DUET: DUAL-SPARSE INFERENCE FOR BLOCK DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) enable parallel token generation through iterative denoising, yet their inference remains costly due to repeated attention over a growing historical prefix and repeated processing of the entire current generation block. Historical attention becomes increasingly expensive as generation progresses, while all current-block positions are propagated through the Transformer at every denoising step even though only a small fraction are typically resolved at that step. In this paper, we introduce Duet, a training-free dual-sparse inference framework that addresses these challenges through prefix Key-Value (KV) sparsity and early query sparsity, respectively. Specifically, Duet exploits the concentration and cross-step stability of historical attention within each generation block to construct a compact historical KV subset that is reused across denoising steps. For query sparsity, Duet predicts update positions from shallow representations and propagates only the corresponding queries through the suffix Transformer layers, avoiding deep computation at positions unlikely to be updated in the current step. Experiments show that Duet maintains competitive accuracy while achieving 1.34×/2.83× end-to-end/per-step speedups on LLaDA2.1-mini and 2.41×/4.72× on SDAR-8B-Chat-b32 at a 32K context length.

1 INTRODUCTION

DLMs have recently emerged as an alternative paradigm for text generation, replacing strictly left-to-right autoregressive decoding with iterative denoising over multiple token positions (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). By updating several positions in parallel, DLMs offer a promising path toward more flexible and potentially faster generation (Wang et al., 2026). Recent advances have demonstrated competitive performance across general language understanding, mathematical reasoning, and code generation tasks (Bie et al., 2025), further motivating interest in efficient DLM inference.

However, the iterative nature of denoising introduces substantial computational redundancy. In block DLM (BDLM) decoding, a current generation block is iteratively refined over multiple denoising steps while conditioning on an generated prefix (Arriola et al., 2025). This inference pattern repeatedly incurs two major sources of unnecessary computation. First, every denoising step attends over the full historical prefix, making attention increasingly expensive as the context grows (Long et al., 2026). Second, all positions in the current block are typically propagated through the full Transformer stack at every denoising step, although only a subset of unresolved positions are ultimately selected for the current update (Liang et al., 2026). These two forms of redundancy jointly make BDLM inference increasingly expensive.

Motivated by these inefficiencies, we introduce Duet, a training-free dual-sparse inference framework for BDLMs. Duet addresses the two sources of redundant computation with complementary sparsity mechanisms. First, current-block queries concentrate much of their historical attention on a compact prefix subset that remains stable across denoising steps within the same generation block. Duet exploits this structure through historical *prefix KV sparsity*, selecting a compact historical KV subset once for each generation block and reusing it throughout the block. Second, the positions selected by the final decoding update become predictable from early Transformer representations. Duet leverages this predictability through *early query sparsity* within block, applying the model's original decoding policy at an early selection layer to identify active query rows and propagating

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

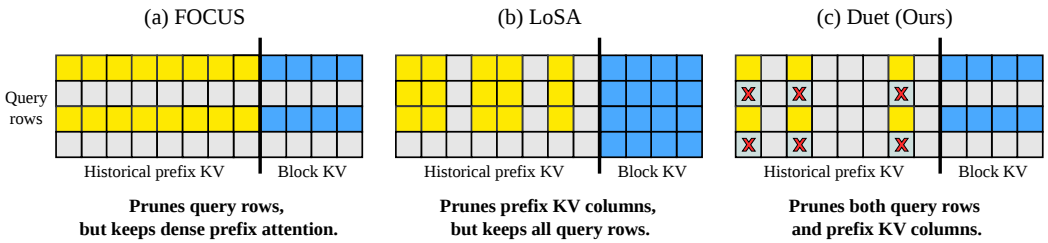


Figure 1: Illustration of different sparsity patterns in BDLM inference. FOCUS only reduces computation along the query dimension, whereas LoSA only sparsifies the historical prefix. Our method is designed around the two orthogonal computation dimensions, selectively reducing both the query rows propagated through deep layers and the historical prefix KV states they attend to.

only these rows through the remaining layers. Together, the two mechanisms reduce both historical attention and deep token-wise computation, including attention projections, FFN/MoE modules, and other token-wise operations. We evaluate Duet on LLaDA2.1-mini (Bie et al., 2026) and SDAR-8B-Chat-b32 (Cheng et al., 2025) across long-context and standard reasoning benchmarks. Duet consistently reduces inference cost while preserving competitive generation quality. At a 32K context length, Duet achieves 1.34×/2.83× end-to-end/per-step speedups on LLaDA2.1-mini and 2.41×/4.72× on SDAR-8B-Chat-b32. We further analyze the two empirical observations underlying Duet, validating that compact historical prefix subsets remain stable within a generation block and that final update positions become highly predictable at early Transformer layers.

Our contributions are summarized as follows:

- We identify two complementary sources of redundancy in BDLM inference and show that historical-prefix relevance remains stable within a generation block, while final update positions can be predicted from early Transformer representations.
- We propose Duet, a training-free dual-sparse inference framework that combines prefix KV sparsity with early query sparsity to reduce both historical attention and deep token-wise computation.
- We conduct extensive experiments on two representative BDLMs across long-context, reasoning, code generation, and knowledge benchmarks, demonstrating competitive generation quality and substantial inference speedups.

2 PRELIMINARIES

BDLM Inference. DLMs generate text through iterative denoising rather than left-to-right autoregressive decoding. We consider BDLM inference, where generation proceeds one block at a time. Let P denote the length of the currently generated prefix and B the size of the current generation block. At denoising step s, the input sequence is

$$\mathbf{x}^{(s)} = \left(\underbrace{x_1, \ldots, x_P}_{\text{Prefix}}, \underbrace{x_{P+1}^{(s)}, \ldots, x_{P+B}^{(s)}}_{\text{Block}} \right). \tag{1}$$

After the final layer, the model applies its decoding policy to the current-block representations and resolves a subset of positions, while the unresolved positions are processed again in subsequent denoising steps. Once all positions in the block are resolved, the decoded block is appended to the historical prefix and generation proceeds to the next block.

Within a generation block, the historical prefix remains fixed while the current-block representations evolve across denoising steps. Therefore, historical prefix KV states can be cached and reused, whereas the B current-block positions are repeatedly passed through the Transformer (Wu et al., 2026b;a). At layer l and step s, we denote the current-block queries by $\mathbf{Q}^{(l,s)} \in \mathbb{R}^{B \times H_q \times d_h}$ and the cached historical keys and values by $\mathbf{K}_P^{(l)}, \mathbf{V}_P^{(l)} \in \mathbb{R}^{P \times H_{kv} \times d_h}$. Each current-block query attends to the P cached prefix positions and the B positions in its own block. Suppressing head counts

and constant factors, the per-layer, per-step costs of dense attention and the feed-forward (FFN) or mixture-of-experts (MoE) module are

$$T^{(l)}_{\text{attn,dense}} = O\Big(B(P+B)d_h + Bc^{(l)}_{\text{proj}}\Big), \qquad T^{(l)}_{\text{ff,dense}} = O\Big(Bc^{(l)}_{\text{ff}}\Big), \tag{2}$$

where $c^{(l)}_{\text{proj}}$ is the per-row attention projection cost and $c^{(l)}_{\text{ff}}$ is the per-row cost of the FFN or activated MoE experts, including routing. The first attention term accounts for query–key interactions and value aggregation. These costs expose two complementary reduction opportunities: attending to fewer historical KV states and propagating fewer current-block query rows through deep Transformer layers.

Prefix-Sparse Attention. We first consider sparsity along the historical-prefix dimension. For clarity, consider a single attention head and omit layer and denoising step superscripts. Let $\mathbf{q}_i \in \mathbb{R}^{d_h}$ be a current-block query and $\mathbf{K}_P, \mathbf{V}_P \in \mathbb{R}^{P\times d_h}$ and $\mathbf{K}_B, \mathbf{V}_B \in \mathbb{R}^{B\times d_h}$ be the historical-prefix and current-block KV states, respectively. Dense attention computes

$$\mathbf{o}_i = \text{softmax}\left(\frac{\mathbf{q}_i[\mathbf{K}_P;\mathbf{K}_B]^\top}{\sqrt{d_h}}\right)[\mathbf{V}_P;\mathbf{V}_B]. \tag{3}$$

Prefix-sparse attention instead retains only a subset $\mathcal{S}_P \subseteq \{1,\dots,P\}$ of historical positions and computes

$$\widetilde{\mathbf{o}}_i = \text{softmax}\left(\frac{\mathbf{q}_i[\mathbf{K}_P[\mathcal{S}_P];\mathbf{K}_B]^\top}{\sqrt{d_h}}\right)[\mathbf{V}_P[\mathcal{S}_P];\mathbf{V}_B], \tag{4}$$

where the semicolon concatenates along the KV sequence dimension and $[\mathcal{S}_P]$ gathers the indexed historical rows. The original attention mask, when present, applies to both expressions. Only the historical KV portion changes; prefix sparsity does not remove the current-block KV states. Prefix sparsity therefore targets the context-length-dependent component of BDLM inference, which becomes increasingly important as P grows.

Query-Sparse Deep Computation. BDLM inference also repeatedly propagates all B current-block positions through the Transformer, even though only a subset may require deep computation at a given denoising step. We denote the active block positions at step s by $\mathcal{S}^{(s)}_Q \subseteq \{1,\dots,B\}$ and their number by $Q_s = |\mathcal{S}^{(s)}_Q|$. Query sparsity omits the inactive query rows from subsequent layers: these rows do not incur deep attention projections or query-side attention, nor the corresponding FFN/MoE computation. The current block still provides its KV states to the active queries. Thus, prefix sparsity reduces the historical KV context, while query sparsity reduces the number of current-block positions receiving deep computation. The next section describes how Duet combines them during BDLM decoding.

3 METHOD

In this section, we present Duet, a training-free dual-sparse inference framework that reduces redundant computation in BDLM decoding through prefix KV sparsity and early query sparsity. We first provide an overview of Duet, and then describe the two sparsity mechanisms in detail.

3.1 OVERVIEW

As shown in Figure 2, Duet applies two complementary forms of sparsity to a generation block of B positions conditioned on a prefix of length P. The first denoising step ($s = 0$) runs densely, providing the block queries and the full historical KV states needed for prefix selection. Duet then constructs a compact prefix cache from these states without an additional dense model forward. In subsequent steps, it processes the block densely through an early query-selection layer l_Q and propagates only active positions through the remaining layers. We use zero-based indices for layers and denoising steps.

For each suffix layer $l > l_Q$, *prefix KV sparsity* uses block queries and historical keys to construct a subset $\mathcal{S}^{(l)}_P \subseteq \{1,\dots,P\}$. This subset remains fixed within the current block; at the next block, selection is repeated over the full historical KV cache. Layers $0,\dots,l_Q$ retain the full prefix KV

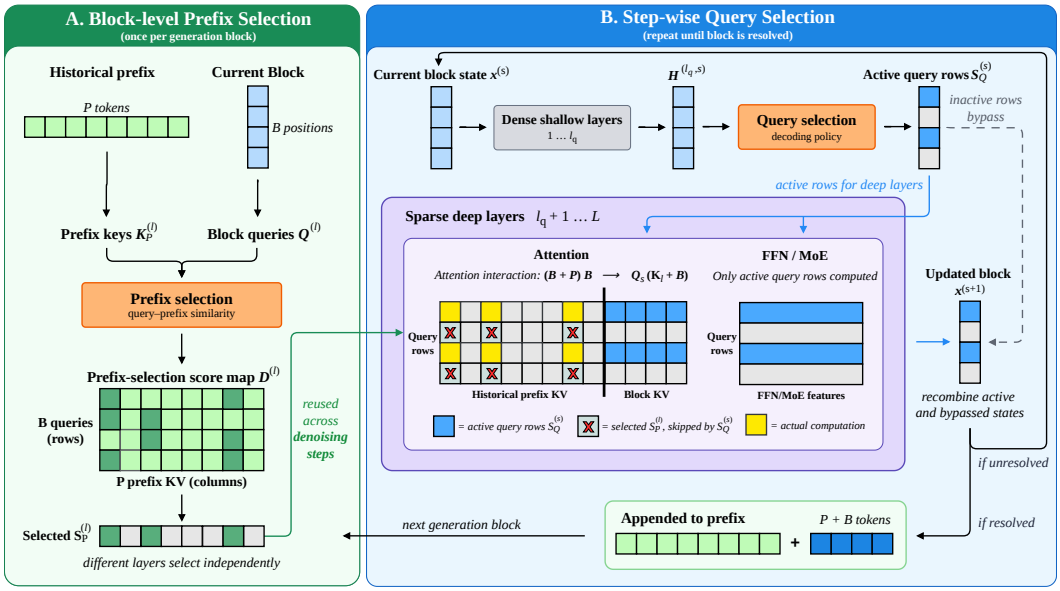


Figure 2: **Overview of Duet. (A) Block-level prefix selection:** After the first dense step, Duet selects a compact historical prefix subset for each sparse suffix layer and reuses the resulting KV cache within the generation block. **(B) Step-wise query selection:** in subsequent denoising steps, Duet identifies active query rows at an early selection layer and propagates only these rows through the remaining Transformer layers, reducing redundant computation in both attention and FFN/MoE. Active and bypassed states are then recombined for the next denoising step.

states. At each subsequent step $s \geq 1$, *early query sparsity* identifies an active set $\mathcal{S}_Q^{(s)} \subseteq \{1, \dots, B\}$ at layer l_Q. Only these Q_s rows traverse layers $l_Q + 1, \dots, L - 1$, while the other rows bypass the suffix computation.

Within the sparse suffix, active queries attend to the selected historical KV states together with the current block KVs. Restricting the suffix to $\mathcal{S}_Q^{(s)}$ also avoids attention projections and FFN/MoE computation for inactive rows. The resulting active states are scattered back to their original block positions and combined with the bypassed states. The process repeats until the block is resolved, after which it is appended to the historical prefix.

3.2 Prefix KV Sparsity

Design intuition. Within a generation block, the historical prefix remains fixed while the current block is iteratively refined. Empirically, we observe that block queries concentrate a large fraction of their prefix-attention mass on a compact subset of historical tokens, and that this subset remains largely stable throughout the denoising steps within the same block. This motivates selecting a compact historical context once at the beginning of each generation block and reusing the resulting KV cache throughout that block. We validate this block-level stability in Section 4.3.

Query–prefix similarity. Prefix selection is performed for each suffix layer $l > l_Q$, using queries captured during the first dense step. Let $\mathbf{q}_{h,i}^{(l,0)} \in \mathbb{R}^{d_h}$ be the attention-space query at block position i and query head h, and let $\mathbf{k}_{g(h),j}^{(l)} \in \mathbb{R}^{d_h}$ be the historical key at position j in its corresponding KV head $g(h)$ under grouped-query attention. We measure their similarity by

$$a_{hij}^{(l)} = \left\langle \mathbf{q}_{h,i}^{(l,0)}, \mathbf{k}_{g(h),j}^{(l)} \right\rangle. \tag{5}$$

Block-shared prefix selection. Rather than constructing a separate historical context for each query, Duet selects one prefix subset shared across the block and query heads. Given a target budget B_P, each of the BH_q query-head rows selects its $m = \max(1, \lceil B_P/(BH_q) \rceil)$ highest-scoring historical positions, and the candidates are merged. We score each historical position across all block queries

and query heads by

$$r_j^{(l)} = \max_{\substack{1\le i\le B\\ 1\le h\le H_q}} a_{hij}^{(l)}, \tag{6}$$

which captures the strongest match to any query-head row. If the candidate union contains fewer than $\min(B_P, P)$ distinct positions, we fill it with the highest-scoring remaining positions according to $r_j^{(l)}$; if it contains more, we retain the $\min(B_P, P)$ highest-scoring candidates. Thus, under a strict prefix budget, the shared set has size $K_l = |\mathcal{S}_P^{(l)}| = \min(B_P, P)$. Ranking by the unscaled dot product is equivalent to ranking by the scaled attention logit for a fixed head dimension. Scoring the full prefix costs $O(BPd_h)$ per suffix layer, suppressing head counts, and is performed once per generation block.

The selected indices gather the cached keys and values, $\widetilde{\mathbf{K}}_P^{(l)} = \mathbf{K}_P^{(l)}[\mathcal{S}_P^{(l)}]$ and $\widetilde{\mathbf{V}}_P^{(l)} = \mathbf{V}_P^{(l)}[\mathcal{S}_P^{(l)}]$. This compact cache is reused throughout the current block; the full historical cache remains available for selection at the next block.

3.3 Early Query Sparsity

Design intuition. At each denoising step, standard BDLM inference propagates all positions in the current generation block through every Transformer layer, even though only a subset of unresolved positions are ultimately selected for the current update. We observe that these update positions can already be predicted from early Transformer representations. This suggests that query selection can be performed before the final layer, allowing Duet to avoid redundant deep computation on positions unlikely to be updated. Accordingly, Duet designates an early Transformer layer l_Q as the selection layer and propagates only the predicted active query rows through the remaining deep layers. We empirically validate this early predictability in Section 4.3.

Early prediction of query positions. At a subsequent denoising step $s \ge 1$, Duet evaluates layers $0, \dots, l_Q$ densely over the current block, including attention to the full historical prefix, and obtains the intermediate representations $\mathbf{H}^{(l_Q,s)}$. The model's original decoding rule is then applied to $\mathbf{H}^{(l_Q,s)}$ to estimate which masked positions are likely to be selected for the current update. Let $\mathcal{U}^{(s)}$ denote the unresolved positions at step s. The query retention ratio ρ_Q sets a nominal target of $\lceil \rho_Q |\mathcal{U}^{(s)}| \rceil$ active unresolved queries, selected according to the model's decoding policy (confidence-based for LLaDA and in block order for sequential SDAR); previously resolved positions are always retained as well. Together, these positions form $\mathcal{S}_Q^{(s)}$, whose Q_s rows are propagated through the suffix layers. In this way, the retention ratio controls the amount of deep computation devoted to unresolved positions, while reusing the original decoding rule keeps early query selection consistent with the underlying BDLM inference procedure without introducing an additional learned predictor.

Sparse suffix computation. For $s \ge 1$ and $l > l_Q$, only the Q_s active query rows attend to the compact historical KV cache indexed by $\mathcal{S}_P^{(l)}$ and the B current-block KV positions. The first dense step initializes the current-block KV cache at all B positions. Subsequent sparse steps update the selected entries and reuse the others, so active queries still attend to all B block positions. Using the cost conventions of Equation 2, the per-layer sparse costs are

$$T_{\text{attn,sparse}}^{(l,s)} = O\Big(Q_s(K_l + B)d_h + Q_s c_{\text{proj}}^{(l)}\Big), \qquad T_{\text{ff,sparse}}^{(l,s)} = O\Big(Q_s c_{\text{ff}}^{(l)}\Big). \tag{7}$$

Compared with dense computation, the suffix attends over $K_l + B$ rather than $P + B$ KV positions and processes Q_s rather than B rows through attention projections and FFN/MoE. These expressions apply only after the first dense step and beyond l_Q; the first step and the preceding layers retain the dense costs of Equation 2. Because selection at the next block revisits the full prefix, the shorter attended KV sequence does not imply a reduction in the stored full historical KV cache.

Position-preserving update. Sparse suffix computation preserves the original block indices of all active query rows, allowing their updated representations to be written back to the corresponding positions after the deep layers. At the final decoding stage, the original decoding rule is applied only to the active unresolved positions $\mathcal{S}_Q^{(s)} \cap \mathcal{U}^{(s)}$. Unselected unresolved positions bypass the suffix computation, remain masked at this step, and are reconsidered at the next step. Thus, the per-step

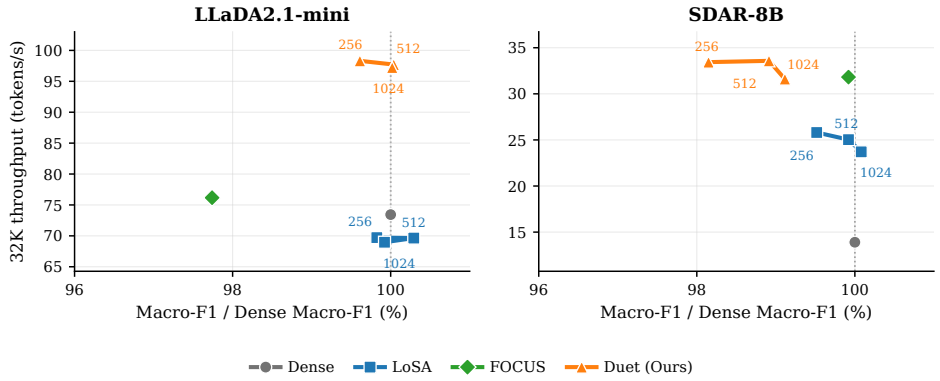


Figure 3: **LongBench quality–throughput trade-off.** Five-task Macro-F1 relative to each model's dense score (100%) versus 32K end-to-end throughput. Absolute Macro-F1 scores are reported in Table 1; task-wise results appear in Appendix Table 4.

Table 1: **LongBench Macro-F1 (%).** Column headings give nominal prefix budgets. For LoSA, parentheses give measured prefix KV counts, which can exceed those budgets. Higher is better.

			LoSA			Duet (Ours)		
Model	Dense	FOCUS	256	512	1024	256	512	1024
LLaDA2.1-mini	51.33	50.17	51.24 (434)	51.48 (833)	51.29 (1566)	51.13	51.35	51.34
SDAR-8B	49.69	49.65	49.45 (405)	49.65 (794)	49.73 (1507)	48.77	49.15	49.25

update set may differ from dense decoding. The updated active rows, bypassed unresolved rows, and previously resolved rows are recombined to reconstruct the full block state for the next denoising step. Query selection is repeated until all positions are resolved. Once the current block is fully decoded, it is appended to the historical context, and Duet proceeds to the next generation block with an updated prefix.

4 Experiments

In this section, we evaluate Duet in terms of generation quality and inference efficiency across multiple models and benchmarks. We then analyze the two empirical observations underlying its design: the stability of prefix sparsity and the early predictability of query positions. Code is publicly available at `https://anonymous.4open.science/r/Duet-45AF`.

4.1 Experimental Setup

Models. We evaluate Duet on two representative BDLMs, LLaDA2.1-mini (Bie et al., 2026) and SDAR-8B-Chat-b32 (Cheng et al., 2025). Because the predictability of decoding positions emerges at different depths across models, we use model-specific query-sparsity configurations. For LLaDA2.1-mini, we set the query retention ratio to $\rho_Q = 0.7$ and the selection layer to $l_Q = 1$, while retaining its original decoding policy. For SDAR-8B-Chat-b32, we set $\rho_Q = 0.5$ and $l_Q = 5$, and use sequential decoding with 16 denoising steps. At the selection layer, Duet applies the same decoding policy used for the final update to predict active query rows.

Benchmarks. We evaluate accuracy on both long- and shorter-context tasks. For long-context evaluation, we use LongBench (Bai et al., 2024), which spans a diverse set of long-document understanding and generation tasks. We additionally evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021b), code generation on HumanEval (Chen et al., 2021), and knowledge-intensive question answering on MMLU (Hendrycks et al., 2021a).

Table 2: Generation quality on short-context benchmarks. We report accuracy (%) on GSM8K, MMLU, and MATH-500, and pass@1 (%) on HumanEval. For LoSA, parentheses give the average number of selected KV tokens per task. Higher is better.

Model	Method	Budget	GSM8K	MMLU	HumanEval	MATH-500
LLaDA2.1-mini	Dense	–	85.82	**67.79**	78.66	91.80
	LoSA	256	84.61 (216)	66.99 (873)	70.73 (417)	88.80 (512)
	FOCUS	–	**86.05**	66.76	79.88	**92.60**
	Duet (Ours)	256	84.31	66.90	**80.49**	87.60
SDAR-8B	Dense	–	**88.25**	74.52	61.59	63.60
	LoSA	256	87.11 (188)	**74.55** (458)	54.88 (332)	58.40 (294)
	FOCUS	–	86.88	74.52	**64.63**	**65.80**
	Duet (Ours)	256	87.64	74.53	62.20	63.60

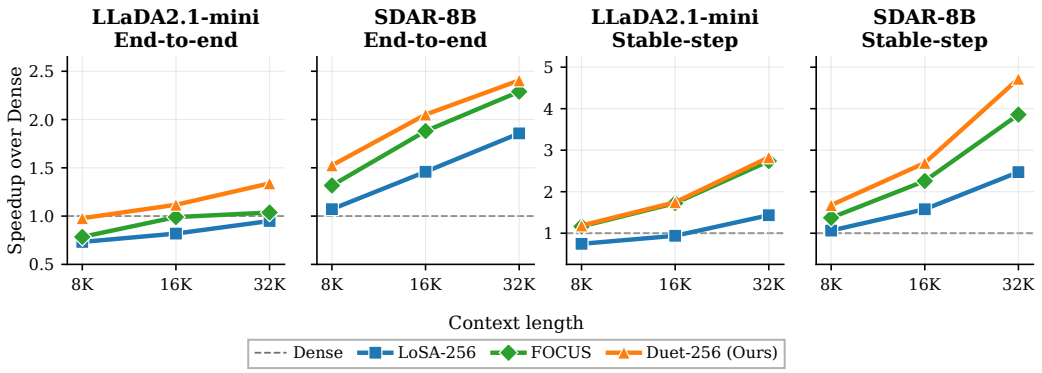


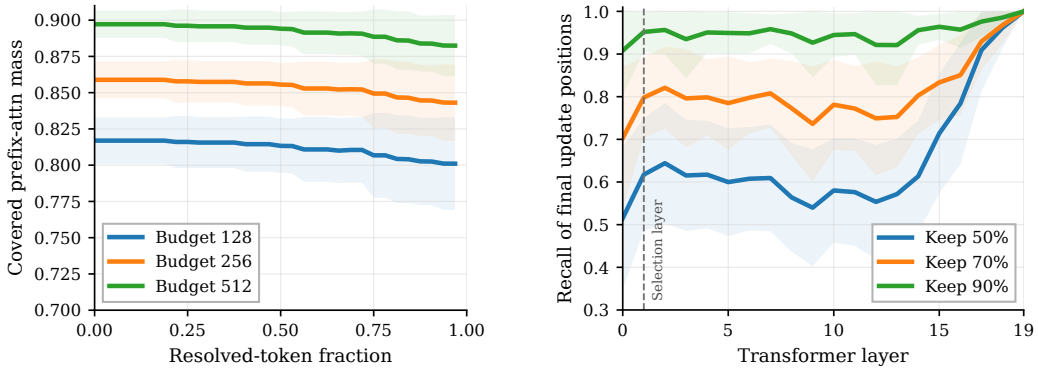
Figure 4: **Inference speedup across context lengths.** End-to-end and stable-step speedups over dense inference on LLaDA2.1-mini and SDAR-8B at 8K, 16K, and 32K context lengths. Stable-step latency averages all denoising steps except the first step, highlighting the execution efficiency of sparse decoding. Duet achieves increasingly larger speedups as the context grows and consistently provides the strongest stable-step acceleration among the compared methods.

Baselines and implementation. We compare Duet against dense inference and two sparse-inference baselines, FOCUS (Liang et al., 2026) and LoSA (Xi et al., 2026). FOCUS primarily sparsifies computation within the current generation block, whereas LoSA focuses on sparsifying attention over the historical prefix. Both baselines use their default configurations. To ensure a fair systems comparison, we implement all methods within the same inference framework and preserve the algorithmic behavior described in the original papers. We further optimize the core sparse operators with Triton kernels and use the same runtime stack across methods. Since the original FOCUS implementation is optimized for shorter-context, larger-batch settings, we adapt its execution path to long-context decoding while keeping its selection and sparse-computation logic unchanged.

Efficiency evaluation. We evaluate inference efficiency on 80 NarrativeQA samples at 8K, 16K, and 32K context lengths with a fixed generation length of 256 tokens on an NVIDIA H800 GPU. We report end-to-end and stable per-step speedups in the main text, with detailed results provided in the appendix; stable per-step latency excludes the first denoising step to remove initialization overhead.

4.2 Main Results

Generation quality. Figure 3 shows the LongBench quality–throughput trade-off, Table 1 reports the corresponding Macro-F1 scores, and Table 2 reports short-context quality. At nominal budget 512, Duet matches dense Macro-F1 on LLaDA2.1-mini (51.35 vs. 51.33) with a 1.33× end-to-end speedup. On SDAR-8B, it reaches 49.15 versus 49.69 with a 2.42× speedup. LoSA also preserves quality, but at nominal budget 256 its measured prefix KV use averages 434 and 405 tokens on the



(a) Persistent prefix-attention concentration.

(b) Early predictability of query positions.

Figure 5: **Two sparsity opportunities in BDLM inference.** (a) A compact prefix subset selected at the beginning of a generation block continues to cover most of the prefix-attention mass throughout denoising. (b) Intermediate representations can identify, with high recall, the positions selected by the final-layer decoding policy. Results are shown on LLaDA2.1-mini as a representative example.

two models, respectively, exceeding the configured budget. Full task-wise scores and KV counts are in Appendix Table 4.

Duet also maintains comparable quality on shorter-context benchmarks. Across GSM8K, MMLU, HumanEval, and MATH-500, both LLaDA2.1-mini and SDAR-8B closely track their dense counterparts, with LLaDA2.1-mini additionally outperforming Dense on HumanEval. These results show that Duet preserves generation quality across diverse tasks while substantially reducing inference cost, as evaluated next.

Inference efficiency. As shown in Figure 4 and Table 3, Duet provides increasingly larger end-to-end gains as the context grows. On LLaDA2.1-mini, Duet with a prefix budget of 256 achieves a $1.12\times$ speedup at 16K and a $1.34\times$ speedup at 32K, while FOCUS provides only a $1.04\times$ speedup and LoSA remains slower than dense inference at 32K. The same trend is more pronounced on SDAR-8B: Duet reaches $1.53\times$, $2.05\times$, and $2.41\times$ end-to-end speedups at 8K, 16K, and 32K, respectively. These results show that Duet remains effective as the historical prefix grows, where repeatedly attending to dense historical KV states becomes increasingly expensive.

The stable-step results in Table 3 further isolate the computation saved during sparse decoding. At 32K, Duet reduces the stable-step latency from 58.12 ms to 20.55 ms on LLaDA2.1-mini, corresponding to a $2.83\times$ speedup, and from 99.53 ms to 21.09 ms on SDAR-8B, corresponding to a $4.72\times$ speedup. In comparison, LoSA reaches only $1.43\times$ and $2.47\times$ speedups on the two models at 32K, respectively, indicating that its prefix-selection overhead substantially limits the benefit of sparse attention. FOCUS achieves stable-step speedups closer to Duet—$2.74\times$ on LLaDA2.1-mini and $3.86\times$ on SDAR-8B at 32K—but this advantage translates less effectively to end-to-end generation. FOCUS changes the decoding trajectory and requires more denoising steps to generate the same number of output tokens; we quantify this effect in Table 3. As a result, its end-to-end speedup is lower than what its per-step latency alone would suggest.

4.3 Analysis

We analyze stability of prefix sparsity and early predictability of query positions on LLaDA2.1-mini.

Stability of prefix sparsity. We examine whether a compact prefix subset selected at the beginning of a generation block remains representative throughout the subsequent denoising process. Take LLaDA2.1-mini as the For each block, we keep the selected historical prefix subset fixed and measure the fraction of prefix-attention mass covered by this subset as the block is progressively resolved. Figure 5a shows that even a prefix budget of 256 consistently captures around 85% of the historical attention mass, while increasing the budget to 512 raises the coverage to nearly 90%.

More importantly, the covered attention mass remains stable throughout block decoding: across all evaluated budgets, the variation over the denoising trajectory is within approximately 2.5 percentage points. These results indicate that the current-block queries maintain a largely stable focus on a compact portion of the historical prefix, supporting our design of selecting prefix KV states once before decoding each block and reusing the resulting cache throughout its denoising steps.

Early predictability of query positions. We analyze whether the positions selected by the final decoding update can already be predicted from early Transformer representations. At each denoising step, we apply the model's original decoding policy to hidden states from different layers and measure the recall of the positions selected by the final-layer decoding result. Figure 5b shows that the final update positions become highly predictable at surprisingly early layers. At layer 1, retaining 70% of the masked positions already recovers approximately 80% of the final update positions, while a 90% retention ratio reaches about 95% recall. The recall remains high through most of the network and approaches 100% near the final layers. These results demonstrate that the locations of the final decoding updates can be identified well before the full Transformer depth is evaluated, providing direct support for selecting active query rows at an early layer and avoiding unnecessary deep computation on the remaining positions.

5 Related Work

DLMs. DLMs generate text by iteratively denoising discrete token sequences. Prior work explored discrete-state diffusion, continuous diffusion in embedding space, and diffusion with pretrained masked language models (Austin et al., 2021; Li et al., 2022; He et al., 2023). Masked discrete diffusion provided a scalable formulation, later demonstrated at scale by LLaDA and Dream (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). Fully parallel masked DLMs predict multiple positions per step, enabling bidirectional context modeling and parallel generation. However, repeatedly processing the full sequence incurs redundant computation on largely unchanged tokens, while bidirectional attention prevents direct reuse of the append-only KV cache used by autoregressive models (Wu et al., 2026b; Liu et al., 2026).

From DLMs to BDLMs. Fast-dLLM demonstrates the potential of block-level organization for DLM acceleration through block-wise approximate KV caching and confidence-aware parallel decoding (Wu et al., 2026b). Subsequent methods explore adaptive cache reuse at different granularities, including segment-level updates, delayed updates, and token-level state selection (Liu et al., 2026; Ma et al., 2025; Jiang et al., 2026). BDLMs make this structure part of the generation paradigm: Block Diffusion generates blocks autoregressively while refining tokens through diffusion within each block (Arriola et al., 2025); SDAR adapts pretrained autoregressive models to block-wise diffusion (Cheng et al., 2025); and LLaDA2.0 and LLaDA2.1 extend the scale and token-editing capabilities of this model family (Bie et al., 2025; 2026). Fast-dLLM v2 further combines block-diffusion model design with hierarchical caching and parallel decoding (Wu et al., 2026a).

Inference optimization for existing BDLMs. Recent work optimizes existing BDLMs, including SDAR and LLaDA2.x, by reducing current-block computation or historical-prefix attention (Liang et al., 2026; Xi et al., 2026). FOCUS estimates token-wise decodability to skip positions unlikely to be updated, but under confidence-based decoding this may increase the number of denoising steps, limiting end-to-end gains (Liang et al., 2026). LoSA sparsifies long-context prefix attention by recomputing attention for active queries and reusing outputs for stable ones, while repeated query-dependent KV selection still incurs nontrivial overhead (Xi et al., 2026). As a result, the reduction in attention computation does not always translate into substantial end-to-end speedup.

6 Conclusion

We presented Duet, a training-free dual-sparse framework for BDLM inference that combines block-wise prefix KV selection with early query selection to avoid repeated historical attention and deep processing of inactive positions. The selected prefix is reused across denoising steps, while only predicted active queries traverse the remaining Transformer layers. Duet maintains competitive accuracy across long-context and standard benchmarks, achieving 1.34×/2.83× end-to-end/per-step speedups on LLaDA2.1-mini and 2.41×/4.72× on SDAR-8B-Chat-b32 at 32K. Stable prefix relevance within blocks and early predictability of update positions support both sparsity mechanisms.z

AI USE STATEMENT

The core idea and algorithmic design of Duet were developed by the authors. We used generative AI tools primarily to modify and optimize portions of the implementation and evaluation code, monitor experimental runs and inspect logs, and edit the manuscript for clarity and presentation. The tools also assisted in reviewing the presentation of mathematical claims and interpreting benchmark results; they did not originate the core method. The authors reviewed the AI-assisted work and take full responsibility for the final code, results, and claims in this paper.

REPRODUCIBILITY STATEMENT

We release the anonymized source code and testing procedures for Duet at the repository linked in Section 4. The release includes the implementation and evaluation scripts for the short- and long-context benchmarks and inference-speed measurements. Section 3 specifies the inference algorithm, while Section 4 describes the models, decoding settings, baselines, benchmarks, and evaluation protocol. The appendix provides task-wise LongBench results together with detailed latency and decoding-step measurements. Together, these materials support reproduction of the reported quality and efficiency comparisons.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/arriola2025iclr-block/.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

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.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Junlin Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. Llada2.1: Speeding up text diffusion via token editing, 2026. URL https://arxiv.org/abs/2602.08676.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian,

Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

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

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

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving generative masked language models with diffusion models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4521–4534, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.248. URL https://aclanthology.org/2023.acl-long.248/.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, CHONGHAN LIU, and xu yang. dˆ2cache: Accelerating diffusion-based llms via dual adaptive caching. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 137511–137531, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/de4f2e3b8263aff39f0dd7571a8dc4b1-Paper-Conference.pdf.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Kaihua Liang, Xin Tan, An Zhong, Hong Xu, and Marco Canini. FOCUS: DLLMs know how to tame their compute bound. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=40fUEdwvH3.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-cache: Accelerating diffusion large language models with adaptive caching, 2026. URL https://openreview.net/forum?id=DriG3hgh42.

Lingkun Long, Yushi Huang, Shihao Bai, Ruihao Gong, Jun Zhang, Ao Zhou, and Jianlei Yang. Focus-dLLM: Accelerating long-context diffusion LLM inference via confidence-guided context focusing. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12146–12160, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.556. URL https://aclanthology.org/2026.acl-long.556/.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 149009–149033. Curran Associates, Inc., 2025. doi: 10.52202/085713-4984. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/db0ee27cb50dd9087b133f6e7d28a90e-Paper-Conference.pdf.

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. In *Advances in Neural Information Processing Systems*, 2025. URL https://mlanthology.org/neurips/2025/nie2025neurips-large/.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=t5uLZSRjhF.

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. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 128353–128370, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d0865cbe51d35ec322f9af9db7806fc7-Paper-Conference.pdf.

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. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Haocheng Xi, Harman Singh, Yuezhou Hu, Coleman Richard Charles Hooper, Rishabh Tiwari, Aditya Tomar, Minjae Lee, Wonjun Kang, Michael W Mahoney, Chenfeng Xu, et al. Losa: Locality aware sparse attention in diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SMhHoUepeR.

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.

A Efficiency Evaluations

We evaluate inference efficiency on 80 NarrativeQA samples at 8K, 16K, and 32K context lengths, generating 256 tokens per sample on an NVIDIA H800 GPU. Table 3 reports three complementary measurements: mean end-to-end (E2E) generation latency, stable-step latency excluding the first denoising step, and the mean number of decoding steps per sample. End-to-end latency includes prefill and generation overheads, so its speedup need not equal the stable-step speedup.

Table 3: **Detailed inference efficiency results.** E2E is end-to-end latency in seconds; Step is stable-step latency in milliseconds; Steps is the mean number of decoding steps per sample. Lower latency is better.

		8K context			16K context			32K context		
Method	Budget	E2E (s)	Step (ms)	Steps	E2E (s)	Step (ms)	Steps	E2E (s)	Step (ms)	Steps
LLaDA2.1-mini										
Dense	–	0.841	24.03	16.06	1.447	35.47	16.00	3.486	58.12	25.05
FOCUS	–	1.072	20.65	24.05	1.462	20.58	25.15	3.361	21.23	64.53
LoSA	256	1.148	32.25	16.65	1.767	37.78	16.18	3.672	40.54	27.81
	512	1.171	33.12	16.59	1.774	38.00	16.18	3.676	40.89	28.06
	1024	1.188	34.25	16.51	1.789	39.09	16.10	3.712	40.71	28.81
Duet(Ours)	256	0.860	20.26	15.60	1.295	20.32	15.88	2.604	20.55	23.29
	512	0.840	19.54	15.58	1.292	20.22	15.94	2.620	20.57	24.08
	1024	0.871	20.24	15.48	1.303	20.36	15.70	2.633	20.46	24.16
SDAR-8B										
Dense	–	5.350	35.42	128	9.279	56.88	128	18.416	99.53	128
FOCUS	–	4.062	25.80	128	4.934	25.18	128	8.046	25.80	128
LoSA	256	4.992	33.30	128	6.360	36.05	128	9.920	40.26	128
	512	5.252	35.37	128	6.616	38.33	128	10.227	42.46	128
	1024	5.684	38.76	128	7.050	41.77	128	10.798	46.64	128
Duet(Ours)	256	3.508	21.12	128	4.523	21.12	128	7.656	21.09	128
	512	3.524	21.21	128	4.563	21.46	128	7.624	20.84	128
	1024	3.526	21.05	128	4.588	20.96	128	8.100	20.58	128

At 32K context length, Duet with a prefix budget of 256 reduces LLaDA2.1-mini's stable-step latency from 58.12 to 20.55 ms and its end-to-end latency from 3.486 to 2.604 s. Its mean step count decreases from 25.05 to 23.29. FOCUS reaches a similar stable-step latency of 21.23 ms but takes 64.53 steps on average, partly explaining its smaller end-to-end gain. SDAR-8B uses its default sequential decoding policy, which fixes the number of decoding steps for a given generation length. Thus, all methods take 128 steps in this experiment. On SDAR-8B, Duet reduces stable-step latency from 99.53 to 21.09 ms and end-to-end latency from 18.416 to 7.656 s at 32K.

B DETAILED LONGBENCH RESULTS

Table 4 reports the full task-wise F1 results underlying the LongBench Macro-F1 values in Figure 3. Each task contributes equally to Macro-F1. LoSA's parenthesized values are measured averages of attended prefix KV tokens, not its configured budget; the parenthesized value in the Macro-F1 column is the mean of the five task-level KV counts.

Table 4: Task-wise results on LongBench. We report F1 (%) on five long-context tasks and Macro-F1, the unweighted average across tasks. For LoSA, parentheses give the measured average number of attended prefix KV tokens per task; the Macro-F1 column gives the mean of these five KV counts. Higher F1 is better.

Method	Budget	HotpotQA	TriviaQA	NarrativeQA	Qasper	MultiFieldQA	Macro-F1
LLaDA2.1-mini							
Dense	–	54.60	85.91	24.71	42.15	49.26	51.33
FOCUS	–	55.23	84.74	21.96	41.89	47.03	50.17
LoSA	256	54.74 (430)	85.51 (426)	23.87 (447)	43.20 (435)	48.90 (433)	51.24 (434)
	512	56.13 (833)	85.13 (820)	23.23 (873)	43.32 (813)	49.60 (825)	51.48 (833)
	1024	55.60 (1594)	85.86 (1549)	23.02 (1677)	42.80 (1483)	49.15 (1529)	51.29 (1566)
Duet (Ours)	256	54.02	84.40	24.69	43.27	49.29	51.13
	512	55.36	85.11	24.98	42.29	49.01	51.35
	1024	55.13	85.45	23.97	42.90	49.25	51.34
SDAR-8B							
Dense	–	52.11	83.15	25.03	34.85	53.29	49.69
FOCUS	–	51.55	83.65	24.78	34.93	53.34	49.65
LoSA	256	51.96 (412)	83.12 (405)	25.25 (398)	34.38 (403)	52.56 (406)	49.45 (405)
	512	51.92 (809)	83.12 (784)	25.31 (804)	34.49 (785)	53.42 (786)	49.65 (794)
	1024	51.93 (1546)	83.20 (1490)	25.37 (1571)	34.47 (1461)	53.66 (1467)	49.73 (1507)
Duet (Ours)	256	50.34	83.37	24.91	34.37	50.85	48.77
	512	51.11	83.12	24.76	34.74	52.01	49.15
	1024	50.87	82.87	25.87	34.78	51.86	49.25