Inverse Speculation: Structural Anchoring from Diffusion Language Models for Edge-Scale Generation

Ben Wade
Independent Researcher
Weymouth, Massachusetts, USA
bestbenwade@gmail.com

Abstract

Speculative decoding accelerates large language model inference by using a small model to draft tokens that a large model verifies. We invert this paradigm: rather than using a small model to approximate a large one, we use a Diffusion Language Model (DLM) to structurally elevate a small one. We exploit the permanence property of absorbing-state masked diffusion—tokens committed during denoising are irrevocable—to extract anchor skeletons from as few as 10% of denoising steps. A 0.5B-parameter autoregressive model fills gaps between these anchors via forced decoding, achieving 0.82–0.93 F1 against the DLM's full output (N=190 across four benchmarks). Ablation experiments (N=50) demonstrate that token identity, not position, drives anchor effectiveness: correct tokens at random positions yield 0.90 F1, while random tokens at correct positions yield 0.002. Gap-only decomposition shows the gap-filler more than doubles its unconstrained performance at non-anchor positions (0.475 vs. 0.231 word F1), with no correlation between anchor density and gap quality ($\rho = -0.037$, $p = 0.876$), confirming genuine conditional modeling rather than density inflation. We further show that a 0.5B gap-filler matches a 1.5B gap-filler when anchors are provided ($\Delta = -0.009$), suggesting the DLM provides sufficient semantic structure. These findings establish an inverse speculation framework for cloud/edge deployment where a DLM transmits a compressed anchor-template payload—43% smaller than gzip-compressed full text—to enable high-fidelity reconstruction of DLM output on sub-billion-parameter edge devices. Component profiling on A100 hardware confirms a 2.3× sequential pipeline speedup, with cloud compute reduced by 89.7%.

1 Introduction

Speculative decoding has established a dominant paradigm for inference acceleration: a small, fast model drafts tokens that a large model verifies (Leviathan et al., 2023; Li et al., 2024). This architecture assumes the small model approximates the large model's distribution well enough to be useful, and breaks down on hard reasoning tasks where small drafters hallucinate confidently. We invert this paradigm entirely. Rather than using a small model to speed up a large one, we use a large Diffusion Language Model to elevate a small one.

We exploit a mathematical property of absorbing-state masked diffusion—that token commitments during denoising are permanent and structurally load-bearing (Austin et al., 2021; Saini et al., 2026)—to extract a sparse skeleton of anchor tokens from the earliest stages of generation. A sub-billion-parameter autoregressive model then fills the gaps between these anchors via forced decoding, reproducing the DLM's output while running entirely on edge hardware. We demonstrate that a shallow pass through a Diffusion Language Model—10% of its denoising trajectory—extracts enough structural information for a sub-billion-parameter model to reconstruct high-fidelity DLM output at a fraction of the original compute cost. The depth is in the architecture, not the inference.

Our contributions are:

1. We demonstrate that 10–25% of DLM denoising steps produce token anchors sufficient for a 0.5B model to achieve 0.82–0.93 F1 against the DLM's full output (N=190 prompts across four benchmarks).
2. We show via controlled ablation (N=50) that anchor effectiveness is driven by token identity, not position—establishing the DLM as a content architect rather than a positional template.
3. We decompose pipeline F1 into anchor density and gap-filler contribution, showing that anchors condition the AR model to more than double its gap performance (0.475 vs. 0.231 F1) independent of coverage density ($\rho = -0.037$).
4. We find that gap-filler model size has minimal effect when anchors are provided: 0.5B matches 1.5B (Δ=-0.009, N=50), suggesting the DLM provides sufficient semantic structure.
5. We propose an inverse speculation framework for cloud/edge inference where a DLM transmits 100–400 bytes of structural anchors to enable high-fidelity output reconstruction on commodity hardware.

Deployment motivation. Current approaches to LLM inference on compute-constrained devices—smartphones, embedded systems, IoT endpoints—face a trilemma: run a large model locally (impossible at 7B+ parameters), stream full generated text from the cloud (latency-sensitive and bandwidth-intensive), or run a small local model and accept degraded quality. Inverse speculation offers a fourth option: the cloud DLM performs 10% of its denoising trajectory (552 ms on A100, versus 5383 ms for full generation) and transmits the resulting anchor skeleton as a compressed text template—approximately 119 bytes per response after gzip, 43% smaller than gzip-compressed full text. The local device runs a 0.5B model (which fits comfortably in $\sim$1 GB of RAM) to fill the blank positions. This is particularly relevant for high-latency or low-bandwidth environments (satellite links, rural connectivity, offline-capable devices) where minimizing both cloud compute and payload size matters.

2 Background

2.1 Absorbing-State Masked Diffusion

Masked Diffusion Models (MDMs) define a forward noising process that progressively replaces tokens with a `[MASK]` symbol, and a learned reverse process that reconstructs the sequence (Austin et al., 2021; Sahoo et al., 2024). A key structural property is absorbing masking: once a token becomes `[MASK]`, it remains `[MASK]` for all later forward times. In the reverse (generative) direction, this creates a permanence property: if a position is unmasked at step t, it remains fixed for all subsequent steps $t{-}1, t{-}2, \ldots$ (Saini et al., 2026, Eq. 4). We empirically verify this on Dream-7B (Ye et al., 2025): across 310 prompts and 128 denoising steps, no token reversions were observed. While this permanence is expected from the absorbing-state formulation, we note it is an empirical observation on this architecture, not a theoretical guarantee across all MDM implementations.

This permanence means that tokens committed early in denoising carry outsized structural importance—they form the skeleton around which all subsequent tokens crystallize. The DLM's bidirectional attention ensures these early commits are globally coherent, not just locally plausible.

2.2 Speculative Decoding and Its Inverse

Standard speculative decoding (Leviathan et al., 2023) uses a small draft model to generate candidate tokens that a large target model verifies in parallel. Extensions such as EAGLE-3 train specialized draft heads for improved acceptance rates. In all cases, the direction is fixed: small model helps large model go faster. Critically, speculative decoding has a failure mode on hard reasoning: when the small drafter hallucinates confidently, the verifier rejects those tokens, wasting compute and falling back to sequential generation. On long-horizon reasoning tasks, acceptance rates can drop below 50%, negating the parallelism advantage.

We reverse this direction entirely. A large DLM provides structural waypoints that a small AR model fleshes out. This is not verification—the small model never checks the DLM's work. Instead, the DLM's committed tokens are injected as hard constraints into the AR model's autoregressive

Table 1: Comparison of inference paradigms.

Method	Direction	Mechanism
Speculative decoding	Small $\rightarrow$ Large	Small drafts, large verifies
EAGLE-3	Small $\rightarrow$ Large	Trained drafter, parallel verify
Inverse Speculation	**Large $\rightarrow$ Small**	**Large anchors, small fills gaps**

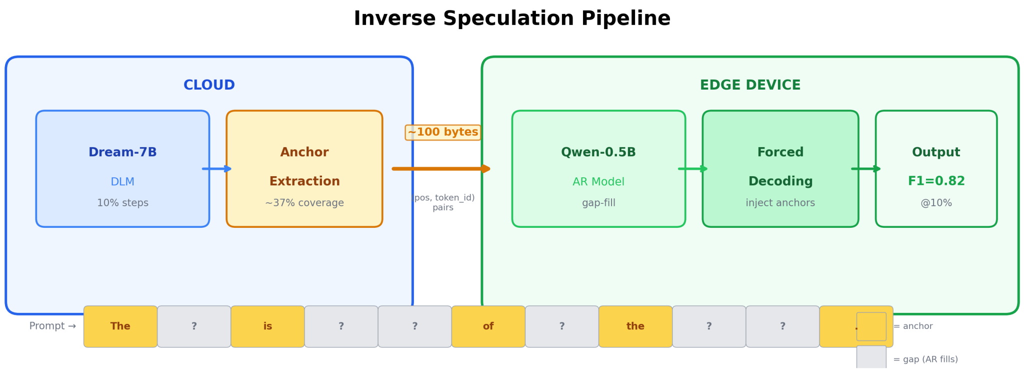

Figure 1: Inverse speculation pipeline. The cloud DLM runs a partial denoising pass (10% of steps), extracting committed anchor tokens. These are transmitted as $\sim$100 bytes of (position, token_id) pairs to the edge device, where a 0.5B AR model fills gaps via forced decoding.

generation via forced decoding, steering it toward an output it could never produce independently. There is no rejection step, no wasted compute, and the forced anchor structure substantially reduces the risk of hallucination cascades where a confident-but-wrong draft leads generation down an irrecoverable path. (The gap-filler may still produce inconsistent connective tissue between anchors, but cannot derail the structural trajectory.)

3 Method

3.1 Anchor Extraction

Given a prompt, we run Dream-7B for T_{partial} steps (10–25% of the full $T{=}128$ schedule) using the standard entropy-based unmasking schedule. At step T_{partial}, we record all positions where a non-`[MASK]` token has been committed. These (position, token_id) pairs constitute the anchor set $\mathcal{A}$.

For a typical response of $\sim$100 tokens, running 13 steps (10%) yields $\sim$37% coverage (37 anchors), while 32 steps (25%) yields $\sim$86% coverage. The anchor set $\mathcal{A}$ requires only $|\mathcal{A}| \times (\text{position\_index} + \text{token\_id}) \approx$ 100–400 bytes for transmission, depending on response length and anchor fraction.

3.2 Forced Decoding

The anchor set $\mathcal{A}$ is injected into a standard autoregressive generation loop. At each generation position p, if $p \in \mathcal{A}$, the anchor token is emitted regardless of the AR model's own prediction. Otherwise, the AR model generates freely via greedy decoding. The AR model's context window sees the full sequence including previously forced anchor tokens, so its predictions for gap positions are conditioned on the DLM's structural skeleton.

No modification to the AR model's weights, architecture, or tokenizer is required. The only requirement is a shared vocabulary between the DLM and AR model (in our case, Dream-7B and Qwen-0.5B share compatible tokenizer vocabularies after mapping).

The pipeline is not equivalent to early-stopping the DLM. Running Dream-7B for 13 of 128 steps and decoding directly yields partially masked, incoherent sequences—not usable text. The AR gap-filler is required to transform these sparse structural commits into fluent output—the anchors serve as structural waypoints that provide a semantic floor for the tiny realizer. Formally, let C_{DLM} denote the cost of one DLM forward pass and C_{AR} the cost of one AR forward pass. Full DLM inference costs $T \cdot C_{\text{DLM}} = 128 \cdot C_{\text{DLM}}$. The pipeline costs:

$$C_{\text{pipeline}} = T_{\text{partial}} \cdot C_{\text{DLM}} + L \cdot C_{\text{AR}} \tag{1}$$

where L is the sequence length. (The AR model performs a forward pass at every position to maintain its context state, even where the anchor token overrides its prediction.) Since $C_{\text{AR}} \approx \frac{1}{14} \cdot C_{\text{DLM}}$ and $T_{\text{partial}} = 13$, the pipeline requires approximately 10–15% of the full DLM's compute budget. The system performs conditional compression of inference-time computation: the DLM's early structural decisions are preserved and transmitted, while the expensive realization work is offloaded to a model that is orders of magnitude cheaper to run.

Evaluation metrics. We report two complementary metrics: (1) word-level F1 against the DLM's full deterministic output (a fixed reference generated once per prompt), measuring output reproduction fidelity, and (2) ground-truth benchmark accuracy, measuring task correctness. F1 is computed as bag-of-words overlap (precision $\times$ recall harmonic mean over whitespace-tokenized words). We note that bag-of-words F1 does not penalize word reordering; however, ground-truth accuracy—which requires exact answer-string matching and is therefore order-sensitive—confirms the same performance patterns (Section 4.2), providing an independent check that bag-of-words F1 is not inflating the signal.

3.3 End-Marker Optimization

When anchor coverage exceeds 40%, the last committed anchor position provides a reliable end-of-generation marker. We truncate AR generation at this position, preventing the common failure mode where small models generate excessive trailing content. This improves F1 by an average of $+0.22$ in high-coverage conditions. Note that F1 is computed against the DLM's full untruncated output; the improvement reflects elimination of spurious trailing tokens from the AR model, not metric shaping via reference truncation.

4 Experiments

4.1 Setup

We evaluate on 310 prompts spanning four benchmarks: MMLU (100: 50 easy, 50 hard), ARC (100: 50 Easy, 50 Challenge), GSM8K (60: 20 easy, 20 medium, 20 hard), and HumanEval (50 mixed). All experiments use Dream-7B (bf16) as the DLM, Qwen2.5-0.5B-Instruct and Qwen2.5-1.5B-Instruct as gap-fillers, running on a single A100-SXM4-80GB. The DLM runs for 128 steps with entropy-based unmasking, temperature 0.2, top_p 0.95. The N=190 evaluation set (Section 4.2) comprises the hard/medium/mixed prompts from the 310; the N=50 ablation and gap-only subsets (Sections 4.3–4.5) are drawn from this 190. No prompt appears in a test set without also appearing in the parent set; all subsets are proper subsets with no external contamination.

4.2 Anchor Pipeline Performance (N=190)

We evaluate forced decoding on the 190 hard/medium/mixed prompts (a subset of the 310 Phase A prompts) at three anchor fractions:

F1 is computed against a fixed deterministic Dream-7B output (generated once per prompt with temperature 0.2, greedy argmax). This reference is not ground truth—it is the DLM's own output, used to measure pipeline reproduction fidelity. Ground-truth accuracy is evaluated independently.

At 25% denoising, the pipeline preserves 87 of 92 correct Dream answers (94.6%; Wilson 95% CI: 87.9%–98.0%). We note that at 25% anchor fraction, mean anchor coverage is $\sim$86% of response tokens; the high F1 partially reflects this density. Section 4.5 decomposes this via gap-only F1 analysis.

Table 2: Anchor pipeline performance on N=190 hard/medium/mixed prompts.

Condition	**Mean F1**	**Std**	**GT Accuracy**
0.5B alone	0.162	0.137	28.4% (54/190)
0.5B + anchors @10%	0.820	0.220	—
0.5B + anchors @15%	0.888	0.195	—
0.5B + anchors @25%	0.934	0.164	45.8% (87/190)
Dream-7B full (reference)	1.000	—	48.4% (92/190)

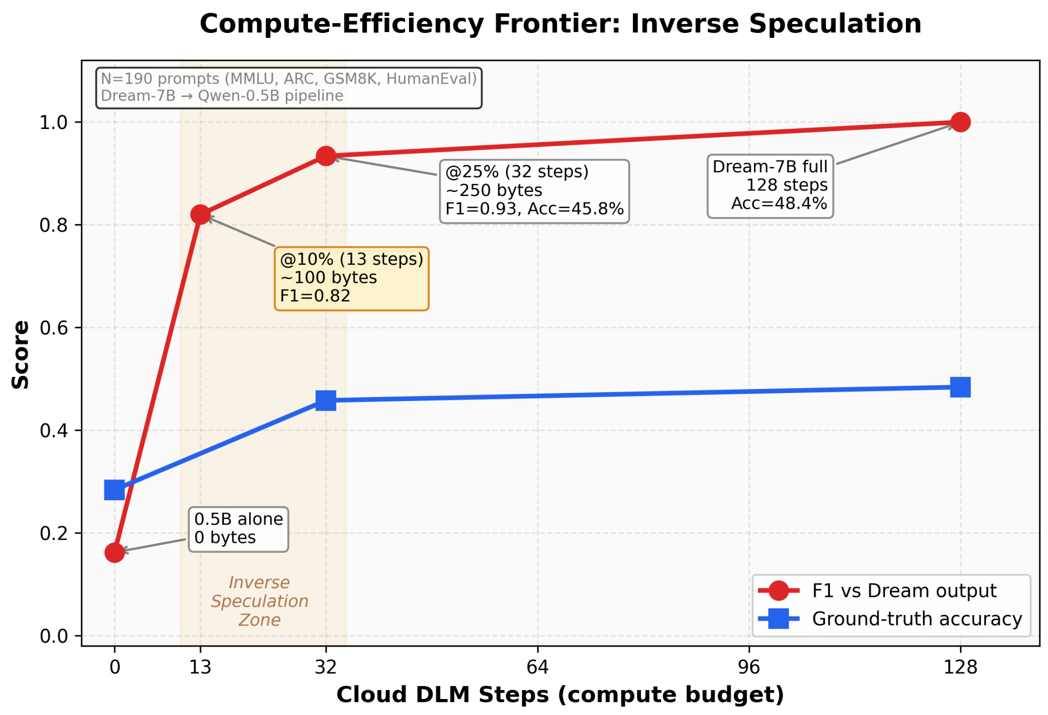

Figure 2: Compute-efficiency frontier. F1 against Dream output (red) and ground-truth accuracy (blue) as a function of cloud DLM steps. At 13 steps (10%), the pipeline achieves 0.82 F1 while transmitting only ∼100 bytes. The shaded region marks the inverse speculation operating zone.

4.3 Gap-Filler Size Is Irrelevant (N=50)

On a 50-prompt subset (drawn from the 190 hard/medium prompts), we compare 0.5B and 1.5B as gap-fillers:

Without anchors, the 1.5B model is substantially better. With anchors, the gap vanishes—the 0.5B model is marginally better at both fractions. This demonstrates that the DLM provides sufficient semantic structure; the gap-filler's own capability contributes minimally once structural anchors are present. (We avoid the stronger claim that the DLM provides "all" semantic content, as the high anchor density at 25% reduces gap-filler entropy sufficiently that model size effects may be masked.)

4.4 Random Anchor Ablation (N=50)

The critical control experiment. We test whether Dream's specific committed tokens matter, or whether any structural constraint would suffice:

Token identity is everything; position is secondary. Random tokens at correct positions produce output worse than no anchors at all (0.002 vs. 0.125 F1)—wrong tokens actively derail generation. Correct tokens at random positions still achieve 0.90 F1, demonstrating the AR model's robustness

Table 3: Gap-filler model size comparison (N=50).

Condition	0.5B F1	1.5B F1	Δ
Alone (no anchors)	0.068 ± 0.091	0.149 ± 0.142	$+0.081$
+ anchors @10%	0.841 ± 0.215	0.832 ± 0.227	-0.009
+ anchors @25%	0.953 ± 0.118	0.944 ± 0.131	-0.009

Table 4: Ablation: token identity vs. position (N=50). Statistical tests: Wilcoxon signed-rank; Cohen's d computed on per-prompt F1 distributions.

Condition	F1 @10%	F1 @25%
0.5B alone	0.125	0.125
REAL anchors (Dream tokens at Dream positions)	0.840	0.921
RANDOM-TOKEN (Dream positions, random tokens)	0.002	0.002
RANDOM-POSITION (Dream tokens, random positions)	0.774	0.900
Real vs Random-Token: $p = 1.21 \times 10^{-10}$, $d = 6.41$		
Real vs Random-Position: $p = 1.52 \times 10^{-2}$, $d = 0.10$		

to positional perturbation. We note that bag-of-words F1 does not penalize reordering, which may inflate the random-position score; nevertheless, the pattern is unambiguous: the DLM functions as a content architect that determines *what* to say, while the AR model handles arrangement.

4.5 Gap-Only F1 Decomposition (N=50)

The critical question: does the gap-filler contribute beyond anchor density? We tracked every generated token as either *forced* (anchor position) or *free* (gap position) and computed F1 exclusively on gap tokens. Of the 50 prompts, 30 have 100% anchor coverage at @0.1 (short responses where every token commits), leaving 20 partial-coverage prompts where the gap-filler must generate non-trivial content.

At 10% denoising, the gap-filler more than doubles its unconstrained performance on non-anchor tokens (0.475 vs. 0.231, $\Delta = +0.244$). Crucially, there is no correlation between anchor coverage and gap-only F1 (Spearman $\rho = -0.037$, $p = 0.876$), demonstrating that the improvement reflects genuine conditional modeling—the anchors constrain the semantic space, steering the AR model toward topically aligned content—rather than density inflation. At 25%, gap-only evaluation becomes unreliable due to sparse gap counts (mean 6.5 tokens across only 13 partial-coverage prompts).

The mechanism is constrained semantic space, not token passthrough: anchors do not merely inflate F1 by occupying positions; they condition the AR model to produce better gap content than it generates independently.

4.6 Commit Rate and Task Difficulty

As a secondary finding, we observe that the DLM's commit rate correlates with task difficulty. Across 260 labeled prompts, harder tasks show later S_{50} (the step at which 50% of tokens are committed): Spearman $\rho = 0.182$, $p = 0.003$. However, response length dominates ($\rho = 0.929$ for S_{50} vs. length), and the partial correlation controlling for length is modest ($\rho = 0.117$). Within GSM8K, solution step count is a stronger continuous predictor ($\rho = 0.315$, $p = 0.014$). This difficulty signal, while real, is primarily a supporting observation motivating adaptive compute allocation rather than a primary contribution.

4.7 Negative Results

Quantization destroys anchor quality. Dream-7B at 4-bit (GGUF Q4_K_M) produces $\sim$33% token overlap with bf16 and exhibits repetitive loops. DLMs are more quantization-sensitive than AR models because each denoising step compounds errors across all positions simultaneously.

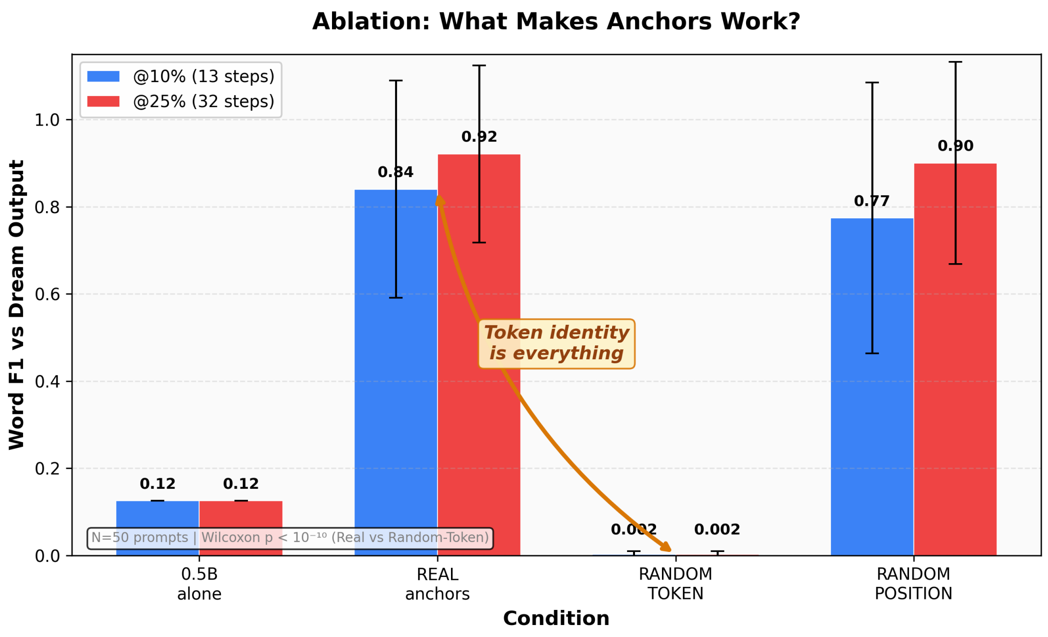

Figure 3: Ablation results. Token identity drives anchor effectiveness: real anchors achieve 0.84–0.92 F1, while random tokens at correct positions collapse to 0.002—worse than no anchors at all. Correct tokens at random positions retain 0.77–0.90 F1. Wilcoxon signed-rank $p < 10^{-10}$ for Real vs. Random-Token.

Table 5: Gap-only F1 decomposition. Partial-coverage prompts only (N=20, mean 45% coverage, mean 44 gap tokens).

Metric	**@0.1**	**@0.25** (N=13, 6.5 gap tok.)
Gap-only word F1	0.475 ± 0.392	0.358 ± 0.424
Gap token accuracy	0.340 ± 0.363	0.146 ± 0.255
0.5B alone (same prompts)	0.231 ± 0.163	0.221 ± 0.159
Full F1 (with anchors)	0.599 ± 0.241	0.697 ± 0.300

Template prompting fails. Sub-billion AR models cannot follow instructions to “fill in blanks” in a template. Forced decoding bypasses instruction-following entirely, treating the AR model as a conditional text generator rather than an instruction follower.

4.8 Supplementary: Compute Profiling

To validate the compute efficiency claims, we profiled both pipeline components on an NVIDIA A100-SXM4-80GB with `torch.cuda.synchronize()` timing across 20 prompts per configuration. Both models were loaded in bf16 (total VRAM: 15.2 GB).

The DLM per-step cost (42.0 ms) is only 2.3× the AR per-token cost (18.3 ms), despite the 14× parameter ratio, reflecting the DLM’s parallel token updates per step. In a sequential pipeline, 13 DLM steps plus 100 AR tokens costs 2371 ms versus 5383 ms for full Dream generation—a 2.3× speedup. In an asynchronous cloud/edge deployment where the DLM and AR model run on separate hardware, the wall-clock time is $\max(552, 1825) = 1825$ ms, yielding a 2.9× speedup with the cloud completing its work in 552 ms (89.7% compute reduction).

Anchor payload as masked-text compression. A natural encoding for the anchor skeleton is a text template: the DLM’s partial output with a single-byte placeholder (e.g., a null byte) at each uncommitted position, compressed with gzip. Because the placeholder is a single repeated byte pat-

Table 6: Component timing on A100-80GB (N=20 prompts, 256 max tokens).

Component	Time (ms)	Per unit
Dream-7B $\times$ 1 step	42.0	42.0 ms/step
Dream-7B $\times$ 13 steps	552	42.4 ms/step
Dream-7B $\times$ 128 steps	5383	42.1 ms/step
Qwen-0.5B $\times$ 1 token	18.3	18.3 ms/tok
Qwen-0.5B $\times$ 100 tokens	1825	18.3 ms/tok
Pipeline (13 steps + 100 AR tokens)	**2371**	**2.3$\times$ faster**

tern, gzip compresses gap positions to near-zero cost, while the surviving anchor words are natural language that gzip handles efficiently. Positional information is implicit in the sequence, eliminating the 2-byte position overhead of binary packing. Across our 8-prompt evaluation set at 10% denoising, this format averages **119 bytes**—43.1% smaller than gzip-compressed full text (209 bytes) and 70.4% smaller than raw text (402 bytes). The anchor payload thus achieves genuine compression while simultaneously enabling edge reconstruction: fewer bytes transmitted, with the structural skeleton intact for the 0.5B gap-filler. This encoding also provides natural error resilience—corrupted bytes in the text region can be inferred from surrounding context, unlike opaque binary (position, token_id) pairs.

5 Related Work

Anchored Diffusion Language Model (Rout et al., 2025) is the closest related work. ADLM trains a dedicated anchor network to predict important tokens, then conditions the main denoiser on these predictions. Our approach requires no additional training—we extract natural commits from an unmodified DLM and hand them to a separate, much smaller AR model. ADLM improves the DLM itself; we use the DLM to improve a different model.

Trajectory Lock-in and Load-Bearing Tokens (Saini et al., 2026) formalizes the permanence property we exploit and introduces the Token Influence Score (TIS), proving via gradient analysis that certain positions are "load-bearing pivots" whose early commitment maximally reduces future uncertainty. We treat lock-in as constructive rather than problematic: the tokens TABES identifies as structurally critical via gradients are precisely the anchors our pipeline transmits. Our work is the systems realization of their theoretical insight—we show that these load-bearing tokens, extracted from just 10% of the denoising trajectory, suffice to steer a 14$\times$ smaller model to 82% output fidelity.

Prophet (Li et al., 2025) demonstrates early answer convergence in DLMs—correct answers are internally identified by step 64 of 128 on 97% of GSM8K instances. This validates our S_{50} observations and suggests that Prophet's confidence-gap criterion could optimize the cloud DLM's step budget in our pipeline.

DiffuSpec (Li et al., 2025b) uses Dream-7B as a drafter for large AR targets (Qwen-32B), achieving 3$\times$ wall-clock speedup via one-shot parallel drafting. DiffuSpec goes big$\rightarrow$big: a 7B DLM accelerates a 32B AR model. We go big$\rightarrow$tiny: a 7B DLM elevates a 0.5B AR model to high-fidelity reconstruction, targeting edge deployment rather than datacenter throughput.

Plan-Verify-Fill (PVF) (Jiang et al., 2026) constructs hierarchical skeletons from structural anchors (discourse markers like "Therefore," "Firstly") to accelerate DLM decoding by 65%. PVF is a homogeneous system: anchors steer the same DLM that produced them. Our work demonstrates that structural waypoints transfer across architectures—from a DLM to a separately-trained AR model—proving that the anchor signal is not architecture-specific but a general property of the committed token distribution.

DLM-Scope (Wang et al., 2026) shows that SAE features in DLM early layers predict decoding order and, when inserted, reduce cross-entropy loss. This points toward a future version of our pipeline where hidden-state representations, not just committed tokens, are transmitted as anchors.

6 Toward Factored Inference

Our results establish one link in what we believe is a larger architectural decomposition of language model inference. The key observation is that generation involves at least three distinct computational functions—reasoning, structuring, and realization—and no single architecture is optimal for all three.

Empirical support for decomposition. DLM-Scope (Wang et al., 2026) demonstrates that diffusion language models maintain "semantic superposition" through most of their depth, crystallizing into discrete tokens only in the final $\sim$20% of layers. This means DLMs internally separate planning from realization. Prophet (Li et al., 2025) shows that DLMs converge on correct answers by step 64 of 128—half the denoising trajectory contains nearly all the reasoning content. And our own results show that 10% of the DLM's trajectory provides enough structural information for a $14\times$ smaller model to reproduce 82% of its output, with the gap-filler more than doubling its independent performance under anchor conditioning.

Each finding addresses a different stage of a factored pipeline:

1. **Thinker** (large AR model, early layers): Deep sequential reasoning—chain-of-thought, planning, multi-step logic. AR models excel here because their autoregressive structure naturally supports the kind of step-by-step dependency tracking that reasoning requires. The Thinker need not generate final tokens; it produces a semantic plan, either as hidden-state representations or as a structured prompt for the next stage.
2. **Crystallizer** (DLM, partial denoising): Parallel token arrangement with bidirectional coherence. The DLM takes the Thinker's plan and commits structural tokens—the load-bearing vocabulary that determines *what* the output says. Prophet's early-convergence finding suggests this stage requires far fewer steps than full denoising. Our permanence results confirm that these early commits are irrevocable and globally coherent.
3. **Polisher** (tiny AR model, forced decoding): Gap-filling under structural constraint. This paper proves this stage works: a 0.5B model conditioned on DLM anchors more than doubles its gap performance and achieves high-fidelity output reconstruction. The Polisher runs on edge hardware at negligible cost.

What exists vs. what remains. Stage 3 is empirically validated (this work). Stage 2 is empirically supported by Prophet's early convergence and our anchor extraction results. Stage 1—using a large AR model's representations as input to a DLM—is the missing link. DiffuSpec (Li et al., 2025b) demonstrates that DLM-to-AR handoff is viable for acceleration; the reverse direction (AR-to-DLM) for *reasoning transfer* has not been demonstrated but is architecturally straightforward: the Thinker's hidden states or generated plan-text could serve as the DLM's conditioning input.

The complete pipeline—massive AR early layers $\rightarrow$ DLM partial denoising $\rightarrow$ tiny AR gap-fill—would allow each architecture to operate exclusively in its optimal regime, with total compute potentially far below running any single model end-to-end.

Nearer-term improvements. Combining BoE Steering (Saini et al., 2026) with our anchor extraction should improve anchor quality at very low step fractions. Prophet's confidence-gap criterion could dynamically determine the optimal cloud compute budget per query, and entropy-based adaptive schedules (e.g., Swordsman, BACD) offer a principled stopping rule: halt the DLM when the commit rate gradient plateaus, signaling that structural work is complete and realization can be offloaded to the edge. Additionally, pseudo-trajectory distillation (d3LLM) suggests that 1–3B DLMs could be trained to match the anchor-commit quality of larger models by learning from their denoising trajectories, potentially eliminating the need for a 7B cloud model entirely. Our component profiling (Table 6) establishes a $2.3\times$ sequential speedup and 89.7% cloud compute reduction; an integrated implementation with asynchronous anchor streaming would be required to measure end-to-end latency under realistic network conditions.

Limitations. This work evaluates a single DLM (Dream-7B) and AR family (Qwen); cross-architecture generalization remains to be tested. Extending to AR models with different tokenizers (e.g., Llama-family) would require a token-translation layer that decodes anchors to text and re-encodes in the target vocabulary, introducing subword boundary noise that may degrade anchor precision. We measure reproduction fidelity against the DLM's own output rather than independent reasoning gains. Our timing results (Table 6) profile components independently; integrated end-to-end latency under realistic network conditions remains to be measured. Chain-of-thought preservation poses an additional risk: if the DLM commits final-answer tokens without intermediate reasoning steps, the gap-filler may produce logically incoherent bridges—ensuring anchor sets include "pivotal tokens" (reasoning markers like "Therefore" or "Wait," which DLM-Scope identifies as early-committing structural sinks) is likely critical for CoT-heavy tasks. These are natural next steps.

We have shown that the inverse speculation paradigm—using a large model's structural commitments to guide a small model—is both effective and practical. A shallow pass through a Diffusion Language Model extracts enough structural information for a sub-billion-parameter model to reconstruct high-fidelity output at a fraction of the original compute cost. The depth is in the architecture, not the inference.

References

Austin, J., Johnson, D.D., Ho, J., Tarlow, D., & Van Den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *NeurIPS*, 34.

Leviathan, Y., Kalman, M., & Matias, Y. (2023). Fast inference from transformers via speculative decoding. *ICML 2023*.

Li, Y., Cai, T., Zhang, Y., Chen, D., & Narasimhan, K. (2024). EAGLE: Speculative sampling requires rethinking feature uncertainty. *ICML 2024*.

Li, P., Zhou, Y., Muhtar, D., et al. (2025). Diffusion language models know the answer before decoding. *arXiv:2508.19982*.

Li, G., Fu, Z., Fang, M., Zhao, Q., Tang, M., Yuan, C., & Wang, J. (2025). DiffuSpec: Unlocking diffusion language models for speculative decoding. *arXiv:2510.02358*.

Jiang, H., et al. (2026). Plan, verify and fill: A structured parallel decoding approach for diffusion language models. *arXiv:2601.12247*.

Nie, S., Zhu, F., You, Z., et al. (2025). Large language diffusion models. *arXiv:2502.09992*.

Rout, L., Caramanis, C., & Shakkottai, S. (2025). Anchored diffusion language model. *NeurIPS 2025*.

Sahoo, S., Arriola, M., Schiff, Y., et al. (2024). Simple and effective masked diffusion language models. *NeurIPS*, 37.

Saini, S., Saha, A., Adsumilli, B., Birkbeck, N., Wang, Y., & Bovik, A.C. (2026). TABES: Trajectory-aware backward-on-entropy steering for masked diffusion models. *arXiv:2602.00250*.

Wang, G., Schiff, Y., Sahoo, S.S., & Kuleshov, V. (2025). Remasking discrete diffusion models with inference-time scaling. *arXiv:2503.00307*.

Wang, X., et al. (2026). DLM-Scope: Mechanistic interpretability of diffusion language models via sparse autoencoders. *arXiv:2602.05859*.

Ye, J., Xie, Z., Zheng, L., et al. (2025). Dream 7B: Diffusion large language models. *arXiv:2508.15487*.