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

Anonymous ACL submission

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 benchmarks spanning mathematical reasoning, knowledge, and code generation, PVF preserves final accuracy while reducing decoding cost. Measured on LLaDA-8B-Instruct, PVF achieves up to $2.3\times$ higher throughput than confidence-threshold decoding.

1 Introduction

DLMs (Gong et al., 2024; 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 (Khanna et al., 2025; Nie et al., 2025; Google DeepMind, 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., 2025b; 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., 2026; Fu et al., 2025). 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 tokens. 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., 2025).

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., 2025). 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., 2025a; Cheng et al., 2025; 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., 2025b), 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; Lou et al., 2023; Ou et al., 2024) 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 process is

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

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 $\mathbf{y}_{t-1}$:

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

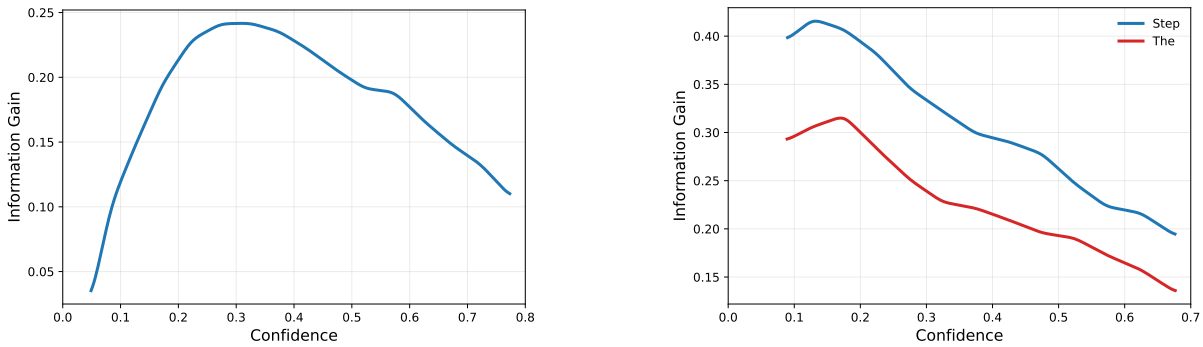
In its most canonical form, the algorithm progressively fills the sequence by committing to tokens where the model's confidence exceeds a static threshold τ_{high}:

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

While this static heuristic serves as a basic building block for parallel decoding, later approaches (Chen et al., 2025; 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

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

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

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

The base set and base branch are

$$S_t^{\text{base}} = \{i \in M_t \mid p_t^i \geq \tau_{\text{high}}\},$$

$$z_t^{i,\text{base}} = \begin{cases} \hat{y}_t^i, & i \in S_t^{\text{base}}, \\ y_{t-1}^i, & i \notin S_t^{\text{base}}. \end{cases}$$

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

$$S_t^{\text{impact}} = \left\{ i \in B_t \;\middle|\; \begin{array}{l} z_t^{i,\text{base}} = \texttt{[MASK]}, \\ \max_{w \in V} P_\theta(y^i = w \mid \mathbf{z}_t^{\text{base}}) \geq \tau_{\text{high}} \end{array} \right\}. \tag{1}$$

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:

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

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 $A_t^j \in \{0,1\}$:

$$A_t^j = 1 \iff |S_t^{\text{impact}}| > 0 \text{ and } \mathbf{z}_t^{\text{plan},j} \text{ preserves top-1 base predictions on } S_t^{\text{impact}}. \tag{3}$$

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

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

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

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

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 A.2.

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

We propose **Plan-Verify-Fill (PVF)** (Figure 3), 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.

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:

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

The lower bound filters highly unstable predictions, while the upper bound removes near-confident posi-

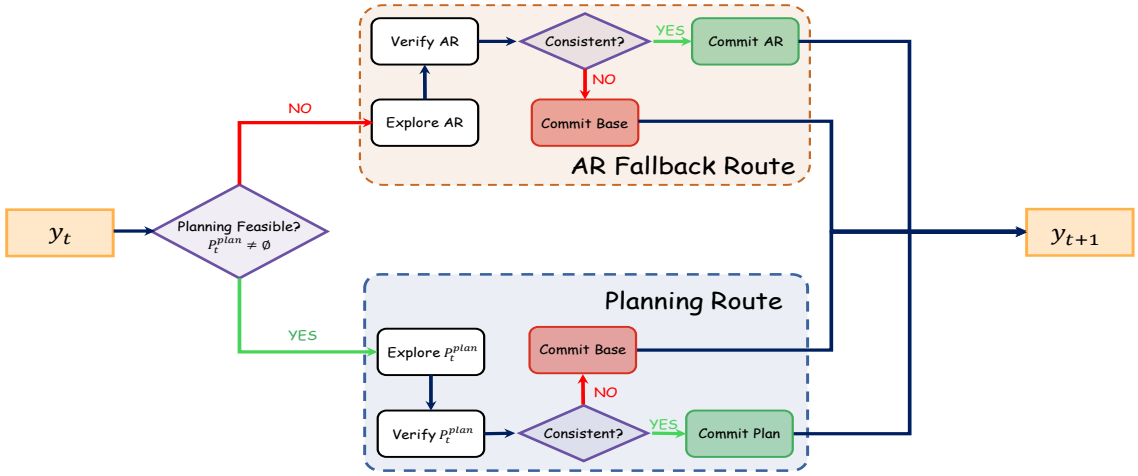


Figure 3: Overview of the **Plan–Verify–Fill (PVF)** decoding pipeline.

tions that offer limited additional information. The band therefore targets candidates with meaningful uncertainty but enough model support to be verification-worthy.

PVF keeps the top-K candidates by confidence:

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

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

$$\left\{\mathbf{z}_t^{\mathrm{base}}, \mathbf{z}_t^{\mathrm{plan},1}, \ldots, \mathbf{z}_t^{\mathrm{plan},K_t}\right\}.$$

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., 2025). If $P_t^{\mathrm{plan}} = \emptyset$, 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 token types 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 $\mathbf{z}_t^{\mathrm{plan},j}$, PVF computes the verification indicator A_t^j defined in Equation (3). Only verified candidates are preserved:

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

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 $\mathcal{J}_t^{\mathrm{plan}} = \emptyset$, PVF treats planning as premature for the current state, and commits $\mathbf{y}_t \leftarrow \mathbf{z}_t^{\mathrm{base}}$.

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:

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

PVF selects

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

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

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

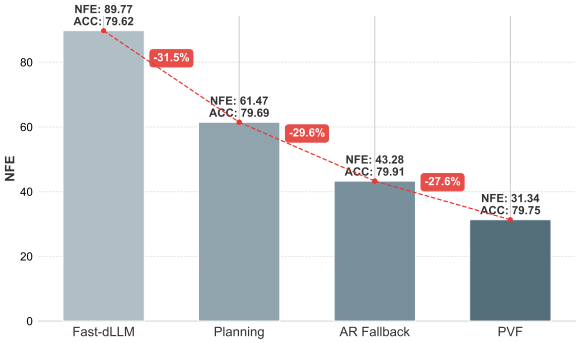


Figure 4: Ablation study on GSM8k, evaluating the contribution of each PVF component. Accuracy scores are displayed in the bracket above each bar to confirm they remain comparable across methods.

4.3 AR Fallback Route: Local Filling

4.3.1 AR Fallback Route

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

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

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

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

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

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

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

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., 2025b) as the realized throughput metric, allowing us to verify whether fewer forward passes translate into faster generation. To ensure a fair comparison, we fix $\tau_{\mathrm{high}} = 0.9$ across all experiments unless stated otherwise. Unless otherwise specified, we follow (Fu et al., 2025; Wu and Zhang, 2025) and use a batch size of 4 ($k = 3$), which lies within the "free lunch" region.

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 Section A.4. All experiments are conducted on a single NVIDIA H200 GPU. Additional experimental details are deferred to section A.6.

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., 2025b) 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. Additional experiment results are deferred to section A.3.

Accuracy and NFE. Table 1 summarizes accuracy and NFE across four benchmarks. PVF consistently reduces forward passes while preserving comparable accuracy. Relative to Fast-dLLM, PVF achieves a 1.8×–2.9× NFE speedup across datasets and model backbones, corresponding to roughly 43%–65% fewer forward passes. The accuracy changes remain small: on LLaDA-8B-Instruct, PVF is within one point of Fast-dLLM on all bench-

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.

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×)

marks, and on Dream-7B-Instruct, PVF improves accuracy on all four benchmarks. Compared with FreeDave, PVF also consistently uses fewer forward passes, showing that selective planning and AR fallback provide additional savings beyond draft-and-verify decoding alone.

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× 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.

Ablation study. Figure 4 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.

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 decoding 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 accelerator 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.

References

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. 2025. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. 2024. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv 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, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Venkat Srinivasan, and Jiantao Jiao. 2025. From bits to rounds: Parallel decoding with exploration for diffusion language models. *arXiv preprint arXiv:2511.21103*.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. 2025. Self speculative decoding for diffusion large language models. *arXiv preprint arXiv:2510.04147*.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

Google DeepMind. 2025. Gemini diffusion. Accessed: 2025.

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

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

Xiangqi Jin, Yuxuan Wang, Yifeng Gao, Zichen Wen, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2025. Thinking inside the mask: In-place prompting in diffusion llms. *arXiv preprint arXiv:2508.10736*.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, and 1 others. 2025. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 1.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. 2026. Klass: Kl-guided fast inference in masked diffusion models. *Advances in Neural Information Processing Systems*, 38:92267–92301.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2023. Discrete diffusion language modeling by estimating the ratios of the data distribution.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and 1 others. 2026. The flexibility trap: Why arbitrary order limits reasoning potential in diffusion language models. *arXiv preprint arXiv:2601.15165*.

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. *arXiv preprint arXiv:2502.09992*.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. 2025. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025a. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. 2025b. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren,

Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2025a. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025b. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.

Shutong Wu and Jiawei Zhang. 2025. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. *arXiv preprint arXiv:2510.00294*.

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. *arXiv 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.

A Appendix

A.1 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., 2025a,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 $\mathbf{y}_t = [y_t^{1:L/b}, y_t^{L/b+1:2(L/b)}, \ldots, y_t^{L-b+1:L}]$. 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.

A.2 Planning Token Set 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 $\mathcal{D}_{\mathrm{cal}}$ 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 $\mathbf{z}_t^{\mathrm{base}}$, 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:

$$\mathcal{C}_t^{\mathrm{cal}} = \left\{ i \in M_t \setminus S_t^{\mathrm{base}} \,\middle|\, \tau_{\mathrm{cal}}^{\ell} \le p_t^i < \tau_{\mathrm{cal}}^{u} \right\}.$$

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 $i \in \mathcal{C}_t^{\mathrm{cal}}$, 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 $\mathbf{z}_t^{\mathrm{plan},j}$ 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:

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

For each vocabulary token v, we compute four statistics:

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

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:

$$P^{\text{plan}} = \left\{ v \in V \;\middle|\; \begin{array}{ll} n_v \geq n_{\min}, & m_v \geq m_{\min}, \\ r_v^{\text{ver}} \geq \rho_{\text{ver}}, & \widehat{\text{VIG}}(v) \geq \gamma_{\text{vig}} \end{array} \right\}.$$

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

Calibration data and thresholds. We construct P^{plan} separately for each task family. For mathematical reasoning tasks, we sample from the MATH training split. For code-generation tasks, we sample from the OpenCoder training corpus. For MMLU-style knowledge tasks, we sample from the `auxiliary_train` split of CAIS MMLU. In all cases, we use $N_{\text{cal}} = 1000$ calibration examples, and labels are not used during calibration.

Unless otherwise stated, we use the calibration band

$$[\tau_{\text{cal}}^{\ell}, \tau_{\text{cal}}^{u}) = [0.15, 0.25),$$

and the filtering thresholds

$$n_{\min} = 50, m_{\min} = 20, \rho_{\text{ver}} = 0.9, \gamma_{\text{vig}} = 0.03.$$

The selected vocabulary is fixed before evaluation and is used only to restrict the planning proposal space. Online verified compatibility remains the final acceptance criterion for every planning commitment.

A.3 Additional Experiments

A.3.1 *Planning* Tokens Beyond High-Confidence Decoding

We evaluate two exploration variants—*Random* and *Planning*—under the same Semi-Autoregressive Blockwise Decoding framework introduced in Section A.1. At each decoding iteration, we proceed as follows.

1. **High-confidence commits.** We first commit all masked positions that satisfy the static Fast-dLLM unmasking rule (Wu et al., 2025b) with threshold $\tau_{\text{high}} = 0.9$.

2. **Low-confidence exploration within a confidence bin.** Let $[\tau^l, \tau^u]$ denote the active confidence range (bin). Among the remaining masked positions in the current active block whose top-1 posterior confidence lies in $[\tau^l, \tau^u]$, 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 $[\tau^l, \tau^u]$. 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 $[\tau^l, \tau^u]$).

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 9). 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., 2025b) as the baseline. 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 6, we fix $\tau^u = 0.6$ for simplicity. The

Table 3: 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

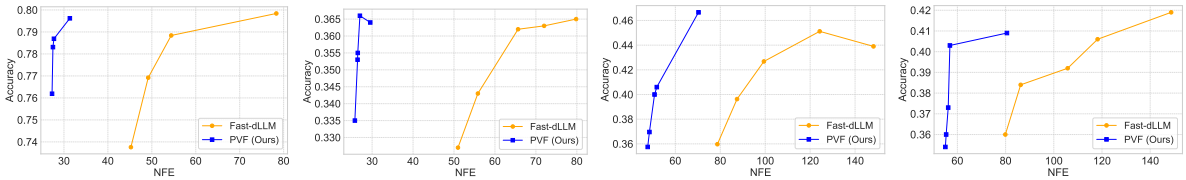


Figure 5: Pareto frontiers for PVF (blue) and Fast-dLLM (orange) on the GSM8K, MMLU-Pro, HumanEval, and Math datasets (from left to right) with LLaDA-8B-Instruct as the base model.

results show that unrestricted low-confidence commitments, labeled *Random* in Figure 6, cause substantial performance degradation, whereas prioritizing our content-neutral planning token set P^{plan} for such commitments, labeled *Planning* in Figure 6, markedly reduces the accuracy loss while also improving efficiency.

Additionally, Table A.2 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.

A.3.2 Pareto Frontier

Figure 5 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

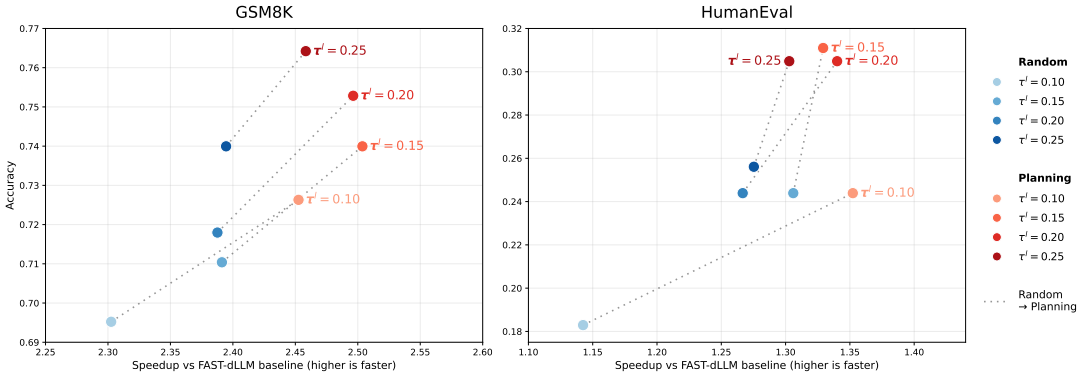


Figure 6: Ablation on GSM8K and HumanEval comparing lower-confidence commits of planning tokens versus random tokens (i.e., without prioritizing planning tokens). Across confidence bins, prioritizing planning tokens consistently yields faster decoding and improved accuracy; points closer to the upper-right indicate better performance on both axes.

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.

A.3.3 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 4, 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.

A.3.4 Sensitivity Analysis: Planning token thresholds τ^l_{plan}, τ^u_{plan}

We further investigate the sensitivity of PVF to the planning-token thresholds, specifically the lower bound τ^l_{plan} and upper bound τ^u_{plan}. To ensure a fair comparison, we fix all other hyperparameters to match those used in Table 1 for both datasets. Across the range of values we test, accuracy remains largely stable, indicating that the planning-token procedure is reasonably reliable. We nonetheless observe a mild trade-off between algorithmic efficiency and generation quality: lowering both threshold bounds consistently reduces NFE (improving efficiency) but leads to a corresponding minor decline in accuracy. This behavior occurs because lowering the thresholds effectively forces the model to commit to planning tokens with lower confidence. While this aggressive commitment typically provides a stronger contextual signal that boosts the confidence of subsequent tokens, which accelerates the decoding process, it simultaneously increases the chance of making mistakes.

We further examine the percentage of planning tokens relative to the total token count (defined as the sum of planning and AR fallback tokens) under varying lower bound thresholds τ^l_{plan}, while fixing the upper bound at $\tau^u_{\text{plan}} = 0.9$. Figure 9 illustrates this relationship, showing that a lower threshold results in a higher proportion of selected planning tokens, which in turn typically leads to better algorithmic efficiency.

A.4 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

Table 4: 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

components as the batch size varies, ensuring that accuracy remains lossless relative to static decoding. In Figure 10, 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.

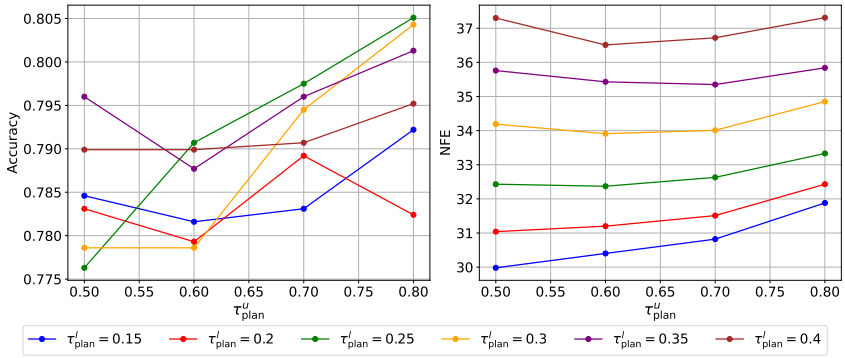


Figure 7: Sensitivity analysis of planning-token thresholds on GSM8K. Each curve corresponds to a fixed lower bound τ^l_{plan} and shows accuracy and NFE as τ^u_{plan} varies.

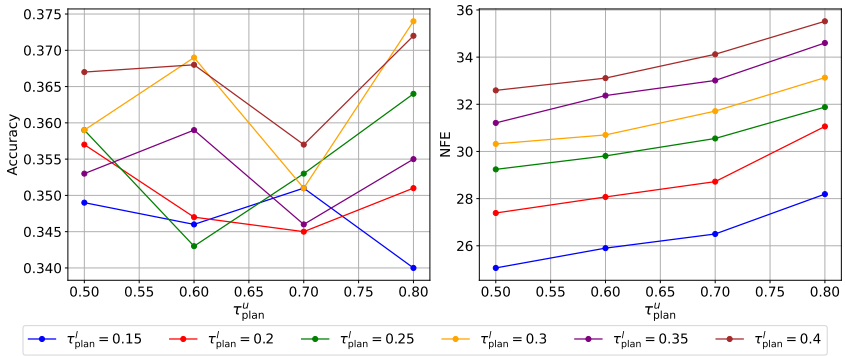


Figure 8: Sensitivity analysis of planning-token thresholds on MMLU-Pro. Each curve corresponds to a fixed lower bound τ^l_{plan} and shows accuracy and NFE as τ^u_{plan} varies.

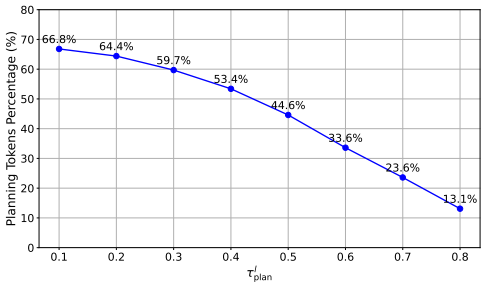


Figure 9: Impact of the planning token threshold lower bound (τ^l_{plan}) on the percentage of selected planning tokens.

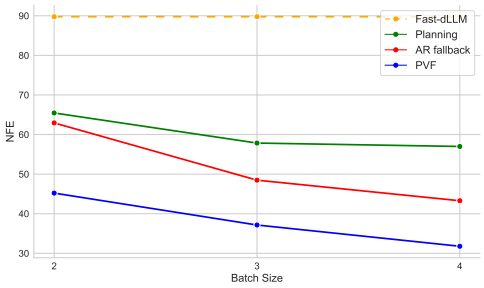


Figure 10: Ablation study on the full GSM8k dataset evaluating the impact of PVF components across varying batch sizes in the "free lunch" regime. All results maintain lossless accuracy compared to static decoding.

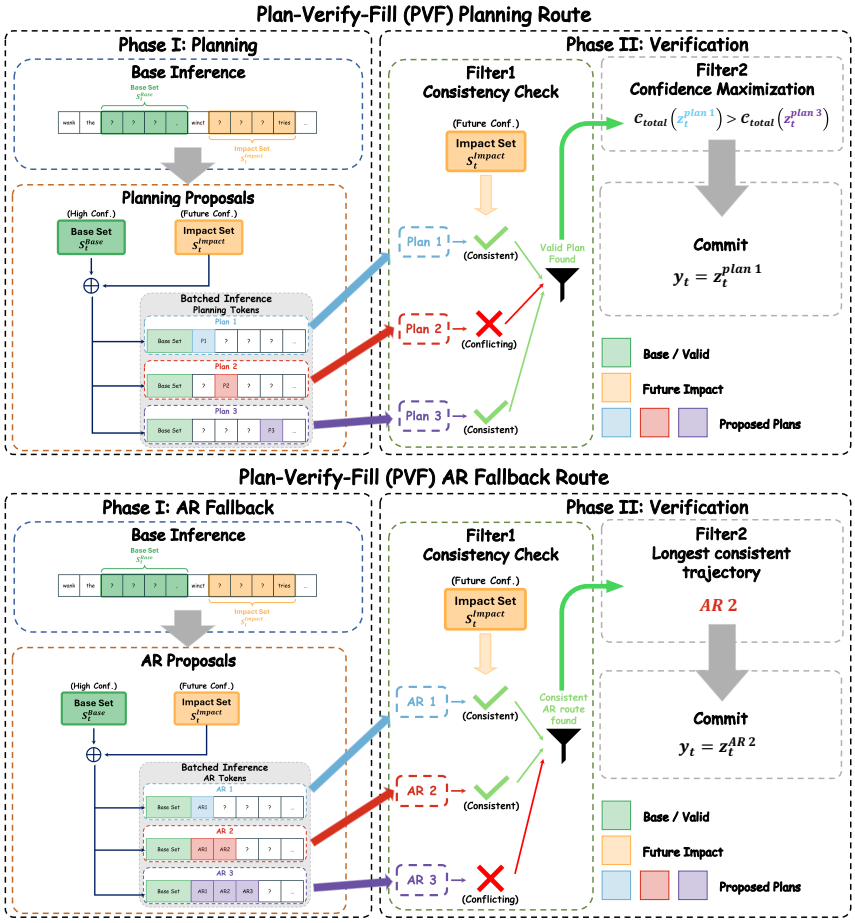


Figure 11: Overview of the **Planning Route** and the **AR Fallback Route**.

A.5 Additional Details of PVF

A.5.1 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., 2025b; Fu et al., 2025), 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., 2025b), 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 $\mathcal{M}_t^{\text{active}} = \{i | y_{t-1}^i = \texttt{[MASK]}\} \bigcap \{S(k_t - 1) + 1, \ldots, Sk_t\}$. 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}:

$$\mathcal{B}_t = \begin{cases} \mathcal{M}_t^{\text{active}} \bigcup \mathcal{I}_{k_t+1}, & \text{if } \left|\mathcal{M}_t^{\text{active}}\right| \leq N_{\text{s}} \\ \mathcal{M}_t^{\text{active}}, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{I}_{k_t+1} = \{k_t S + 1, \ldots, (k_t + 1)S\}$ 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.

A.6 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 5 summarizes the hyperparameters used across benchmarks.

Benchmark	**Base Budget** (L, B)	**Planning Thresholds** $(\tau^l_{\text{plan}}, \tau^u_{\text{plan}})$	**AR Fallback Threshold** τ^l_{AR}	**Early Cross-block Threshold** N_s
MMLU-Pro	L=512, B=64	(0.2, 0.65)	0.1	5
GSM8K	L=512, B=64	(0.2, 0.65)	0.1	5
HumanEval	L=512, B=64	(0.8, 0.9)	0.3	0
MATH	L=512, B=64	(0.8, 0.9)	0.3	0

Table 5: Report of Hyperparameters Used

For the Pareto sweeps in Fig. 5 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 $(\tau^l_{\text{plan}}, \tau^u_{\text{plan}}) = (0.4, 0.6)$ for $\tau_{\text{high}} < 0.9$.

A.7 Case study

In this section, we present two example generations from LLaDA-8B-Instruct using Fast-dLLM and our PVF strategy. For PVF, we highlight tokens committed via the planning route in yellow and tokens committed via the AR fallback route 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:
\[
5 + \frac{2}{5} \times 5 = 5 + 2 = 7
\]

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:
\[
2 \times 7 = 14
\]

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:
\[
5 + 7 + 14 = 26
\]

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:
\[
26 \times 4 = 104
\]

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

LLaDA Prediction with PVF (36 steps)

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 \(\boxed{104}\).

Algorithm 1 Plan-Verify-Fill (PVF) Decoding

Input: model P_θ, length L, thresholds $\tau_{\text{high}}, \tau_{\text{plan}}$, width K
Initialize: $\mathbf{y}_0 \leftarrow [\texttt{MASK}]^L$, $t \leftarrow 0$
while $\mathbf{y}_t$ contains `[MASK]` **do**
  Compute top predictions and confidences on the current active block.
  Construct the high-confidence base branch $\mathbf{z}_t^{\text{base}}$.
  Identify planning candidates $\mathcal{P}_t^{\text{plan}}$.
  Set $\mathcal{J}_t^{\text{plan}} \leftarrow \emptyset$.
  if $\mathcal{P}_t^{\text{plan}} \neq \emptyset$ **then**
    Planning Route.
    Construct up to K planning branches $\{\mathbf{z}_t^{\text{plan},j}\}_{j=1}^{K_t}$.
    Run one batched forward pass on $\{\mathbf{z}_t^{\text{base}}, \mathbf{z}_t^{\text{plan},1}, \ldots, \mathbf{z}_t^{\text{plan},K_t}\}$.
    Apply impact-set verification.
    Let $\mathcal{J}_t^{\text{plan}}$ be the set of verified planning branches.
    if $\mathcal{J}_t^{\text{plan}} \neq \emptyset$ **then**
      Select j^* among verified planning branches.
      Commit $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{plan},j^*}$.
    else
      Commit $\mathbf{z}_t^{\text{base}}$.
    end if
  else
    AR Fallback Route.
    Construct nested fallback branches $\{\mathbf{z}_t^{\text{AR},k}\}_{k=1}^{K_t^{\text{AR}}}$.
    Verify fallback branches by the local consistency.
    if no fallback branch passes verification **then**
      Commit $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{base}}$.
    else
      Let k^* be the largest verified fallback length.
      Commit $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t^{\text{AR},k^*}$.
    end if
  end if
  $t \leftarrow t + 1$
end while
Return $\mathbf{y}_t$