SUPPORTING TOKENS FIRST: DIRICHLET PRIOR-GUIDED DECODING FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unlike auto-regressive language models, diffusion language models (DLMs) iteratively refine masked sequences and can reveal multiple positions in parallel, offering the potential for faster generation. However, unresolved positions can remain conditionally dependent, making parallel decoding potentially unreliable. Explicitly modelling these dependencies through latent-variable inference is computationally expensive, while existing DLM decoding strategies primarily rely on instance-level signals, such as model confidence or attention, which can be noisy at capturing a token's influence on subsequent predictions. To address these limitations, we hypothesise the existence of supporting tokens that, when revealed early, reduce conditional dependence among the remaining masked positions. Under an idealised Bayesian latent-variable model, we show that prioritising tokens with greater prior support can reduce residual dependence in expectation. Building on this insight, we propose Prior-Guided Diffusion Decoding (PGDD), a training-free framework that exploits stable, population-level vocabulary statistics to identify candidate supporting tokens without costly latent-variable inference. Specifically, an answer-free Dirichlet Prior Bank captures query-conditioned vocabulary priors from an external corpus, which are combined with the DLM's current predictions through a Dirichlet posterior update to guide token commitment. By complementing noisy instance-level confidence with population-level prior information, PGDD prioritises supporting tokens, reducing residual conditional dependence and enabling more reliable parallel decoding. Empirical analysis confirms that higher-prior tokens tend to yield greater confidence gains at other masked positions early in denoising, while prior-guided ordering reduces residual conditional dependence. Experiments on mathematical reasoning, code generation, and planning demonstrate improved generation quality and decoding efficiency with limited computational overhead compared with recent baselines.

1 INTRODUCTION

Diffusion language models (DLMs) have the potential for faster and more flexible reasoning than auto-regressive counterparts, as they predict and reveal multiple positions and tokens in parallel (Nie et al., 2025; Sahoo et al., 2024; Ye et al., 2025b). However, this flexibility has been proved to bring harm to model capability during both training and inference time sampling (Ni et al., 2026; Wu et al., 2025). In particular, positions decoded in parallel are not necessarily independent, and unresolved tokens may still depend on information from one another. Decoding such conditionally dependent tokens together can introduce factorization errors and lead to suboptimal generation performance (Wu et al., 2025; Chen et al., 2026a; Dmitriev et al., 2026).

To mitigate errors caused by conditional dependencies in DLM sampling, existing work schedules decoding using model-internal signals such as token probability (Cai & Li, 2026; Wu et al., 2025) and attention scores (Li et al., 2026). While both approaches improve empirical sampling efficiency, they rely primarily on information extracted from the current inference instance. Confidence-based criteria can be sensitive to instance-level predictive uncertainty, whereas attention-based approaches may require additional training. More importantly, these signals do not directly indicate which tokens are most useful for resolving conditional dependencies among the remaining masked posi-

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

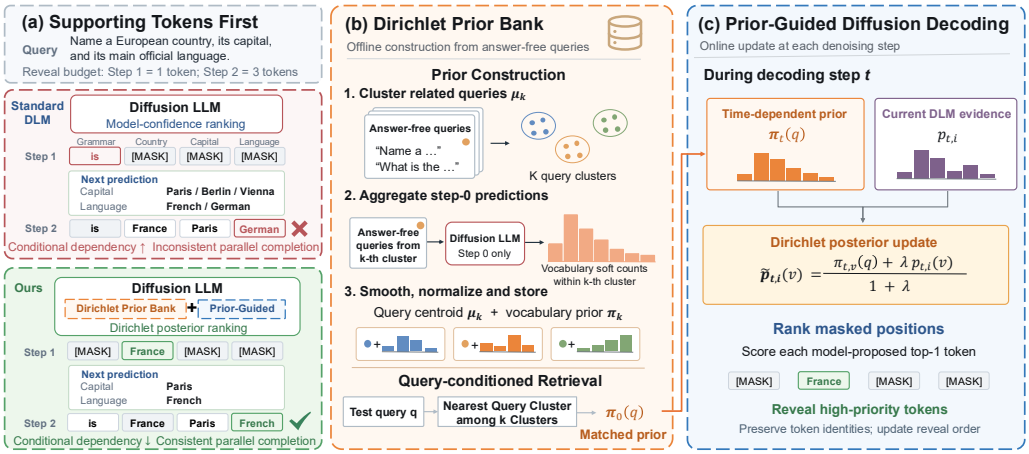


Figure 1: **Overview of prior-guided diffusion decoding. (a)** Supporting Tokens First. An illustrative example shows how revealing *France* before *is* supports consistent parallel completion under the same budget. **(b)** Dirichlet Prior Bank. Cluster-level vocabulary priors are constructed from step-0 predictions on answer-free queries and retrieved for each test query. **(c)** Prior-Guided Diffusion Decoding. A Dirichlet posterior update combines the annealed prior with DLM evidence to guide commitment order while preserving model-proposed token identities.

tions. Prediction confidence describes how certain the model is about a token at a given position, but a highly confident token is not necessarily the most informative one for subsequent predictions.

In this paper, we introduce a Bayesian latent-variable view of DLM decoding and propose a training-free simple yet effective decoding scheme. The premise is that a substantial fraction of token dependencies is mediated by shared latent structure. Rather than explicitly inferring this latent structure, we consider a set of *supporting tokens* that are disproportionately informative about this structure. Decoding such tokens with higher priority can reduce uncertainty over the latent state, weaken residual dependencies among the remaining positions, and consequently reduce the factorization error incurred by parallel sampling.

To identify the supporting tokens, we use an external, task-related corpus of answer-free queries to estimate a population-level vocabulary prior, capturing regularities that are unavailable from the current decoding trajectory alone. At inference time, this prior is fused with the DLM's contextual predictions to prioritize informative supporting positions during decoding. The resulting procedure performs implicit latent-state inference before aggressive parallel decoding, directly targeting the conditional dependencies that limit DLM sampling efficiency.

Our main contributions can be summarized as follows.

- We introduce a *Bayesian view of supporting tokens*, hypothesising that prioritising informative tokens can reduce conditional dependence among unresolved positions. Under an idealised latent-variable model, we show that prior-weighted token selection reduces residual dependence in expectation, motivating an efficient alternative to explicit latent-variable inference.
- We propose *Prior-Guided Diffusion Decoding* (PGDD, Fig. 1), a training-free framework that exploits stable, population-level vocabulary statistics to identify candidate supporting tokens. An answer-free *Dirichlet Prior Bank (DPB)* provides query-conditioned priors, which are combined with instance-level DLM predictions through a Dirichlet posterior update to guide reveal order, without additional model forward passes or retraining.
- We empirically validate the supporting-token hypothesis, showing that higher-prior tokens yield greater confidence gains early in denoising and that prior-guided decoding reduces residual conditional dependence. Across mathematical reasoning, code generation, and planning tasks, PGDD improves generation quality and decoding efficiency with limited computational overhead.

2 Related Work

Diffusion Language Models and Decoding. Diffusion language models generate text by iteratively denoising masked tokens rather than through left-to-right factorization (Nie et al., 2025; Ye

et al., 2025b; Sahoo et al., 2024). Because multiple positions are predicted and revealed in parallel, the choice of which positions to commit at each step becomes an important part of inference. Existing decoding strategies include fixed-count unmasking (Nie et al., 2025), confidence-threshold decoding (Liang et al., 2026), entropy-based selection (Ye et al., 2025b), margin-based selection (Li et al., 2025), and stability-based commitment (Zou et al., 2026; Chen et al., 2026b). Recent work also explicitly considers dependencies among positions during parallel decoding. PUNT (Azangulov et al., 2026) uses contextual independence tests to identify positions that can be safely unmasked together and removes lower-confidence positions from conflicting groups, while CLAD (Qi et al., 2026) uses attention-guided dependencies among high-confidence token clusters to coordinate parallel commitment. Other work improves inference efficiency using delayed KV caching (Ma et al.; Liu et al., 2025), block-wise approximate KV caching with confidence-aware parallel decoding (Wu et al., 2025), adaptive parallel decoding (Israel et al., 2025), and self-distillation to improve parallel decoding efficiency (Xu et al., 2026). These methods primarily derive reveal decisions from signals available at the instance level and not necessarily stable. Our work instead uses an external, query-conditioned vocabulary prior to prioritize more informative tokens and reduce residual dependence during denoising.

Training-Free Inference and Bayesian Priors. Language model predictions can be augmented at inference time without updating model parameters, for example through continuous caches or nearest-neighbour retrieval to provide richer contextual information (Grave et al., 2016; Khandelwal et al., 2019; Shi et al., 2024). In contrast, our method stores domain-specific information as a vocabulary-level prior to alleviate the memory usage of additional contextual tokens and reduce DLM inference-time errors resulting from conditional dependencies. We use Dirichlet priors as a representation for such information, where prior mass can be interpreted as pseudo-counts and combined with categorical distributions during DLM inference through a Bayesian update (Zhai & Lafferty, 2004; Malinin & Gales, 2018).

3 METHOD

Fig. 1 provides an overview of PGDD, a training-free decoding framework that uses a query-conditioned vocabulary prior constructed from answer-free queries to guide the reveal order of supporting tokens while preserving the model's token predictions and parameters. The following sections introduce the Bayesian view of supporting tokens (Sec. 3.1), the answer-free DPB (Sec. 3.2), and prior-guided diffusion decoding (Sec. 3.3).

3.1 A BAYESIAN VIEW OF SUPPORTING TOKENS IN DLMS

Masked Diffusion Language Models. Masked diffusion language models generate text by iteratively denoising a masked answer region and progressively revealing its tokens (Nie et al., 2025; Ye et al., 2025b). Let q denote an input query, `[MASK]` the mask token, and L the generation length. The initial sequence is

$$\mathbf{x}_0 = q \oplus \texttt{[MASK]}^L. \tag{1}$$

At denoising step t, let W_t denote the set of revealed answer positions and M_t the remaining masked positions. For each unresolved position $i \in M_t$, the model predicts a distribution over the vocabulary $\mathcal{V}$, conditioned on the query and the currently revealed answer tokens

$$p_{t,i}(v) = p_\theta\left(X_i = v \mid q, X_{W_t}\right), \qquad v \in \mathcal{V}, \tag{2}$$

where $\mathcal{V}$ denotes the vocabulary and $V = |\mathcal{V}|$ its size.The decoder then determines which masked positions are revealed at the current step. Existing DLM decoders use rules such as random position selection or model confidence to determine the reveal order (Nie et al., 2025; Ye et al., 2025b; Wu et al., 2025). For example, confidence-based decoding prioritizes positions whose current token predictions have high confidence, while uncertain positions remain masked for later denoising. Since each newly revealed token becomes part of the conditioning context for subsequent predictions, the reveal order can change the conditional dependencies among the remaining masked positions.

For a selected position i, the token identity is still determined by the original DLM distribution. Under top-1 decoding,

$$\hat{v}_{t,i} = \arg\max_{v \in \mathcal{V}} p_{t,i}(v), \qquad x_{t+1,i} = \hat{v}_{t,i}. \tag{3}$$

Selected positions are committed, while the remaining positions stay masked and are predicted again at the next denoising step. The decoding problem is therefore not only which token the DLM predicts, but also which predicted positions should be committed first.

Conditional dependence and supporting tokens. Although diffusion decoding allows multiple masked positions to be committed in parallel, the predictions at unresolved positions can still be conditionally dependent. For two positions $j, k \in M_t$, we quantify how much revealing $X_j = x_j$ changes the predictive distribution at position k by

$$C_{j\to k}(W_t) = D_{\mathrm{KL}}\left(p_\theta\left(X_k \mid q, X_{W_t}, X_j = x_j\right) \,\|\, p_\theta\left(X_k \mid q, X_{W_t}\right)\right). \tag{4}$$

A larger value indicates stronger conditional dependence between the two positions. If dependent positions are committed in parallel from the same model prediction, the prediction at k cannot incorporate the information provided by x_j, which can introduce factorization errors and degrade generation quality (Ni et al., 2026; Li et al., 2026).

Classical latent-variable models address such dependence by introducing hidden variables, followed by inference or marginalization over the latent variables Dempster et al. (1977); Rabiner (1989), often at substantial computational cost. Our paper aims to address such dependence in the tokenization representation space to avoid extra inference latency. The core research question in our work is, *whether there exist tokens that, when decoded early, reduce the conditional dependence among the remaining unresolved positions and therefore reduce decoding latency.* We refer to tokens that provide useful information for other unresolved positions as *supporting tokens*.

Standard DLM decoding selects early tokens mainly from the model's current confidence scores. However, confidence only reflects the current prediction and does not capture which tokens are generally informative for the task (Ni et al., 2026; He et al., 2026). Such decoding rules therefore lack an explicit mechanism for identifying and prioritizing supporting tokens. We therefore hypothesis that token frequency in an external, answer-free corpus can provide a complementary and robust task-level signal. Tokens that repeatedly appear across related examples are more likely to represent recurring semantic or structural patterns of the task; once revealed, such tokens can constrain the plausible values of other unresolved positions. More formally, we define the Dirichlet vocabulary prior to represent this task-level vocabulary support as below.

Dirichlet vocabulary prior. Let $\boldsymbol{\vartheta} \in \Delta^{V-1}$ denote a categorical distribution over the vocabulary, where ϑ_v denotes the probability assigned to token v within a related group of queries. We place a Dirichlet prior on $\boldsymbol{\vartheta}$,

$$\boldsymbol{\vartheta} \sim \mathrm{Dir}(\boldsymbol{\alpha}), \tag{5}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_V)$ is the positive parameter vector of the Dirichlet distribution, and α_v represents prior support for vocabulary item v. As the conjugate prior to the categorical likelihood, the Dirichlet distribution incorporates token evidence through closed-form additive updates. Observing token counts $\mathbf{c}$ updates the parameters from $\boldsymbol{\alpha}$ to $\boldsymbol{\alpha} + \mathbf{c}$. Thus, α_v acts as a prior pseudo-count for token v, without necessarily corresponding to an integer number of observations. Further derivations of Dirichlet conjugacy and the posterior predictive distribution are provided in Appx. A.1.

Under an idealized token generation model, we show that prior-weighted selection favors tokens that are more informative about the shared latent structure and can reduce residual dependence in expectation in Appx. A.2. Empirically, the next subsection introduces a task-related Dirichlet prior bank constructed from an external corpus, which encourages prior-supported predictions to serve as supporting tokens during DLM inference.

3.2 Answer-Free Dirichlet Prior Bank

The Bayesian formulation above leaves one practical question: *how to estimate an informative vocabulary prior for a test query without access to its answer*. We address this by constructing a *Dirichlet Prior Bank* from an external corpus of answer-free queries. The bank stores query-conditioned vocabulary priors that are retrieved to steer inference-time generation.

Prior construction. Let $\mathrm{D}_u = \{q_m\}_{m=1}^{M}$ denote the answer-free corpus. A query q_m is represented by averaging its input token embeddings from the DLM, denoted as $E(q_m)$. The resulting embeddings are grouped into K clusters using unsupervised k-means clustering, where $c_m \in \{1, \ldots, K\}$ denotes the cluster assignment of q_m and $\boldsymbol{\mu}_k$ the centroid of cluster k.

The vocabulary prior for each cluster is estimated from the DLM's initial predictions on the queries assigned to that cluster. For each query q_m, a masked answer region is appended and the DLM is evaluated once at the fully masked state. For each answer position $i \in \mathrm{A}_m$ at the 0-th step, the model produces

$$p_{0,i}^{(m)}(v) = p_\theta\left(X_i = v \mid q_m\right). \tag{6}$$

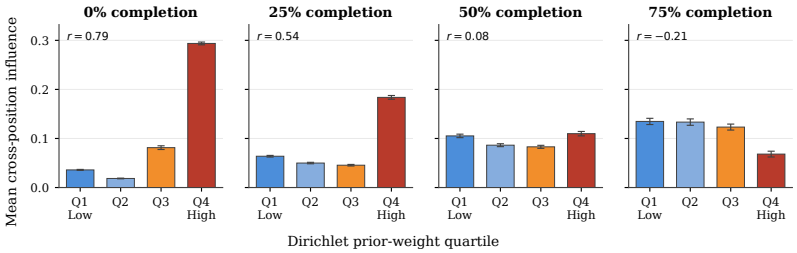


Figure 2: Prior weight and confidence gains on MATH-500 (generation length 128, block length 32). Q1–Q4 indicate increasing prior-weight quartiles. Bars show means with standard errors; r denotes Pearson correlation.

Rather than using later denoising states, which are increasingly influenced by previously revealed answer tokens, we construct the prior from the fully masked step-0 predictions. At this stage, repeated predictive mass across answer positions provides an answer-free, query-conditioned signal that is aggregated into a coarse task-related vocabulary profile. Alternative prior construction strategies are demonstrated to be less promising than DPB in Appx. D.2.

Within cluster k, these step-0 predictions are aggregated into vocabulary-level soft counts $a_{k,v}$. With smoothing constant $\delta > 0$, the Dirichlet parameters are defined as $\alpha_{k,v} = a_{k,v} + \delta$. Normalizing these parameters gives the cluster-level vocabulary prior,

$$a_{k,v} = \sum_{m:c_m=k} \sum_{i \in \mathrm{A}_m} p_{0,i}^{(m)}(v), \quad \pi_{k,v} = \frac{\alpha_{k,v}}{\sum_{u=1}^{V} \alpha_{k,u}}. \tag{7}$$

Each prior-bank entry therefore stores a cluster centroid together with a vocabulary prior estimated entirely from answer-free queries.

Prior retrieval. For a test query q, its embedding $E(q)$ is matched to the nearest cluster centroid, $k(q) = \arg\min_k \|E(q) - \boldsymbol{\mu}_k\|$, using Euclidean distance. No additional embedding normalization is applied during retrieval. The corresponding prior $\boldsymbol{\pi}_{k(q)}$ is retrieved as $\boldsymbol{\pi}_0(q)$. Retrieval requires only a query embedding and a nearest-centroid lookup in a small prior-bank containing at most a few dozen clusters.

Can the retrieved vocabulary prior help identify supporting tokens? To examine this question, we measure how revealing an individual candidate token affects confidence at other remaining masked positions. The analysis in Sec. 4.5 (Fig. 2) shows that higher-prior tokens tend to yield greater confidence gains early in denoising, while this association weakens at later stages. These findings support using the retrieved prior to identify candidate supporting tokens when little answer context is available. The next subsection describes how this task-level signal is combined with current DLM evidence to guide token commitment.

3.3 Prior-Guided Diffusion Decoding

At decoding time, the retrieved prior provides task-level vocabulary support, while $\mathbf{p}_{t,i}$ provides instance-specific evidence from the current partially denoised sequence. We combine these sources through a Dirichlet posterior update and use the posterior predictive probabilities to determine reveal order under a fixed schedule. The same formulation also supports adaptive decoding through a calibrated commitment threshold.

Time-dependent prior. As more answer tokens are revealed, the current sequence provides increasingly specific context for the DLM. We therefore gradually remove the task-specific vocabulary bias of the retrieved prior by annealing it toward the uniform distribution over the decoding trajectory. For $0 \le t \le T$, let $\rho_t = 1 - t/T$. The prior at step t at vth word is defined as

$$\pi_{t,v}(q) = \rho_t \pi_{0,v}(q) + (1 - \rho_t)\frac{1}{V}, \tag{8}$$

This keeps $\sum_v \pi_{t,v}(q) = 1$ while flattening the prior over time.

Dirichlet posterior update. Given the time-dependent prior, the current DLM prediction provides instance-specific evidence for each masked position. For position i, the probability vector $\mathbf{p}_{t,i}$ is treated as fractional categorical evidence with total mass $\lambda > 0$. This gives the Dirichlet posterior

$$\boldsymbol{\vartheta}_{t,i} \mid \mathbf{p}_{t,i}, \boldsymbol{\pi}_t(q) \sim \mathrm{Dir}\left(\boldsymbol{\pi}_t(q) + \lambda \mathbf{p}_{t,i}\right). \tag{9}$$

Since both $\boldsymbol{\pi}_t(q)$ and $\mathbf{p}_{t,i}$ are normalized distributions, the total posterior concentration is $1 + \lambda$. Taking the mean of the Dirichlet posterior gives the posterior predictive probability

$$\widetilde{p}_{t,i}(v) = \frac{\pi_{t,v}(q) + \lambda p_{t,i}(v)}{1 + \lambda}. \tag{10}$$

Here, λ controls the relative strength of the current DLM evidence, with larger values placing more weight on the model prediction and smaller values giving greater relative influence to the retrieved prior.

Decoding strategies. By default, we use *fixed-step decoding* with a predefined reveal budget at each step. For each eligible masked position i, we evaluate the posterior predictive probability in Eq. 10 at the original DLM prediction $\hat{v}_{t,i}$ defined in Eq. 3. Positions are then ranked by $\widetilde{p}_{t,i}(\hat{v}_{t,i})$, and the highest-ranked positions are selected according to the scheduled budget. Thus, the posterior predictive distribution is used only as a commitment-confidence score for the DLM's top-1 token, rather than as a replacement token distribution.

We additionally support *adaptive decoding*, where the number of positions revealed at each step is determined by applying a threshold to their posterior predictive confidence. For an original confidence threshold τ, we use uniform prior support as the reference to obtain the corresponding posterior threshold

$$\widetilde{\tau} = \frac{1/V + \lambda\tau}{1 + \lambda} \approx \frac{\lambda}{1 + \lambda}\tau, \tag{11}$$

where $1/V$ is negligible for a large vocabulary. Positions whose posterior predictive confidence exceeds $\widetilde{\tau}$ are committed. If none crosses the threshold, the scheduled reveal budget is followed using the same confidence for ranking. Thus, the same prior-guided confidence is used for ranking in fixed-step decoding and for both thresholding and ranking in adaptive decoding.

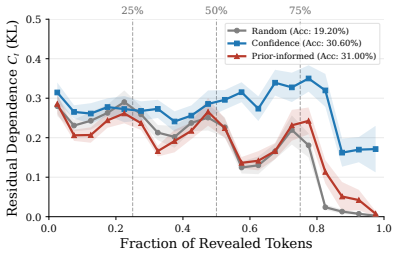


Figure 3: Residual conditional dependence on MATH-500 with generation length 128 and block length 32. Shaded regions denote standard errors.

Does prior-guided reveal ordering reduce conditional dependence among the remaining masked positions? Analysis in Sec. 4.5 (Fig. 3) shows that, under matched reveal budgets, prior-guided decoding generally yields lower average residual dependence than confidence-based decoding, as measured by Eq. 4. This finding supports the view that prioritising candidate supporting tokens provides context that makes the remaining predictions less sensitive to individual token revelations.

4 Experiments

We evaluate PGDD across tasks and decoding settings in terms of generation quality and efficiency, and further study the effect of different prior choices.

4.1 Experimental Setup

We evaluate our method on five benchmarks spanning three categories of reasoning tasks. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024); for code generation, HumanEval (Chen, 2021) and MBPP (Austin et al., 2021); and for planning-style symbolic reasoning, Countdown following (Zhao et al., 2026). Unless otherwise stated, the prior bank uses $K = 8$ clusters and the block size is fixed to 32, consistent with common practice in prior DLM decoding work.

We use LLaDA-8B-Instruct (Nie et al., 2025) as the main backbone and evaluate prior-guided decoding under both fixed-step and adaptive decoding. For adaptive decoding, the baseline commits tokens whose confidence exceeds $\tau = 0.95$ before the maximum decoding budget is reached. Additionally, we evaluate our method using Fast-dLLM (Wu et al., 2025) to test compatibility with an accelerated decoder based on KV caching and parallel token commitment. The backbone model and decoding configuration are kept unchanged between each baseline and its prior-guided counterpart. We report the standard task-specific quality metric together with decoding steps and speed-up. Addi-

Table 1: Task performance and decoding efficiency of LLaDA-8B-Instruct under fixed-step and adaptive decoding. Each cell reports performance and average decoding steps in blue, with speedup relative to the corresponding maximum-step budget shown in orange parentheses. Bold indicates the best performance within each setting, shaded cells denote our PGDD, and "–" denotes configurations that terminated with a runtime crash under Fast-dLLM.

Benchmark	Max. Steps	Fixed-step decoding		Adaptive decoding			
		LLaDA	PGDD	LLaDA	PGDD	Fast-dLLM	Fast-dLLM + PGDD
HumanEval	128	26.83 128.0 (1.0×)	**32.32** 128.0 (1.0×)	26.22 60.0 (2.1×)	**29.27** 47.5 (2.7×)	**28.05** 49.9 (2.6×)	27.44 45.6 (2.8×)
	256	38.41 256.0 (1.0×)	**39.63** 256.0 (1.0×)	37.80 121.3 (2.1×)	**39.02** 108.1 (2.4×)	35.34 103.2 (2.5×)	**36.59** 91.1 (2.8×)
GSM8K	128	69.98 128.0 (1.0×)	**71.11** 128.0 (1.0×)	69.83 60.8 (2.1×)	**70.13** 55.7 (2.3×)	69.45 65.3 (2.0×)	**69.67** 58.8 (2.2×)
	256	**75.06** 256.0 (1.0×)	74.08 256.0 (1.0×)	**74.60** 89.7 (2.9×)	**74.60** 81.7 (3.1×)	71.27 97.9 (2.6×)	**72.71** 84.5 (3.0×)
MBPP	128	**37.80** 128.0 (1.0×)	37.20 128.0 (1.0×)	**37.40** 45.4 (2.8×)	37.20 43.5 (2.9×)	37.80 44.4 (2.9×)	**39.80** 47.5 (2.7×)
	256	36.00 256.0 (1.0×)	**36.20** 256.0 (1.0×)	36.00 51.2 (5.0×)	**38.20** 43.8 (5.8×)	25.80 80.1 (3.2×)	**26.00** 86.1 (3.0×)
MATH-500	128	30.80 128.0 (1.0×)	**31.40** 128.0 (1.0×)	30.60 74.1 (1.7×)	**31.00** 69.9 (1.8×)	27.20 77.9 (1.6×)	**27.80** 75.5 (1.7×)
	256	31.20 256.0 (1.0×)	**32.80** 256.0 (1.0×)	**32.00** 122.1 (2.1×)	31.40 115.6 (2.2×)	27.40 131.5 (1.9×)	**29.69** 122.3 (2.1×)
Countdown	64	37.89 64.0 (1.0×)	**41.41** 64.0 (1.0×)	39.84 41.1 (1.6×)	**40.62** 34.6 (1.8×)	–	–
	128	22.27 128.0 (1.0×)	**23.05** 128.0 (1.0×)	20.31 90.5 (1.4×)	**23.44** 89.5 (1.4×)	23.05 92.6 (1.4×)	**24.61** 85.1 (1.5×)
	256	16.80 256.0 (1.0×)	**20.31** 256.0 (1.0×)	16.80 182.3 (1.4×)	**17.58** 177.6 (1.4×)	20.31 193.1 (1.3×)	**20.70** 175.8 (1.5×)

tional details on prior sources, prior-bank construction, and hyperparameter selection and sensitivity analysis are provided in Appx. B.

4.2 Main Results and Analysis

Overall performance and efficiency. Tab. 1 compares PGDD with the original LLaDA decoder under both fixed-step and adaptive decoding. Under fixed-step decoding, incorporating the prior generally improves task performance, with particularly clear gains on HumanEval and Countdown. For example, HumanEval improves from 26.83 to 32.32 with a 128-step budget, while Countdown improves from 37.89 to 41.41 with a 64-step budget. Across all 32 comparable settings in Tab. 1, PGDD improves performance in 26 settings, with an average absolute gain of 1.15 percentage points over the corresponding baseline. As a distribution-free aggregate check, a two-sided sign test over the comparable settings gives a p value of 1.9e-4. Since these settings span heterogeneous evaluation setups, this test should be interpreted only as an aggregate descriptive check.

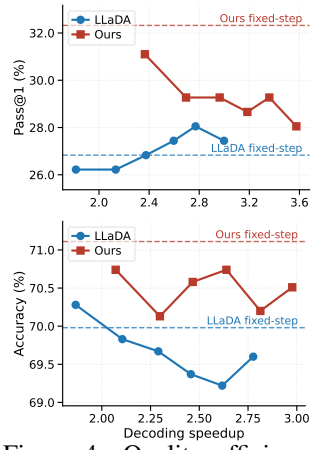


Figure 4: Quality–efficiency trade-off under different commitment thresholds. Dashed lines denote fixed-step decoding. *Top:* HumanEval; *Bottom:* GSM8K.

The benefit of PGDD is more evident under adaptive decoding. Under the same confidence threshold ($\tau = 0.95$), PGDD generally maintains or improves task performance while consistently requiring fewer decoding steps. On HumanEval with a 128-step budget, the average number of steps decreases from 60.0 to 47.5, while Pass@1 increases from 26.22 to 29.27. Similarly, on MBPP with a 256-step budget, PGDD improves the score from 36.00 to 38.20 while reducing the average number of decoding steps from 51.2 to 43.8.

Beyond the vanilla decoder, applying PGDD to Fast-dLLM (Wu et al., 2025) shows that prior guidance is complementary to existing acceleration methods, achieving the best performance in several settings, including MBPP-128 and Countdown-128/256.

The gains vary across tasks, which may partly reflect differences in how well parallel denoising suits the underlying generation process (Ye et al., 2025a). Planning-style tasks such as Countdown involve stronger global constraints across multiple output positions, making reveal order particu-

Table 2: PGDD as a plug-in for existing decoding methods. *Left:* Prophet; *Right:* AHD.

Benchmark	Len.	Prophet	Prophet+PGDD
HumanEval	128	24.39 81.30 (1.57×)	**31.10** 84.41 (1.52×)
	256	37.80 177.76 (1.44×)	**40.24** 176.65 (1.45×)
GSM8K	128	52.01 96.09 (1.33×)	**64.06** 110.71 (1.16×)
	256	**69.67** 202.68 (1.26×)	69.07 202.61 (1.26×)

Benchmark	Len.	AHD	AHD+PGDD
HumanEval	128	27.44 34.82 (3.68×)	**28.05** 33.35 (3.84×)
	256	36.59 90.91 (2.82×)	**37.80** 73.16 (3.50×)
MATH-500	128	23.20 57.87 (2.21×)	**23.60** 52.65 (2.43×)
	256	31.00 105.54 (2.43×)	**31.20** 102.19 (2.51×)

Note: Prophet and AHD are evaluated using their original code, so their prompt templates and implementation details may differ slightly from the main experiments. Following the original AHD setting, an early-commit threshold of 0.90 is used for AHD and AHD+PGDD, while $\tau = 0.95$ is used in all other experiments.

larly important. In such cases, prioritizing informative tokens provides more useful context for subsequent predictions, leaving more room for PGDD to improve generation quality. Note that these improvements come with limited additional computation. Prior retrieval requires only a K-way nearest-centroid lookup once per query, with $K = 8$ in the main experiments, followed by a lightweight confidence update during decoding, without additional DLM forward passes.

Quality–efficiency trade-off. Fig. 4 compares decoding quality and efficiency at generation length 128 across commitment thresholds $\tau \in \{0.975, 0.95, 0.925, 0.90, 0.875, 0.85\}$. At every evaluated threshold, PGDD achieves higher generation quality with fewer decoding steps than the LLaDA baseline on both HumanEval and GSM8K. The improvement is particularly pronounced on HumanEval, where PGDD maintains substantially higher Pass@1 across a broad range of decoding speedups. Full numerical results, including the corresponding decoding steps for each threshold, are reported in Appx. D.1.

4.3 Compatibility with Other Decoding Methods

PGDD is plugged into recent DLM decoding methods Prophet (Li et al., 2025) and AHD (Zou et al., 2026) to test its effectiveness. Prophet commits tokens based on the confidence margin between top two candidates, while AHD uses anchor tokens and historical prediction stability to unlock tokens beyond the current block boundary. Tab. 2 reports each method and its PGDD version.

The results show that PGDD improves Prophet on HumanEval across the reported generation settings and improves GSM8K in one setting. With AHD, PGDD matches or improves the reported scores in all settings and reduces the average number of decoding steps. The smaller accuracy gains with AHD may be because AHD already uses anchor tokens and historical stability to make aggressive commitment decisions, leaving less room for an additional confidence prior to change the token order. Overall, these results suggest that PGDD is largely orthogonal to existing commitment rules.

4.4 What Makes a Useful Prior?

We study how prior relevance affects performance, whether localized priors provide an advantage over a single global prior, and how sensitive the method is to the prior-bank structure.

Does prior source matter? We first test whether the gain depends on the information encoded by the prior rather than simply modifying the confidence score. All variants use the same decoding rule. The no-prior baseline uses the original model confidence, while the remaining variants use Dirichlet priors constructed from different question sources. Tab. 3 shows that performance depends strongly on the relevance of the retrieved prior. Task-related priors generally yield the largest gains. The MBPP prior improves HumanEval from 26.83 to 32.32, the GSM8K prior improves GSM8K from 69.98 to 71.11, and the MATH-hard prior improves MATH-500 from 31.20 to 32.80. In contrast, a general-domain prior constructed from SQuAD (Rajpurkar et al., 2016) consistently performs worse than the task-related priors. We further construct `gsm8k_mbpp_c16` by combining the GSM8K and MBPP $K = 8$ prior banks. Despite spanning two domains, the combined bank achieves the best GSM8K performance of 71.34 and also improves over the no-prior baseline on HumanEval and MATH-500. This indicates that the prior bank need not be restricted to a single task domain, as nearest-centroid retrieval selects a relevant prior for each query from the available clusters.

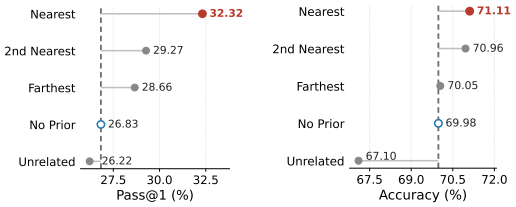


Figure 5: Effect of prior proximity on downstream performance. The dashed line denotes the no-prior baseline. *Left:* HumanEval; *Right:* GSM8K.

Table 3: Effect of prior source on downstream performance. Parentheses show changes from the no-prior baseline. **Bold** and underlined values denote the best and second-best results.

Prior Source	HumanEval	GSM8K	MATH-500
No prior	26.83	69.98	31.20
pycode_c8	27.44 (+0.61)	67.93 (−2.05)	30.80 (−0.40)
mbpp_c8	**32.32** (+5.49)	69.67 (−0.31)	31.40 (+0.20)
squad_c8	26.22 (−0.61)	67.10 (−2.88)	30.80 (−0.40)
gsm8k_c8	28.06 (+1.23)	71.11 (+1.13)	31.40 (+0.20)
math_hard_c8	24.39 (−2.44)	69.75 (−0.23)	**32.80** (+1.60)
gsm8k_mbpp_c16	28.05 (+1.22)	**71.34** (+1.36)	31.80 (+0.60)

Table 4: Sensitivity to the number of prior clusters across downstream tasks. **Bold** and underlined values denote the best and second-best results, and † marks the setting used in the main experiments.

K	HumanEval	GSM8K	MBPP	MATH-500	Countdown
No prior	26.83	69.98	**37.80**	30.80	37.89
1	**33.54**	70.66	36.40	31.60	40.62
4	31.10	70.28	35.60	31.60	41.02
8†	32.32	**71.11**	37.20	31.40	**41.41**
16	32.32	70.66	36.60	**31.80**	39.84

Does prior proximity predict utility? We next examine whether prior utility depends on its proximity to the target query within the DPB. Fig. 5 shows closer priors generally lead to better performance. On HumanEval, the nearest prior achieves a Pass@1 of 32.32, compared with 26.83 without a prior, while the second-nearest and farthest priors achieve 29.27 and 28.66. The unrelated-domain prior further drops performance to 26.22. A similar pattern appears on GSM8K. The nearest and second-nearest priors achieve 71.11 and 70.96 accuracy, compared with 69.98 without a prior. The farthest prior provides only a small improvement at 70.05, while the unrelated-domain prior reduces accuracy to 67.10. These results show that prior utility decreases as retrieval becomes less relevant, supporting nearest-prior retrieval rather than the use of an arbitrary prior.

Sensitivity to prior-bank size. We further vary the number of clusters used to construct the prior bank. As shown in Tab. 4, performance remains relatively stable across different values of K, indicating that PGDD does not require a fine partition of the query space. A single cluster already yields substantial gains on HumanEval, while larger prior banks provide additional benefits on other tasks. We use $K = 8$ in the main experiments as it offers a good balance between performance and prior-bank size.

4.5 Empirical Analysis of Prior-Guided Denoising

Prior weight and token influence. To examine whether high-prior supporting tokens provide more useful information to other positions, we group masked positions into quartiles according to the prior weight of their top-1 predicted tokens at different stages of denoising. For each position, we reveal its predicted token while keeping all other positions unchanged and measure the mean change in top-1 confidence across other remaining masked positions. Fig. 2 shows that revealing higher-prior tokens tends to produce greater confidence gains at other masked positions early in decoding. In particular, the highest prior-weight quartile yields the largest average gains at 0% and 25% decoding progress. This stage-dependent behavior suggests that task-level prior information is particularly useful when the sequence is still highly under-specified, while its contribution naturally decreases as more context becomes available.

Residual conditional dependence. We then examine whether prior-guided reveal ordering reduces the conditional dependence defined in Eq. 4. We average $C_{j\to k}(W_t)$ over sampled masked pairs j and k to obtain the trajectory-level residual dependence score C_t at step t. Three reveal schedules are compared under the same model and reveal budget. Random selects masked positions uniformly at random. Confidence ranks positions using the original DLM confidence, while Prior-informed ranks them using the confidence after incorporating the retrieved prior. Fig. 3 shows that, under the default block size used in the main experiments, the prior-informed schedule generally leaves lower residual dependence than Confidence-only decoding. The relation, $C_t^{\text{Prior}} < C_t^{\text{Confidence}}$, holds over substantial portions of the denoising trajectory. Together with Fig. 2, these results provide two complementary observations. High-prior tokens tend to have greater influence during early denoising, while prior-guided reveal ordering leaves lower residual dependence among the remaining masked positions. The non-monotonic pattern mainly reflects block-boundary effects under block-wise decoding (See Appx. C.1).

5 Conclusion

We propose PGDD, a training-free decoding framework for masked diffusion language models that uses an answer-free, query-conditioned vocabulary prior to guide token reveal order. Across reasoning, code generation, and planning tasks, PGDD generally improves generation quality and decoding efficiency and remains compatible with existing accelerated decoding methods. Empirical analysis further shows that higher-prior tokens tend to have greater influence during early denoising, while prior-guided reveal ordering leaves lower residual conditional dependence among unresolved positions. Future work will explore more adaptive prior construction and retrieval, including dynamic prior updates during denoising and finer-grained query matching.

AI USE STATEMENT

ChatGPT (GPT-5.6 Sol and Astra) has been used for paraphrasing and improving manuscript clarity, as well as assisting with verification of the mathematical proof in Sec. A.2. All AI-assisted content was reviewed and revised by the authors, who take full responsibility for the final manuscript.

REFERENCES

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

Iskander Azangulov, Teodora Pandeva, Niranjani Prasad, Javier Zazo, and Sushrut Karmalkar. Parallel sampling from masked diffusion models via conditional independence testing. In *International Conference on Learning Representations*, volume 2026, pp. 140485–140521, 2026.

Changxiao Cai and Gen Li. Confidence-based decoding is provably efficient for diffusion language models. *arXiv preprint arXiv:2603.22248*, 2026.

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

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. In Steve Hanneke and Tor Lattimore (eds.), *Proceedings of Thirty Ninth Conference on Learning Theory*, volume 336 of *Proceedings of Machine Learning Research*, pp. 1279–1311. PMLR, 29 Jun–03 Jul 2026a. URL https://proceedings.mlr.press/v336/chen26e.html.

Yuzhe Chen, Jiale Cao, Xuyang Liu, Jin Xie, Aiping Yang, and Yanwei Pang. Stdec: Spatio-temporal stability guided decoding for dllms. *arXiv preprint arXiv:2604.06330*, 2026b.

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

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1): 1–22, 1977.

Daniil Dmitriev, Zhihan Huang, and Yuting Wei. Efficient sampling with discrete diffusion models: Sharp and adaptive guarantees. In Steve Hanneke and Tor Lattimore (eds.), *Proceedings of Thirty Ninth Conference on Learning Theory*, volume 336 of *Proceedings of Machine Learning Research*, pp. 2038–2104. PMLR, 29 Jun–03 Jul 2026. URL https://proceedings.mlr.press/v336/dmitriev26a.html.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

Siyang He, Qiqi Wang, Xiaoran Liu, Hongnan Ma, Yiwei Shi, Yuerong Song, Ying Zhu, Tianyi Liang, Zengfeng Huang, Ziwei He, et al. Fouriersampler: Unlocking non-autoregressive potential in diffusion language models via frequency-guided generation. *arXiv preprint arXiv:2601.23182*, 2026.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Ian Li, Zilei Shao, Benjie Wang, Rose Yu, Guy Van den Broeck, and Anji Liu. Breaking the factorization barrier in diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=euv8IoKFbh.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*, 2025.

Kaihua Liang, Xin Tan, An Zhong, Hong Xu, and Marco Canini. Focus: Dllms know how to tame their compute bound. *arXiv preprint arXiv:2601.23278*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

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

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models, 2025. *URL https://arxiv. org/abs/2505.15781*.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

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

Heqiang Qi, Wei Huang, Mingyuan Bai, and Xiangming Meng. Cluster-level attention-guided parallel decoding for masked diffusion language models. *arXiv preprint arXiv:2605.29607*, 2026.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

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

Luohe Shi, Yao Yao, Zuchao Li, Lefei Zhang, and Hai Zhao. Reference trustable decoding: A training-free augmentation paradigm for large language models. *Advances in Neural Information Processing Systems*, 37:80034–80055, 2024.

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

Shijian Xu, Andrea Miele, Metod Jazbec, Volker Roth, Eric Nalisnick, and Ilija Bogunovic. Temporal self-distillation: Faster inference in discrete diffusion language models. *arXiv preprint arXiv:2609.15177*, 2026.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning, 2025a. URL https://arxiv.org/abs/2410.14157.

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

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:56729–56762, 2026.

Shun Zou, Yong Wang, Zehui Chen, Lin Chen, Chongyang Tao, Feng Zhao, and Xiangxiang Chu. Breaking block boundaries: Anchor-based history-stable decoding for diffusion large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20637–20658, 2026.

A Mathematical Details

A.1 Dirichlet Conjugacy and Posterior Predictive Distribution

Our vocabulary prior is based on the conjugacy between the Dirichlet distribution and categorical token observations. This choice is especially convenient in diffusion language models, where each unresolved position is repeatedly represented by a categorical distribution over the vocabulary during denoising. Let

$$\boldsymbol{\vartheta} = (\vartheta_1, \ldots, \vartheta_{|\mathrm{V}|}), \qquad \sum_{v=1}^{|\mathrm{V}|} \vartheta_v = 1, \tag{12}$$

denote the latent vocabulary distribution associated with a group of related queries. We assume

$$p(\boldsymbol{\vartheta}) = \mathrm{Dir}(\boldsymbol{\vartheta}; \boldsymbol{\alpha}) \propto \prod_{v=1}^{|\mathrm{V}|} \vartheta_v^{\alpha_v - 1}. \tag{13}$$

Suppose the retrieved related examples provide token observations $\mathcal{D} = z_1, \ldots, z_N$, with vocabulary counts

$$c_v = \sum_{n=1}^{N} \mathbf{1}[z_n = v]. \tag{14}$$

Under a categorical likelihood,

$$p(\mathcal{D} \mid \boldsymbol{\vartheta}) \propto \prod_{v=1}^{|\mathrm{V}|} \vartheta_v^{c_v}. \tag{15}$$

Combining the likelihood with the Dirichlet prior yields

$$p(\boldsymbol{\vartheta} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{\vartheta}) p(\boldsymbol{\vartheta}) \qquad \propto \prod_{v=1}^{|\mathrm{V}|} \vartheta_v^{\alpha_v + c_v - 1}, \tag{16}$$

and therefore

$$\boldsymbol{\vartheta} \mid \mathcal{D} \sim \mathrm{Dir}(\boldsymbol{\alpha} + \mathbf{c}). \tag{17}$$

Hence, incorporating additional evidence amounts to an additive pseudo-count update,

$$\alpha_v \leftarrow \alpha_v + c_v, \tag{18}$$

without requiring iterative Bayesian inference or optimization. The posterior predictive probability of observing vocabulary item v is available in closed form:

$$p(z_{\mathrm{new}} = v \mid \mathcal{D}) = \mathbb{E}[\vartheta_v \mid \mathcal{D}] = \frac{\alpha_v + c_v}{\sum_{u=1}^{|\mathrm{V}|} (\alpha_u + c_u)}. \tag{19}$$

This conjugate structure is useful for diffusion decoding for two reasons. First, the model prediction at every unresolved position is itself categorical, so the Bayesian support signal has the same vocabulary-level representation as the DLM output. Second, because diffusion decoding repeatedly revisits unresolved positions, the support statistics can be reused or updated across denoising steps through closed-form count updates rather than through an additional learned estimator.

In our decoding rule, we use only the normalized posterior support

$$\pi(v) = \frac{\alpha_v}{\alpha_0}, \qquad \alpha_0 = \sum_{u=1}^{|\mathrm{V}|} \alpha_u. \tag{20}$$

as an auxiliary signal for determining which positions to reveal. In particular, $\pi(v)$ affects the reveal policy but does not modify the DLM distribution $p_\theta(x_i \mid x_t)$. This separation allows the vocabulary prior to inject task-related supporting information while preserving the pretrained generative distribution of the diffusion language model.

A.2 Frequency-informed decoding with soft support

Model. Fix the decoding context c for a generation and suppress it throughout. Let U denote the set of currently masked positions, with $n = |U|$, and let the vocabulary $\mathcal{V}$ have size $V = |\mathcal{V}| \geq 2$.

The external Dirichlet parameters $\alpha_v > 0$ define the frequency profile $\pi_v = \alpha_v / \sum_{u \in \mathcal{V}} \alpha_u$. Let $b_v = b(v \mid c) > 0$ be a model-induced background distribution, shared across the positions being compared for simplicity. To analyze cross-position dependence, we introduce an auxiliary shared latent vocabulary state Z, centered on the vocabulary prior $\boldsymbol{\pi}$ used in the main method. We assume

$$Z \sim \mathrm{Dir}(\kappa\pi), \qquad X_i \mid Z \overset{\text{ind}}{\sim} \mathrm{Cat}\big(a_i Z + (1 - a_i)b\big), \tag{21}$$

where $\kappa > 0$, $a_i \in [0, 1]$, and the parameter κ controls the concentration of Z around its mean π. The shared vector Z describes uncertain word propensities for one generation; a_i measures how strongly position i depends on it. Thus dependencies arise by marginalising the same latent state. The shared structural distribution is centred on the external prior, $\mathbb{E}[Z \mid c] = \pi$, while the background retains the model's nonuniform word probabilities.

Equivalently, draw $A_i \sim \mathrm{Bernoulli}(a_i)$ independently across positions and independently of Z: position i samples from Z when $A_i = 1$ and from b otherwise. For $0 < a_i < 1$, every word can arise through either source, and several positions can sample different words from Z. Bayes' rule gives

$$\Pr(A_i = 1 \mid X_i = v) = \frac{a_i \pi_v}{a_i \pi_v + (1 - a_i) b_v}. \tag{22}$$

For fixed $0 < a_i < 1$, this probability increases with the prior enrichment π_v / b_v: a word recurrent in the task population relative to the model background is more likely to originate from shared structure.

Objective. Let p denote the joint law in 21 at the fixed context c. For a fixed budget $1 \le m < n$, select $S \subseteq U$ with $|S| = m$ before observing its tokens, and write $R = U \setminus S$. We first sample X_S jointly from p, then sample the remaining tokens independently from their exact conditional marginals, yielding $Q_S(x_U) = p(x_S) \prod_{j \in R} p(x_j \mid x_S)$. The expected residual factorisation error is

$$\mathcal{E}(S) := \mathbb{E}_{X_S}\left[D_{\mathrm{KL}}\left(p(X_R \mid X_S) \,\middle\|\, \prod_{j \in R} p(X_j \mid X_S) \right) \right]. \tag{23}$$

Equivalently, $\mathcal{E}(S) = D_{\mathrm{KL}}(p \| Q_S) = \mathrm{TC}(X_R \mid X_S)$, the sum of conditional marginal entropies minus conditional joint entropy. The expectation is over the revealed words; position selection still uses a maximum.

Theorem A.1 (Frequency-guided prioritisation)**.** *Under 21, suppose there exists*

$$v_0 \in \arg\max_v b_v \pi_v \quad \textit{such that} \quad \pi_{v_0} > b_{v_0}. \tag{24}$$

Thus a word favoured jointly by the prior and background is enriched in the task population. With predictive evidence $p_i(v) = p(X_i = v)$, define

$$s_i := \max_v \{p_i(v)\pi_v\} = \max_v \big\{ b_v \pi_v + a_i \pi_v (\pi_v - b_v) \big\}. \tag{25}$$

For any $1 \le m < n$, a set S_m of the m largest-scoring positions satisfies

$$S_m \in \operatorname*{arg\,min}_{S \subseteq U : |S| = m} \mathcal{E}(S). \tag{26}$$

Moreover, for any $T \subseteq U$ and $i \notin T$,

$$\mathcal{E}(T) - \mathcal{E}(T \cup \{i\}) = \sum_{j \notin T \cup \{i\}} I(X_i; X_j \mid X_T) \ge 0. \tag{27}$$

Proof. The score ranks structural strength. Integrating out Z gives $p_i = a_i \pi + (1 - a_i) b$, so 25 follows. Write this score as $s(a_i)$. The function $s(a)$ is convex, being the maximum of affine functions of a. At $a = 0$, the word v_0 is a maximiser with positive slope $\pi_{v_0}(\pi_{v_0} - b_{v_0})$. Hence $s'_+(0) > 0$; convexity makes its slopes nondecreasing, so $s(a)$ is strictly increasing on $[0, 1]$. The maximum score therefore ranks a_i.

Weaker observations contain less information. Fix distinct $i, k \notin T$ with $a_i \ge a_k$ and $a_i > 0$. Construct $\widetilde{X}_k$ by keeping X_i with probability $\lambda = a_k / a_i$ and otherwise redrawing independently from b. For every Z, its distribution is

$$\lambda\big[a_i Z + (1 - a_i)b\big] + (1 - \lambda)b = a_k Z + (1 - a_k)b. \tag{28}$$

Conditional independence given Z implies that $(\widetilde{X}_k, X_j) \mid X_T$ and $(X_k, X_j) \mid X_T$ have the same law for $j \notin T \cup \{i, k\}$. Since $\widetilde{X}_k$ uses only X_i and fresh randomness, data processing gives

$$I(X_i; X_j \mid X_T) \geq I(\widetilde{X}_k; X_j \mid X_T) = I(X_k; X_j \mid X_T). \tag{29}$$

If $a_i = a_k = 0$, both information terms are zero.

The information ordering determines decoding error. Set $R = U \setminus (T \cup \{i\})$. Expanding 23 and using $H(X_i, X_R \mid X_T) = H(X_i \mid X_T) + H(X_R \mid X_T, X_i)$ yields

$$\mathcal{E}(T) = \sum_{j \in R} H(X_j \mid X_T) - H(X_R \mid X_T, X_i),$$

$$\mathcal{E}(T \cup \{i\}) = \sum_{j \in R} H(X_j \mid X_T, X_i) - H(X_R \mid X_T, X_i).$$

Subtracting proves 27. Comparing additions of i and k, the common $I(X_i; X_k \mid X_T)$ term cancels:

$$\mathcal{E}(T \cup \{k\}) - \mathcal{E}(T \cup \{i\}) = \sum_{j \notin T \cup \{i,k\}} \big[I(X_i; X_j \mid X_T) - I(X_k; X_j \mid X_T)\big] \geq 0.$$

Replacing a selected lower-scoring position with an unselected higher-scoring one therefore cannot increase the error. Repeated exchanges prove 26. □

Implication. A model can assign high confidence to a background word that carries little information about Z. Under the observable enrichment condition 24, the prior-weighted maximum instead ranks structural strength, minimising residual dependence in expectation. Several different words can supply partial evidence about Z. In practice, the predictive evidence at position i is given by the current DLM distribution $\mathbf{p}_{t,i}$ defined in Eq. 2.

B Implementation Details

This section provides additional details on the evaluation protocol, prior sources, construction and retrieval of the DPB, and selection of the prior-strength parameter λ.

Evaluation protocol. All benchmarks are evaluated in the zero-shot setting except MBPP, for which three-shot prompting is used to specify the expected Python function format. The main experiments use a block size of 32, and the Prior Bank contains $K = 8$ entries unless otherwise specified. For each comparison, the backbone model, generation length, and decoding configuration are held fixed between the baseline and the prior-guided variant.

Prior sources. Priors are constructed from answer-free training questions. GSM8K serves as the prior source for GSM8K, MATH-500, and Countdown, while MBPP serves as the source for MBPP and HumanEval. SQuAD (Rajpurkar et al., 2016) is used as a general-domain reference. No evaluation answers are accessed.

Prior-bank construction and retrieval. Queries used for prior construction are represented by mean-pooling their input-token embeddings from the underlying DLM. The resulting representations are partitioned into $K = 8$ groups using k-means clustering. Each cluster is associated with a vocabulary-level Dirichlet prior estimated from the DLM predictions of the queries assigned to that cluster. At inference time, the test query is represented in the same embedding space and matched to the nearest cluster to retrieve the corresponding prior.

Selection of λ. The parameter λ controls the relative strength of the current DLM prediction in the Dirichlet posterior update. We select λ separately for each benchmark and generation length using a small validation set and fix it for the corresponding test setting. Tab. 5 reports the values used in the main experiments. The sensitivity to λ is further examined in the ablation study (Fig. 8).

Table 5: λ settings across benchmarks and generation lengths.

Benchmark	64	128	256
HumanEval	–	1.0	3.0
GSM8K	–	1.0	2.0
MBPP	–	13.0	2.0
MATH-500	–	1.0	5.0
Countdown	0.5	1.0	2.0

Table 6: Storage cost of the Dirichlet Prior Bank on GSM8K.

K	1	4	8	16
Size	0.524 MB	2.1 MB	4.2 MB	8.4 MB

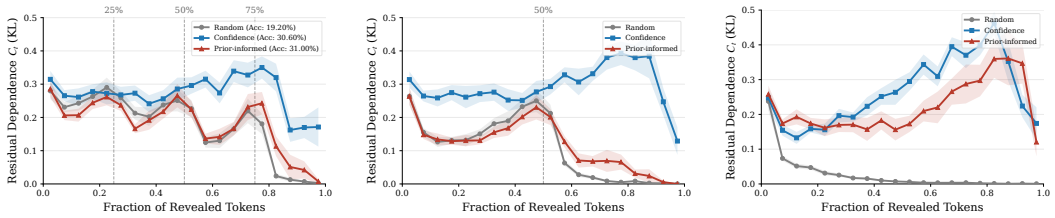
Computational and storage overhead. PGDD is constructed before evaluation and adds no DLM forward passes during inference. At test time, prior retrieval only requires mean pooling over input embeddings and a K-way centroid lookup, with $K = 8$ by default. During denoising, PGDD only applies a vectorized gather-and-rescale operation to the model's top-1 confidence.

The storage cost is small and grows approximately linearly with K, as shown in Table 6.Table 6 reports the saved file sizes for the prior banks.

C Additional Denoising-Dynamics Analysis

C.1 Conditional Dependence Across Block Sizes

To examine how the position-selection space affects conditional dependence during decoding, we compare block sizes 32, 64, and 128 while fixing the generation length to 128. Block size 32 is the default configuration used in the main experiments. Larger blocks allow the decoder to rank unresolved positions over a wider candidate set at each step. Note that the conditional-dependence scores in this analysis require additional intervention forward passes only for evaluation and are not used by PGDD during decoding.



(a) Block length 32. (b) Block length 64. (c) Block length 128.

Figure 6: Residual conditional dependence on MATH-500 under different block lengths. The generation length is fixed to 128. Random, Confidence, and Prior-informed use the same reveal budget and underlying DLM predictions and differ only in the position-selection rule. Shaded regions denote standard errors. Lower C_t indicates weaker conditional dependence among unresolved tokens.

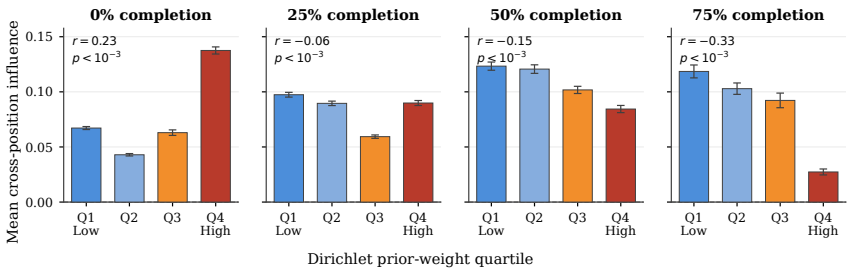
Fig. 6 shows the dependence trajectories across different block sizes. With block size 32 (Fig. 6a), decoding proceeds through four blocks. The transitions between blocks change the set of active unresolved positions, resulting in a more non-monotonic trajectory. The difference between Confidence and Prior-informed decoding is nevertheless visible over substantial parts of the trajectory.

With block size 64 (Fig. 6b), decoding requires only two blocks and each reveal decision is made over a larger candidate set. The separation between Confidence and Prior-informed decoding becomes more pronounced, particularly in the later stages of denoising.

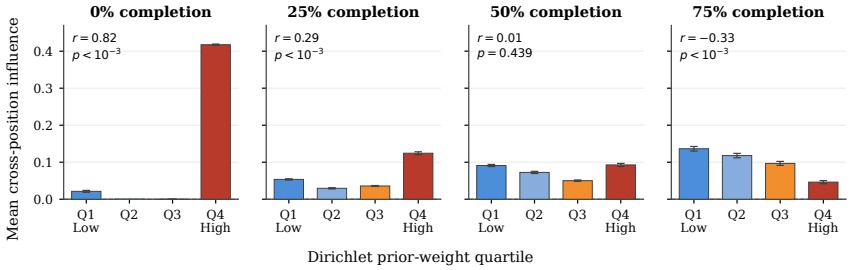
With block size 128 (Fig. 6c), all unresolved positions can be ranked jointly. Prior-informed decoding maintains lower residual dependence than Confidence over most of the trajectory, indicating that the benefit of prior-guided ordering becomes clearer when the decoder has greater flexibility in selecting positions.

Random decoding is included only as a reference. Its low residual dependence at some stages does not imply better generation quality, since C_t measures dependence among the remaining positions rather than the correctness of revealed tokens.

Overall, Prior-informed decoding leaves lower residual dependence than Confidence over substantial portions of the trajectory across all three block sizes. The separation becomes more visible with larger blocks, where the decoder has a wider set of unresolved positions to rank at each step.



(a) GSM8K



(b) Countdown

Figure 7: Relationship between Dirichlet prior weight and influence on other masked tokens during denoising across four additional tasks, with generation length 128 and block length 32. Within each task, the four panels correspond to different denoising stages. Q1 and Q4 denote the lowest and highest prior-weight quartiles, respectively. Bars show the mean influence with standard error.

C.2 Prior-Weight Influence Across Denoising Stages

We further analyze the relationship between Dirichlet prior weight and cross-position influence during denoising. At each stage, masked tokens are grouped by prior-weight quartile, and we measure the average confidence change induced at the remaining masked positions after committing each token. Fig. 7 extends the MATH-500 analysis to two additional tasks.

D Ablation Studies

D.1 Full Quality-Efficiency Results

Tab. 7 reports the complete results underlying Fig. 4. Each entry shows generation quality followed by the average number of inference steps in parentheses. All experiments use generation length 128.

Across $\tau \in \{0.975, 0.950, 0.925, 0.900, 0.875, 0.850\}$, prior-guided decoding achieves higher generation quality with fewer inference steps on both benchmarks.

D.2 Alternative Heuristic Priors

The vocabulary prior can be constructed from different lexical or linguistic signals. To separate the contribution of DLM-derived predictive information from simpler alternatives, three other heuristic priors are considered:

- **Uniform Prior.** All vocabulary tokens receive the same score,
$$\pi_v^{\text{uni}} = c, \qquad \forall v \in \mathcal{V},$$
where c is constant across the vocabulary. This prior contains no token-specific information and serves as a neutral control.
- **POS Prior.** We construct two sparse linguistic priors based on part-of-speech (POS) categories using the NLTK POS tagger. The noun prior retains noun tokens (`NN`, `NNS`, `NNP`,

Table 7: Full quality–efficiency results on HumanEval and GSM8K. Values in parentheses denote the average number of inference steps.

Setting	Method	HumanEval Pass@1 (%)	GSM8K Acc. (%)
Fixed-step	LLaDA	26.83 (128)	69.98 (128)
	Ours	32.32 (128)	71.11 (128)
$\tau = 0.975$	LLaDA	26.22 (70.5)	70.28 (68.6)
	Ours	31.10 (54.1)	70.74 (61.8)
$\tau = 0.950$	LLaDA	26.22 (60.0)	69.83 (60.8)
	Ours	29.27 (47.5)	70.13 (55.7)
$\tau = 0.925$	LLaDA	26.83 (53.9)	69.67 (55.9)
	Ours	29.27 (43.2)	70.58 (51.9)
$\tau = 0.900$	LLaDA	27.44 (49.3)	69.37 (52.1)
	Ours	28.66 (40.2)	70.74 (48.5)
$\tau = 0.875$	LLaDA	28.05 (46.2)	69.22 (48.9)
	Ours	29.27 (38.1)	70.20 (45.5)
$\tau = 0.850$	LLaDA	27.44 (42.7)	69.60 (46.1)
	Ours	28.05 (35.8)	70.51 (43.0)
$\tau = 0.800$	LLaDA	24.39 (32.3)	68.99 (38.6)
	Ours	28.05 (37.7)	68.69 (41.4)

and `NNPS`), while the verb prior retains verb tokens (`VB`, `VBD`, `VBG`, `VBN`, `VBP`, and `VBZ`). For each variant, we keep the 50 eligible tokens with the highest document frequency in the answer-free query corpus and assign them equal non-zero prior weight; all remaining vocabulary tokens receive zero weight. Document frequency is used only for token selection, rather than as the prior weight itself.

- **Frequency Prior.** We consider two frequency-based priors constructed from the training set. The *Query Frequency Prior* accumulates tokenizer-token frequencies from the answer-free training queries, while the *Answer Frequency Prior* uses the corresponding training answers. Both variants retain the same query-conditioned clustering as the main method. Within each cluster, token counts are accumulated from the selected text source and normalized across the vocabulary. The answer-based variant is included as an answer-informed reference for assessing whether direct answer statistics provide additional benefit over query-only information. For source $s \in \{\mathrm{query}, \mathrm{answer}\}$, the frequency prior for cluster k is

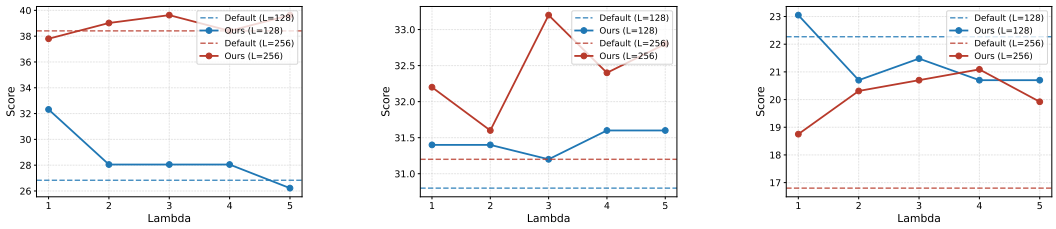
$$\pi_{k,v}^{(s)} = \frac{c_{k,v}^{(s)}}{\sum_{u \in \mathcal{V}} c_{k,u}^{(s)}},$$

where $c_{k,v}^{(s)}$ denotes the number of occurrences of token v in the corresponding training texts assigned to cluster k.

Comparison of heuristic priors. All alternative priors are evaluated under fixed-step decoding with generation length 128, block length 32, and $\lambda = 1$, while keeping the remaining decoding settings unchanged. The Uniform and POS priors are global, whereas the Query and Answer Frequency Priors use query-conditioned clustering with $K = 8$. The two frequency variants differ only in whether token statistics are computed from training queries or training answers.

Table 8: Comparison of alternative heuristic priors under fixed-step decoding with generation length 128 and block length 32.

Method	HumanEval Pass@1 (%)	GSM8K Acc. (%)
LLaDA	26.83	69.98
Uniform Prior	26.22	69.98
POS Prior (Noun)	26.22	70.13
POS Prior (Verb)	26.83	69.75
Query Frequency Prior	27.44	70.74
Answer Frequency Prior	26.83	**71.11**
PGDD	**32.32**	**71.11**



(a) HumanEval (b) MATH-500 (c) Countdown

Figure 8: Sensitivity to λ across representative benchmarks and generation lengths.

As shown in Tab. 8, the Uniform and POS priors provide little or inconsistent improvement over the vanilla decoder. The frequency-based priors are more competitive, although their gains remain task dependent. The Query Frequency Prior improves both benchmarks modestly, while the Answer Frequency Prior does not provide a consistent advantage over its query-based counterpart. In comparison, PGDD achieves the largest gain on HumanEval while matching the best GSM8K accuracy.

D.3 Hyperparameter Sensitivity

We further examine the sensitivity of PGDD to λ across HumanEval, MATH-500, and Countdown. Fig. 8 shows that the preferred value varies across tasks and generation lengths. HumanEval at length 128 is more sensitive to the prior strength, whereas MATH-500 and the longer generation settings remain relatively stable over a broader range. This variation supports selecting λ separately for each setting using a small validation set rather than adopting a single value across all tasks and generation lengths.

D.4 Qualitative Examples

We provide qualitative examples from all five benchmarks to illustrate how PGDD changes the reveal order and affects subsequent predictions. For each success case, we compare confidence-based decoding with PGDD and highlight the key supporting tokens that help resolve later predictions. We also include a failure case to illustrate a limitation of prior guidance.

Success Case 1: GSM8K

Question: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

Response with Confidence Decoding:
Step 1: Calculate the new value of the house after the repairs.
The original value of the house is $80,000, and the repairs increased the value by 150%.

$$\text{New value} = \mathbf{80{,}000} \times \mathbf{1.50} = \mathbf{120{,}000}.$$

The total cost is

$$80{,}000 + 50{,}000 = 130{,}000,$$

so the resulting profit is

$$120{,}000 - 130{,}000 = \mathbf{-10{,}000}.$$

Response with PGDD:
Step 1: Calculate the new value of the house after the repairs.
The original value of the house is $80,000, and the repairs increased the value by 150%.

$$\text{New value} = \mathbf{80{,}000} + (\mathbf{1.50} \times \mathbf{80{,}000}) = \mathbf{200{,}000}.$$

The total cost is

$$80{,}000 + 50{,}000 = 130{,}000,$$

so the resulting profit is

$$200{,}000 - 130{,}000 = \mathbf{70{,}000}.$$

Observation: Color intensity indicates the decoding order, with darker tokens revealed earlier and lighter tokens revealed later. In the baseline, the decoding trajectory forms the incorrect relation $80{,}000 \times 1.50$, yielding \$120,000. In contrast, PGDD reveals tokens that support retaining the original \$80,000 and adding the 150% increase, leading to the correct intermediate value of \$200,000 and the correct final profit.

Success Case 2: MATH-500

Question: Find the constant term in the expansion of

$$\left(10x^3 - \frac{1}{2x^2}\right)^5.$$

Response with Confidence Decoding:
The confidence-based decoder identifies

$$10x^3 \cdot \left(-\frac{1}{2x^2}\right) = \mathbf{-5},$$

and directly predicts the constant term as

$$\boxed{\mathbf{-5}}.$$

Response with PGDD:
PGDD continues to account for repeated contributions from the expansion and obtains

$$(-5)^3 = \mathbf{-125},$$

leading to the correct constant term

$$\boxed{\mathbf{-125}}.$$

Observation: The baseline stops after a single intermediate product and predicts -5. In contrast, PGDD retains the repeated contribution in the expansion, leading to the correct constant term of -125.

Success Case 3: HumanEval

Question: For a given number n, find the largest divisor of n that is strictly smaller than n.

```
>>> largest_divisor(15)
5
```

Response with Confidence Decoding:

```
def largest_divisor(n: int) -> int:
    for i in range(n, 0, -1):
        if n % i == 0:
            return i
    return 1
```

The baseline starts the search from n. Since n always divides itself, the function immediately returns n, violating the requirement that the divisor must be smaller than n.

Response with PGDD:

```
def largest_divisor(n: int) -> int:
    if n == 1:
        return 1
    i = n - 1
    while i > 0:
        if n % i == 0:
            return i
        i -= 1
    return 1
```

PGDD instead starts the search from $n - 1$, correctly enforcing the strict upper-bound constraint and returning the largest proper divisor.

Observation: The two generations differ in a critical boundary decision. Confidence-based decoding uses n as the initial search point and therefore returns the input itself, whereas PGDD uses $n - 1$ and correctly excludes n from the candidate divisors. For example, the baseline returns 3 for `largest_divisor(3)`, while PGDD correctly returns 1.

Success Case 4: Countdown

Question: Using the numbers 69, 31, and 78, construct an arithmetic expression that equals 40. Use each number exactly once.

Response with Confidence Decoding:
The confidence-based decoder produces

$$(\mathbf{69} + \mathbf{31}) - \mathbf{78},$$

with the reasoning

$$69 + 31 = 100, \qquad 100 - 78 = \mathbf{22}.$$

Thus, although all three numbers are used exactly once, the generated expression does not reach the target value of 40.

Response with PGDD:
PGDD instead produces

$$\mathbf{78} - \mathbf{69} + \mathbf{31},$$

which gives

$$78 - 69 = 9, \qquad 9 + 31 = \mathbf{40}.$$

The resulting expression uses each number exactly once and reaches the target value.

Observation: The baseline produces a plausible intermediate calculation but fails to satisfy the target value. In contrast, PGDD forms a more suitable intermediate structure, leading to a valid expression that satisfies the global arithmetic constraint.

Failure Case: HumanEval

Question: Return all prefixes of the input string from shortest to longest.

```
>>> all_prefixes('abc')
['a', 'ab', 'abc']
```

Response with Confidence Decoding:

```
def all_prefixes(string: str) -> List[str]:
    result = []
    for i in range(len(string)):
        result.append(string[:i+1])
    return result
```

The confidence-based decoder correctly includes the current character at each step and returns all prefixes from shortest to longest.

Response with PGDD:

```
def all_prefixes(string: str) -> List[str]:
    result = []
    for i in range(len(string)):
        result.append(string[:i])
    return result
```

The generated implementation follows the correct overall structure but uses `string[:i]` instead of `string[:i+1]`. As a result, `all_prefixes('abc')` produces **['', 'a', 'ab']** instead of **['a', 'ab', 'abc']**.

Observation: This example illustrates a failure mode of prior-guided decoding. Although PGDD preserves the correct high-level program structure, the reveal order leads to an off-by-one error in the slice boundary. This suggests that prior guidance does not always improve fine-grained, instance-specific code dependencies.