

Plan, Verify and Fill: A Structured Parallel Decoding Approach for Diffusion Language Models

Miao Li^{1,*} Hanyang Jiang^{1,*} Sikai Cheng¹ Hengyu Fu² Yuhang Cai²
Baihe Huang² Zhongzhu Chen³ Tinghan Ye¹ Xuanzhou Chen¹ Pascal Van Hentenryck¹
¹Georgia Institute of Technology ²University of California, Berkeley ³University of Michigan

*Equal contribution

{mli746, scottjhy}@gatech.edu

Abstract

Diffusion Language Models (DLMs) enable parallel text generation by denoising multiple masked positions in a single pass, but confidence-threshold decoding often commits only the safest predictions, leaving informative uncertainty underused. We propose **Plan-Verify-Fill (PVF)**, a training-free draft-and-verification decoding strategy that preserves the high-confidence decoding backbone while selectively testing lower-confidence candidates with high expected verified utility. PVF uses a calibrated planning-token vocabulary to allocate its limited verification budget, accepts proposed drafts only when they remain consistent with the backbone trajectory, and otherwise defaults to conservative confidence-gated unmasking. Across mathematical reasoning, knowledge, and code-generation benchmarks on LLaDA-8B-Instruct and Dream-7B-Instruct, PVF maintains comparable accuracy while delivering up to $2.3\times$ higher throughput than baseline methods.

1 Introduction

DLMs (Gong et al., 2025; Shi et al., 2024; Lou et al., 2024; Ye et al., 2025; Nie et al., 2025) extend the iterative denoising paradigm of diffusion models to discrete text generation, offering an alternative to the strictly sequential decoding employed by AR models. By operating over a global canvas and iteratively refining token predictions, DLMs unlock substantial potential for parallel token updates. Recent studies, together with real-world deployments (Inception Labs et al., 2025; Nie et al., 2025), demonstrate that masked diffusion architectures scale well and deliver competitive performance across a range of tasks.

The appeal of DLMs lies in their ability to generate in parallel (Prabhudesai et al., 2025; Chen et al., 2024; Arriola et al., 2025): a single denoising pass predicts many masked positions, enabling multiple tokens to be committed per iteration rather than

decoding strictly left-to-right as in autoregressive models. To realize this parallelism while maintaining generation accuracy, existing strategies commonly commit only sufficiently high-confidence tokens (Wu et al., 2026b; Nie et al., 2025). This conservative gate is reliable, but it can limit the achievable tokens-per-forward-pass (TPF). We hypothesize that this limitation partly reflects trajectory multiplicity: before surrounding context is resolved, a masked position may remain compatible with several coherent continuations, causing the model to distribute probability mass across alternatives. Consequently, confidence-threshold decoding may underutilize the parallel capacity of each denoising pass, making inference-time decoding a key bottleneck for unlocking the speed potential of DLMs.

A complementary line of work instead exploits uncertainty for speed, relaxing confidence-greedy decoding to expose informative high-entropy or low-confidence positions earlier (Ben-Hamu et al., 2025; Kim et al., 2025; Fu et al., 2026). While this can reduce decoding rounds, it also sharpens the accuracy–efficiency trade-off: the positions that accelerate convergence are often less reliable commitments, and premature unmasking can propagate errors or constrain later denoising steps. This raises the central question of our work:

Can fast DLM decoding move beyond confidence-greedy commitment and instead perform selective exploration that is both informative and reliable?

More concretely, in addition to committing high-confidence tokens, can we identify lower-confidence candidates whose early unmasking provides useful planning information while remaining compatible with the model’s current high-confidence denoising trajectory?

Motivated by this question, we observe that the risk of early unmasking is not uniform across to-

kens. Confidence-threshold decoding already provides a reliable backbone: high-confidence tokens can be safely committed. The opportunity lies in the lower-confidence region, where some tokens are too uncertain for ordinary confidence-based decoding but still useful for accelerating the denoising trajectory. To exploit this asymmetry, we propose **Plan-Verify-Fill (PVF)**, a training-free draft-and-verification strategy that preserves the high-confidence backbone while selectively testing a few lower-confidence drafts. PVF commits a draft only when verification shows it remains compatible with the model’s current denoising trajectory; otherwise, it falls back to safer local fills. Since verification is useful only within a small efficiency-preserving budget, PVF prioritizes candidates that offer both downstream benefit and a high chance of passing verification.

PVF operationalizes this principle through two complementary routes. The **Planning Route** uses this separation to look beyond the high-confidence backbone and identify which lower-confidence tokens are worth verifying. Rather than exploring the lower-confidence pool uniformly, it prioritizes candidates with high expected verified utility: tokens whose early unmasking can reduce downstream uncertainty while remaining compatible with the model’s current high-confidence base trajectory. Such candidates can resolve ambiguities that confidence-based decoding would otherwise defer, but their lower confidence makes direct commitment risky. PVF therefore treats them as drafts, accepting them only after impact-set checks confirm that the proposed commitment is supported by the predictions it is expected to influence. This targeted proposal process turns limited verification attempts into effective decoding progress.

When reliable planning candidates are unavailable, the **AR Fallback Route** switches to a safer local strategy: it proposes locally coherent fills that densify the context for later rounds, but accepts them only when they pass an immediate-context consistency check. Thus, PVF reframes fast DLM decoding as verified selective exploration, choosing between global planning and local filling according to where each route offers the most favorable risk–benefit trade-off.

Empirically, we evaluate PVF on four benchmark datasets using LLaDA-8B-Instruct and Dream-7B-Instruct, demonstrating robustness across two representative DLM backbones. Our contributions can be summarized as follows:

- We introduce **PVF**, a novel training-free parallel decoding strategy that treats DLM acceleration as verification-aware selective exploration rather than reactive high-confidence token harvesting.
- We design a **dual-route verified commitment** mechanism that spends the limited verification budget selectively: a Planning Route pursues high-value and verifiable drafts, while an AR Fallback Route commits safer local fills when no such planning opportunity is available.
- We demonstrate that PVF consistently improves decoding efficiency while preserving accuracy, showing gains over strong baselines across multiple DLMs and benchmarks.

1.1 Related Work

Draft-and-Verification-Based Decoding. Speculative decoding is a widely adopted family of efficient LLM inference methods built around proposal–verification: a drafter proposes candidate continuations, and a target model verifies them in parallel; self-speculative variants use the same model for both roles (Leviathan et al., 2023; Chen et al., 2023; Zhang et al., 2024). Its gains depend on a favorable acceptance regime, where accepted candidates amortize the verifier call. For DLM-style batched verification, recent work further identifies a narrow “free-lunch” region: a small number of candidates can be checked with little additional wall-clock latency because small-batch GPU inference can remain memory-bandwidth limited, but larger verification width increases latency, memory pressure, and wasted computation from rejected candidates (Wu and Zhang, 2025; Fu et al., 2026).

Verification-Guided DLM Decoding. Recent DLM methods adapt this principle to masked denoising: FreeDAVE verifies candidates along the static DLM trajectory, SSD self-drafts masked positions with hierarchical verification, and WINO drafts broadly but re-masks suspicious tokens under stricter verification (Wu and Zhang, 2025; Gao et al., 2025; Hong et al., 2026). These methods enlarge the safe acceptance window, but primarily organize verification around agreement with the high-confidence base trajectory. PVF instead treats the limited free-lunch budget as verification-aware selective exploration, spending it on candidates with high expected verified utility: informative planning tokens when they are likely to pass

impact-set checks, and safer AR-style fallback fills otherwise.

Token-level Planning. The limited low-overhead verification regime in proposal–verification decoding puts pressure on candidate selection for DLMs: unlike AR decoding, which extends a single left-to-right prefix, masked decoding exposes a combinatorial space of positions and generation orders, so useful drafts should be both likely to pass and worth revealing. Recent analyses of reasoning traces suggest that the downstream utility of revealing a position is highly non-uniform: some high-entropy positions are not merely uncertain residues, but trajectory-defining choices that can shape the remaining generation (Wang et al., 2025; Cheng et al., 2026; Ni et al., 2026).

Purely confidence-based DLM decoding can therefore be too locally conservative: by filling easy tokens first, it may bypass high-entropy decision points, and Ni et al. (2026) shows that such bypassing hurts pass@ k accuracy compared with decoding orders that confront these positions directly. PVF adapts this insight to inference-time decoding by treating selected high-entropy positions as verifiable drafts: instead of globally lowering the confidence threshold, which can trade accuracy for TPF (Wu et al., 2026b), PVF targets uncertain candidates with high expected information gain and accepts them only when they remain compatible with the current denoising consensus, turning limited verification budget into reliable progress before their ambiguity is eliminated by surrounding context.

2 Preliminary

Departing from the left-to-right decoding of AR models, DLMs (Austin et al., 2021; Ou et al., 2025) formulate generation as iterative denoising. They define a forward corruption process that gradually masks a clean sequence, and learn a reverse process that reconstructs text from partially masked states.

Forward Process. Given a clean sequence $\mathbf{x}_0$, the forward process independently corrupts each token over continuous time $s \in [0, 1]$ by replacing it with a special absorbing token [MASK]. Once masked, a token remains masked, so $\mathbf{x}_1$ is fully masked. Following (Austin et al., 2021), this pro-

cess is

$$q(\mathbf{x}_s | \mathbf{x}_0) = \prod_{i=1}^{|\mathbf{x}_0|} q(x_s^i | x_0^i) \\ = \prod_{i=1}^{|\mathbf{x}_0|} \text{Cat}\left(x_s^i; (1-s)\delta_{x_0^i} + s\delta_{[\text{MASK}]}\right).$$

Reverse Process. Given $\mathbf{y} = [\mathbf{y}_{\text{prompt}}, \mathbf{y}_{\text{gen}}]$, where $\mathbf{y}_{\text{prompt}}$ is fixed and $\mathbf{y}_{\text{gen}}$ is initialized as masked tokens. A denoising model P_θ is trained typically with an ELBO-based objective, to recover clean tokens from corrupted states (Nie et al., 2025; Sahoo et al., 2024). At inference time, generation iteratively applies $P_\theta(\cdot | \mathbf{y}_t)$ to a partially masked sequence $\mathbf{y}_t$, producing token distributions over the vocabulary for all masked positions in parallel. A decoding rule then selects which predictions to commit, yielding a progressively less-masked sequence until generation completes.

Confidence-based Parallel Decoding: One of the most prominent decoding strategies in DLMs is confidence-based decoding (Yu et al., 2025), which governs the generation process based on prediction certainty. For each masked position i at step t , define the model’s top prediction $\hat{y}_t^i$ conditioned on the previous state $\mathbf{y}_{t-1}$:

$$\hat{y}_t^i = \arg \max_{w \in \mathcal{V}} P_\theta(y^i = w | \mathbf{y}_{t-1}).$$

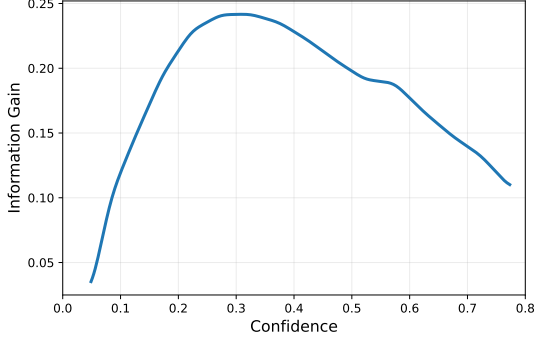
In its most canonical form, the algorithm progressively fills the sequence by committing to tokens where the model’s confidence exceeds a static threshold τ_{high} :

$$y_t^i = \begin{cases} \hat{y}_t^i, & \text{if } P_\theta(y^i = \hat{y}_t^i | \mathbf{y}_{t-1}) \geq \tau_{\text{high}}, \\ y_{t-1}^i, & \text{otherwise.} \end{cases}$$

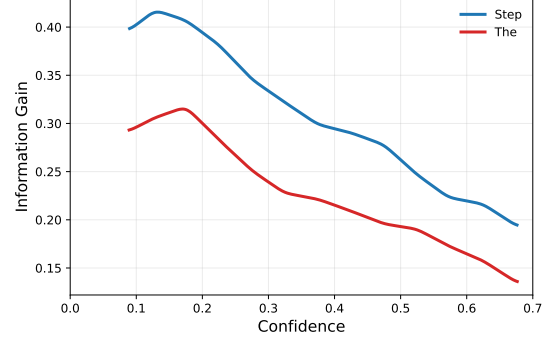
While this static heuristic serves as a basic building block for parallel decoding, later approaches (Chen et al., 2026; Jin et al., 2025; Lou et al., 2024) adopt more complex decoding mechanisms that extend beyond heuristic gating.

3 Selecting High-Entropy Candidates for Planning

Confidence-based parallel decoding provides a reliable base trajectory by committing positions whose confidence exceeds τ_{high} . Our PVF preserves this base behavior: high-confidence tokens are committed by default and serve as the backbone of the current step. However, because these tokens



(a) Average over all candidate tokens.



(b) Representative tokens Step and The.

Figure 1: Relationship of information gain of candidate token commitments versus model confidence on GSM8K. **(a)** Average over all candidate tokens, showing a sweet spot in the low to moderate confidence region where candidates provide high marginal information gain. **(b)** Two representative vocabulary tokens, Step and The, showing that information gain varies by token even at similar confidence.

are already near-certain, their marginal information value is limited. As Figure 1a shows, candidate commitments in a low to moderate confidence region can yield larger information gain than high-confidence tokens, while extremely low-confidence candidates are less useful. This suggests that the main acceleration opportunity lies in carefully selected uncertain positions that help resolve the surrounding trajectory earlier.

Figure 1b further shows that information gain is not determined by confidence alone. On GSM8K, Step frequently appears in step-by-step solutions such as Step 1 and Step 2, and yields consistently higher information gain than a generic token such as The. This motivates selecting a compact proposal set of uncertain candidates that are both compatible with the high-confidence trajectory and informative for the remaining masked positions. Since only a few candidates can be checked within the small-batch free-lunch regime, we use a planning-token vocabulary P^{plan} to restrict the proposal space. This vocabulary only controls eligibility. The full PVF algorithm still verifies each proposal online before accepting it.

Base branch and impact set. We use the high-confidence decoding path as a local reference for evaluating riskier high-entropy commitments. Intuitively, the base set contains the positions that confidence-only decoding would safely commit at the current step. The base branch is the resulting sequence after making only these safe commits. It is not ground truth, but it is the most reliable local trajectory available to a training-free decoder. A planning draft should therefore preserve the future predictions that this base path already makes with

high confidence.

Let B_t denote the active block at step t , and let

$$M_t = \{i \in B_t \mid y_{t-1}^i = [\text{MASK}]\}$$

be the currently masked positions. For each $i \in M_t$, define the top prediction and its confidence as

$$\begin{aligned} \hat{y}_t^i &= \arg \max_{w \in V} P_\theta(y^i = w \mid \mathbf{y}_{t-1}), \\ p_t^i &= P_\theta(y^i = \hat{y}_t^i \mid \mathbf{y}_{t-1}). \end{aligned}$$

The base set and base branch are

$$\begin{aligned} S_t^{\text{base}} &= \{i \in M_t \mid p_t^i \geq \tau_{\text{high}}\}, \\ z_t^{i,\text{base}} &= \begin{cases} \hat{y}_t^i, & i \in S_t^{\text{base}}, \\ y_{t-1}^i, & i \notin S_t^{\text{base}}. \end{cases} \end{aligned}$$

We define the impact set as the positions that remain masked after the base commit but become high-confidence under the base branch:

$$\begin{aligned} S_t^{\text{impact}} &= \{i \in B_t \mid z_t^{i,\text{base}} = [\text{MASK}], \\ &\quad \max_{w \in V} P_\theta(y^i = w \mid \mathbf{z}_t^{\text{base}}) \geq \tau_{\text{high}}\}. \end{aligned} \quad (1)$$

These positions form the high-confidence future that additional commitments should preserve.

Verified compatibility. A high-entropy candidate should not disturb the high-confidence path established by the base branch. For candidate j , let $i_{(j)}$ denote its position. We form a temporary planning branch by starting from the base branch and additionally committing this candidate:

$$z_t^{i,\text{plan},j} = \begin{cases} \hat{y}_t^{i_{(j)}}, & i = i_{(j)}, \\ z_t^{i,\text{base}}, & \text{otherwise.} \end{cases} \quad (2)$$

This planning branch is only a draft. It is not committed unless it passes verification.

The candidate is verified-compatible if the impact set is nonempty and the high-confidence top predictions on all positions in S_t^{impact} remain the same under $\mathbf{z}_t^{\text{plan},j}$ as under $\mathbf{z}_t^{\text{base}}$. We denote this binary outcome by $A_t^j \in \{0, 1\}$:

$$A_t^j = 1 \iff \begin{aligned} &|S_t^{\text{impact}}| > 0 \text{ and} \\ &\mathbf{z}_t^{\text{plan},j} \text{ preserves top-1} \\ &\text{base predictions on } S_t^{\text{impact}}. \end{aligned} \quad (3)$$

Thus, $A_t^j = 1$ means that the candidate is locally consistent with the model’s high-confidence reference path. This is not a guarantee of global correctness. It is a lightweight stability check used to avoid commitments that immediately contradict the base trajectory.

Verified information gain. Verified compatibility screens for risk, but it does not measure usefulness. Since high-confidence tokens are already committed by the base decoder, useful additional commitments should make the unresolved part of the block less uncertain. We therefore score a candidate by the entropy reduction it produces over the remaining masked positions after passing verification.

Let

$$H_i(\mathbf{z}) = H(P_\theta(y^i | \mathbf{z}))$$

be the entropy of the model’s token distribution at position i under state $\mathbf{z}$. For candidate j , define the evaluation region as

$$R_t^j = \left\{ i \in B_t \mid z_t^{i,\text{base}} = [\text{MASK}], i \neq i_{(j)} \right\}.$$

This excludes the candidate position itself, so the score measures how the candidate affects the remaining masked positions. Define

$$d_{t,i}^j = H_i(\mathbf{z}_t^{\text{base}}) - H_i(\mathbf{z}_t^{\text{plan},j}).$$

When R_t^j is nonempty, the entropy reduction induced by candidate j is

$$\Delta H_t^j = \frac{1}{|R_t^j|} \sum_{i \in R_t^j} d_{t,i}^j.$$

If R_t^j is empty, we set $\Delta H_t^j = 0$. The event-level verified information gain is

$$\text{VIG}_t^j = A_t^j \Delta H_t^j.$$

A positive VIG_t^j means that the candidate both preserves the high-confidence path and reduces uncertainty over the remaining block.

Planning-token vocabulary. We compute verified compatibility and verified information gain for each vocabulary token on an offline calibration set. Tokens that appear often enough and achieve high average scores on both metrics are retained as the planning-token vocabulary P^{plan} . This selection is data-driven rather than manually specified: it identifies tokens that are empirically likely to pass verification and reduce uncertainty when proposed in high-entropy positions. The resulting vocabulary is frozen before evaluation and used only as an eligibility filter for the planning route. Every proposed commitment is still verified online by PVF. The full calibration procedure, thresholds, and data sources are provided in Appendix B.

4 Methodology: Plan-Verify-Fill (PVF) Decoding

We propose **Plan-Verify-Fill (PVF)** (Figure 2), a training-free draft-and-verification decoder for DLMs. PVF keeps confidence-threshold decoding as a high-confidence backbone: at each decoding step t , it forms $\mathbf{z}_t^{\text{base}}$ from S_t^{base} , then uses S_t^{impact} defined in Equation (1) to verify a small number of additional drafts. These drafts come from two routes: the **Planning Route**, which targets lower-confidence candidates from P^{plan} , and the **AR Fallback Route**, which supplies safer local fills when planning is unavailable. Detailed route-level schematics and the complete decoding procedure are provided in Appendix G (Figure 10 and Algorithm 1).

4.1 Planning Route Phase I: Candidate Proposal

PVF exploits the small-batch verification regime by testing only a few additional drafts alongside the base branch. The Planning Route selects these exploratory drafts from lower-confidence candidates whose predicted token lies in P^{plan} , treating them as proposals that are committed only after impact-set verification supports their compatibility with the current denoising trajectory.

Planning candidates. Using the planning-token set P^{plan} constructed in Section 3, PVF selects non-base masked positions whose top prediction is a planning token and whose confidence falls in a planning band:

$$P_t^{\text{plan}} = \left\{ i \in M_t \setminus S_t^{\text{base}} \mid \hat{y}_t^i \in P^{\text{plan}}, \right. \\ \left. \tau_{\text{plan}}^\ell \leq p_t^i < \tau_{\text{plan}}^u \right\}.$$

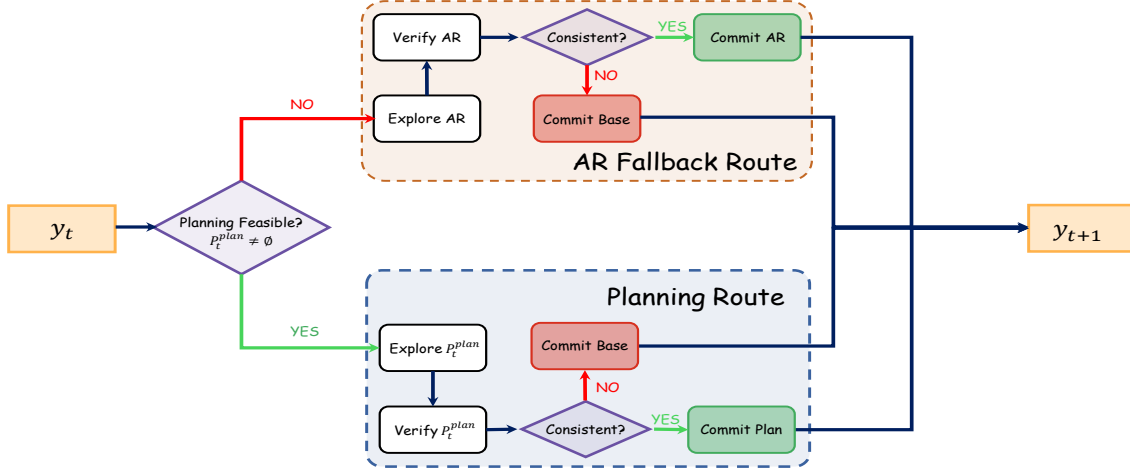


Figure 2: Overview of the **Plan-Verify-Fill (PVF)** decoding pipeline.

The lower bound filters highly unstable predictions, while the upper bound removes near-confident positions that offer limited additional information. The band therefore targets candidates with meaningful uncertainty but enough model support to be verification-worthy. As described in Section 3, P^{plan} acts as an eligibility list: a masked position can be considered for planning only when its top-predicted token belongs to this vocabulary. The planning band then filters these positions by confidence; its lower and upper bounds control how aggressively eligible candidates are proposed during inference, while online verification remains the final acceptance test. Appendix B details the vocabulary construction, the role of the planning band, and sensitivity to its two bounds.

PVF keeps the top- K candidates by confidence:

$$\{i_{(1)}, \dots, i_{(K_t)}\} = \text{TopK}_{i \in P_t^{\text{plan}}}(p_t^i),$$

where $K_t = \min(K, |P_t^{\text{plan}}|)$. For each selected candidate $i_{(j)}$, PVF instantiates the temporary planning branch $\mathbf{z}_t^{\text{plan},j}$ defined in Equation (2), and evaluates all branches in one batched forward pass:

$$\{\mathbf{z}_t^{\text{base}}, \mathbf{z}_t^{\text{plan},1}, \dots, \mathbf{z}_t^{\text{plan},K_t}\}.$$

We use $K = 3$ by default, keeping verification within the small-width regime used by prior draft-and-verification DLM methods (Wu and Zhang, 2025; Gao et al., 2025; Hong et al., 2026). If $P_t^{\text{plan}} = \emptyset$, PVF skips the Planning Route and uses the AR Fallback Route.

4.2 Planning Route Phase II: Verification and Selection

The Planning Route separates offline usefulness from online acceptability. The offline VIG score is used only to build P^{plan} , identifying vocabulary tokens that are promising to explore. During decoding, however, every concrete proposal is re-verified in context: PVF accepts a planning draft only if its impact set remains compatible with the current high-confidence trajectory. In this sense, P^{plan} provides the proposal prior, while online verification provides the commitment test.

Filter 1: Impact-set verification. For each planning branch $\mathbf{z}_t^{\text{plan},j}$, PVF computes the verification indicator A_t^j defined in Equation (3). Only verified candidates are preserved:

$$\mathcal{J}_t^{\text{plan}} = \{j \in [K_t] \mid A_t^j = 1\}.$$

Equivalently, a planning branch is accepted by Filter 1 only if it preserves the base branch’s top-1 predictions on the nonempty impact set S_t^{impact} . This makes the proposed commitment consistent with the model-internal high-confidence base trajectory used as the reference trajectory. If $\mathcal{J}_t^{\text{plan}} = \emptyset$, PVF treats planning as premature for the current state, and commits $\mathbf{y}_t \leftarrow \mathbf{z}_t^{\text{base}}$.

Filter 2: Runtime confidence maximization. If at least one planning branch passes Filter 1, PVF selects among the verified branches using a runtime confidence objective:

$$\mathcal{C}_{\text{total}}(\mathbf{z}) = \sum_{\substack{i \in B_t \\ z^i = [\text{MASK}]}} \max_{w \in V} P_\theta(y^i = w \mid \mathbf{z}).$$

PVF selects

$$j^* = \arg \max_{j \in \mathcal{J}_t^{\text{plan}}} \mathcal{C}_{\text{total}}(\mathbf{z}_t^{\text{plan},j})$$

and commits $\mathbf{y}_t \leftarrow \mathbf{z}_t^{\text{plan},j^*}$.

This final selection favors the verified branch that makes the remaining masked positions in the current decoding horizon most ready for future commitment.

4.3 AR Fallback Route: Local Filling

When no planning draft is available, PVF uses an AR fallback route to make conservative local progress. The fallback proposes short left-to-right extensions from the remaining non-base masked positions and accepts only the longest extension that remains stable under verification.

AR candidate construction. Let $i_1^{\text{AR}}, \dots, i_{K_t^{\text{AR}}}^{\text{AR}}$ be the leftmost positions in $M_t \setminus S_t^{\text{base}}$ whose confidence satisfies $p_t^i \geq \tau_{\text{AR}}^\ell$, ordered by position, where $K_t^{\text{AR}} \leq K$. For each $k \in [K_t^{\text{AR}}]$, PVF constructs an AR branch by adding the first k proposed fills to the base branch:

$$z_t^{i,\text{AR},k} = \begin{cases} \hat{y}_t^i, & i \in \{i_1^{\text{AR}}, \dots, i_k^{\text{AR}}\}, \\ z_t^{i,\text{base}}, & \text{otherwise.} \end{cases}$$

AR verification and selection. PVF accepts an AR branch only if its proposed fills remain unchanged under the verified trajectory:

$$\mathcal{K}_t^{\text{AR}} = \left\{ k \in [K_t^{\text{AR}}] \mid \forall r \leq k, \arg \max_{w \in V} P_\theta(y_r^{i_r^{\text{AR}}} = w \mid \mathbf{z}_t^{\text{AR},k}) = \hat{y}_t^{i_r^{\text{AR}}} \right\}.$$

If $\mathcal{K}_t^{\text{AR}} \neq \emptyset$, PVF commits the longest verified extension,

$$k^* = \max \mathcal{K}_t^{\text{AR}}, \quad \mathbf{y}_t \leftarrow \mathbf{z}_t^{\text{AR},k^*}.$$

Otherwise, it falls back to the base branch, $\mathbf{y}_t \leftarrow \mathbf{z}_t^{\text{base}}$.

5 Experiment Results

5.1 Experiment Settings

We evaluate PVF on four benchmarks datasets spanning mathematical reasoning, coding synthesis, and broad knowledge: GSM8K (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024), HumanEval (Chen et al., 2021), and MATH (Hendrycks et al., 2021). LLaDA-8B-Instruct and Dream-7B-Instruct are

used as base models to verify that PVF’s performance gains are robust across two representative backbones. We report both accuracy and efficiency, using the Number of Function Evaluations (NFE), the total number of model forward passes required to complete a sequence as our main efficiency metric. Besides, we also report tokens generated per second (TPS) under KV cache (Wu et al., 2026b) as the realized throughput metric, allowing us to verify whether fewer forward passes translate into faster generation. To ensure a fair comparison, we fix $\tau_{\text{high}} = 0.9$ across all experiments unless stated otherwise.

A batched forward pass counts as one NFE even when it jointly evaluates the base branch and multiple candidate branches. With the default verification width $K = 3$, a verification batch contains one base branch and three candidates (batch size 4), which lies within the small-width “free-lunch” region (Fu et al., 2026; Wu and Zhang, 2025). This convention records sequential decoding rounds, while TPS reports the actual wall-clock cost of the batched computation.

Across all datasets, we use a fixed maximum generation length of $L = 512$ and a block size of $B = 64$. Decoding is training-free (no finetuning, no additional supervision, and no external verifier). Sensitivity to L and B is reported in Appendix D.2. All experiments are conducted on a single NVIDIA H200 GPU. Additional experimental details are provided in Appendix E.2.

5.2 Main Results

Baselines. We compare PVF with three training-free DLM decoding baselines. **Static** greedily commits the most confident token at each iteration. **Fast-dLLM** (Wu et al., 2026b) performs confidence-threshold parallel decoding by committing all tokens whose posterior confidence exceeds τ_{high} . **FreeDave** (Wu and Zhang, 2025) follows a draft-and-verify design, generating look-ahead drafts and committing the longest consecutively verified span under top-1 decoding to reduce the number of model forward passes. Further evaluations and analyses are provided in Appendices C–D; mechanism and experimental details, qualitative examples, and the complete decoding algorithm appear in Appendices E–G.

Accuracy and NFE. Table 1 summarizes accuracy and NFE across four benchmarks. Compared with Fast-dLLM, PVF achieves a $1.8\times$ – $2.9\times$ NFE

Table 1: Accuracy and NFE comparison with baseline methods. Brackets report accuracy change from Static and NFE speedup over Static.

Model	Method	GSM8K		MMLU-Pro		HumanEval		Math	
		Acc.	NFE ↓	Acc.	NFE ↓	Acc.	NFE ↓	Acc.	NFE ↓
LLaDA-8B Instruct	Static	79.46	512.00	36.80	512.00	46.95	512.00	41.06	512.00
	Fast-dLLM	79.62 (+0.16)	89.77 (5.70×)	36.20 (-0.60)	80.26 (6.38×)	47.56 (+0.61)	124.26 (4.12×)	40.84 (-0.22)	147.16 (3.48×)
	FreeDave	79.08 (-0.38)	135.65 (3.77×)	36.70 (-0.10)	136.36 (3.75×)	45.73 (-1.22)	118.72 (4.31×)	40.90 (-0.16)	142.82 (3.58×)
	PVF	79.75 (+0.29)	31.34 (16.34×)	36.40 (-0.40)	29.55 (17.33×)	46.66 (-0.29)	70.31 (7.28×)	40.24 (-0.82)	73.56 (6.96×)
Dream-7B Instruct	Static	78.00	512.00	41.30	512.00	60.37	512.00	42.20	512.00
	Fast-dLLM	77.71 (-0.29)	75.83 (6.75×)	40.18 (-1.12)	92.16 (5.56×)	59.14 (-1.23)	112.78 (4.54×)	43.46 (+1.26)	129.77 (3.95×)
	FreeDave	79.92 (+1.92)	141.01 (3.63×)	41.10 (-0.20)	91.74 (5.58×)	59.15 (-1.22)	112.82 (4.54×)	42.00 (-0.20)	130.25 (3.93×)
	PVF	79.20 (+1.20)	32.34 (15.83×)	43.30 (+2.00)	35.22 (14.54×)	62.66 (+2.29)	50.89 (10.06×)	43.66 (+1.46)	64.80 (7.90×)

Table 2: TPS comparison on LLaDA-8B-Instruct. Brackets report speedup over Static. The corresponding TPS comparison on Dream-7B-Instruct is reported in Appendix C.3.

Dataset	Static	Fast-dLLM	FreeDave	PVF
GSM8K	31.74	111.37 (3.51×)	71.24 (2.24×)	259.35 (8.17×)
MMLU-Pro	30.48	64.20 (2.11×)	53.89 (1.77×)	113.78 (3.73×)
HumanEval	30.25	100.04 (3.31×)	108.10 (3.57×)	163.26 (5.40×)
Math	32.41	96.75 (2.99×)	87.34 (2.69×)	161.12 (4.97×)

speedup across datasets and model backbones. On LLaDA-8B-Instruct, all accuracy differences between PVF and Fast-dLLM are within one percentage point. On Dream-7B-Instruct, PVF is numerically higher than Fast-dLLM on all four benchmarks, with differences ranging from 0.20 to 3.52 percentage points. Compared with FreeDave, PVF achieves a $1.7\times$ – $4.6\times$ NFE speedup across the same eight model–dataset comparisons, while accuracy differences range from -0.72 to $+3.51$ percentage points. Overall, these results support substantially higher decoding efficiency without a systematic loss in accuracy. Appendix C.1 evaluates whether PVF’s efficiency advantage extends to the recent verification-based decoders SSD and WINO under a unified setting on LLaDA-8B-Instruct. Appendix C.2 expands the multiple-choice evaluation from MMLU-Pro to ARC-Challenge and WinoGrande, testing whether the efficiency gains persist for concise outputs.

Throughput. NFE is an algorithmic efficiency metric, but a lower NFE is useful only if it translates into actual throughput. Table 2 confirms that PVF improves TPS over Fast-dLLM on every benchmark, corresponding to roughly 1.6 – $2.3\times$ higher throughput than Fast-dLLM. This indicates that the additional verification work does not erase the gains from fewer decoding rounds. Within the small-batch regime, the NFE reduction is reflected in real generation speed.

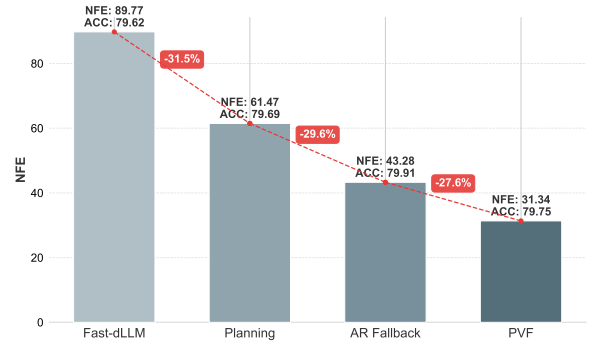


Figure 3: Ablation study on GSM8k, evaluating the contribution of each PVF component. Accuracy scores are displayed in the bracket above each bar to confirm they remain comparable across methods.

Ablation of the Two PVF Routes. Figure 3 decouples the two PVF routes. Planning-only is activated only when eligible planning candidates exist, whereas AR Fallback can operate at every decoding round, giving it more frequent opportunities to add verified tokens and explaining its lower standalone NFE. Still, Planning reduces NFE from 89.77 to 61.47 with comparable accuracy, showing

that high-quality planning in the low-confidence region is valuable when available. Combining both routes gives the strongest result: full PVF reaches 31.34 NFE, a further 27.6% reduction over AR Fallback alone. This suggests that frequent local filling and selective planning are complementary: fallback supplies steady decoding progress, while planning adds extra gains when the low-confidence region contains high-value candidates. In other words, the result highlights the importance of exploiting high-value low-confidence positions when they are verifiable, which is precisely the role of PVF’s Planning Route.

Ablation of Verification. Table 3 isolates candidate verification while keeping the proposal mechanism and all other settings fixed at the default width $K = 3$. Verification is essential for preserving accuracy, especially on HumanEval. On GSM8K, removing all verification lowers accuracy from 79.75% to 76.00% and does not improve NFE (31.34 versus 33.19). On HumanEval, removing AR verification or all verification lowers accuracy from 46.66% to 20.73%, while NFE changes only from 70.31 to 67.34 or 67.03. Thus, verification prevents harmful commitments with little or no sacrifice in decoding efficiency. Full results, including force-accept controls, are reported in Appendix D.4.

Table 3: Compact verification ablation on LLaDA-8B-Instruct at $K = 3$. Each entry reports Accuracy/NFE.

Variant	GSM8K	HumanEval
Full PVF	79.75 / 31.34	46.66 / 70.31
No planning verification	78.13 / 36.72	44.51 / 70.27
No AR verification	78.45 / 27.58	20.73 / 67.34
No verification	76.00 / 33.19	20.73 / 67.03

Route Behavior. To characterize route behavior, PVF decoding trajectories are logged and the resulting events are aggregated over four benchmarks. A Planning attempt, recorded when the current state contains at least one eligible planning candidate and one or more planning branches are evaluated, occurs on 14.09%–57.68% of decoding steps across tasks. Among the proposed planning branches, 41.12%–73.65% pass Filter 1 by preserving the base branch’s top-1 predictions on a nonempty impact set. Empty impact sets occur on 43.23%–59.87% of steps, which helps explain why Planning remains selective. For the AR Fallback Route, the most frequent outcome is an accepted

length of 3, meaning that the longest default extension passes local consistency and all three proposed tokens are committed. This outcome accounts for 58.83%–66.48% of the accepted-length distribution. Together, these statistics characterize selective planning and frequent local filling; complete results are provided in Appendix D.6.

Verification-Width Sensitivity. A sensitivity analysis varies K , the maximum number of candidate branches evaluated with the base branch, from 1 to 3 on GSM8K and HumanEval. Increasing K reduces NFE from 45.83 to 31.34 on GSM8K and from 102.36 to 70.31 on HumanEval. TPS correspondingly increases from 187.60 to 259.35 and from 116.26 to 163.26, while accuracy remains comparable. Thus, a wider verification budget improves decoding efficiency without a clear accuracy loss; complete results are provided in Appendix D.5.

Memory Overhead. A separate memory analysis compares per-example peak allocated GPU memory for confidence-only decoding and Full PVF under the same inference setting. At the default $K = 3$, mean peak memory increases from 15.974 to 17.699 GiB (10.80%), and the maximum overhead is 11.87%. The efficiency gains at the default verification width therefore require moderate additional memory; complete results are provided in Appendix D.7.

6 Conclusion

We introduced **Plan-Verify-Fill (PVF)**, a training-free decoding strategy for DLMs that uses limited verification budget to turn selected low-confidence candidates into reliable decoding progress. PVF preserves the high-confidence decoding backbone, verifies high-value planning drafts when informative uncertain positions are available, and falls back to conservative local filling otherwise. Experiments show consistent efficiency improvements while maintaining comparable accuracy, highlighting that low-confidence planning and local fill-in are complementary: effective DLM acceleration comes not from uniformly lowering confidence thresholds, but from spending verification on high-value uncertain positions when they can be safely accepted.

Limitations

Scope. PVF is a training-free inference-time framework for adapting proposal–verification de-

coding to DLMs. Its central message is that masked decoding exposes a much larger space of candidate positions and generation orders than left-to-right decoding, while only a small number of additional drafts can be verified within the low-overhead regime. PVF should therefore be viewed as an initial step toward targeted draft selection, rather than a universal solution for all DLM decoding settings.

Model-internal verification. PVF relies on model-internal verification. Impact-set checks test whether an exploratory commitment remains compatible with the model’s current high-confidence denoising trajectory, while local consistency checks play a similar role for AR-style fallback fills. These checks are lightweight and training-free, but they are not external correctness guarantees: a draft that passes verification is locally consistent with the model’s own trajectory, not necessarily globally correct. Thus, PVF improves the inference-time accuracy–efficiency trade-off of an existing DLM, but does not replace stronger training-time alignment, external verifiers, or self-correction mechanisms.

Fixed calibration. The current implementation uses fixed calibration choices. The planning candidate set, confidence bands, and verification width are chosen before evaluation rather than adapted dynamically to each prompt, model state, or task instance. This keeps PVF simple and inexpensive, and every proposed commitment is still verified online before acceptance. Nevertheless, different model families, tokenizers, languages, or domains may benefit from different proposal policies. A natural extension is to learn or adapt the proposal distribution online, so that the limited verification budget is allocated to candidates with higher expected verified utility.

Systems dependence. Batched verification is not literally cost-free. The “free-lunch” region refers to a small-width systems regime in which evaluating a few additional branches adds little incremental wall-clock latency, often because GPU inference remains memory-bandwidth limited and accepted drafts amortize the verifier call. Outside this regime, larger verification widths can increase memory pressure and latency, while rejected candidates can turn parallelism into wasted computation. PVF deliberately keeps the verification width small, but realized wall-clock speedups may vary with ac-

celerator type, inference backend, batching policy, memory layout, and serving load.

Evaluation coverage. Our experiments focus on a limited set of DLM backbones, benchmarks, block sizes, and generation lengths. While these settings cover representative reasoning and knowledge tasks, broader deployment may require retuning confidence bands, block configurations, and verification width for the target environment. We view this as part of the broader direction opened by PVF: efficient DLM decoding is not only about accepting more tokens, but about deciding which small set of exploratory commitments is worth verifying.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039655 and the National Science Foundation Artificial Intelligence Institute for Advances in Optimization under Grant No. 2112533. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. [Block diffusion: Interpolating between autoregressive and diffusion language models](#). In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. [Structured denoising diffusion models in discrete state-spaces](#). In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc.
- Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. 2025. [Accelerated sampling from masked diffusion models via entropy bounded unmasking](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 55981–56007. Curran Associates, Inc.
- Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. 2024. [Diffusion forcing: Next-token prediction meets full-sequence diffusion](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 24081–24125. Curran Associates, Inc.

- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. [Accelerating large language model decoding with speculative sampling](#). *Preprint*, arXiv:2302.01318.
- 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, and 39 others. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. 2026. [dparallel: Learnable parallel decoding for dLLMs](#). In *The Fourteenth International Conference on Learning Representations*.
- Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. 2026. [Reasoning with exploration: An entropy perspective](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30377–30385.
- 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. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Junkeun Yi, Mohammad Mahdi Kamani, Venkat Krishna Srinivasan, and Jiantao Jiao. 2026. [From bits to rounds: Parallel decoding with exploration for diffusion language models](#). In *Forty-third International Conference on Machine Learning*.
- Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. 2025. [Self speculative decoding for diffusion large language models](#). *Preprint*, arXiv:2510.04147.
- Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. 2025. [Scaling diffusion language models via adaptation from autoregressive models](#). In *The Thirteenth International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the math dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. 2026. [Wide-in, narrow-out: Revokable decoding for efficient and effective dllms](#). In *International Conference on Learning Representations*, volume 2026, pages 40132–40150.
- Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. 2025. [Mercury: Ultra-fast language models based on diffusion](#). *Preprint*, arXiv:2506.17298.
- Xiangqi Jin, Yuxuan Wang, Yifeng Gao, Zichen Wen, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2025. [Thinking inside the mask: In-place prompting in diffusion llms](#). *Preprint*, arXiv:2508.10736.
- Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. 2025. [KLASS: KL-guided fast inference in masked diffusion models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. [Fast inference from transformers via speculative decoding](#). In *Proceedings of the 40th International Conference on Machine Learning, ICML'23*. JMLR.org.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. [Discrete diffusion modeling by estimating the ratios of the data distribution](#). In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org.
- Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. 2026. [The flexibility trap: Rethinking the value of arbitrary order in diffusion language models](#). In *Forty-third International Conference on Machine Learning*.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan LI. 2025. [Large language diffusion models](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 50608–50646. Curran Associates, Inc.
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. [Your absorbing discrete diffusion secretly models the conditional distributions of clean data](#). In *The Thirteenth International Conference on Learning Representations*.
- Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. 2025. [Diffusion beats autoregressive in data-constrained settings](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 10581–10606. Curran Associates, Inc.
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. [Simple and effective masked diffusion language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 130136–130184. Curran Associates, Inc.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. [Simplified and generalized masked diffusion for discrete data](#). In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA. Curran Associates Inc.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. [Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. 2026. [Diffusion llms can do faster-than-ar inference via discrete diffusion forcing](#). In *International Conference on Learning Representations*, volume 2026, pages 2282–2299.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. [Mmlu-pro: A more robust and challenging multi-task language understanding benchmark](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290. Curran Associates, Inc.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2026a. [Fast-dllm v2: Efficient block-diffusion llm](#). In *International Conference on Learning Representations*, volume 2026, pages 128353–128370.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2026b. [Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding](#). In *The Fourteenth International Conference on Learning Representations*.

Shutong Wu and Jiawei Zhang. 2025. [Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models](#). In *NeurIPS 2025 Workshop on Efficient Reasoning*.

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

Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025. [Dimple: Discrete diffusion multimodal large language model with parallel decoding](#). *Preprint*, arXiv:2505.16990.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. [Draft & verify: Lossless large language model acceleration via self-speculative decoding](#). In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11263–11282. Association for Computational Linguistics.

Appendices

A Additional Background of Semi-Autoregressive Blockwise Decoding

Semi-autoregressive diffusion adopts a hybrid structure combining *inter-block* autoregression with *intra-block* parallel diffusion. Widely adopted in modern DLMs (Arriola et al., 2025; Nie et al., 2025; Sahoo et al., 2024; Wu et al., 2026a,b), this strategy restricts attention to local regions, thereby reducing inference cost while preserving the local context. First partition the fixed canvas length L into B blocks, writing $\mathbf{y}_t = [y_t^{1:L/b}, y_t^{L/b+1:2(L/b)}, \dots, y_t^{L-b+1:L}]$. The k -th block encompasses the token segment spanning from index $(k-1)\frac{L}{B} + 1$ to $k\frac{L}{B}$. During inference, the decoding process adheres to a block-causal constraint: the state $\mathbf{y}_t$ can commit tokens in the $(k+1)$ -th block only if the k -th block is fully resolved; otherwise, all subsequent blocks $j > k$ must remain masked. In this paper, we follow the framework of blockwise decoding, and define $\mathcal{B}_t$ to represent the set of indices for the current working block where [MASK] tokens are eligible for replacement at step t .

B Planning Vocabulary: Construction and Confidence Bands

This appendix consolidates the role, construction, contents, and robustness of the planning-token vocabulary, and explains how the calibration band used during construction relates to the planning band used during inference.

B.1 Role and Offline Construction

Overview. Section 3 defines two event-level quantities for a candidate planning draft: verified compatibility A_t^j and verified information gain VIG_t^j . Here we describe how these event-level scores are aggregated into the planning-token vocabulary P^{plan} . The construction is performed offline on non-evaluation data, and the resulting vocabulary is frozen before evaluation. Importantly, P^{plan} is only an eligibility filter for proposing candidates. It does not approve any commitment by itself, since every proposed planning draft is still verified online by PVF.

Calibration trajectory. For each task family, we run the target DLM on a calibration set $\mathcal{D}_{\text{cal}}$ using the same confidence-based base decoding rule as in Section 3. At each calibration step t , we form the base set S_t^{base} , the base branch $\mathbf{z}_t^{\text{base}}$, and the impact set S_t^{impact} . We then inspect only non-base masked positions whose top-token confidence lies in a fixed calibration band:

$$\mathcal{C}_t^{\text{cal}} = \left\{ i \in M_t \setminus S_t^{\text{base}} \mid \tau_{\text{cal}}^{\ell} \leq p_t^i < \tau_{\text{cal}}^u \right\}.$$

This band focuses calibration on the uncertain region where planning is potentially useful, while avoiding candidates that are either too unstable or already close to the high-confidence base rule.

For each candidate position $i \in \mathcal{C}_t^{\text{cal}}$, we let $v = \hat{y}_t^i$ be its predicted vocabulary token and temporarily index it as candidate j with $i_{(j)} = i$. We then construct the planning branch $\mathbf{z}_t^{\text{plan},j}$ as in Eq. 2, compute the verified-compatibility outcome A_t^j as in Eq. 3, and compute the event-level verified information gain VIG_t^j as defined in Section 3. These branches are used only for measurement and are not committed to the calibration trajectory.

Token-level aggregation. Let $\mathcal{E}_v$ denote all calibration candidate events whose predicted vocabulary token is v :

$$\mathcal{E}_v = \left\{ (t, j) \mid \hat{y}_t^{i_{(j)}} = v, i_{(j)} \in \mathcal{C}_t^{\text{cal}} \right\}.$$

For each vocabulary token v , we compute four statistics:

$$\begin{aligned} n_v &= |\mathcal{E}_v|, \\ m_v &= \sum_{(t,j) \in \mathcal{E}_v} \mathbf{1} \left[|S_t^{\text{impact}}| > 0 \right], \\ r_v^{\text{ver}} &= \frac{\sum_{(t,j) \in \mathcal{E}_v} A_t^j}{m_v}, \\ \widehat{\text{VIG}}(v) &= \frac{1}{n_v} \sum_{(t,j) \in \mathcal{E}_v} \text{VIG}_t^j. \end{aligned}$$

Here, n_v measures support, m_v counts non-vacuous verification opportunities, r_v^{ver} measures how often the token passes verified compatibility, and $\widehat{\text{VIG}}(v)$ measures its average verified information gain. If $m_v = 0$, we set $r_v^{\text{ver}} = 0$. Since VIG_t^j already includes the factor A_t^j , failed or vacuous candidates contribute zero information gain.

Filtering rule. We retain a vocabulary token only if it has enough support, enough non-vacuous verification evidence, high verified-compatibility rate,

and sufficient average verified information gain:

$$P^{\text{plan}} = \left\{ v \in V \mid n_v \geq n_{\min}, m_v \geq m_{\min}, r_v^{\text{ver}} \geq \rho_{\text{ver}}, \widehat{\text{VIG}}(v) \geq \gamma_{\text{vig}} \right\}.$$

This rule selects tokens by measured behavior under the target model, rather than by manual semantic labels.

Practical Details. A separate planning vocabulary is constructed for each task family using 1,000 randomly sampled, unlabeled examples from non-evaluation data: the MATH training split for mathematical reasoning, the OpenCoder training corpus for code generation, and the auxiliary_train split of CAIS MMLU for knowledge tasks. Construction uses the fixed calibration band $[\tau_{\text{cal}}^{\ell}, \tau_{\text{cal}}^u] = [0.15, 0.25]$ and filtering thresholds $n_{\min} = 50$, $m_{\min} = 20$, $\rho_{\text{ver}} = 0.9$, and $\gamma_{\text{vig}} = 0.03$. Constructing one task-family-specific vocabulary incurs a one-time offline cost of approximately 0.8 H200 GPU-hours.

B.2 Why Prioritize Planning Tokens?

We evaluate two exploration variants—*Random* and *Planning*—under the same Semi-Autoregressive Blockwise Decoding framework introduced in Appendix A. At each decoding iteration, we proceed as follows.

1. **High-confidence commits.** We first commit all masked positions that satisfy the static FastdLLM unmasking rule (Wu et al., 2026b) with threshold $\tau_{\text{high}} = 0.9$.
2. **Low-confidence exploration within a confidence bin.** Let $[\tau^{\ell}, \tau^u]$ denote the active confidence range (bin). Among the remaining masked positions in the current active block whose top-1 posterior confidence lies in $[\tau^{\ell}, \tau^u]$, we commit exactly one additional token using one of the following strategies:

- (a) **Random.** Uniformly sample one eligible masked position (within the block) and commit its top-1 token.
- (b) **Planning.** We form the subset of eligible masked positions whose top-1 token is in $\mathcal{P}$ and whose confidence lies in $[\tau^{\ell}, \tau^u]$. If this subset is non-empty, we uniformly sample one element from it and commit its top-1 token. If it is empty, we fall back to the same Random rule above (sampling from all eligible positions in $[\tau^{\ell}, \tau^u]$).

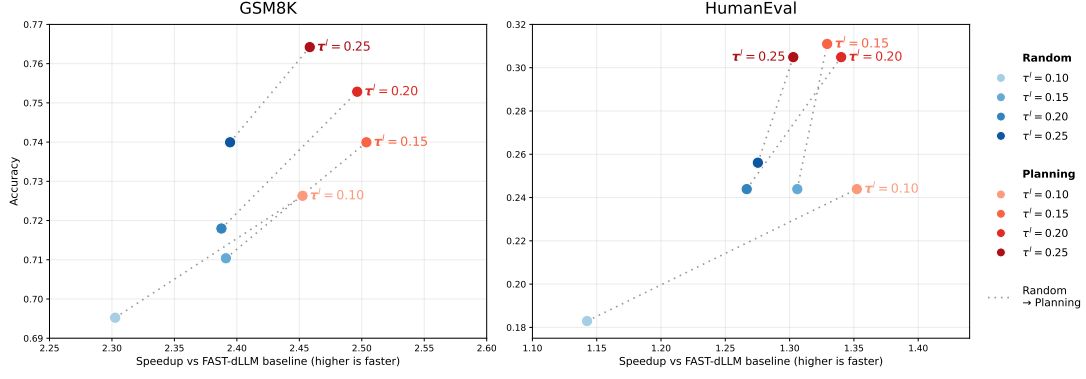


Figure 4: Controlled comparison on GSM8K and HumanEval between planning-token prioritization and confidence-matched random low-confidence commitments. Across the evaluated bins, Planning yields lower NFE and higher accuracy than Random; points closer to the upper right indicate better performance on both axes.

Table 4: Comparison of Planning versus Random token strategies in low-confidence regimes. The average confidence (*Avg Token Conf*) of the selected tokens is maintained at comparable levels to ensure a controlled comparison.

Dataset	τ^l	τ^u	Method	Acc.	NFE	Avg. Token Conf.
GSM8K	0.10	0.60	Planning	0.73	50.67	0.38
			Random	0.70	53.97	0.38
	0.15	0.60	Planning	0.74	49.63	0.40
			Random	0.71	51.97	0.39
	0.20	0.60	Planning	0.75	49.78	0.42
			Random	0.72	52.04	0.41
	0.25	0.60	Planning	0.76	50.55	0.43
			Random	0.74	51.89	0.44
HumanEval	0.10	0.60	Planning	0.24	66.38	0.37
			Random	0.18	78.57	0.36
	0.15	0.60	Planning	0.31	67.54	0.39
			Random	0.24	68.74	0.38
	0.20	0.60	Planning	0.30	66.99	0.41
			Random	0.24	70.87	0.40
	0.25	0.60	Planning	0.30	68.90	0.43
			Random	0.26	70.39	0.43

The fallback in the Planning variant is important for a fair comparison: since P^{plan} is deliberately small, the planning-eligible subset is sometimes empty (see Figure 7). Without a fallback, the Planning variant would make fewer low-confidence commits, confounding accuracy and runtime with a different number of exploration steps. By enforcing a matched exploration rate via fallback, improvements can be attributed to *which* low-confidence tokens are committed rather than *how many* are committed. Empirically, we adopt the static variant of Fast-DLLM (Wu et al., 2026b) as the base-

line. At each decoding step, in addition to the tokens selected by Fast-DLLM, we randomly commit one extra token whose confidence lies within a prescribed low-confidence interval $[\tau^l, \tau^u]$. In Figure 4, we fix $\tau^u = 0.6$ for simplicity. The results show that unrestricted low-confidence commitments, labeled *Random* in Figure 4, cause substantial performance degradation, whereas prioritizing our content-neutral planning token set P^{plan} for such commitments, labeled *Planning* in Figure 4, markedly reduces the accuracy loss while also improving efficiency. Additionally, Table 4

reports the average confidence of the extra lower-confidence tokens committed under both the **Random** and **Planning** strategies. Across all datasets, the average values differ by at most 0.01, indicating that the gains in efficiency and accuracy are driven by prioritizing commits to planning tokens, rather than due to a statistical bias whereby planning tokens exhibit systematically higher posterior confidence within the same confidence band.

Speedup metric. We quantify decoding speed by the number of function evaluations (NFE), i.e., the total number of forward passes. For each setting, we report speedup as

$$\text{Speedup} = \frac{\text{NFE}_{\text{Fast-dLLM}}}{\text{NFE}_{(\text{Random/Planning})}},$$

so that larger values indicate fewer forward passes and faster decoding.

Table 5: Sensitivity to planning-vocabulary size.

Dataset	Size	Accuracy	NFE	TPS
GSM8K	50	79.75	31.34	259.35
GSM8K	150	79.62	32.75	248.45
HumanEval	46	46.66	70.31	163.26
HumanEval	312	46.34	74.17	154.89

B.3 Robustness to Vocabulary Size

Since the planning vocabulary only prioritizes candidates for subsequent online verification, vocabulary mismatch primarily affects how efficiently PVF allocates its limited verification budget. When no eligible planning candidate is available, PVF instead uses the verified AR Fallback Route. To evaluate sensitivity to the vocabulary, nested enlarged sets are constructed by relaxing the calibration filters.

Relative to the compact vocabularies, expanding the GSM8K vocabulary from 50 to 150 tokens and the HumanEval vocabulary from 46 to 312 tokens changes accuracy by only 0.13 and 0.32 percentage points, respectively, while TPS decreases by 4.2% and 5.1%. PVF therefore retains most of its efficiency advantage with substantially larger planning vocabularies. For unknown or mixed-domain deployment, a pooled or union vocabulary calibrated from heterogeneous unlabeled data is a practical option. Because PVF verifies only the top K candidates, enlarging the vocabulary does not increase the verification width; when no reliable proposal

prior is available, the verified AR-only configuration provides a domain-agnostic fallback. These deployment options are not evaluated here.

B.4 Calibration Band and Planning Band

The calibration band and planning band serve different stages of PVF. The fixed calibration band $[\tau_{\text{cal}}^l, \tau_{\text{cal}}^u] = [0.15, 0.25]$ is used only during offline construction of P^{plan} , where it identifies vocabulary tokens that remain reliable and informative under deliberately challenging low-confidence conditions. The resulting vocabulary is then fixed before evaluation. During inference, the planning band $[\tau_{\text{plan}}^l, \tau_{\text{plan}}^u]$ determines which confidence values are eligible for planning proposals and therefore controls how aggressively P^{plan} is used. The two bands need not coincide because they serve different purposes, and every concrete proposal remains subject to online verification.

B.5 Planning-Band Selection and Sensitivity

For each benchmark, the lower and upper bounds of the planning band are selected from a fixed 0.05 grid using benchmark-matched non-evaluation data where available and OpenCoder for HumanEval. The selected planning band minimizes NFE subject to no observed accuracy degradation relative to Static decoding on the selection data. This selection protocol is separate from the label-free calibration procedure used to construct P^{plan} .

Figures 5 and 6 evaluate sensitivity by varying the lower and upper bounds of the planning band while fixing all other hyperparameters to the main-result settings. Across nearby settings, the favorable accuracy–efficiency trade-off persists, although lower values generally reduce NFE at the cost of more variable accuracy by admitting less-confident proposals.

We further examine the percentage of planning tokens relative to the total number of tokens accepted through the Planning and AR Fallback Routes (i.e., planning plus AR-fallback tokens) under varying τ_{plan}^l , while fixing $\tau_{\text{plan}}^u = 0.9$. Figure 7 shows that lower bounds increase the proportion of selected planning tokens, which typically leads to better algorithmic efficiency.

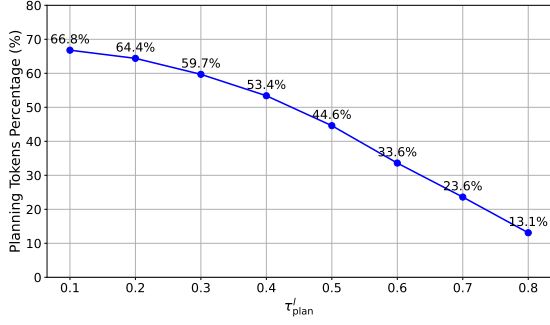


Figure 7: Impact of the planning-band lower bound on the percentage of selected planning tokens.

C Additional Evaluation Results

This appendix provides complementary evidence along three dimensions: comparison with recent verification-based decoders on LLaDA, expanded multiple-choice evaluation on both backbones, and realized throughput on Dream. Together, these experiments test whether PVF’s efficiency gains persist across decoding strategies, concise-output tasks, and DLM backbones.

C.1 Comparison with Verification-Based Decoders

Table 6 compares PVF with the recent verification-based DLM decoders SSD (Gao et al., 2025) and WINO (Hong et al., 2026). This comparison tests whether PVF’s efficiency advantage extends beyond confidence-threshold decoding and FreeDAVE to a broader set of verification-based methods. Under the same hardware and implementation settings, including KV caching and verification width, PVF requires fewer sequential model evaluations and achieves higher realized throughput at comparable numerical accuracy across all four LLaDA-8B-Instruct benchmarks.

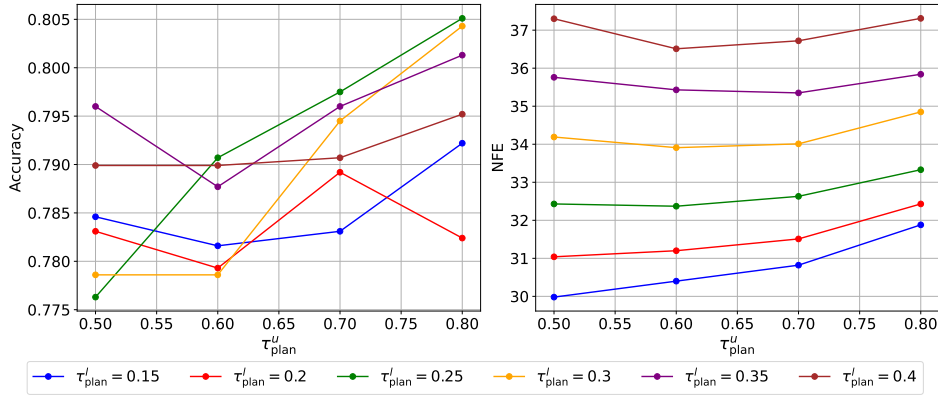


Figure 5: Sensitivity analysis of the planning band on GSM8K. Each curve fixes τ_{plan}^l and varies τ_{plan}^u .

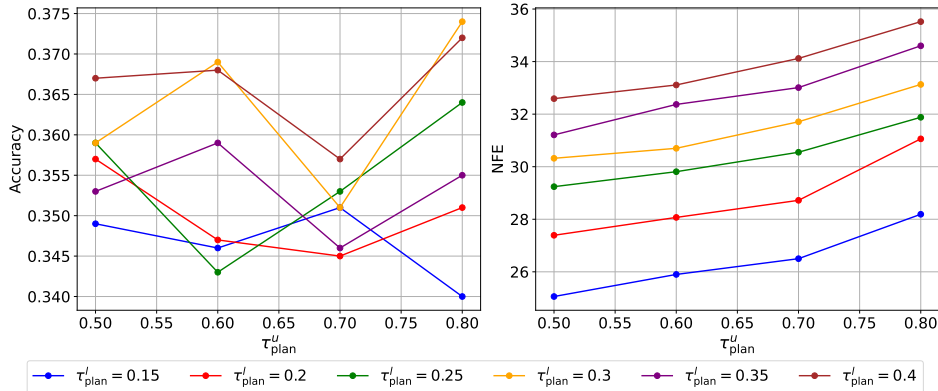


Figure 6: Sensitivity analysis of the planning band on MMLU-Pro. Each curve fixes τ_{plan}^l and varies τ_{plan}^u .

Table 6: Same-system comparison with verification-based decoders on LLaDA-8B-Instruct. All methods use the same hardware, KV-cache setting, and verification width.

Dataset	Method	Accuracy	NFE ↓	TPS ↑
GSM8K	PVF	79.75	31.34	259.35
	SSD	79.23	213.15	45.95
	WINO	77.15	67.75	124.66
MMLU-Pro	PVF	36.40	29.55	113.78
	SSD	26.50	213.17	21.97
	WINO	35.30	77.34	52.25
HumanEval	PVF	46.66	70.31	163.26
	SSD	45.73	216.99	57.80
	WINO	44.63	98.13	104.24
MATH	PVF	40.24	73.56	161.12
	SSD	40.20	218.68	55.38
	WINO	39.00	101.43	106.20

C.2 Additional Multiple-Choice Evaluation

To test whether PVF remains useful when the required output is concise, Table 7 extends the existing MMLU-Pro evaluation with ARC-Challenge and WinoGrande. Across both LLaDA-8B-Instruct and Dream-7B-Instruct, PVF reduces NFE by 53.8%–64.7% relative to Fast-dLLM while maintaining comparable multiple-choice accuracy.

Table 7: Multiple-choice results on MMLU-Pro, ARC-Challenge, and WinoGrande.

Dataset	Fast Acc.	Fast NFE	PVF Acc.	PVF NFE	Reduction
<i>LLaDA-8B-Instruct</i>					
MMLU-Pro	36.20	80.26	36.40	29.55	63.2%
ARC-Challenge	83.02	48.75	84.30	17.23	64.7%
WinoGrande	70.24	48.57	69.93	18.81	61.3%
<i>Dream-7B-Instruct</i>					
MMLU-Pro	40.18	92.16	43.30	35.22	61.8%
ARC-Challenge	86.78	56.13	87.55	25.93	53.8%
WinoGrande	70.65	43.01	70.81	16.92	60.7%

C.3 Throughput on Dream-7B-Instruct

Table 8 confirms that the reduction in sequential denoising rounds translates into wall-clock gains beyond LLaDA: on Dream-7B-Instruct, PVF achieves $1.6\times$ – $2.3\times$ higher TPS than Fast-dLLM across the four main benchmarks.

Table 8: TPS comparison on Dream-7B-Instruct.

Dataset	Fast-dLLM	FreeDave	PVF
GSM8K	88.20	38.40	153.10
MMLU-Pro	42.65	32.49	97.86
HumanEval	63.51	39.69	113.46
MATH	98.59	71.71	158.51

D Additional Efficiency and Robustness Analyses

This appendix reports the accuracy–efficiency frontier, robustness across generation lengths and block sizes, and sensitivity to verification batch size.

D.1 Pareto Frontier

Figure 8 plots the accuracy–NFE trade-off of PVF and Fast-dLLM under a sweep of the shared high-confidence threshold τ_{high} . For a fair comparison, we keep PVF-specific hyperparameters fixed and vary only the shared threshold used by both methods. Across four benchmark datasets, PVF consistently shifts the frontier toward lower NFE at comparable accuracy. This means that for the same accuracy target, PVF generally requires substantially fewer forward passes than Fast-dLLM. Conversely, for a fixed NFE budget, PVF can often operate at a higher-accuracy point. The advantage is most visible near the high-accuracy regime, where Fast-dLLM must use a conservative threshold and therefore commits fewer tokens per denoising step, while PVF can still make additional verified progress through planning and fallback routes. These results indicate that PVF improves the accuracy–efficiency frontier rather than merely tuning the confidence threshold more aggressively.

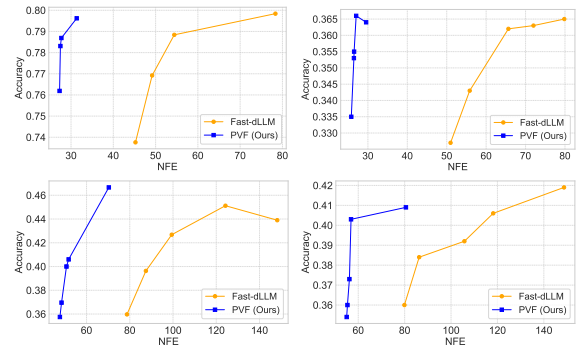


Figure 8: Pareto frontiers for PVF (blue) and Fast-dLLM (orange) on the GSM8K, MMLU-Pro, HumanEval, and Math datasets (from left to right) with LLaDA-8B-Instruct as the base model.

D.2 Sensitivity Analysis: Generation Length and Block Size

To evaluate the robustness of our approach, we benchmark PVF against the Fast-dLLM baseline across varying maximum generation lengths ($L \in \{256, 512\}$) and block sizes ($k \in \{16, 32, 64\}$) on the GSM8K dataset. As shown in Table 9, PVF consistently demonstrates a substantial improvement in algorithmic efficiency, achieving significantly

lower NFE than Fast-dLLM across all configurations while maintaining near-identical accuracy. Furthermore, we observe a positive correlation between block size and efficiency gains: as the block size increases, the reduction in NFE becomes more pronounced. This trend validates the efficacy of our planning mechanism, which leverages larger blocks to secure longer verified decoding trajectories, thereby minimizing the number of required forward passes more effectively.

Table 9: Comparison of PVF and Fast-dLLM performance across different Generation Lengths and Block Sizes on GSM8K for LLaDA-8B-Instruct.

Gen Length	Block Size	PVF		Fast-dLLM	
		Accuracy	NFE	Accuracy	NFE
256	16	76.46	37.76	76.30	77.73
	32	75.92	26.75	75.76	71.23
	64	75.84	24.52	75.69	67.85
512	16	79.53	58.34	80.07	110.32
	32	79.92	41.55	79.69	96.86
	64	79.62	31.34	79.62	89.77

D.3 Ablation Study: Sensitivity to Batch Size

To assess whether our algorithm retains efficiency even with more constrained batch sizes, we evaluate the NFE of our PVF method and its individual components as the batch size varies, ensuring that accuracy remains lossless relative to static decoding. In Figure 9, we observe that the planning stage delivers stable performance even at a batch size of 2, whereas the AR fallback benefits more significantly from larger batches due to its cumulative nature. Notably, the full PVF method consistently outperforms both individual components across all settings, and demonstrates substantial efficiency gains compared to the baseline method even at a batch size of 2.

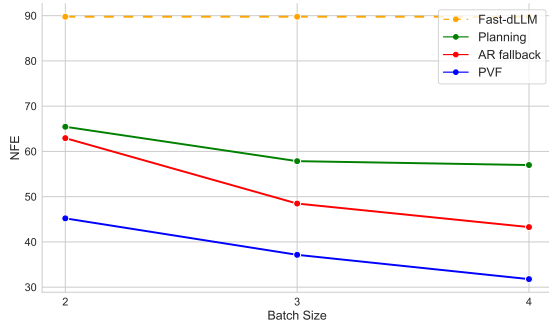


Figure 9: Ablation study on the full GSM8k dataset evaluating the impact of PVF components across varying batch sizes in the “free lunch” regime. All results maintain lossless accuracy compared to static decoding.

D.4 Effect of Verification

The complete verification ablation in Table 10 includes route-specific removal and force-accept controls. Verification has a modest but consistent effect on GSM8K and is critical on HumanEval: variants without AR verification fall from 46.66% to approximately 21% accuracy. Because the Planning and AR proposal routes remain active, removing verification yields little or no NFE improvement, showing that verification protects accuracy without materially weakening efficiency.

Table 10: Full verification ablation on LLaDA-8B-Instruct at $K = 3$.

Variant	GSM8K		HumanEval	
	Accuracy	NFE	Accuracy	NFE
Full PVF	79.75	31.34	46.66	70.31
No planning verification	78.13	36.72	44.51	70.27
No AR verification	78.45	27.58	20.73	67.34
No verification	76.00	33.19	20.73	67.03
Force-accept planning rejects	78.13	32.41	45.12	70.45
Force-accept AR rejects	77.21	31.78	21.95	69.18

D.5 Verification-Width Sensitivity

Table 11 reports the effect of increasing K from 1 to 3. Wider verification substantially reduces NFE and increases TPS at comparable accuracy on both tasks. Although rejection rates rise as more candidates are tested, accepted-token information gain also increases, indicating that the wider candidate budget finds more informative verified commitments.

Table 11: Verification-width sensitivity on LLaDA-8B-Instruct.

Dataset	K	Accuracy	NFE	TPS
GSM8K	1	79.46	45.83	187.60
	2	79.62	37.56	227.76
	3	79.75	31.34	259.35
HumanEval	1	46.95	102.36	116.26
	2	45.73	83.54	142.51
	3	46.66	70.31	163.26

Dataset	K	Plan Reject	AR Reject	Accepted Gain
GSM8K	1	7.94%	1.78%	0.169
	2	14.46%	6.58%	0.191
	3	18.65%	10.97%	0.203
HumanEval	1	5.89%	2.48%	0.044
	2	8.14%	9.36%	0.045
	3	10.18%	16.20%	0.047

D.6 Route Statistics

The statistics in Table 12 highlight the complementary behavior of the two routes. Planning makes a selective commitment whose use varies substantially by task, whereas the AR Fallback Route most

frequently accepts the full length-3 extension in the reported accepted-length distribution.

Table 12: Route-use and acceptance statistics.

Dataset	Plan Attempted	Filter-1 Accepted	Empty Impact
GSM8K	57.68%	54.97%	43.23%
HumanEval	14.09%	73.65%	48.69%
MATH	18.91%	71.89%	56.04%
MMLU-Pro	37.78%	41.12%	59.87%

Dataset	AR Length 0	AR Length 1	AR Length 2	AR Length 3
GSM8K	3.75%	15.81%	15.91%	64.54%
HumanEval	2.39%	19.78%	16.70%	61.13%
MATH	2.16%	22.12%	16.89%	58.83%
MMLU-Pro	2.15%	15.68%	15.68%	66.48%

D.7 Memory Overhead

Table 13 reports per-example peak allocated GPU memory under the same inference setting. At the default verification width $K = 3$, PVF increases mean peak allocated memory by 10.80% and maximum peak allocated memory by 11.87% relative to confidence-only decoding. The increase is moderate because model parameters are shared across branches, while K provides a direct memory–efficiency trade-off for constrained deployments.

Table 13: Peak allocated GPU-memory comparison at $K = 3$.

Memory Metric	Baseline (GiB)	PVF (GiB)	Overhead (GiB)
Mean peak allocated	15.974	17.699	+1.725 (+10.80%)
Maximum peak allocated	16.094	18.004	+1.910 (+11.87%)

E Additional Mechanism and Experimental Details

This appendix documents planning-guided early cross-block revelation together with the implementation, evaluation protocol, and hyperparameter settings used in the experiments.

E.1 Planning-Guided Early Cross-Block Revelation

While retaining block-wise efficiency, PVF discards the rigid requirement of strictly sequential convergence. Our motivation is structural: as the active block enters a *saturated regime* of mask scarcity, the constricted search space limits the planner’s optimization capacity. To counteract this, we adopt a *dynamic expansion strategy* that reveals the subsequent block preemptively when the current block’s sparsity hinders effective planning.

Consistent with recent approaches that relax rigid block boundaries (Wang et al., 2026; Fu et al., 2026), we allow for early cross-block revelation to facilitate information flow. However, to mitigate the accuracy degradation often observed with aggressive expansion (Wang et al., 2026), this strategy is implemented conservatively. Expansion is only triggered when the current block enters a saturated regime of mask scarcity, ensuring that the early decoding mechanism serves solely to maintain a valid planning horizon.

Formally, the block size is denoted as $S = L/B$. We identify the index of the current active block k_t as the first block containing masked tokens. Define the index set of masked tokens in the current active block $\mathcal{M}_t^{\text{active}} = \{i | y_{t-1}^i = [\text{MASK}]\} \cap \{S(k_t - 1) + 1, \dots, Sk_t\}$. We define the working index set $\mathcal{B}_t$ by checking if the remaining masks in the current block fall below a sparsity threshold N_s :

$$\mathcal{B}_t = \begin{cases} \mathcal{M}_t^{\text{active}} \cup \mathcal{I}_{k_t+1}, & \text{if } |\mathcal{M}_t^{\text{active}}| \leq N_s \\ \mathcal{M}_t^{\text{active}}, & \text{otherwise} \end{cases} \quad (4)$$

where $\mathcal{I}_{k_t+1} = \{k_t S + 1, \dots, (k_t + 1)S\}$ represents the full index range of the next block. This condition ensures that the planner always operates on a sufficiently broad horizon, seamlessly transitioning focus to $k + 1$ as block k_t stabilizes.

E.2 Additional Experimental Details

Experimental details. Across all datasets, we use a fixed maximum generation length of $L = 512$

and a block size of $B = 64$. Decoding is training-free (no finetuning, no additional supervision, and no external verifier). All experiments are conducted on a single NVIDIA H200 GPU.

Implementation and evaluation. All methods share the same prompt templates per dataset, the same stopping criterion, and the same maximum generation length. For HumanEval, we follow the standard unit-test protocol and report pass@1. For MATH and GSM8K, we use exact-match evaluation on the final answer. For MMLU-Pro, we report multiple-choice accuracy.

Hyper-parameters. Table 14 summarizes the hyperparameters used across benchmarks.

Benchmark	(L, B)	Planning Band	τ_{AR}^l	N_s
MMLU-Pro	(512, 64)	(0.2, 0.65)	0.1	5
GSM8K	(512, 64)	(0.2, 0.65)	0.1	5
HumanEval	(512, 64)	(0.8, 0.9)	0.3	0
MATH	(512, 64)	(0.8, 0.9)	0.3	0

Table 14: Report of Hyperparameters Used

The construction of P^{plan} , the relationship between the calibration and planning bands, and the planning-band selection protocol are detailed in Appendix B, particularly Sections B.4–B.5. For the Pareto sweeps in Fig. 8 on HumanEval and MATH, we additionally enforce a feasibility constraint that the planning range does not overlap with the high-confidence region $(\tau_{\text{high}}, 1)$, since overlapping these zones would make the roles of “planning” versus “high-confidence” commits ill-defined. Concretely, we set $(\tau_{\text{plan}}^l, \tau_{\text{plan}}^u) = (0.4, 0.6)$ for $\tau_{\text{high}} < 0.9$.

F Case Study

This appendix presents two example generations from LLaDA-8B-Instruct using Fast-dLLM and PVF. For PVF, tokens committed through the Planning Route are highlighted in yellow, while tokens committed through the AR Fallback Route are highlighted in pink.

LLaDA Prediction of Fast-dLLM (89 steps)

To determine the total number of hours Meredith spent writing articles over the three days, we need to follow these steps:

1. Calculate the number of articles she wrote on Tuesday.
2. Calculate the number of articles she wrote on Wednesday.
3. Find the total number of articles she wrote over the three days.
4. Calculate the total number of hours she spent writing the articles.

****Step 1: Calculate the number of articles she wrote on Tuesday.****

Meredith wrote 5 articles on Monday. On Tuesday, she wrote $\frac{2}{5}$ times more articles than on Monday. Therefore, the number of articles she wrote on Tuesday is:

$$\begin{aligned} & 5 + \frac{2}{5} \times 5 = 5 + 2 = 7 \end{aligned}$$

****Step 2: Calculate the number of articles she wrote on Wednesday.****

On Wednesday, she wrote twice the number of articles she wrote on Tuesday. Therefore, the number of articles she wrote on Wednesday is:

$$\begin{aligned} & 2 \times 7 = 14 \end{aligned}$$

****Step 3: Find the total number of articles she wrote over the three days.****

The total number of articles she wrote over the three days is the sum of the articles she wrote on Monday, Tuesday, and Wednesday:

$$\begin{aligned} & 5 + 7 + 14 = 26 \end{aligned}$$

****Step 4: Calculate the total number of hours she spent writing the articles.****

Each article takes an average of 4 hours to research and write. Therefore, the total number of hours she spent writing the articles is:

$$\begin{aligned} & 26 \times 4 = 104 \end{aligned}$$

Thus, the total number of hours she spent writing articles in the three days is $\boxed{104}$.

To determine the total number of hours Meredith spent writing articles over the three days, we need to follow these steps:

Step 1: Calculate the number of articles written on Tuesday.

- On Monday, Meredith wrote 5 articles.
- On Tuesday, she wrote $\frac{2}{5}$ times more articles than on Monday.
- To find the number of articles written on Tuesday, we calculate $\frac{2}{5}$ of 5:

$$\frac{2}{5} \times 5 = 2$$
- Therefore, the number of articles written on Tuesday is:

$$5 + 2 = 7$$

Step 2: Calculate the number of articles written on Wednesday.

- On Wednesday, she wrote twice the number of articles she wrote on Tuesday.
- On Tuesday, she wrote 7 articles.
- Therefore, the number of articles written on Wednesday is:

$$2 \times 7 = 14$$

Step 3: Calculate the total number of articles written over the three days.

- On Monday, she wrote 5 articles.
- On Tuesday, she wrote 7 articles.
- On Wednesday, she wrote 14 articles.
- The total number of articles written over the three days is:

$$5 + 7 + 14 = 26$$

Step 4: Calculate the total number of hours spent writing articles.

- Each article takes an average of 4 hours to research and write.
- Therefore, the total number of hours spent writing articles is:

$$26 \times 4 = 104$$

Thus, the total number of hours Meredith spent writing articles in the three days is $\boxed{104}$.

G Decoding Algorithm

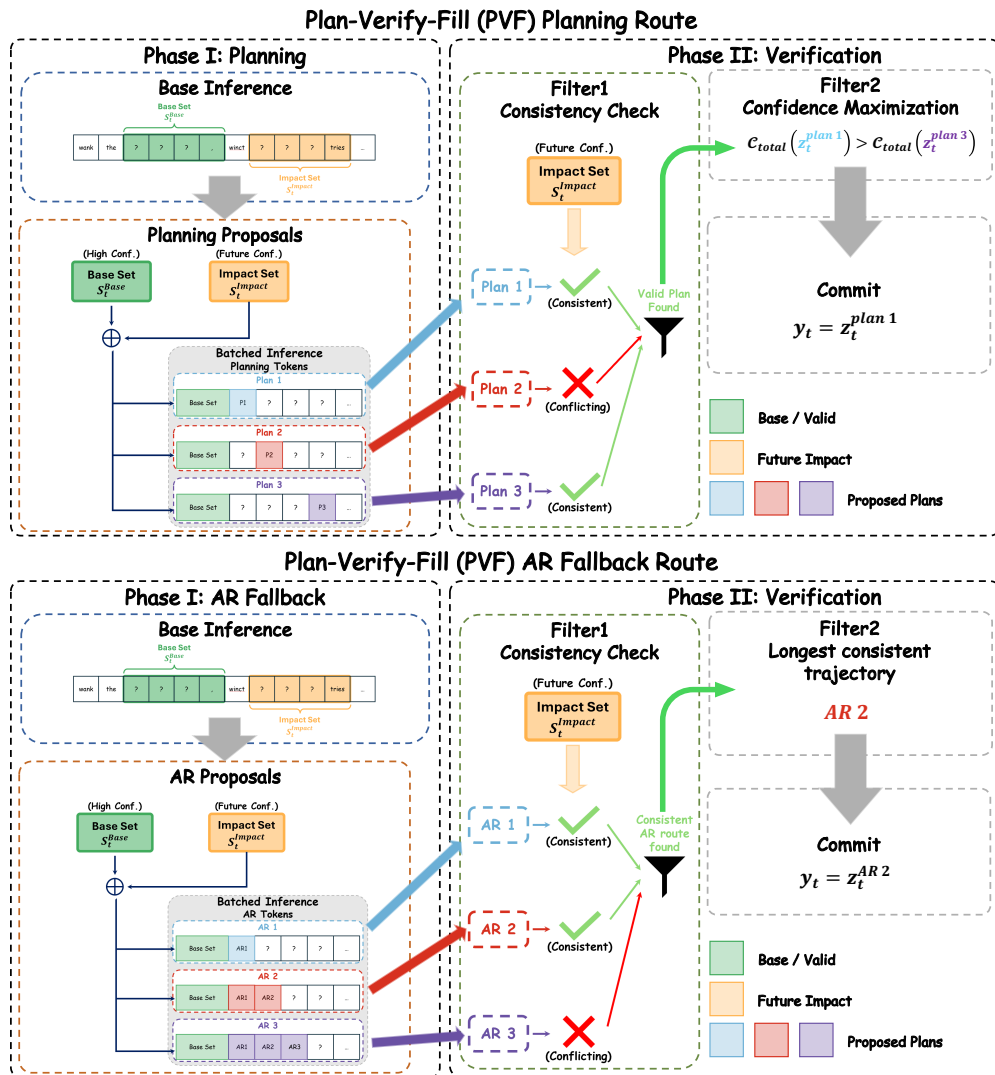


Figure 10: Overview of the **Planning Route** and the **AR Fallback Route**.

Algorithm 1 Plan-Verify-Fill (PVF) Decoding

```
1: Input: model  $P_\theta$ , length  $L$ , thresholds  $\tau_{\text{high}}, \tau_{\text{plan}}$ , width  $K$ 
2: Initialize:  $\mathbf{y}_0 \leftarrow [\text{MASK}]^L, t \leftarrow 0$ 
3: while  $\mathbf{y}_t$  contains  $[\text{MASK}]$  do
4:   Compute top predictions and confidences on the current active block.
5:   Construct the high-confidence base branch  $\mathbf{z}_t^{\text{base}}$ .
6:   Identify planning candidates  $\mathcal{P}_t^{\text{plan}}$ .
7:   Set  $\mathcal{J}_t^{\text{plan}} \leftarrow \emptyset$ .
8:   if  $\mathcal{P}_t^{\text{plan}} \neq \emptyset$  then
9:     Planning Route.
10:    Construct up to  $K$  planning branches  $\{\mathbf{z}_t^{\text{plan},j}\}_{j=1}^{K_t}$ .
11:    Run one batched forward pass on  $\{\mathbf{z}_t^{\text{base}}, \mathbf{z}_t^{\text{plan},1}, \dots, \mathbf{z}_t^{\text{plan},K_t}\}$ .
12:    Apply impact-set verification.
13:    Let  $\mathcal{J}_t^{\text{plan}}$  be the set of verified planning branches.
14:    if  $\mathcal{J}_t^{\text{plan}} \neq \emptyset$  then
15:      Select  $j^*$  among verified planning branches.
16:      Commit  $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{plan},j^*}$ .
17:    else
18:      Commit  $\mathbf{z}_t^{\text{base}}$ .
19:    end if
20:  else
21:    AR Fallback Route.
22:    Construct nested fallback branches  $\{\mathbf{z}_t^{\text{AR},k}\}_{k=1}^{K_t^{\text{AR}}}$ .
23:    Verify fallback branches by the local consistency.
24:    if no fallback branch passes verification then
25:      Commit  $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{base}}$ .
26:    else
27:      Let  $k^*$  be the largest verified fallback length.
28:      Commit  $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{AR},k^*}$ .
29:    end if
30:  end if
31:   $t \leftarrow t + 1$ 
32: end while
33: Return  $\mathbf{y}_t$ 
```
