$q = \sum_{k=1}^{\infty} \frac{1}{k^3}$$
Find a way to write

$$\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$$
in terms of p and q .

Solution:
+ GENERATED CONTENT.

Context for block 4 with three examples:

Problem:
Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.

Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$.

Final Answer: The final answer is $[2,5)$. I hope it is correct.

Problem:
If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det (\mathbf{A} \mathbf{B})$.

Solution: We have that $\det (\mathbf{A} \mathbf{B}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$.

Final Answer: The final answer is 24 . I hope it is correct.

Problem:
Terrell usually lifts two 20-pound weights 12 times. If he uses two 15-pound weights instead, how many times must Terrell lift them in order to lift the same total weight?

Solution: If Terrell lifts two 20-pound weights 12 times, he lifts a total of $2 \cdot 12 \cdot 20 = 480$ pounds of weight. If he lifts two 15-pound weights instead for n times, he will lift a total of $2 \cdot 15 \cdot n = 30n$ pounds of weight. Equating this to 480 pounds, we can solve for n :

$$\begin{aligned} 30n &= 480 \\ \Rightarrow n &= 480/30 = \boxed{16} \end{aligned}$$

Final Answer: The final answer is 16 . I hope it is correct.

Problem:
Define

$$p = \sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$q = \sum_{k=1}^{\infty} \frac{1}{k^3}$$
Find a way to write

$$\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$$
in terms of p and q .

Solution:
+ GENERATED CONTENT.

B Appendix: Algorithm for Fast-dLLM and DIP

Algorithm 1 presents the decoding process of Fast-dLLM (Wu et al., 2026). Building on Fast-dLLM, we present the algorithm of DIP in Algorithm 2.

C Appendix: Additional Metrics Comparison

In this section, we define metrics specific to DLMs’ generation process and present the impact of the number of in-context examples on metrics and accuracy. We collect all data from the same 1000 sampled questions in the GSM8K training set, with 0 to 8 in-context examples.

(1) **Speculative Entropy**: the entropy at the unmasked index, which is obtained at the Non-KV cache call. In Fig. 6, this metric shows a positive correlation ($r = +0.76$) with accuracy.

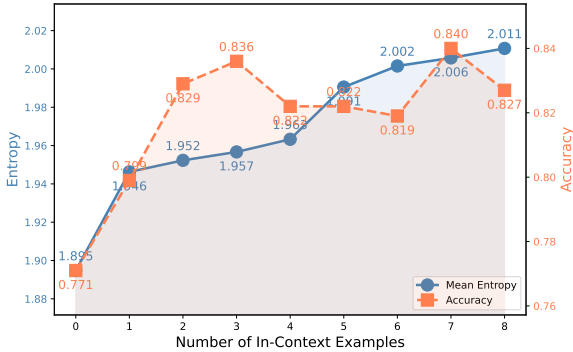


Figure 6: Impact of number of in-context examples vs. average speculative entropy and accuracy.

(2) **Speculative Confidence**: the highest confidence at a masked index, which is obtained at the Non-KV cache call. In Fig. 7, this metric shows a negative correlation ($r = -0.79$) with accuracy.

(3) **Selected Confidence**: the confidence of the selected token, which is obtained when the token gets unmasked. This metric is similar to token confidence in ARLMs (Fu et al., 2026). In Fig. 8, this metric shows a negative correlation ($r = -0.71$) with accuracy.

(4) **Verified Entropy**: the entropy of the last call over the index, which is usually the last call on the block level in Fast-dLLM. In Fig. 9, this

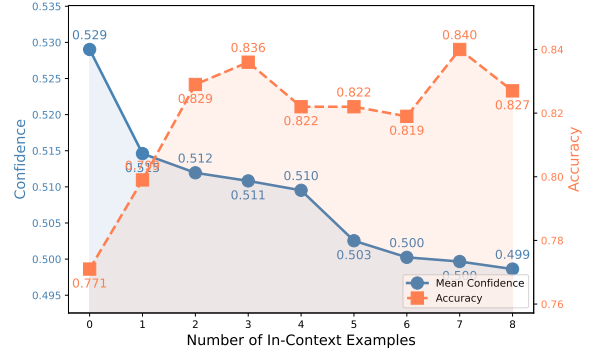


Figure 7: Impact of number of in-context examples vs. average speculative confidence and accuracy.

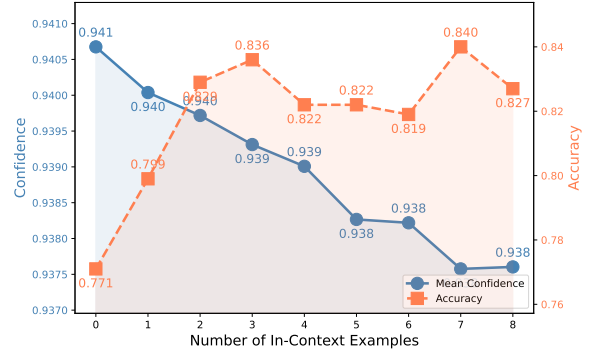


Figure 8: Impact of number of in-context examples vs. average selected confidence and accuracy.

metric shows a negative correlation ($r = -0.80$) with accuracy.

(5) **Verified Confidence**: the verified confidence of the selected token at the last call over the token, which is usually the last call on the block level in Fast-dLLM. In Fig. 3, this metric shows a positive correlation ($r = 0.92$) with accuracy.

Summary: By comparing those relationships from different metrics, average verified confidence (Fig. 2) shows the strongest correlation with accuracy among all those predictive metrics derived from DLMs’ internal token distributions. Hence, we used the average verified confidence metric as the quality signal throughout our experiments.

D Appendix: MBPP Results

We evaluate DIP on the MBPP benchmark using both the LLaDA-8B-Instruct and LLaDA-1.5 models. As detailed in Tables 7 and 8, DIP consistently accelerates inference, achieving speedups of up to $1.45\times$ while maintaining comparable generation quality. Notably, because code generation tasks inherently demand rigorous logical and syntactic reasoning, empirical results indicate that DIP requires

Algorithm 1: Fast-dLLM Decoding with Threshold

Input: model p_θ , prompt p , answer length L , blocks N , block size B , step per block T , threshold τ

```
1  $x \leftarrow [p; [\text{MASK}], \dots, [\text{MASK}]]$ 
2 Initialize KV Cache for  $x$ . ▷▷▷ KV Cache Init
3 for  $n = 1$  to  $N$  do
4    $s \leftarrow \lfloor |p| + (n - 1)B \rfloor$ ,  $e = |p| + nB$ 
5   Update KV Cache with non-KV cache call ▷▷▷ KV Cache Update
6   for  $t = 1$  to  $T$  do
7     Use cache, call  $p_\theta$  on  $x^{[s,e]}$  ▷▷▷ KV Cache Reuse
8     For each masked  $x^i$ , confidence  $c^i = \max_x p_\theta(x^i)$ 
9     Unmask all  $i$  in  $[s, e)$  with  $c^i \geq \tau$ , with index  $\max c^i$ 
10    if all  $x^{[s,e]} \neq [\text{MASK}]$  then
11      break, move to next block
12 return  $x$ ;
```

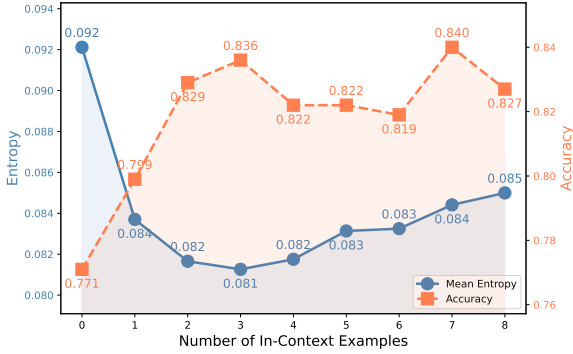


Figure 9: Impact of number of in-context examples vs. average verified entropy and accuracy.

a larger initial number of in-context examples to effectively bootstrap the generation process.

E Appendix: Stability Results

To further investigate DIP stability and account for the inherent noise in DLM generation, we conducted multi-run evaluations in two representative settings.

First, we evaluated the configuration that showed the most significant drop in accuracy in our main results: LLaDA-8B-Instruct in the MATH500 8-shot setting. Across 5 independent runs, the Fast-dLLM fixed-prompt baseline achieved an accuracy of 37.5 ± 1.34 , while DIP ($n_0 = 1$, $RT = 0.00$) achieved 36.0 ± 2.21 . Although DIP shows higher variance in this boundary-case configuration, its overall performance remains competitive.

Second, we evaluated our optimal configuration on GSM8K with LLaDA-1.5 using DIP ($n_0 = 2$, $RT = 0.00$) across 3 independent runs for both

the 5-shot and 8-shot settings. As detailed in Table 5, the average accuracy and throughput are highly stable and consistent with the single-run values reported in our main results, demonstrating that DIP reliably maintains generation quality while preserving its speedup advantages.

F Appendix: Additional MMR Results

In this section, we present additional results with the MMR example ranking method. Following the results in Table 3, we experiment with MMR example ranking with $\lambda = 0.2$. In Table 6, we present results on 8-shot GSM8K using the LLaDA-8B-Instruct model alone, with the fixed prompt baseline. In Table 10, we present results on the 8-shot GSM8K task using the LLaDA-1.5 model alone, with the fixed prompt baseline.

G Appendix: Hyperparameter Sensitivity Results

We present the sensitivity results over DIP’s hyperparameter choices on 8-shot GSM8K using LLaDA-1.5 (Zhu et al., 2026) in Table 9, Table 11, and Table 12. Results show that a wide range of parameters can yield reasonable performance compared to the Fast-dLLM Fixed Prompt with speedups on the GSM8K task.

²Random ordered examples

Algorithm 2: DIP with Fast-dLLM

Input: model p_θ , answer length L , blocks N , block size B , step per block T , threshold τ , example insertion policy π , examples set E , number of examples K , query q

```
1 Rank  $E$  using defined ranking method,  $E_{rank} = [E_1, E_2, \dots, E_K]$   $\triangleright\triangleright\triangleright$  Example Ranking Stage
2  $p \leftarrow [E_1; q]$ ,  $k = 1$   $\triangleright\triangleright\triangleright$  Initialize Prompt With  $\pi$ 
3  $x \leftarrow [p; [\text{MASK}], \dots, [\text{MASK}]]$ 
4 Initialize KV Cache for  $x$ .  $\triangleright\triangleright\triangleright$  KV Cache Init
5  $\bar{\mu} = 0$   $\triangleright\triangleright\triangleright$  Average Verified Confidence Over All Generated Tokens
6  $\mu = 0$   $\triangleright\triangleright\triangleright$  Verified Confidence Mean Of Current Block
7 for  $n = 1$  to  $N$  do
8    $s \leftarrow \lfloor |p| + (n - 1)B \rfloor$ ,  $e = |p| + nB$ 
9   if  $n! = 1$  then
10     Insertion =  $\pi(\mu, \bar{\mu})$   $\triangleright\triangleright\triangleright$  Get Action Following  $\pi$ 
11     if Insertion == True then
12        $p \leftarrow [E_1, E_2, \dots, E_k; q]$ ,  $k++ = 1$   $\triangleright\triangleright\triangleright$  Update Prompt
13        $lp = \text{length of } p$ 
14        $x \leftarrow [p; [\text{MASK}], \dots, [\text{MASK}]]$ 
15     Update KV Cache with non-KV cache call  $\triangleright\triangleright\triangleright$  KV Cache Update
16   for  $t = 1$  to  $T$  do
17     Use cache, call  $p_\theta$  on  $x^{[s,e]}$   $\triangleright\triangleright\triangleright$  KV Cache Reuse
18     For each masked  $x^i$ , confidence  $c^i = \max_x p_\theta(x^i)$ 
19     Unmask all  $i$  in  $[s, e)$  with  $c^i \geq \tau$ , with index  $\max c^i$ 
20     if all  $x^{[s,e]} \neq [\text{MASK}]$  then
21        $\mu = \text{mean}(c^i \text{ of } x^{[s,e]})$   $\triangleright\triangleright\triangleright$  Save Confidence Mean Of Current Block
22        $\bar{\mu} = \text{mean}(c^i \text{ of } x^{[lp,e]})$   $\triangleright\triangleright\triangleright$  Save Confidence Mean Of Generated Tokens
23       break, move to next block
24      $\mu = \text{mean}(c^i \text{ of } x^{[s,e]})$   $\triangleright\triangleright\triangleright$  Save Confidence Mean Of Current Block
25      $\bar{\mu} = \text{mean}(c^i \text{ of } x^{[lp,e]})$   $\triangleright\triangleright\triangleright$  Save Confidence Mean Of Generated Tokens
26 return  $x$ ;
```

Shot	Method	Throughput	Accuracy
5	Baseline	72.52 ± 0.9	80.1 ± 0.9
5	DIP	88.78 ± 0.7	79.9 ± 1.5
8	Baseline	61.96 ± 0.3	80.6 ± 0.5
8	DIP	85.87 ± 0.9	80.2 ± 0.4

Table 5: Results with LLaDA-1.5 on 5 and 8-shot GSM8K across 3 independent runs. Values denote Mean $\pm$ Standard Deviation.

DIP Setting	Throughput	Acc
Baseline	66.58	77.3
$n_0=1$, RT=0.00	87.13	75.4
$n_0=2$, RT=0.00	83.98	78.4
$n_0=1$, RT=0.05	89.89	76.0
$n_0=2$, RT=0.05	86.91	77.3

Table 6: Additional results on 8-shot GSM8K with MMR example ranking where $\lambda = 0.2$ on LLaDA-8B-Instruct.

H Appendix: Comparing Against Naive Insertion Policy and Fixed Few-Shot Policy

We also evaluate DIP against a naive insertion policy that starts with minimal context and simply adds one example at each block boundary. As shown in Table 13, this naive policy yields suboptimal inference throughput. By indiscriminately increasing

the prompt length at every step, the naive policy incurs unnecessary computational costs. DIP, by contrast, dynamically injects examples only when needed, demonstrating that strategically timed insertions are essential for maximizing throughput without sacrificing generation quality.

We also compared with a fixed few-shot pol-

	Dataset		MBPP	
Model	Method	# of Shot	Throughput	Acc
LLaDA-8B Instruct	Baseline	5	41.70(1.0 $\times$)	26.6
	DIP($n_0=3$, RT=0.00)	5	49.98(1.20 $\times$)	27.6
	DIP($n_0=3$, RT=0.05)	5	52.37 (1.25 $\times$)	27.4
LLaDA-8B Instruct	Baseline	8	34.27(1.0 $\times$)	28.0
	DIP($n_0=3$, RT=0.00)	8	47.80(1.39 $\times$)	26.4
	DIP($n_0=3$, RT=0.05)	8	49.61 (1.45 $\times$)	27.8

Table 7: DIP’s results on MBPP comparing against fixed prompt baseline using LLaDA-8B-Instruct (Nie et al., 2025). **Blue: relative speedup**. We also **bold** the DIP configuration that achieves the highest speedup while maintaining on-par or higher accuracy.

	Dataset		MBPP	
Model	Method	# of Shot	Throughput	Acc
LLaDA 1.5	Baseline	5	40.70(1.0 $\times$)	26.0
	DIP($n_0=3$, RT=0.00)	5	48.78 (1.20 $\times$)	25.0
	DIP($n_0=3$, RT=0.05)	5	50.02(1.23 $\times$)	23.8
LLaDA 1.5 ²	Baseline	8	35.29(1.0 $\times$)	29.4
	DIP($n_0=4$, RT=0.00)	8	42.98 (1.22 $\times$)	27.8
	DIP($n_0=4$, RT=0.05)	8	44.10(1.25 $\times$)	26.2

Table 8: DIP’s results on MBPP comparing against fixed prompt baseline using LLaDA-1.5 (Zhu et al., 2026). **Blue: relative speedup**. We also **bold** the DIP configuration that achieves the highest speedup while maintaining on-par or higher accuracy.

n_0	Throughput	Acc
Baseline	58.60	80.1
1	90.65	78.9
2	85.86	80.3
3	80.82	81.5
4	77.46	80.2
5	74.00	79.4

Table 9: Ablation study of initial examples n_0 on 8-shot GSM8K with LLaDA-1.5 (Zhu et al., 2026).

DIP Setting	Throughput	Acc
Baseline	58.60	80.1
$n_0=1$, RT=0.00	85.56	79.2
$n_0=2$, RT=0.00	81.20	79.2
$n_0=1$, RT=0.05	89.68	79.6
$n_0=2$, RT=0.05	85.80	79.6

Table 10: Additional results on 8-shot GSM8K with MMR example ranking where $\lambda = 0.2$ on LLaDA-1.5.

RT	Throughput	Acc
Baseline	58.60	80.1
-0.1	79.11	79.7
-0.05	81.35	79.4
0	86.56	80.3
0.05	89.28	80.2
0.01	88.90	80.5

Table 11: Ablation study of ratio threshold RT on 8-shot GSM8K with LLaDA-1.5 (Zhu et al., 2026).

icy on GSM8K in both the 5-shot and 8-shot settings. This apples-to-apples comparison compares fixed few-shot k vs. DIP with the same initial context ($n_0 = k$), isolating exactly what dynamic insertion adds. As shown in Table 14, for every matched starting context, DIP either matches the fixed prompt or improves accuracy by using additional examples exactly when needed, at the expense of some inference throughput. However, DIP can balance accuracy and example usage (throughput) in DLM inference. This is exactly the flexibility we aim to demonstrate. Moreover, the best k is only knowable post hoc and does not transfer.

I Appendix: Use of AI Assistants

In this paper, we use AI to help with grammar and fluency. We also use AI for coding syntax assistance.

CST	Throughput	Acc
Baseline	58.60	80.1
0.75	87.36	78.9
0.8	88.62	78.9
0.85	88.38	78.9
0.9	86.68	79.5
0.95	84.10	80.7

Table 12: Ablation study of cold start confidence threshold CST on 8-shot GSM8K with LLaDA-1.5(Zhu et al., 2026).

Shot	Method	Throughput	Acc
5	Baseline	75.87(1.00 $\times$)	77.7
	Naive ($n_0=1$)	81.32(1.07 $\times$)	77.5
	DIP ($n_0=1$, RT=0)	87.08(1.15 $\times$)	77.5
8	Baseline	66.58 (1.00 $\times$)	77.3
	Naive ($n_0=1$)	73.40 (1.10 $\times$)	77.6
	DIP ($n_0=1$, RT=0)	87.80 (1.32 $\times$)	76.2
	DIP ($n_0=2$, RT=0)	84.08 (1.26 $\times$)	78.6

Table 13: Comparison of DIP against a naive monotonic-increasing policy on GSM8K with LLaDA-8B-Instruct.

	Dataset		GSM8K	
Model	Method	Max # of Example	Throughput	Acc
LLaDA 8B-Instruct	Baseline (fixed $k = 1$)	5	95.67	73.0
	DIP($n_0=1$)	5	87.08	77.5
	Baseline (fixed $k = 2$)	5	90.05	77.1
	DIP($n_0=2$)	5	83.85	77.1
	Baseline (fixed $k = 1$)	8	95.76	74.4
	DIP($n_0=1$)	8	90.24	76.4
LLaDA 1.5	Baseline (fixed $k = 2$)	8	89.35	74.5
	DIP($n_0=2$)	8	84.08	78.6
	Baseline (fixed $k = 1$)	5	100.40	76.6
	DIP($n_0=1$)	5	86.55	78.9
	Baseline (fixed $k = 2$)	5	94.18	78.6
	DIP($n_0=2$)	5	85.63	79.5
	Baseline (fixed $k = 1$)	8	98.01	77.1
	DIP($n_0=1$)	8	87.73	79.2
	Baseline (fixed $k = 2$)	8	93.12	79.8
	DIP($n_0=2$)	8	82.92	80.3

Table 14: DIP’s results comparing against the fixed-shot policy using LLaDA-1.5 and LLaDA-8B-Instruct.