DAWN: Dependency-Aware Fast Inference for Diffusion LLMs

Lizhuo Luo[1] **Zhuoran Shi**[2] **Jiajun Luo**[3] **Zhi Wang**[3]
Shen Ren[4] **Wenya Wang**[1] **Tianwei Zhang**[1*]
[1] Nanyang Technological University
[2] Southern University of Science and Technology
[3] Shenzhen International Graduate School, Tsinghua University
[4] AUMOVIO Singapore Pte. Ltd.

Abstract

Diffusion large language models (dLLMs) have shown advantages in text generation, particularly due to their inherent ability for parallel decoding. However, constrained by the quality–speed trade-off, existing inference solutions adopt conservative parallel strategies, leaving substantial efficiency potential underexplored. A core challenge is that parallel decoding assumes each position can be filled independently, but tokens are often semantically coupled. Thus, the correct choice at one position constrains valid choices at others. Without modeling these positional dependencies, parallel strategies produce deteriorated outputs. Motivated by this insight, we propose `DAWN`, a training-free, dependency-aware decoding method for fast dLLM inference. `DAWN` extracts dependency relations and leverages two key observations: (1) masked positions dependent on unmasked certain positions become more reliable; (2) coupled masked positions strongly influence each other's predictions. Given those findings, `DAWN` leverages a dependency graph to select more reliable unmasking positions at each iteration, achieving high parallelism with negligible loss in generation quality. Extensive experiments across multiple models and datasets demonstrate that `DAWN` speedups the inference by **1.80 - 8.06**$\times$ over baselines while preserving the generation quality. Code is released at https://github.com/lizhuo-luo/DAWN.

1 Introduction

Diffusion models have achieved remarkable success in image [29, 44] and video [6, 19] generation, and have recently been extended to text generation. Unlike autoregressive (AR) models [28, 31] that generate sequentially, diffusion large language models (dLLMs) [27, 47] adopt full attention over all positions and refine an entire sequence through multiple denoising iterations, achieving surprisingly strong performance on text generation tasks. dLLMs offer potential solutions to longstanding limitations of AR models, including the inability to decode in parallel and the reversal curse [4]. These characteristics have drawn growing research interest in further advancing dLLMs [13, 38, 20, 5].

Despite these efforts, dLLMs still exhibit performance gaps in practical deployments compared to state-of-the-art AR models [41, 22]. These gaps are largely attributed to two main factors: KV-Cache management [41, 26, 24] and nonindependent position predictions [36, 41]. First, dLLMs employ bidirectional attention, which fundamentally contradicts the causal assumption underlying standard KV-Cache mechanisms. Second, the marginal distributions at each position produced by dLLMs often violate the independence assumption underlying parallel decoding.

*Corresponding author: `tianwei.zhang@ntu.edu.sg`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This work aims to improve the efficiency of parallel decoding in dLLMs, with a particular focus on *nonindependent position predictions*. Most existing parallel decoding strategies [41, 3] select masked positions using heuristics and relatively coarse-grained criteria, such as confidence and entropy, to ensure that the selected positions behave safely and approximately independently. However, overly conservative selection criteria can substantially limit the achievable parallelism, leaving much of the potential efficiency of parallel decoding underexploited. To address this problem, a natural alternative is to account for positional relationships more directly. Since the main difficulty stems from position coupling, improving parallel decoding requires approximating positional dependencies during inference rather than evaluating each position in isolation. Attention maps [10, 32, 16, 50, 49] provide an approximate yet cheap signal of token interactions that is readily available from each forward pass, making them a practical proxy for dependency estimation. In fact, dependencies between positions of different states yield different effects. From this perspective, two observations are particularly relevant: (i) For unmasked-masked position pairs, masked positions that are strongly dependent on previously unmasked high-confidence tokens can remain highly consistent with the final output even when their confidence is relatively low; and (ii) For masked–masked position pairs, committing one position induces a substantial shift in the predictive distribution of its coupled counterpart. These observations further validate that attention maps can serve as a reasonable proxy for positional interactions and provide a new insight: *dependency-aware inference rules can expand safe parallelism beyond those conservative threshold methods*.

Motivated by this, we propose DAWN, a training-free Dependency-AWare fast inference method for diffusioN LLMs. It improves parallel decoding by explicitly accounting for positional dependencies. DAWN consists of three pipelined components: *Dependency Graph Construction*, *Anchor-Guided Decoding*, and *Conflict-Based Scheduling*. At each denoising iteration, a dependency graph is constructed from the processed attention maps via thresholding to capture salient coupling relations. Based on this graph, the next two components select disjoint sets of positions for parallel updates: $\mathcal{U}_{\text{anchor}}$ and $\mathcal{U}_{\text{conflict}}$. *Anchor-Guided Decoding* first selects high-confidence positions that will probably serve as anchors in subsequent steps, and then previously unmasked high-confidence positions as anchors and relaxes the confidence threshold for strongly anchor-coupled masked positions. Then *Conflict-Based Scheduling* identifies conflicts in the dependency graph and greedily constructs a large non-conflicting update set from the remaining candidates that satisfy a lower confidence threshold.

Compared with prior approaches that rely on strict positional criteria, DAWN relaxes selection constraints by incorporating dependency information, thereby enabling additional updates that would otherwise be filtered out while preserving generation quality. Meanwhile, it substantially mitigates failures caused by nonindependent position predictions. Extensive experiments validate that DAWN improves the quality–speed trade-off of dLLM inference across multiple models and settings.

The main contributions are summarized as follows:

- We characterize positional dependencies in dLLM inference through attention-derived estimates and identify two key findings: (1) For unmasked–masked position pairs, once a high-confidence token is committed, positions that depend on it become reliably predictable, even at lower confidence; (2) For masked–masked position pairs, committing one position in a coupled pair substantially shifts the predictive distribution of its partner.
- We propose DAWN, a training-free, dependency-aware method for fast inference of diffusion LLMs. It uses the estimated dependencies in two ways: relaxing confidence thresholds for positions anchored by committed high-confidence tokens, and preventing strongly coupled low-confidence positions from being unmasked together, thereby enabling more efficient inference.
- Extensive experiments across multiple models, datasets, and representative baselines demonstrate the effectiveness and robustness of DAWN, achieving **1.80 - 8.06**$\times$ speedup over the baseline. Ablation studies further validate the contributions of each component and the impact of key hyperparameters.

2 Preliminaries

2.1 Inference Process of dLLMs

Most recent diffusion LLMs adopt the discrete masked diffusion model paradigm [35, 34], where generation is cast as an iterative unmasking process. Unlike autoregressive models that commit

tokens sequentially, dLLMs start from a heavily masked sequence and progressively recover masked positions over several denoising steps until no [MASK] remains. At each step, the model predicts token distributions for all masked positions, conditioned on the current state.

Given a prompt X, we initialize the response as a fully masked sequence of predefined length L:

$$y^{(0)} = (\texttt{[MASK]}, \ldots, \texttt{[MASK]}) \in (\mathcal{V} \cup \{\texttt{[MASK]}\})^L.$$

Here, $\mathcal{V}$ denotes the vocabulary and [MASK] is the special mask token. In the naive setting, the sampler unmasks exactly one token with the highest confidence per step. At each denoising step $t = 0, 1, \ldots, L-1$, it concatenates the prompt X and the current response state $y^{(t)}$ as the model input, and commits the corresponding token at the [MASK] response position with the highest confidence:

$$\begin{aligned} c_i^{(t)} &\triangleq \max_{v \in \mathcal{V}} p_\theta(y_i = v \mid X, y^{(t)}), \quad i \in M^{(t)}, \\ i_t &= \arg\max_{i \in M^{(t)}} c_i^{(t)}, \\ y_i^{(t+1)} &= \begin{cases} \arg\max_{v \in \mathcal{V}} p_\theta(y_i = v \mid X, y^{(t)}), & \text{if } i = i_t, \\ y_i^{(t)}, & \text{otherwise.} \end{cases} \end{aligned}$$

where $M^{(t)} \triangleq \{i \mid y_i^{(t)} = \texttt{[MASK]}\}$ denotes the set of masked response positions at step t. Repeating this procedure for L steps yields a fully unmasked response $y^{(L)}$.

Despite the inherent parallelism within each step, practical dLLM inference exhibits a quality–speed trade-off [30]: directly unmasking multiple tokens in one step leads to substantial quality degradation.

2.2 Nonindependent Position Predictions

Unmasking a single token per step is inherently slow, and runs counter to the full position computation paradigm of dLLMs, where each denoising step produces predictions for all positions simultaneously. A common approach is to commit tokens at different positions independently according to the model's predictive distributions at each position:

$$p_\theta\left(\{y_i\}_{i \in U^{(t)}} \mid X, y^{(t)}\right) \approx \prod_{i \in U^{(t)}} p_\theta\left(y_i \mid X, y^{(t)}\right),$$

where $U^{(t)} \subseteq M^{(t)}$ is a set of parallel unmasking positions. However, position predictions in masked refinement are often statistically coupled. Committing multiple tokens at strongly coupled positions in the same step can introduce inconsistencies and degrade generation quality. As the classic example in prior work [36, 41], two-word poker hands (e.g., "high card," "two pair," "full house," "straight flush") must match as a pair, while parallel sampling can produce invalid combinations such as "high house". However, if we first decode one of them like "[MASK] house", then the probability of recovering a valid pair in the next step becomes much higher, which aligns with the conditional probability factorization: $p(y_i, y_j \mid X, y^{(t)}) = p(y_j \mid X, y^{(t)})\, p(y_i \mid y_j, X, y^{(t)})$. This nature of dLLMs suggests that parallel updates should account for position dependencies, avoiding strongly coupled positions while increasing the number of safely updated positions per step.

2.3 Attention Sinks Bias Dependency Proxies

Attention sinks [43, 33] are common in dLLMs. Across multiple methods and diverse samples, we repeatedly observe an abnormal concentration of attention on a small subset of keys. Moreover, the concentration distribution is not static. The specific sink tokens and the strength of aggregation can shift as the denoising step progresses. However, this phenomenon appears largely unrelated to the semantics of sink tokens. We provide a detailed empirical validation of this phenomenon in App. A.

Attention sinks bias attention-derived dependency proxies. Those sink positions attract considerable attention and can be misinterpreted as exerting a strong influence on many other tokens.

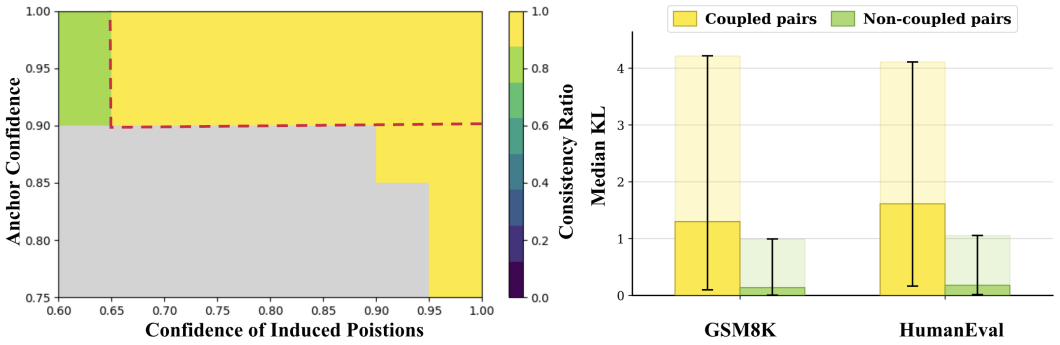


Figure 1: **Left: Heatmap of Induced Consistency.** Color indicates the consistency ratio (same as the final output) between induced predictions and the final output. Gray cells indicate bins with a negligible fraction of samples and are excluded from analysis. Implementation details are in App. B.1. **Right: Influence Between Masked Positions.** Bar height indicates the median KL divergence between a masked position's predictive distribution before and after its paired position is committed, reported separately for coupled and non-coupled pairs. For coupled pairs, the committed position is the source of the directed edge in the dependency graph; for non-coupled pairs, it is chosen at random. Error bars denote the [25%–75%] percentile range. Implementation details are in App. B.2. Additional results are in App. B.3, further validating this observation.

3 Observations

The effect of positional dependencies varies with the position state. We examine two cases: dependencies between unmasked–masked position pairs, and between masked–masked position pairs. Details follow in the subsequent subsections.

3.1 High-Confidence Unmasked Positions as Anchors

For unmasked-masked position pairs. During dLLM inference, masked positions can exhibit high consistency even when their confidence is not particularly high. Inspired by LocalLeap [18], we find that the induced positions that are strongly dependent on prompts or previously unmasked tokens with high confidence (anchors) often exhibit high consistency despite relatively low instantaneous confidence. This is validated by the highlighted region in the upper area of the left of Fig. 1: the induced positions can remain consistent with the final output when their confidence is low. In particular, when the corresponding anchors have confidence above 0.9, the induced tokens are consistently correct at a relatively low confidence. Therefore, the safety of parallel updates depends not only on confidence but also on whether the prediction is sufficiently conditioned on reliable context.

3.2 Coupled Masked Positions Exhibit Strong Influence

For masked–masked position pairs, since we represent positional relationships as a directed graph derived from attention maps (details at Sec. 4.1), the right of Fig. 1 shows that committing the source of a directed edge in a coupled pair induces a much larger KL shift in the target's predictive distribution than committing a randomly chosen position in a non-coupled pair. This suggests that sequentially decoding coupled positions yields more consistent joint outputs (the remaining position will be influenced), alleviating the *nonindependent parallel predictions* issue. It also validates the use of attention as a dependency proxy.

4 Methodology

Driven by the observations, we propose `DAWN`, a training-free, dependency-aware solution to accelerate dLLM inference while maintaining the generation quality. The key idea is to *extract positional dependencies and leverage them to select a larger set of reliable positions for update at each iteration*. `DAWN` realizes efficient inference by three pipelined modules, as shown in Fig. 2. *Dependency Graph*

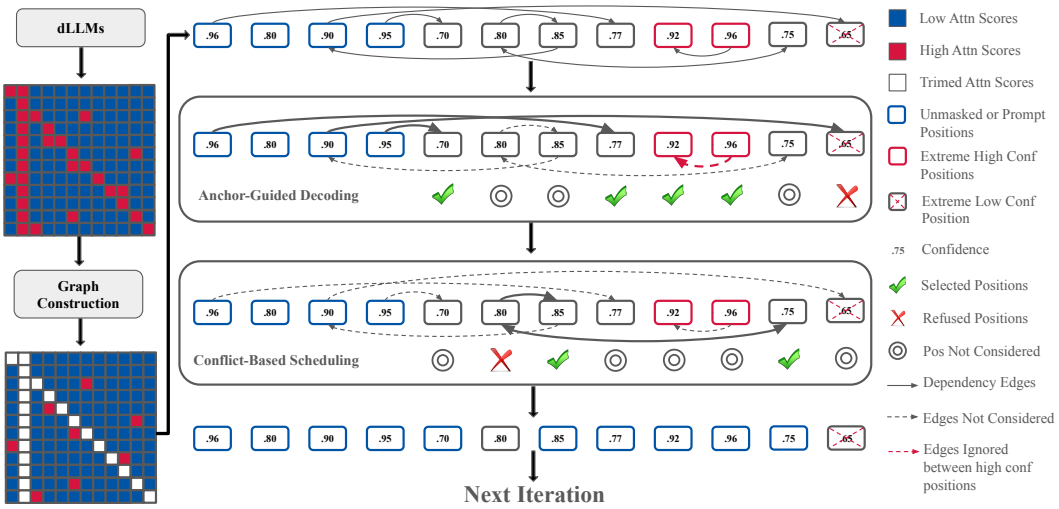


Figure 2: **Overview of** DAWN**.** Left: *Dependency Graph Construction* preprocesses the attention map and extracts a sparse directed dependency graph by retaining only salient (high-score) attention links. Middle: guided by this graph, *Anchor-Guided Decoding* and *Conflict-Based Scheduling* select two sets of positions, and the union of selected positions is unmasked simultaneously.

Construction builds a sparse directed graph from a lightweight proxy of token dependencies based on attention maps [10, 32, 16]. *Anchor-Guided Decoding* first selects high-confidence masked positions that can serve as anchors for the next forward and are approximately reliable [41], and then leverages high-confidence prompts or committed positions as anchors, enabling strongly coupled positions (induced positions) to be unmasked with a relatively low confidence. These together produce a parallel set $\mathcal{U}^{(t)}_{\text{anchor}}$. For the remaining tokens, *Conflict-Based Scheduling* uses conflict relations induced by the dependency graph to select a maximum independent set $\mathcal{U}^{(t)}_{\text{conflict}}$ from positions that meet a lower confidence threshold. Finally, the positions in $\mathcal{U}^{(t)}_{\text{anchor}} \cup \mathcal{U}^{(t)}_{\text{conflict}}$ are unmasked in parallel in this iteration. These three modules are pipelined within each denoising iteration to enable accurate and efficient inference. The following subsections describe their mechanisms in detail.

4.1 Dependency Graph Construction

Prior studies [10, 32, 16] indicate that attention maps capture information flow and dependency relations among tokens. Meanwhile, many existing works [7, 50, 49, 42] use attention maps to characterize token interactions, thereby enabling more efficient inference. Inspired by these, we treat attention maps as a lightweight, approximate proxy for token dependencies during decoding. This coupling signal is readily available from each forward pass and can be converted into a sparse directed dependency graph, which serves as the basis for subsequent efficient scheduling.

Since attention patterns evolve across denoising iterations, the dependency graph is constructed at each iteration from the model's attention maps. Raganato and Tiedemann [32] shows that deeper layers tend to encode more semantics. So to obtain a signal that is closer to the semantics and final prediction, attention weights are averaged across the last few layers and all heads. As discussed in Section 2.3, dLLMs often exhibit attention sinks, where a small set of positions absorbs most of the attention mass due to systematic bias rather than semantic relevance. This can lead to misleading dependent relations when attention is used as a proxy for dependency. To mitigate this effect, sink positions are identified via outlier detection and filtering of their incoming attention mass: given an aggregated attention matrix $A^{(t)} \in \mathbb{R}^{L\times L}$ at iteration t, the incoming attention mass of position j is defined as

$$\bar{A}^{(t)}_j = \frac{1}{L}\sum_{i=1}^{L} A^{(t)}_{i,j},$$

and position j is marked as a sink if $\bar{A}^{(t)}_j$ is larger than a predefined threshold τ_{sink}. Self-attention on the diagonal is ignored as it does not capture cross-position dependencies.

Given the processed attention proxy at iteration t, a directed sparse dependency graph is constructed to capture salient token couplings during decoding. Specifically, to keep the graph sparse and focus on the most informative relations, edges are retained based on thresholded (τ_{edge}) attention scores. A directed edge $j \to i$ is added if query position i places a sufficiently large attention mass on key position j, indicating that the prediction at token i can be significantly conditioned by token j. The resulting graph provides an approximate representation of positional dependencies and will be used by subsequent modules.

4.2 Anchor-Guided Decoding

In fact, some low-confidence tokens at masked positions can still be consistent with the final response, indicating that focusing only on high-confidence positions remains a limitation for dLLM inference. Motivated by this, *Anchor-Guided Decoding* is introduced to accept approximately independent positions and reliable positions under a relaxed confidence criterion.

As discussed in Sec. 3.1, low-confidence positions that are strongly coupled to previously unmasked positions meet the confidence threshold 0.9 and often match the final result. Following this observation, we define anchors as positions that are prompts or have been unmasked and whose confidence meets τ_{high}, where τ_{high} is observed as 0.9. For each step, we first select positions that meet τ_{high} and will likely serve as anchors in subsequent steps, which is also supported by prior work [41] showing that parallel decoding remains relatively stable among sufficiently high-confidence positions. Then, using the dependency graph, we identify the induced positions $\mathcal{I}$ as masked positions that are reachable via directed edges from anchor tokens. Intuitively, higher-confidence anchors provide more reliable context, allowing the corresponding induced positions to be unmasked at lower confidence. Accordingly, the set of positions eligible to be unmasked at the denoising step t is defined as:

$$\mathcal{U}^{(t)}_{\text{anchor}} = \left\{ i \;\middle|\; \begin{array}{l} i \in M^{(t)},\ c_i \geq \tau_{\text{high}} \text{ or} \\ i \in \mathcal{I}^{(t)},\ c_i \geq \tau_{\text{induced}} \end{array} \right\}.$$

where $\tau_{induced}$ is the confidence threshold for unmasking induced positions. Overall, *Anchor-Guided Decoding* selects approximately reliable positions under a high confidence threshold and relaxes the threshold for induced positions, enabling more efficient inference.

4.3 Conflict-Based Scheduling

For the remaining low-confidence positions that are strongly coupled and satisfy a lower confidence threshold τ_{low}, simultaneously unmasking them may introduce inconsistent commitments, as discussed in Sec. 2.2. To mitigate such errors while retaining high parallelism, *Conflict-Based Scheduling* is introduced to prevent low-confidence but highly coupled positions from being decoded in the same step.

Algorithm 1 Conflict-Based Scheduling

Input: remaining candidate positions $\mathcal{C}^{(t)}$, anchor update set $\mathcal{U}^{(t)}_{\text{anchor}}$, confidence scores $\{c_i\}_{i\in\mathcal{C}^{(t)}}$, conflict neighbors $\{\mathcal{N}(i)\}_{i\in\mathcal{C}^{(t)}}$, lower threshold τ_{low}
Output: parallel update set $\mathcal{U}^{(t)}_{\text{conflict}}$
Initialize $\mathcal{U}^{(t)}_{\text{conflict}} \leftarrow \emptyset$
$\mathcal{X} \leftarrow \mathcal{U}^{(t)}_{anchor} \cup \bigcup_{i\in\mathcal{U}^{(t)}_{anchor}} \mathcal{N}(i)$ {selected positions and their conflicts}
$\mathcal{R} \leftarrow \{\, i \in \mathcal{C}^{(t)} \mid c_i \geq \tau_{\text{low}} \,\} \setminus \mathcal{X}$ {remaining candidates}
while $\mathcal{R} \neq \emptyset$ **do**
  Select $i^\star \leftarrow \arg\max_{i\in\mathcal{R}} c_i$
  $\mathcal{U}^{(t)}_{\text{conflict}} \leftarrow \mathcal{U}^{(t)}_{\text{conflict}} \cup \{i^\star\}$
  $\mathcal{R} \leftarrow \mathcal{R} \setminus (\{i^\star\} \cup \mathcal{N}(i^\star))$
end while
return $\mathcal{U}^{(t)}_{\text{conflict}}$

Since the dependency graph captures salient positional dependencies, it can be used to identify strongly coupled position pairs. Sec. 3.2 indicates that strongly coupled masked positions exert significant influence. By first decoding the source position in a coupled pair, the other can be predicted conditioned on the committed position at the next step, yielding more consistent outputs and mitigating the inconsistency introduced by nonindependent parallel predictions if update them together. Therefore, a conflict relation is defined between two positions connected by an edge in the dependency graph, regardless of direction: if either $i \to j$ or $j \to i$ is present, then positions i and j are considered to be in conflict and should not be unmasked simultaneously.

Algorithm 1 shows the detailed procedure. Specifically, at iteration t, based on these conflicts and the graph topology, a greedy independent set is constructed from the remaining positions that (i) are

Table 1: **Performance comparison between DAWN and baselines across 4 datasets and 4 models**. We report Accuracy, TPS, and Speedup to assess their efficiency and generation quality.

Method	GSM8K			MATH			HumanEval			MBPP		
	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑
LLaDA-8B-Instruct												
Vanilla	77.94	10.32	1.00×	33.20	14.10	1.00×	40.24	26.46	1.00×	29.60	9.46	1.00×
Confidence	78.39	33.44	3.24×	32.98	36.88	2.62×	40.85	86.73	3.28×	30.00	34.81	3.68×
KLASS	76.12	23.23	2.25×	32.32	25.27	1.79×	36.59	56.91	2.15×	27.00	22.62	2.39×
LocalLeap	77.33	44.19	4.28×	32.42	47.12	3.34×	39.63	**109.80**	**4.15**×	30.80	44.13	4.66×
DAWN	77.94	**44.72**	**4.33**×	32.36	**48.16**	**3.42**×	40.24	108.99	4.12×	30.80	**45.17**	**4.77**×
LLaDA-1.5												
Vanilla	81.12	9.65	1.00×	33.36	12.75	1.00×	43.90	9.26	1.00×	38.80	3.45	1.00×
Confidence	80.74	32.49	3.37×	33.38	32.87	2.58×	42.68	23.39	2.53×	38.80	20.40	5.91×
KLASS	77.63	22.63	2.35×	31.80	23.29	1.83×	39.63	16.49	1.78×	30.00	13.25	3.84×
LocalLeap	80.14	42.60	4.41×	32.44	42.36	3.32×	42.07	**30.21**	**3.26**×	39.20	27.00	7.83×
DAWN	80.82	**43.24**	**4.48**×	31.80	**42.84**	**3.36**×	42.07	29.53	3.19×	37.60	**27.80**	**8.06**×
Dream-v0-Base-7B												
Vanilla	76.42	14.37	1.00×	34.70	18.16	1.00×	38.41	33.66	1.00×	53.40	11.85	1.00×
Confidence	73.84	23.41	1.63×	34.38	44.66	2.46×	39.63	61.54	1.83×	53.80	50.37	4.25×
KLASS	72.00	12.58	0.88×	32.80	23.16	1.27×	42.68	31.32	0.93×	55.80	19.46	1.64×
LocalLeap	72.55	25.40	1.77×	34.48	51.32	2.83×	36.59	68.28	2.03×	53.60	59.38	5.01×
DAWN	73.54	**25.88**	**1.80**×	34.00	**51.45**	**2.83**×	39.63	**68.96**	**2.05**×	53.20	**64.55**	**5.45**×
Dream-v0-Instruct-7B												
Vanilla	76.35	7.30	1.00×	37.80	17.73	1.00×	53.66	22.66	1.00×	54.20	8.37	1.00×
Confidence	75.20	28.81	3.95×	38.24	37.92	2.14×	57.32	52.11	2.30×	55.60	39.09	4.67×
KLASS	71.72	12.93	1.77×	34.66	18.74	1.06×	54.88	26.71	1.18×	56.80	18.53	2.21×
LocalLeap	73.24	32.94	4.51×	38.10	43.66	2.46×	54.27	59.28	2.62×	55.00	42.71	5.10×
DAWN	73.16	**32.99**	**4.52**×	38.22	**44.18**	**2.49**×	54.88	**60.23**	**2.66**×	55.80	**44.03**	**5.26**×

not yet included in and not conflicting neighbors of $\mathcal{U}_{\text{anchor}}^{(t)}$, and (ii) satisfy τ_{low}, yielding additional positions that can be decoded in parallel. Concretely, positions are greedily selected in descending order of confidence: the highest-confidence position is added to the update set $\mathcal{U}_{\text{conflict}}^{(t)}$, and all of its conflicting neighbors are removed from further consideration. This procedure repeats until no candidate positions remain.

In practice, quality degradation from naively lowering the confidence threshold is largely driven by positional coupling. By explicitly avoiding simultaneous unmasking of strongly coupled positions, *Conflict-Based Scheduling* helps maintain decoding quality while allowing a lower confidence threshold τ_{low}, thus speeding up the inference.

5 Experiments

5.1 Setups

Models and Benchmarks. We evaluate our approach on several variants of two models: LLaDA-8B-Instruct [27], LLaDA-1.5 [53], Dream-v0-Base-7B [47], Dream-v0-Instruct-7B. Benchmarks include diverse datasets: GSM8K (5-shot) [11], MATH (4-shot) [14], HumanEval (0-shot) [8], and MBPP (3-shot) [2], covering a range of reasoning and code generation tasks. We report tokens per second (TPS), relative speedup ratio (Speedup) and number of function evaluations (NFE) to reflect efficiency, along with task accuracy (Acc.).

Baselines. We compare DAWN against four baselines: the Vanilla sampling method (Top-1 Sampling), which unmasks the top-1 confidence position at each iteration; the Confidence-Aware Parallel proposed by Fast-dLLM [41], selecting positions whose confidence exceeds a predefined threshold; KLASS [17], using both confidence and KL divergence to select positions; and LocalLeap [18], identifying anchors and performing localized relaxed parallel decoding.

Hardware and Implementation Details. Our experiments are conducted on an NVIDIA H100 80G GPU. We set the generation length to 256 and the block length to 32 for all methods except KLASS, which uses its best-performing block length. All baselines are evaluated under their default hyperparameter settings. For DAWN, τ_{high} is set to 0.9 according to our observation and Fast-dLLM. τ_{low} is set from 0.7 to 0.85. Full configurations and the justifications can be found in App. C.1. All

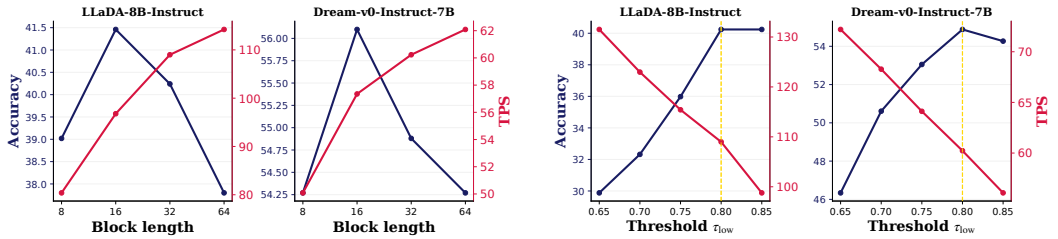


Figure 3: **Left: Effectiveness of DAWN on HumanEval under different block lengths**. We report accuracy (blue, left y-axis) and TPS (red, right y-axis). **Right: We vary the lower threshold τ_{low} and report accuracy (blue, left y-axis) and throughput TPS (red, right y-axis) on HumanEval**. The dashed line (yellow) marks the default setting $\tau_{low} = 0.80$.

evaluations are conducted using the standardized lm-eval [12] library. KV cache integration is trivial and would yield further speedup, but we omit it for consistent comparison with other works.

5.2 Main Results

Tab. 1 reports the accuracy and efficiency of each method across different models on four benchmarks.

Overall, DAWN achieves a substantial improvement in inference speed while maintaining accuracy comparable to or even slightly higher than the vanilla method. DAWN achieves substantially higher throughput than prior decoding baselines, with up to 8.06× speedup on MBPP using the LLaDA-1.5 model. Moreover, these gains do not come at the cost of quality. On LLaDA-8B-Instruct, DAWN achieves 77.94 accuracy on GSM8K, matching the vanilla method, and attains 30.80 on MBPP, slightly higher than the vanilla 29.60. In other settings, DAWN incurs only negligible quality loss, indicating a more favorable quality–speed trade-off.

Compared with confidence-aware parallel and KLASS, DAWN significantly improves throughput while achieving nearly identical accuracy. Compared with LocalLeap, DAWN shows advantages in some cases: it improves throughput by approximately 0.05 – 5.17 tokens per second while enhancing accuracy by up to 3.04. While the overall performance of the two methods is often comparable, they differ fundamentally in both motivation and methodology. LocalLeap leverages local determinism around high-confidence tokens to accelerate decoding, whereas DAWN explicitly models positional dependencies through attention-derived graphs and exploits them for both *Anchor-Guided Decoding* and *Conflict-Based Scheduling*. Further evaluation and statistical results are in App. C.2 and App. D.

5.3 Ablation Study

We conduct extensive ablation studies to assess the contributions of key components in DAWN. We further examine the sensitivity of DAWN to the block length, the lower confidence threshold τ_{low}, and the generation length to understand how these factors affect its effectiveness. All other settings follow the main experiment.

Effectiveness of key components. Since *Dependency Graph Construction* serves as the foundation for the other two modules, we focus on validating the effectiveness of components for selecting update set. Tab. 2 summarizes ablations of DAWN on two representative dLLMs and benchmarks. In particular, adding *Anchor-Guided Decoding* (AGD) alone on top of the vanilla baseline substantially improves throughput while maintaining or even slightly improving accuracy. On LLaDA-8B-Instruct, TPS rises from 10.32 to 33.83 on GSM8K while accuracy

Table 2: **Ablation study on the effectiveness of key modules in DAWN across 2 datasets and 2 models**. We report Accuracy, TPS, and NFE to assess their quality and speed. The settings are the same as in the main experiments.

Method	GSM8K			HumanEval		
	Acc.↑	TPS↑	NFE↓	Acc.↑	TPS↑	NFE↓
LLaDA-8B-Instruct						
DAWN	77.94	**44.72**	**55.76**	40.24	**109.0**	**61.63**
+ CBS	76.80	22.31	112.9	**41.46**	51.51	131.0
+ AGD	**78.47**	33.83	74.69	41.46	92.52	72.93
Vanilla	77.94	10.32	256	40.24	26.46	256
Dream-v0-Instruct-7B						
DAWN	73.16	**32.99**	**51.92**	54.88	**60.23**	**77.12**
+ CBS	73.31	29.33	58.13	54.88	54.79	83.34
+ AGD	75.28	27.63	62.19	**57.31**	51.97	93.01
Vanilla	**76.35**	7.30	256	53.66	22.66	256

improves from 77.94 to 78.47, indicating that AGD accelerates inference with virtually no accuracy loss. Adding *Conflict-Based Scheduling* (CBS) alone instead yields strong efficiency gains at the cost of a small accuracy drop. On Dream-v0-Instruct-7B, CBS even outperforms AGD in speedup, lifting TPS from 22.66 to 54.79 on HumanEval. Overall, the full solution consistently achieves the best efficiency while maintaining comparable accuracy, demonstrating that both modules contribute significantly to the inference acceleration.

Impact of block length. The left of Fig. 3 reports the accuracy and throughput of DAWN with different block lengths. As the block length increases, throughput increases for both methods as higher parallelism, but accuracy first increases and then decreases. Overall, it preserves a robust quality-speed trade-off across a wide range of lengths.

Impact of lower threshold. The right of Fig. 3 reports the evaluation under different values of lower threshold τ_{low}. We observe a clear quality-speed trade-off on both models. Reducing τ_{low} increases TPS by admitting more parallel updates, but can hurt accuracy due to less reliable low-confidence commitments. The default $\tau_{low} = 0.80$ (dashed line) maintains a high generation quality and improved efficiency.

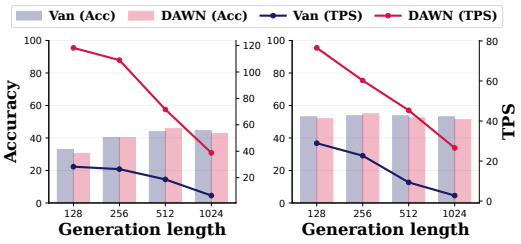


Figure 4: **Effectiveness of DAWN and the vanilla on HumanEval under different generation lengths**. Bars report accuracy (left y-axis) and solid lines report TPS (right y-axis).

Impact of generation length. Fig. 4 reports the accuracy and TPS of DAWN and the vanilla with different generation lengths L. Across both LLaDA and Dream models, DAWN consistently improves TPS over the vanilla while maintaining comparable accuracy. As L increases, TPS decreases for both methods due to higher denoising costs, yet DAWN continues to boost the efficiency. Overall, it preserves a more favorable quality-speed trade-off across a wide range of lengths.

6 Related Work

Diffusion Large Language Models. Recently, diffusion language models have gained renewed interest and emerged as a competitive alternative for text generation, demonstrating promising progress across various tasks. Representative approaches include pretraining dLLMs from scratch [27] and building dLLMs on top of existing AR models [47]. In parallel, several commercial systems [38, 20, 13] highlight the feasibility and practical potential of diffusion-based generation. Recent works [40, 5, 23] continue to advance dLLMs along multiple dimensions, including scaling to larger model size and exploring alternative training-inference paradigms for faster refinement. Moreover, diffusion-based modeling has been extended to multimodal settings, where multimodal dLLMs [48, 45] demonstrate strong performance across various tasks and point toward more unified diffusion-based generative models.

Efficient Inference of dLLMs. Despite dLLMs' ability to parallel decode, practical challenges remain during inference, motivating more efforts on faster and more reliable decoding. Existing works focus on the KV Cache optimization [41, 26], early stopping [46, 21], distillation-based [9] acceleration, and others [51, 37, 25]. These directions have shown promising gains in improving the efficiency of dLLM inference. Beyond these optimizations, numerous studies have explored optimized sampling strategies for dLLM inference. Fast-dLLM [41] adopts a confidence-aware strategy that unmasks multiple positions when their scores are sufficiently high, thereby making the independence approximation more reliable. EB-Sampler [3] uses the entropy of predictive distributions to decide which positions are safe to update in parallel. KLASS [17] further incorporates temporal stability by comparing distributions across iterations via the KL divergence. WINO [15] follows a draft-and-verify style: it drafts many tokens in parallel and selectively regenerates those that fail verification. Spiffy and related works [1, 39] apply speculative decoding style strategies to dLLMs to accelerate diffusion decoding. LocalLeap [18] leverages a local determinism hypothesis, observing that positions adjacent to high-confidence commits tend to stabilize earlier and can therefore be updated more aggressively. Unlike the above methods, this work explicitly approximates token coupling during inference and leverages this approximation to guide efficient parallel sampling.

7 Limitations

DAWN introduces several hyperparameters that require tuning to achieve a favorable performance. However, as discussed in App. C.1, performance is primarily sensitive to τ_{low}, while the remaining parameters can adopt fixed defaults for each model. This reduces the practical tuning burden.

8 Conclusion

This work leads to a better quality-speed trade-off for dLLM inference, narrowing the gap to state-of-the-art language models in practical generation settings. Specifically, we propose DAWN, a training-free, dependency-aware fast inference method for dLLMs. It is primarily motivated by the inherent challenge of nonindependent position predictions in dLLMs and mitigates this issue by adopting a positional-dependency perspective, offering a complementary approach to alleviate failures in parallel unmasking. Guided by a sparse directed dependency graph, DAWN selects unmasking positions at each iteration and enables highly parallel updates while preserving generation quality. Extensive experiments across multiple models and datasets validate the effectiveness of DAWN, demonstrating consistent speedups while maintaining comparable quality.

Acknowledgments and Disclosure of Funding

This research/project is supported by A*STAR under the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative (Award: I2501E0045 (H25-MCP0826)), as well as cash and in-kind contribution from the industry partner(s).

The authors declare no competing interests.

References

[1] Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Mingu Lee, Christopher Lott, and Fatih Porikli. Spiffy: Multiplying diffusion llm acceleration via lossless speculative decoding, 2026. URL https://arxiv.org/abs/2509.18085.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking, 2025. URL https://arxiv.org/abs/2505.24857.

[4] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

[5] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

[6] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[7] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models, 2024. URL https://arxiv.org/abs/2403.06764.

[8] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[9] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *arXiv preprint arXiv:2509.26488*, 2025.

[10] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL workshop BlackboxNLP: analyzing and interpreting neural networks for NLP*, pages 276–286, 2019.

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

[12] 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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

[13] Google DeepMind. Gemini diffusion: Our state-of-the-art, experimental text diffusion model. `https://deepmind.google/models/gemini-diffusion/`, 2025. Accessed: 2026-01-10.

[14] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[15] Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *arXiv preprint arXiv:2507.18578*, 2025.

[16] Shawn Im, Changdae Oh, Zhen Fang, and Sharon Li. How do transformers learn to associate tokens: Gradient leading terms bring mechanistic interpretability, 2026. URL `https://arxiv.org/abs/2601.19208`.

[17] Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models, 2025. URL `https://arxiv.org/abs/2511.05664`.

[18] Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. Accelerating diffusion llm inference via local determinism propagation, 2025. URL `https://arxiv.org/abs/2510.07081`.

[19] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[20] Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

[21] Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models, 2025. URL `https://arxiv.org/abs/2508.00819`.

[22] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL `https://arxiv.org/abs/2503.01840`.

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

[24] Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

[25] Lizhuo Luo, Shenggui Li, Yonggang Wen, and Tianwei Zhang. Dsb: Dynamic sliding block scheduling for diffusion llms, 2026. URL `https://arxiv.org/abs/2602.05992`.

[26] Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

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

[28] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

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

[30] Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*, 2026.

[31] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

[32] Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: analyzing and interpreting neural networks for NLP*, pages 287–297, 2018.

[33] Maximo Eduardo Rulli, Simone Petruzzi, Edoardo Michielon, Fabrizio Silvestri, Simone Scardapane, and Alessio Devoto. Attention sinks in diffusion language models, 2025. URL `https://arxiv.org/abs/2510.15731`.

[34] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

[35] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

[36] Jiaming Song and Linqi Zhou. Ideas in inference-time scaling can benefit generative pre-training algorithms, 2025. URL https://arxiv.org/abs/2503.07154.

[37] Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction, 2025. URL https://arxiv.org/abs/2508.02558.

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

[39] Linye Wei, Wenjue Chen, Pingzhi Tang, Xiaotian Guo, Le Ye, Runsheng Wang, and Meng Li. Orchestrating dual-boundaries: An arithmetic intensity inspired acceleration framework for diffusion language models. *arXiv preprint arXiv:2511.21759*, 2025.

[40] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm, 2025. URL https://arxiv.org/abs/2509.26328.

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

[42] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

[43] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

[44] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers, 2024. URL https://arxiv.org/abs/2410.10629.

[45] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[46] Yicun Yang, Cong Wang, Shaobo Wang, Zichen Wen, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Diffusion llm with native variable generation lengths: Let [eos] lead the way, 2025. URL https://arxiv.org/abs/2510.24605.

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

[48] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

[49] Jintao Zhang, Jia Wei, Haofeng Huang, Pengle Zhang, Jun Zhu, and Jianfei Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. *arXiv preprint arXiv:2410.02367*, 2024.

[50] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025.

[51] Tianao Zhang, Zhiteng Li, Xianglong Yan, Haotong Qin, Yong Guo, and Yulun Zhang. Quant-dllm: Post-training extreme low-bit quantization for diffusion large language models, 2025. URL https://arxiv.org/abs/2510.03274.

[52] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

[53] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models, 2025. URL https://arxiv.org/abs/2505.19223.

A Attention Sinks Validation

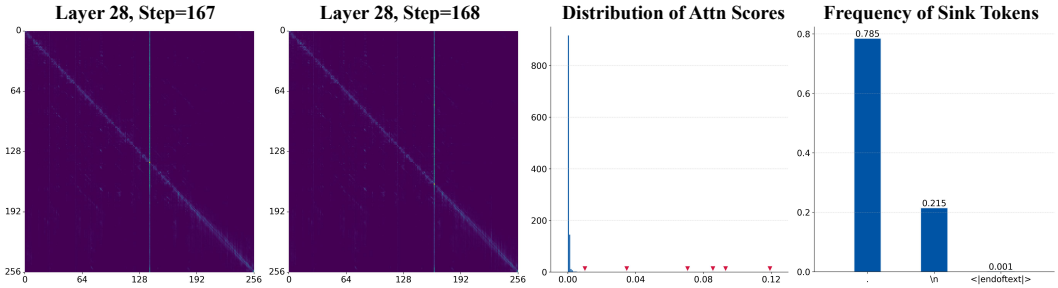


Figure 5: **Attention Sinks in dLLMs.** We conduct experiments on multiple samples using LLaDA-8B-Instruct. Left: The two heatmaps show partial attention maps from the same layer at different denoising steps, illustrating that the attention sink shifts across denoising iterations. Right: The third plot reports the distribution of attention scores corresponding to the first plot, and the rightmost plot reports the frequency of sink tokens from multiple samples.

As shown in Fig. 5, each heatmap exhibits an abnormal concentration of attention mass. Comparing attention maps across denoising steps further indicates that the sink location shifts over iterations. In the rightmost plot, most identified sink tokens are punctuation marks or special tokens with little lexical meaning, suggesting that attention-sink formation is largely independent of token semantics.

B Implementation Details of Observations

B.1 Implementation of Heatmap of Induced Consistency.

We conduct this observation on LLaDA-8B-Instruct using 100 randomly sampled instances from GSM8K and 100 from HumanEval, all decoded under vanilla (top-1) sampling with generation length $L = 256$, block length 32, and temperature 0.

For each request, we first run decoding to completion using vanilla sampling and store the resulting fully unmasked sequence as the reference output. We then replay the same trajectory and, at every denoising step, log per-edge statistics as follows. We form a sparse directed dependency graph as Sec. 4.1. The anchor is restricted to positions and prompt and active blocks, and the induced position is restricted to currently masked positions. For each retained edge, we record the anchor confidence, the induced confidence, and a binary consistency label indicating whether the induced position's current argmax prediction already matches the token at the same position in the reference output.

Records are binned in the (anchor confidence, induced confidence) plane with width 0.05. Since the same anchor–induced pair recurs across consecutive steps while the induced position remains masked, we deduplicate within each bin by retaining only the earliest observation, which provides the most conservative test of consistency at that confidence level. The reported consistency ratio per bin is the fraction of retained records with a positive consistency label, and bins with fewer than 50 records are rendered in gray.

B.2 Implementation of Influence Between Masked Positions.

We conduct this observation on LLaDA-8B-Instruct using 100 randomly sampled instances from GSM8K and 100 from HumanEval, decoded under vanilla (top-1) sampling with generation length $L = 256$, block length 32, and temperature 0. The goal is to assess whether directed edges in our attention-derived dependency graph capture genuine influence: does unmasking the source of an edge induce a larger distributional shift at the target than unmasking an unrelated position?

At nine evenly spaced progress points from 10% to 90% of generation completion per request, we form a sparse directed dependency graph as in Sec. 4.1 and construct a candidate pool R of currently masked positions. Unordered pairs from R are partitioned into *coupled pairs* (those connected by at least one directed edge in the dependency graph) and *non-coupled pairs* (those with no edge between them); coupled pairs are capped at 50 per step, and non-coupled pairs are subsampled to match.

For each pair (p_a, p_b), we choose which position p_i to unmask first: for coupled pairs, p_i is set to the edge source (uniformly random if both directions are present); for non-coupled pairs, p_i is chosen uniformly at random. Let p_j denote the remaining position. We compare the model's existing prediction at p_j (P_{joint}) with the prediction obtained after committing p_i to its argmax and performing a fresh forward pass (P_{seq}), recording

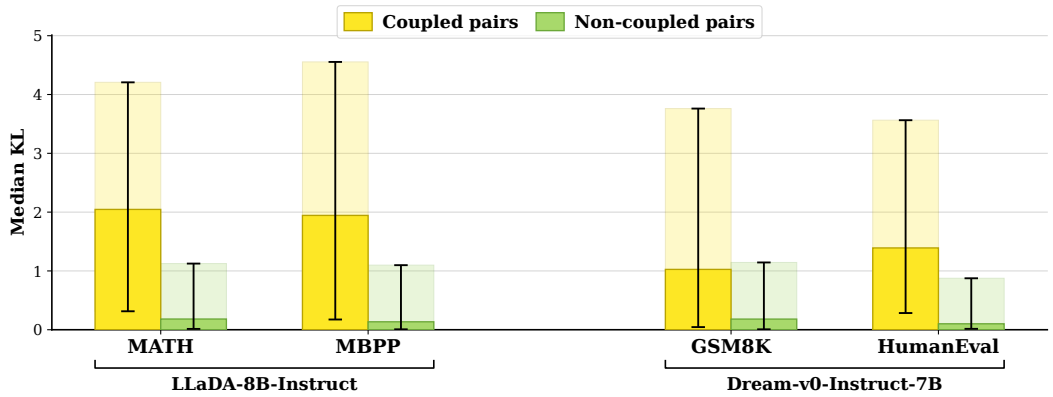


Figure 6: **Additional Validation of Influence Between Masked Positions.** Coupled pairs consistently exhibit larger KL divergence than non-coupled pairs, further validating the attention-derived dependency relations.

$\mathrm{KL}(P_{\text{joint}} \,\|\, P_{\text{seq}})$ as the per-pair coupling measure. All sequential forwards operate on cloned tensors and do not affect the underlying decoding trajectory. Reported statistics are the median KL and its interquartile range, computed separately for coupled and non-coupled pairs.

B.3 Additional Validation of Masked-Position Influence

To further strengthen our validation, we have added results on additional datasets and on the Dream model, which show the further generality. Fig. 6 shows KL statistics consistent with those in Fig. 1 (right), showing that coupled pairs exhibit substantially stronger mutual influence than non-coupled pairs. These additional results provide broader evidence for the reliability of the dependency proxy.

Rigorously establishing the mechanism underlying attention is beyond the scope of this work and would require substantial additional work. We therefore phrase our claims conservatively, ground in prior studies of attention mechanisms, and provide sufficient evidence for the specific properties required by our method. All settings consistently support our central claim that the proposed dependency proxy identifies token pairs with strong mutual influence.

C Experiment Details

C.1 Hyperparameter Settings

We conduct our experiments on several variants of two models: LLaDA-8B-Instruct, LLaDA-1.5, Dream-v0-Base-7B, Dream-v0-Instruct-7B. Benchmarks include diverse datasets: GSM8K (5-shot), MATH (4-shot), HumanEval (0-shot), and MBPP (3-shot), covering a range of reasoning and code generation tasks. Across all settings, we fix the block size to 32 and generation length to 256 tokens except KLASS, which uses its best-performing block length.

All baselines are evaluated under their default hyperparameter settings. To identify the optimal hyperparameter configuration for our method `DAWN` on different models, including τ_{edge}, $\tau_{induced}$ and τ_{sink}, we conduct a grid search on the HumanEval dataset. Specifically, we evaluate a range of candidate values for each parameter and plot the corresponding Accuracy–TPS curves, visualized in Fig. 7, 8, 9, 10.

In most cases, we observe a clear trade-off between throughput and accuracy: higher throughput is generally achieved at the cost of reduced accuracy. To balance this trade-off, we select hyperparameter values that lie near the Pareto frontier, favoring configurations that preserve high accuracy while providing meaningful efficiency gains. For example, on the LLaDA-8B-Instruct model, we select $\tau_{edge} = 0.07$, which achieves the highest accuracy among the tested values while offering higher throughput compared to $\tau_{edge} = 0.08$. This criterion ensures that the chosen configuration does not sacrifice accuracy for marginal speed improvements.

For *the lower confidence threshold* τ_{low} hyperparameter, we perform a grid search over both models and benchmarks, resulting in 16 experimental settings in total. Within each setting, we further tune the method-specific hyperparameters to obtain the best-performing configuration. The final selected results are reported in Tab. 3.

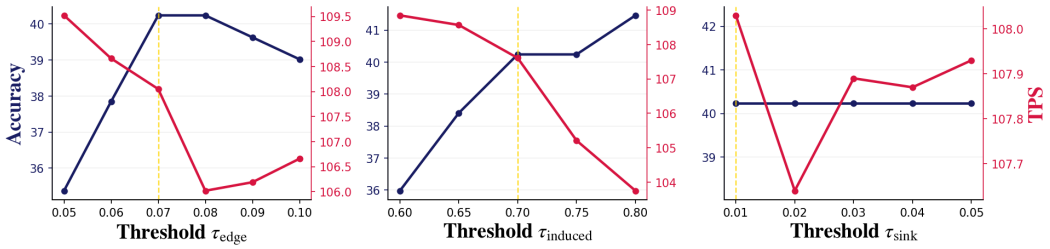


Figure 7: **We vary the threshold τ_{edge}, $\tau_{induced}$, τ_{sink} and report accuracy (blue, left y-axis) and throughput TPS (red, right y-axis) on LLaDA-8B-Instruct**. The dashed lines (yellow) mark the final settings $\tau_{edge} = 0.07$, $\tau_{induced} = 0.70$, $\tau_{sink} = 0.01$.

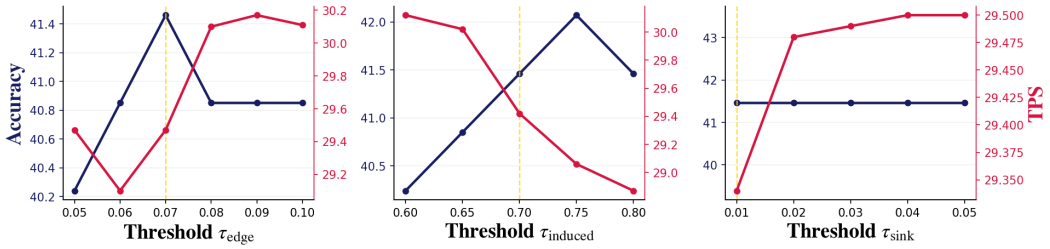


Figure 8: **We vary the threshold τ_{edge}, $\tau_{induced}$, τ_{sink} and report accuracy (blue, left y-axis) and throughput TPS (red, right y-axis) on LLaDA-1.5**. The dashed lines (yellow) mark the final settings $\tau_{edge} = 0.07$, $\tau_{induced} = 0.70$, $\tau_{sink} = 0.01$.

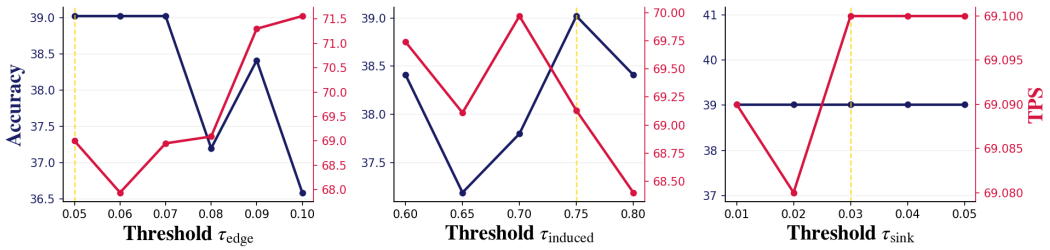


Figure 9: **We vary the threshold τ_{edge}, $\tau_{induced}$, τ_{sink} and report accuracy (blue, left y-axis) and throughput TPS (red, right y-axis) on Dream-v0-Base-7B**. The dashed lines (yellow) mark the final settings $\tau_{edge} = 0.05$, $\tau_{induced} = 0.75$, $\tau_{sink} = 0.03$.

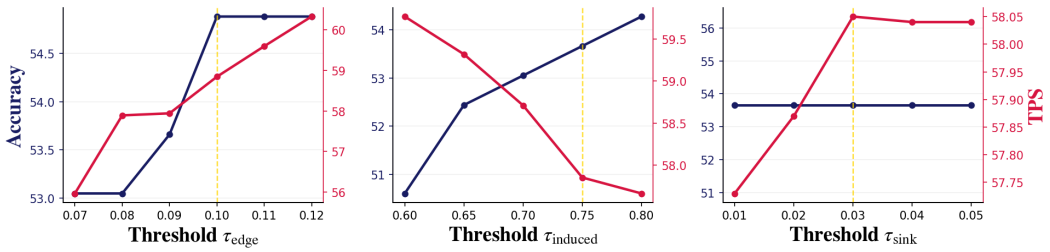


Figure 10: **We vary the threshold τ_{edge}, $\tau_{induced}$, τ_{sink} and report accuracy (blue, left y-axis) and throughput TPS (red, right y-axis) on Dream-v0-Instruct-7B**. The dashed lines (yellow) mark the final settings $\tau_{edge} = 0.10$, $\tau_{induced} = 0.75$, $\tau_{sink} = 0.03$.

However, fine-grained tuning of these thresholds introduces additional overhead. DAWN introduces several thresholds, including τ_{edge}, τ_{induced}, τ_{sink}, and τ_{low}, whose optimal values were lightly tuned in our main experiments to demonstrate the upper-bound performance of the method. Among them, τ_{low} is by far the most influential: variations in this threshold lead to substantial changes in the quality–speed trade-off, while the remaining thresholds exhibit much smaller sensitivity. In practice, users may focus tuning effort on τ_{low} alone and adopt

Table 3: **Final hyperparameter settings for DAWN, where rows denote models and columns denote hyperparameter types**. The $\tau_{induced}$, τ_{sink}, τ_{edge} are shared across benchmarks for each model, while τ_{low} adopts benchmark-specific configurations.

Model	$\tau_{induced}$	τ_{sink}	τ_{edge}	τ_{low}			
				GSM8K	MATH	HumanEval	MBPP
LLaDA-8B-Instruct	0.70	0.01	0.07	0.75	0.75	0.8	0.7
LLaDA-1.5	0.70	0.01	0.07	0.75	0.75	0.8	0.75
Dream-v0-Base-7B	0.75	0.03	0.05	0.75	0.8	0.8	0.8
Dream-v0-Instruct-7B	0.75	0.03	0.10	0.8	0.8	0.8	0.8

Table 4: **Performance comparison between DAWN and baselines on IFEval**. We report Accuracy and TPS to assess generation quality and efficiency.

Metric	Vanilla	Confidence	LocalLeap	DAWN	Vanilla	Confidence	LocalLeap	DAWN
	LLaDA-8B-Instruct, $\tau_{low} = 0.75$				LLaDA-1.5, $\tau_{low} = 0.8$			
Acc.	56.75	57.12	57.30	56.19	54.53	54.71	54.52	55.27
TPS	21.52	40.64	44.10	**45.26**	21.68	40.17	43.23	**43.33**
	Dream-v0-Base-7B, $\tau_{low} = 0.75$				Dream-v0-Instruct-7B, $\tau_{low} = 0.8$			
Acc.	22.55	21.81	21.63	21.81	27.54	28.47	28.10	27.54
TPS	37.28	53.28	55.28	**59.37**	23.74	43.94	45.90	**47.42**

Table 5: **Error-bar analysis of DAWN and representative baselines across 2 datasets and 4 models**. Results are reported as mean $\pm$ standard deviation over three runs.

Method	GSM8K		HumanEval		GSM8K		HumanEval	
	Acc.	TPS	Acc.	TPS	Acc.	TPS	Acc.	TPS
	LLaDA-8B-Instruct				LLaDA-1.5			
Vanilla	77.94	10.40 ± 0.08	40.24	26.73 ± 0.27	81.12	9.73 ± 0.07	43.90	9.29 ± 0.03
LocalLeap	77.33	44.64 ± 0.43	39.63	109.9 ± 0.63	80.14	42.88 ± 0.24	42.07	30.44 ± 0.20
DAWN	77.94	45.12 ± 0.37	40.24	109.2 ± 0.70	80.82	43.46 ± 0.19	42.07	29.72 ± 0.17
	Dream-v0-Base-7B				Dream-v0-Instruct-7B			
Vanilla	76.42	14.41 ± 0.03	38.41	33.55 ± 0.18	76.35	7.320 ± 0.02	53.66	23.00 ± 0.30
LocalLeap	72.55	25.68 ± 0.24	36.59	69.11 ± 0.72	73.24	33.05 ± 0.10	54.27	59.71 ± 0.38
DAWN	73.54	25.68 ± 0.19	39.63	69.68 ± 0.66	73.16	33.07 ± 0.08	54.88	60.98 ± 0.65

fixed defaults for the others. Specifically, we recommend $\tau_{\text{induced}} = 0.7$, $\tau_{\text{sink}} = 0.01$, and $\tau_{\text{edge}} = 0.07$ for the LLaDA family of models, and $\tau_{\text{induced}} = 0.75$, $\tau_{\text{sink}} = 0.03$, $\tau_{\text{edge}} = 0.07$ for the Dream family. These defaults yield results comparable to fully tuned configurations across our benchmarks, suggesting that the practical tuning burden is light despite the apparent number of hyperparameters.

C.2 Additional Evaluation on IFEval

In fact, Reasoning and coding are among the primary task categories for dLLMs. To further broaden the evaluation scope, we additionally report results on IFEval (0-shot) [52], a general instruction-following, open-ended generation benchmark. KLASS is not included as no specific hyperparameter setting for this task. The setting of DAWN follows the main experiments. Tab. 4 demonstrates that DAWN remains effective on instruction-following and open-ended generation benchmarks, improving inference efficiency while preserving output quality.

C.3 Impact of Graph Sparsity

When the dependency graph is sparse, AGD induces fewer masked positions due to the reduced number of dependency relations, while CBS becomes less constrained by graph edges and can therefore select more tokens for decoding. Conversely, when the graph is dense, AGD induces more masked positions, whereas CBS becomes

more conservative because of the stronger dependency constraints. The parameter τ_{edge} controls the sparsity of the dependency graph at a global level: larger values produce sparser graphs. The ablation study of τ_{edge} in these figures shows that performance varies smoothly across different graph sparsities, without any abrupt degradation, demonstrating that DAWN is not overly sensitive to the choice of graph sparsity.

D Statistical Analysis

Most dLLM methods evaluated in our study use greedy decoding and are therefore deterministic with respect to the random seed. KLASS is the only stochastic baseline; however, given its substantial performance gap, we focus our robustness analysis on the strongest and most directly comparable baselines. To further verify the reliability of our results, we repeated representative experiments three times and report the error bar for vanilla decoding, LocalLeap, and DAWN across four models and two datasets.

Tab. 5 confirms the reliability of the results reported in the main experiments. The mean values lead to the same conclusions as the main results, while the small standard deviations indicate that performance remains stable across repeated runs. Overall, DAWN consistently improves inference efficiency while maintaining comparable accuracy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract, Introduction (Sec. 1) and Experiments (Sec. 5)

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitation (Sec. 7)

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Methodology (Sec. 4), Experiments (Sec. 5) and Implementary Details (Sec. B and Sec. C). And the code is submitted.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data is publicly available, and our code has been released as open source.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Experiment (Sec. 5) and Implementation Details (Sec. B and Sec. C) provide all details of experiments and the code is submitted.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the statistical analysis of experiment result on representative settings (Sec. D).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experiment (Sec. 5)

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper complies with the NeurIPS Code of Ethics in all respects.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper focuses on accelerating inference for dLLM models; it is basic research and has no social impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Experiment (Sec. 5)

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The Readme document is provided along with the code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.