BAYESIAN ENTROPY-BASED REORDERING FOR CALIBRATED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked Diffusion Language Models (MDLMs) generate sequences by iteratively replacing masked tokens with model predictions. At each denoising step, the decoder chooses which positions are sufficiently confident to commit. Existing decoding methods typically rely on softmax confidence, which can be miscalibrated. We introduce BAYESER (*BAYES*ian *E*ntropy-based *R*eordering), a post-hoc Bayesian decoding framework that uses predictive uncertainty to guide token commitment. In BAYESER, we construct a lightweight approximate posterior centered at the pretrained checkpoint, similar to *Laplace-LoRA* but without training LoRA adapters. We average predictions over posterior samples and use predictive entropy to prioritize reliable positions. We examine how posterior predictions affect position ordering and token selection across benchmarks spanning code generation, mathematical reasoning, planning, and molecular generation. We show that BAYESER reduces sequence-level calibration error while preserving or improving accuracy relative to common decoding schemes, including confidence-threshold decoding. Additionally, a posterior fitted on one code-generation dataset reduces calibration error on another without refitting, suggesting that Bayesian uncertainty may provide a transferable signal for more reliable MDLM decoding.

1 INTRODUCTION

Language models can make wrong predictions with high confidence. This failure is relatively prevalent, as users often mistake stylistic confidence for evidence without justified certainty. A confidence score should mean what it claims: predictions with confidence q should be correct about a fraction q of the time; this is known as *calibration*. Formally, if the model $\mathbb{M}$ is perfectly calibrated, the following holds: $\mathbb{P}(\mathbb{M}'\text{s prediction is correct} \mid \mathbb{M}'\text{s confidence is } q) = q$, according to Guo et al. (2017). This definition of calibration does not eliminate hallucinations or reasoning errors common in modern language models. However, it makes uncertainty actionable, enabling a user or downstream system to decide whether to trust an answer, verify it, defer to a human, or abstain.

Calibration in masked diffusion language models (MDLMs) plays an unusually direct role. Unlike autoregressive models that generate tokens from left to right, MDLMs iteratively denoise a partially masked sequence. At each step, a bidirectional forward pass predicts a distribution over the vocabulary at every masked position, and the decoder selects a subset of these predictions to commit before continuing. This process enables parallel generation, arbitrary-order decoding, and infilling, and has recently scaled to models such as LLaDA and Dream (Nie et al., 2025; Ye et al., 2025). Consequently, confidence serves two distinct purposes in MDLMs: it should indicate whether the final sequence is likely correct (the usual confidence in the model's prediction), and more interestingly, it can guide which token predictions are reliable enough to commit at a given time-step during generation. This creates a central decoding problem that does not arise in standard autoregressive generation: *Which masked positions should be committed at each denoising step?*

Committing many positions in parallel reduces model evaluations, but once committed, those tokens provide context for all subsequent tokens being unmasked at later denoising steps. Committing too conservatively avoids such problems but reduces the computational advantage of parallel decoding. The token commitment ordering therefore controls both generation quality and inference cost. Existing decoders commonly commit to the tokens at certain positions with the largest top-token probabilities or all positions whose confidence exceeds a fixed threshold (Chang et al., 2022; Wu

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.

et al., 2025). These rules are efficient and often effective, but they rely on the softmax distribution of a single parameter estimate. Neural networks are often overconfident (Guo et al., 2017), and point-estimate confidence does not reveal whether a prediction remains stable under model uncertainty.

In this paper, we investigate whether Bayesian predictive uncertainty provides a better signal for commitment ordering. Given an approximate posterior $q(\boldsymbol{\theta}|\mathcal{D})$ over model parameters $\boldsymbol{\theta}$, the posterior predictive distribution over the output $\boldsymbol{y}$ of a model averages token probabilities across posterior samples, given a noisy state $\boldsymbol{x}_t$ of the denoising process: $\bar{p}(\boldsymbol{y}|\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\mathcal{D})}\left[p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{\theta})\right]$. Its entropy, $H(\bar{p}(\boldsymbol{y}|\boldsymbol{x}_t))$, measures the total uncertainty of the Bayesian prediction. Our method, BAYESER (**BAYES**ian **E**ntropy-based **R**eordering), is a post-hoc decoding framework for MDLMs that uses Bayesian predictive entropy. At each denoising step, BAYESER forms a posterior predictive distribution for every currently masked position, prioritizes positions with low posterior predictive entropy, selects tokens from the posterior predictive mean, and uses posterior confidence to determine how many positions to commit. In this way, BAYESER adjusts both the order and pace of decoding based on the Bayesian predictive distribution.

For typical MDLMs such as LLaDa (Nie et al., 2025) and Dream (Ye et al., 2025), with 7 billion parameters or more, identifying posterior distributions over all parameters is computationally challenging. *Laplace-LoRA* first fine-tunes LoRA adapters to obtain a task-specific maximum a posteriori solution and then approximates a posterior around the adapted model (Yang et al., 2024). While appropriate for downstream adaptation, this procedure changes the mean predictor and makes it difficult to distinguish the benefit of Bayesian uncertainty from that of fine-tuning. We instead construct a curvature-only Laplace-LoRA posterior directly *at the released checkpoint*, *without adapter training*. The LoRA update is zero at the posterior mean, so the deterministic model is unchanged by construction. We fit the posterior covariance with training data, representing each block of a Hessian matrix using a Kronecker product of input and output-gradient second moments to reduce the cost of storing and expressing curvature.

Across several benchmark datasets spanning code generation, mathematical reasoning, planning, and molecule generation, BAYESER substantially reduces expected calibration error (ECE) while preserving—and in several settings improving—task accuracy relative to commonly used confidence-based decoding schemes. Our results show that prior precision and the number of Laplace-LoRA layers govern a three-way trade-off among calibration, task performance, and decoding cost. We further find that a posterior constructed on one code-generation dataset can reduce calibration error on another without refitting, hinting that the learned uncertainty signal may transfer across related tasks.

2 BACKGROUND

We first describe background information on a common decoding scheme in MDLMs, Laplace-LoRA, and expected calibration error (ECE) for evaluating calibration performance.

2.1 MASKED DIFFUSION DECODING

MDLMs generate text by repeatedly predicting and revealing masked tokens (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). We denote a partially decoded, noisy sequence of length L by $\boldsymbol{x}_t^{(1:L)}$, where $\boldsymbol{x}_t^{(l)}$ is a one-hot vector of length $|\mathcal{V}|$ (vocabulary size), representing the l-th position of the sequence with one entry equaling 1 and the rest equaling zero. We also denote which position has masked tokens at time t by $\mathcal{M}_t = \{l : \boldsymbol{x}_t^{(l)} = [\text{MASK}]\}$. We denote a pre-trained model by $f_{\boldsymbol{\theta}_0}$ with fixed trained parameter values $\boldsymbol{\theta}_0$.

In MDLMs, at each time step t, the trained model $f_{\boldsymbol{\theta}_0}$ outputs the token probability over the vocabulary for each position in the sequence: $p_{\boldsymbol{\theta}_0}^{(l)}(\boldsymbol{v}|\boldsymbol{x}_t^{(1:L)})$. The base token is selected by choosing the argmax of this probability: $\hat{y}_t^{(l)} = \arg\max_{\boldsymbol{v}} p_{\boldsymbol{\theta}_0}^{(l)}(\boldsymbol{v}|\boldsymbol{x}_t^{(1:L)})$ and confidence is: $c_t^{(l)} = p_{\boldsymbol{\theta}_0}^{(l)}(\hat{y}_t^{(l)}|\boldsymbol{x}_t^{(1:L)})$.

Confidence-based decoding. MDLMs commonly decode tokens via confidence-based decoding. Given a threshold τ, at each denoising step t, K_t most confident base tokens are selected by: $K_t =$

$\max\left(1, \sum_{l \in \mathcal{M}_t} \mathbb{1}_{\{c_t^{(l)} \geq \tau\}}\right)$, where $\mathbb{1}$ is an indicator function which returns 1, if $\{c_t^{(l)} \geq \tau\}$, and returns 0, if not. The minimum of one ensures that at least one token is unmasked at a time.

2.2 LAPLACE-LORA

LoRA (Low-Rank Adaptation) parameterizes the update of each of the Q, K, V matrices of a transformer architecture at each layer through two low-rank matrices (Hu et al., 2021) (omitting the specification of Q, K, V or layer index for simplicity): $\boldsymbol{W} = \boldsymbol{W}^0 + \frac{\alpha}{r}\boldsymbol{B}\boldsymbol{A}$, with $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$, where r is a rank of the low-rank matrices and α is a scaling hyper-parameter.

Laplace-LoRA To gain posterior uncertainty about the model, Laplace-LoRA first optimizes the LoRA adapter parameters $\boldsymbol{A}$ and $\boldsymbol{B}$ on downstream tasks, then places a Gaussian distribution over $\boldsymbol{A}$ and $\boldsymbol{B}$, while keeping the underlying weights $\boldsymbol{W}^0$ fixed (Yang et al., 2024). Laplace-LoRA uses the *Laplace* approximation to estimate the covariance of the adapter parameters by learning the curvature of the negative log posterior at a posterior mode (MacKay, 1992). For decent-sized large language models, computing the exact curvature is intractable. Laplace-LoRA proposes two levels of approximations: (a) rather than computing the exact curvature, it uses the Fisher information, $\boldsymbol{F} := \sum_{n=1}^{N} \mathbb{E}_{y \sim P(\cdot | f_{\boldsymbol{\theta}}(\mathbf{x}_n))} \left[\nabla_{\boldsymbol{\theta}} \log P\left(y \mid f_{\boldsymbol{\theta}}(\mathbf{x}_n)\right) \left(\nabla_{\boldsymbol{\theta}} \log P\left(y \mid f_{\boldsymbol{\theta}}(\mathbf{x}_n)\right)\right)^{\top}\right]$; (b) furthermore, the Fisher information is computed on each linear layer separately using *Kronecker-factored approximate curvature (KFAC)*, forming a block structure of the Fisher information matrix. KFAC represents each layer's Fisher information using a Kronecker product of input covariance and output covariance, reducing the cost of storing and manipulating curvature (Martens & Grosse, 2015; Ritter et al., 2018). Our method, BAYESER, also specifies the curvature using these approximations, but without fine-tuning adapters, but at a fixed checkpoint as shown in Section 3.1.

2.3 CALIBRATION AND SEQUENCE-LEVEL CONFIDENCE

Expected calibration error (ECE) (Naeini et al., 2015) measures how well the model's prediction accuracy matches its confidence. Large ECE indicates a mismatch between confidence and accuracy, reflecting overconfidence or underconfidence. With N number of samples, predicted labels $\hat{y}_n$ for each sample n, and true label y_n, ECE groups confidences into M bins (each bin is $\mathcal{B}_m$) and measures

$$ECE = \sum_{m=1}^{M} \frac{|\mathcal{B}_m|}{N} \mid acc(\mathcal{B}_m) - conf(\mathcal{B}_m) \mid, \tag{1}$$

where $acc(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|} \sum_{n \in \mathcal{B}_m} \mathbf{1}(\hat{y}_n = y_n), conf(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|} \sum_{n \in \mathcal{B}_m} \hat{p}_n$, and $\hat{p}_n$ is the model's confidence for sample n: $\hat{p}_n = \max_c p(y_n = c|x_n)$. Directly applying ECE to text *sequences* is not straightforward. In Section 3.3, we show two ways to evaluate ECE on the generated sequence.

Another commonly used probabilistic prediction metric which we will also use is the *Brier score* (Brier, 1950). For our binary task-correctness evaluation, $\text{BS} = \frac{1}{N} \sum_{n=1}^{N} (c_n - z_n)^2$, where c_n is the sequence confidence score and $z_n \in \{0, 1\}$ indicates task correctness.

3 METHOD: BAYESER

A decoder in MDLMs makes three decisions at every denoising step: *which positions to commit*, *which tokens to write*, and *how many positions to commit*. BAYESER makes all three decisions from a checkpoint-centered posterior predictive distribution. Specifically, it ranks positions by posterior predictive entropy, selects token identities from the posterior predictive mean, and determines the commitment budget from posterior confidence.

BAYESER consists of two stages: (1) First, we fit the posterior distribution over LoRA adapter parameters using training data, from which we obtain the posterior samples; and (2) Then, using the posterior samples, we compute the predictive posterior entropy at every denoising step t to determine how many tokens to unmask and which positions to fill with which token. We describe each of these two stages below.

3.1 PHASE I: CURVATURE-ONLY LAPLACE-LORA POSTERIOR

Checkpoint-centered LoRA parameterization. We place LoRA adapters on selected query and value matrices in the selected J transformer layers. Below, we omit the indices for Q and V and show only the layer index j, since the LoRA operation is the same for Q and V.

For a selected pretrained matrix $\boldsymbol{W}_{j,0} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the effective weight is

$$\boldsymbol{W}_j(\phi_j) = \boldsymbol{W}_{j,0} + \frac{\alpha}{r}\boldsymbol{B}_j\boldsymbol{A}_j, \qquad \boldsymbol{A}_j \in \mathbb{R}^{r \times d_{\text{in}}}, \quad \boldsymbol{B}_j \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{2}$$

where we let $\boldsymbol{\phi}_0 = \{\boldsymbol{A}_{j,0}, \boldsymbol{B}_{j,0}\}_j$, and $\boldsymbol{B}_{j,0} = 0$, with $\boldsymbol{A}_{j,0}$ given by the standard LoRA initialization. Because $\boldsymbol{B}_{j,0}\boldsymbol{A}_{j,0} = 0$, the posterior-mean model is exactly the released checkpoint:

$$f_{\boldsymbol{\theta}_0,\boldsymbol{\phi}_0} = f_{\boldsymbol{\theta}_0}.$$

Unlike standard Laplace-LoRA, ***we do not optimize the adapter means on a downstream objective.*** We use the training data only to estimate local curvature.

Approximating Fisher information with KFAC For each LoRA adapter matrix index m, where m can vary between layers and Q and V, we denote its input activation for each observation $\widetilde{\mathbf{x}}_n$ by $h_{m,n}$, and let

$$g_{m,n} = \nabla_{a_{m,n}} \log p_{\boldsymbol{\theta}_0,\boldsymbol{\phi}_0}\big(Y_n \mid \widetilde{\mathbf{x}}_n\big)$$

be the gradient with respect to the corresponding pre-activation $a_{m,n}$, where Y_n is distributed according to the model predictive distribution. We approximate the Fisher information matrix separately for each adapter matrix, using Kronecker-factored approximate curvature (KFAC):

$$F_m \approx N_m \left(G_m \otimes C_m\right), \quad \text{where } C_m = \mathbb{E}\big[h_m h_m^\top\big], \text{ and } G_m = \mathbb{E}\big[g_m g_m^\top\big], \tag{3}$$

where N_m counts the contributing source-position observations represented by the sampled rows for block m, rather than calibration sequences or masked targets. The factor C_m captures correlations among matrix inputs, whereas G_m captures correlations among output-gradient coordinates.

Using an isotropic Gaussian prior precision λ, we construct the checkpoint-centered, Laplace-style Gaussian approximation

$$q_\lambda(\boldsymbol{\phi}) = \mathcal{N}\left(\boldsymbol{\phi}_0,\, [\text{blockdiag}(F_1, \ldots, F_R) + \lambda I]^{-1}\right), \tag{4}$$

where R is the total number of individual LoRA adapter matrices across the J we consider. For instance, if we have Q, V matrices per layer and apply LoRA to 10 layers, the posterior covariance consists of small, manageable 40 blocks (since each Q and V has two adapter matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$). Together, these blocks define a single joint Gaussian over all selected adapter parameters. Blocks are independent across matrices, projections, and layers; however, within-matrix KFAC correlations remain, and the leading eigendirections of large factors are retained. Directions omitted by the truncated eigenspace retain prior variance.

Uncertainty in the LoRA factors At the initialization $\boldsymbol{\phi}_0$, we have $\boldsymbol{B}_{j,0}$ initialized as 0. Changing $\boldsymbol{A}_{j,0}$ alone therefore does not affect $\boldsymbol{W}_j(\boldsymbol{\phi}_j)$, so the gradient with respect to $\boldsymbol{A}_j$ and its Fisher block is zero. Under our Gaussian approximation, this gives: $\text{vec}(\boldsymbol{A}_j) \sim \mathcal{N}\big(\text{vec}(\boldsymbol{A}_{j,0}), \lambda^{-1}\boldsymbol{I}\big)$. Therefore, uncertainty in $\boldsymbol{A}_j$ is entirely prior-controlled, while uncertainty in $\boldsymbol{B}_j$ depends on both the calibration curvature and the prior. At prediction time, each posterior draw $\boldsymbol{\phi}^{(s)} \sim q_\lambda(\boldsymbol{\phi})$ gives a corresponding pair of sampled LoRA matrices $\boldsymbol{A}_j^{(s)}$ and $\boldsymbol{B}_j^{(s)}$. We use these matrices to construct the sampled model weights: $\boldsymbol{W}_j(\boldsymbol{\phi}_j^{(s)}) = \boldsymbol{W}_{j,0} + \frac{\alpha}{r}\boldsymbol{B}_j^{(s)}\boldsymbol{A}_j^{(s)}$. Although $\boldsymbol{A}_j$ has zero local Fisher curvature at initialization, its sampled variation still affects predictions because $\boldsymbol{B}_j^{(s)}$ is generally non-zero. Writing $\boldsymbol{A}_j^{(s)} = \boldsymbol{A}_{j,0} + \delta\boldsymbol{A}_j^{(s)}$ and $\boldsymbol{B}_j^{(s)} = \delta\boldsymbol{B}_j^{(s)}$, the sampled update includes the bilinear term $\frac{\alpha}{r}\delta\boldsymbol{B}_j^{(s)}\delta\boldsymbol{A}_j^{(s)}$ in addition to $\frac{\alpha}{r}\delta\boldsymbol{B}_j^{(s)}\boldsymbol{A}_{j,0}$. Thus, predictive variation reflects both data-informed curvature and prior-controlled variation in locally unconstrained directions.

Algorithm 1 BAYESER

Require: Pre-trained model: $f_{\boldsymbol{\theta}_0}$, confidence threshold: $\tau_{\mathrm{B}} \in [0,1]$, how many posterior samples to draw: $S \geq 1$, initial sequence $\mathbf{x}_T^{(1:L)}$, and the number of selected tokens per position: $K_v \geq 1$

Phase I: Fit the parameter posterior q_λ using Algorithm 2
  Then, sample parameters $\boldsymbol{\theta}^{(s)} \sim q_\lambda$ for $s = 1, \ldots, S$
Phase II:
$t \leftarrow T$; $\mathcal{C} \leftarrow \emptyset$; $\mathcal{M}_T \leftarrow \{l : \boldsymbol{x}_T^{(l)} = [\mathrm{MASK}]\}$
while $\mathcal{M}_t \neq \emptyset$ **do**
  Run base model $f_{\boldsymbol{\theta}_0}(\mathbf{x}_t^{(1:L)})$ to obtain token logits and confidences $c_t^{(l)}$ for each position l
  For each masked position l, define a set $\mathcal{V}_t^{(l)}$ containing the K_v highest-scoring tokens
  for $s = 1, \ldots, S$ **do**
    Given each posterior sample, run the sampled model $f_{\boldsymbol{\theta}^{(s)}}(\boldsymbol{x}_t^{(1:L)})$
    $p_{t,s}^{(l)} \leftarrow$ softmax of sampled-model logits over tokens in $\mathcal{V}_t^{(l)}$
  end for
  $\bar{p}_t^{(l)} \leftarrow S^{-1} \sum_s p_{t,s}^{(l)}$
  $H_t^{(l)} \leftarrow -\sum_{v \in \mathcal{V}_t^{(l)}} \bar{p}_t^{(l)}(v) \log \bar{p}_t^{(l)}(v)$
  $\hat{y}_t^{(l)} \leftarrow \arg\max_{v \neq [\mathrm{MASK}]} \bar{p}_t^{(l)}(v)$; and $c_t^{(l)} \leftarrow \bar{p}_t^{(l)}(\hat{y}_t^{(l)})$
  $K_t \leftarrow \max(1, \sum_{l \in \mathcal{M}_t} \mathbb{1}_{c_t^{(l)} \geq \tau_{\mathrm{B}}})$
  $\mathcal{P}_t \leftarrow$ the K_t masked positions with lowest entropy $H_t^{(l)}$
  Record $(t, l, \hat{y}_t^{(l)}, c_t^{(l)})$ in $\mathcal{C}$ for each $l \in \mathcal{P}_t$
  $\boldsymbol{x}_{t-1}^{(l)} \leftarrow \hat{y}_t^{(l)}$ for each $l \in \mathcal{P}_t$
  $t \leftarrow t - 1$
  $\mathcal{M}_t \leftarrow \{l : \boldsymbol{x}_t^{(l)} = [\mathrm{MASK}]\}$
end while
return Updated sequence $\boldsymbol{x}_t^{(l)}$ and saved tokens and their probabilities $\mathcal{C}$

3.2 PHASE II: POSTERIOR PREDICTIVE ENTROPY FOR POSITION ORDERING

At step t, the pre-trained model produces base logits $\ell_{0,t}^{(l)} \in \mathbb{R}^{|\mathcal{V}|}$ for each location l. For each position, we define a candidate set containing the top-K_v highest-scoring non-masked tokens, i.e., $\mathcal{V}_t^{(l)} = \mathrm{TopK}_{v \in \mathcal{V} \setminus \{\texttt{[MASK]}\}} \left(\ell_{0,t}^{(l)}(v); K_v \right)$. For S independent posterior draws $\boldsymbol{\phi}^{(s)} \sim q_\lambda(\boldsymbol{\phi})$, let $\ell_{t,s}^{(l)}$ be the sampled-model logit. We first normalize each draw over the candidate set $\mathcal{V}_t^{(l)}$: $p_{t,s}^{(l)}(v) = \frac{\exp\left(\ell_{t,s}^{(l)}(v)\right)}{\sum_{u \in \mathcal{V}_t^{(l)}} \exp\left(\ell_{t,s}^{(l)}(u)\right)}$, for $v \in \mathcal{V}_t^{(l)}$. Importantly, we average probabilities rather than logits: $\overline{p}_t^{(l)}(v) = \frac{1}{S} \sum_{s=1}^{S} p_{t,s}^{(l)}(v)$. Our position-ordering criterion is the posterior predictive entropy given by

$$H_t^{\mathrm{pred},(l)} = H\left(\overline{p}_t^{(l)}\right) = -\sum_{v \in \mathcal{V}_t^{(l)}} \overline{p}_t^{(l)}(v) \log \overline{p}_t^{(l)}(v). \tag{5}$$

Low entropy indicates that the posterior predictive distribution concentrates on a small number of candidates, making the position comparatively safe to commit. High entropy indicates that no clear token is supported under the current partially masked context, so the position should be deferred.

Commitment Budgets Given a posterior-confidence threshold τ_B, BAYESER sets its per-step commitment budget to $K_t = \max\left(1, \sum_{l \in \mathcal{M}_t} \mathbb{1}_{\left\{c_t^{(l)} \geq \tau_B\right\}}\right)$. It then chooses the K_t positions with the lowest posterior predictive entropy:

$$\mathcal{P}_t = \underset{\substack{\mathcal{P} \subseteq \mathcal{M}_t \\ |\mathcal{P}| = K_t}}{\arg\min} \sum_{l \in \mathcal{P}} H_t^{\mathrm{pred}(l)}. \tag{6}$$

For each $l \in \mathcal{P}_t$, we commit $\boldsymbol{x}_{t-1}^{(l)} = \widehat{y}_t^{(l)}$, where $\hat{y}_t^{(l)} \leftarrow \arg\max_{v \neq [\mathrm{MASK}],} \bar{p}_t^{(l)}(v)$, while all unselected positions remain unchanged. Algorithm 1 summarizes our algorithm.

3.3 Evaluating Sequence-Level Confidence

To evaluate ECE on a generated sequence, we first compute the geometric mean as a length-normalized summary of commitment confidence, for a token scope $\mathcal{S} \subseteq \mathcal{C}$:

$$\pi_{\text{seq}}(\mathcal{S}) = \exp\left(\frac{1}{|\mathcal{S}|}\sum_{(t,l)\in\mathcal{S}} \log c_t^{(l)}\right). \tag{7}$$

using the token probability available at the moment of commitment, before the committed token becomes part of the context for later denoising steps, denoted by $\mathcal{C} = \left\{\left(t, l, y_t^{(l)}, c_t^{(l)}\right)\right\}$, with $c_t^{(l)} = \bar{p}_t^{(l)}\left(y_t^{(l)}\right)$. This aggregation is a candidate confidence estimator, which we use for $\pi_{\text{seq}}(\mathcal{S}_n)$ as the confidence $\hat{p}_n$ for each generated output and compute ECE using equation 1, with correctness determined by the benchmark's task criterion.

In Section 5, we report the score over two scopes: all generated-content tokens (ECE_{gen}) and task-specific final-answer tokens (ECE_{ans}). Note that low ECE_{gen} does not establish that every sentence in a reasoning trace is correct. It means that confidence aggregated over that trace agrees, at the resolution measured, with the frequency of final task success. The generated-content and final-answer variants use different token scopes but share the same task-level correctness event.

This geometric mean is different from Li et al. (2026), where their ECE is an arithmetic mean of token probabilities extracted from a single final forward pass under teacher forcing on the completed sequence. Our geometric mean summarizes the model's confidence at the commitment decisions made along the decoding trajectory.

3.4 Mutual Information and Base Entropy

Mutual Information For comparison, one might consider the token–parameter mutual information (MI) given by $I_t^{(l)} = H_t^{\text{pred}(l)} - \frac{1}{S}\sum_{s=1}^{S} H\left(p_{t,s}^{(l)}\right)$. MI isolates disagreement among posterior draws (Houlsby et al., 2011), but it can be small even when every draw assigns a diffuse token distribution. This makes MI insufficient for deferred commitment: posterior samples may agree that a position remains ambiguous because the current state does not yet provide enough context. We therefore use total posterior predictive entropy as the default ordering score and retain MI-based ordering as an ablation. We empirically compare MI and predictive entropy in Appendix E.

Base Entropy To distinguish the effect of entropy-based ordering from that of Bayesian marginalization, we can also compare our predictive entropy-based selection from base entropy given by: $H_{0,t}^{(l)} = H\left(p_{0,t}^{(l)}\right)$, where $p_{0,t}^{(l)}(v) = \frac{\exp\left(\ell_{0,t}^{(l)}(v)\right)}{\sum_{u\in\mathcal{V}_t^{(l)}}\exp\left(\ell_{0,t}^{(l)}(u)\right)}$. This baseline, Base Entropy, ranks positions by $H_{0,t}^{(l)}$ while retaining the model's tokens and confidence-based commitment budget.

4 Related Work

Decoding and unmasking policies. MaskGIT and Fast-dLLM (Chang et al., 2022; Wu et al., 2025) present confidence-based decoders ; Deferred Commitment Decoding adapts the active window (Shu et al., 2026), and Entropy based sampler uses a point estimate of entropy to control parallel commitments (Ben-Hamu et al., 2025). Lookahead Unmasking and the Information-Gain (InfoGain) Sampler evaluate candidate successor states (Lee et al., 2025; Yang et al., 2026). Other approaches learn unmasking policies through reinforcement learning (Jazbec et al., 2026) or derive ordering corrections from terminal rewards, as in DPRM (Bu et al., 2026). None of the above methods uses *posterior predictive uncertainty* to choose positions, tokens, and commitment counts.

Bayesian inference for pretrained models. Laplace-LoRA fits a posterior around fine-tuned adapters (Yang et al., 2024), BLoB (Wang et al., 2024) learns a variational posterior during adapter training , and Training-Free Bayesianization (Shi et al., 2025) assigns uncertainty to already-trained

Table 1: Evaluation on mathematical reasoning and programming tasks. Accuracy is reported in percent; ECE and Brier values and their standard deviations are multiplied by 100. Subscripts are the standard deviations of three seeds on the corresponding column scale. Numbers in parentheses are relative percentage improvements over confidence-thresholding decoding. Bold values indicate the best result and underlined values indicate the second-best result per metric within each benchmark.

Dataset	Method	Accuracy (%) ↑	ECE_{gen} (×100) ↓	ECE_{ans} (×100) ↓	$\text{Brier}_{\text{gen}}$ (×100) ↓
MBPP	Threshold (Fast-dllm)	**57.60**	29.18	31.24	31.85
	Base Entropy	57.00 (−1.04)	28.99 (+0.65)	31.53 (−0.93)	31.23 (+1.97)
	Info-gain	$54.00_{1.71}$ (−6.25)	$31.60_{1.71}$ (−8.29)	$33.73_{1.71}$ (−7.97)	$33.15_{1.23}$ (−4.07)
	BAYESER (best accuracy)	$\underline{57.53}_{0.12}$ (−0.12)	$\underline{23.54}_{0.11}$ (+19.33)	$\underline{25.70}_{0.09}$ (+17.73)	$\underline{28.40}_{0.02}$ (+10.84)
	BAYESER (best calibration)	$56.73_{1.01}$ (−1.51)	$\mathbf{16.32}_{1.21}$ (+44.07)	$\mathbf{18.17}_{1.02}$ (+41.84)	$\mathbf{25.57}_{0.36}$ (+19.73)
HumanEval	Threshold (Fast-dllm)	54.88	30.90	30.59	33.27
	Base Entropy	**57.32** (+4.45)	27.25 (+11.81)	26.71 (+12.68)	31.80 (+4.42)
	Info-gain	$51.22_{0.61}$ (−6.67)	$33.68_{0.57}$ (−9.00)	$33.28_{0.53}$ (−8.79)	$35.82_{0.82}$ (−7.66)
	BAYESER (best accuracy)	$\underline{56.30}_{3.01}$ (+2.59)	$\underline{24.77}_{3.04}$ (+19.84)	$\underline{24.61}_{3.13}$ (+19.55)	$\underline{30.26}_{1.80}$ (+9.05)
	BAYESER (best calibration)	$55.08_{1.96}$ (+0.36)	$\mathbf{20.16}_{2.04}$ (+34.76)	$\mathbf{20.33}_{1.98}$ (+33.54)	$\mathbf{27.76}_{1.05}$ (+16.56)
GSM8K	Threshold (Fast-dllm)	76.19	10.01	17.91	17.46
	Base Entropy	**78.17** (+2.60)	$\underline{7.14}$ (+28.67)	$\underline{15.98}$ (+10.78)	**16.39** (+6.13)
	Info-gain	$69.70_{0.44}$ (−8.52)	$10.17_{0.18}$ (−1.60)	$18.80_{0.06}$ (−4.97)	$16.62_{0.13}$ (+4.81)
	BAYESER (best accuracy)	$\underline{77.98}_{0.38}$ (+2.35)	$7.17_{0.41}$ (+28.37)	$16.10_{0.20}$ (+10.11)	$\underline{16.53}_{0.18}$ (+5.31)
	BAYESER (best calibration)	$76.89_{0.52}$ (+0.92)	$\mathbf{6.50}_{0.55}$ (+35.06)	$\mathbf{15.75}_{0.52}$ (+12.06)	$16.98_{0.26}$ (+2.73)
MATH500	Threshold (Fast-dllm)	$\underline{42.40}$	$\underline{45.50}$	$\underline{50.08}$	$\underline{45.33}$
	Base Entropy	40.00 (−5.66)	46.27 (−1.69)	53.31 (−6.45)	45.77 (−0.98)
	Info-gain	$40.73_{0.58}$ (−3.94)	$45.96_{0.70}$ (−1.01)	$52.04_{1.13}$ (−3.91)	$45.72_{0.46}$ (−0.87)
	BAYESER (best accuracy)	$\mathbf{43.53}_{1.17}$ (+2.67)	$\mathbf{42.71}_{1.12}$ (+6.13)	$\mathbf{49.55}_{1.09}$ (+1.06)	$\mathbf{43.19}_{0.84}$ (+4.72)

adapters. Building on structured Laplace approximations (Ritter et al., 2018; Daxberger et al., 2021), our curvature-only construction fits uncertainty around zero-update adapters at the released checkpoint. This preserves the deterministic mean model and allows us to study posterior prediction without conflating it with downstream fine-tuning.

Uncertainty in generation. For continuous diffusion, BayesDiff propagates uncertainty using last-layer Laplace (Kou et al., 2023), while Jazbec et al. (2025) develop a Bayesian measure for assessing synthetic samples. Our focus is utilizing the uncertainty at unresolved discrete token positions to guide commitment during generation.

5 EXPERIMENTS

5.1 SETUP

We evaluate Dream-v0-Instruct-7B (Ye et al., 2025) on code generation (MBPP, HumanEval), mathematical reasoning (GSM8K, MATH500), and planning (4×4 Sudoku, Countdown). HumanEval transfers the MBPP posterior without refitting. We also include LLaDA results in Appendix D. We additionally evaluate molecular scaffold decoration with GenMol V2. Detailed configurations are available in Appendix A.2.

We compare against confidence-threshold decoding, Base Entropy, and Info-gain (Yang et al., 2026). Info-gain runs use $N = 8$ candidate paths with fixed $K = 2$ tokens per step. We report task accuracy and sequence-level calibration.

We compare the sequence scores in Eq. 7 with binary task correctness using 15 equal-width bins for ECE (Section 2.3) and report binary Brier scores. We compute scores over generated-content tokens and final-answer tokens, yielding ECE_{gen} and ECE_{ans}. For molecule generation, ECE is evaluated against chemical validity, while quality yield measures the fraction of attempts meeting the QED and SA criteria in Table 3.

Table 2: Evaluation on planning tasks.

Dataset	Method	Accuracy (%) ↑	ECE_{gen} ($\times 100$) ↓	ECE_{ans} ($\times 100$) ↓	$\text{Brier}_{\text{gen}}$ ($\times 100$) ↓
Sudoku	Threshold (Fast-dllm)	**97.60**	3.13	3.13	**2.24**
	Base Entropy	**97.60** (+0.00)	2.98 (+4.79)	2.98 (+4.79)	2.28 (−1.54)
	Info-gain	$96.07_{0.50}$ (−1.57)	$\mathbf{2.54}_{0.08}$ (+18.85)	$\mathbf{2.54}_{0.08}$ (+18.85)	$3.39_{0.37}$ (−51.35)
	BAYESER (best accuracy)	$97.53_{0.31}$ (−0.07)	$3.96_{0.03}$ (−26.52)	$3.96_{0.03}$ (−26.52)	$2.40_{0.17}$ (−7.00)
	BAYESER (best calibration)	$97.40_{0.20}$ (−0.20)	$2.99_{0.27}$ (+4.47)	$2.99_{0.27}$ (+4.47)	$2.39_{0.16}$ (−6.56)
Countdown	Threshold (Fast-dllm)	18.90	66.46	61.54	58.56
	Base Entropy	23.90 (+26.46)	60.07 (+9.61)	54.67 (+11.16)	53.21 (+9.14)
	Info-gain	$\mathbf{41.47}_{2.03}$ (+119.42)	$45.63_{1.77}$ (+31.34)	$43.56_{1.08}$ (+29.22)	$42.40_{1.16}$ (+27.60)
	BAYESER (best accuracy)	$32.13_{0.67}$ (+70.00)	$28.98_{0.21}$ (+56.39)	$\mathbf{27.92}_{0.68}$ (+54.63)	$26.71_{0.18}$ (+54.39)
	BAYESER (best calibration)	$27.80_{0.78}$ (+47.09)	$\mathbf{25.70}_{0.50}$ (+61.33)	$30.24_{0.81}$ (+50.86)	$\mathbf{22.83}_{0.08}$ (+61.01)

Table 3: Evaluation on GenMol V2 Scaffold Decoration. Quality Yield measures the percentage of generated attempts satisfying both drug-likeness (QED ≥ 0.6) and synthetic accessibility (SA ≤ 4.0).

Method	Quality Yield (%) ↑	Frag. Valid (%) ↑	QED ↑	SA ↓	ECE ↓
Native GenMol	11.78 ± 0.69	99.33 ± 0.33	0.6369 ± 0.0104	4.3932 ± 0.0362	45.08 ± 0.29
Base Entropy	14.89 ± 0.69	99.89 ± 0.19	0.6599 ± 0.0011	4.2809 ± 0.0106	35.40 ± 0.59
Info-gain	3.00 ± 0.67	$\mathbf{100.00 \pm 0.00}$	$\mathbf{0.6704 \pm 0.0206}$	$\mathbf{4.1528 \pm 0.0595}$	$\mathbf{24.56 \pm 1.71}$
DPRM (Bu et al., 2026)	12.44 ± 0.69	$\mathbf{100.00 \pm 0.00}$	0.6619 ± 0.0117	4.2682 ± 0.0742	30.62 ± 1.33
BAYESER (Ours)	$\mathbf{15.33 \pm 0.33}$	99.67 ± 0.00	0.6602 ± 0.0006	4.2900 ± 0.0062	35.36 ± 0.73

5.2 MAIN RESULTS

5.2.1 MATHEMATICAL REASONING AND PROGRAMMING

Table 1 shows that BAYESER improves sequence-confidence calibration on several tasks, with task-dependent changes in accuracy. On MBPP, the accuracy-oriented setting approximately preserves threshold accuracy (57.53% versus 57.60%) while reducing generated-content ECE from 0.2918 to 0.2354. The calibration-oriented setting lowers ECE further to 0.1632 at 56.73% accuracy. Transferring the MBPP posterior to HumanEval also reduces ECE and Brier scores without refitting: generated-content ECE falls from 0.3090 to 0.2016 in the calibration-oriented setting, with 55.08% accuracy versus 54.88% for threshold. Base Entropy retains the highest HumanEval accuracy at 57.32%.

On GSM8K, Base Entropy achieves the highest accuracy (78.17%) and lowest generated-content Brier score, while the calibration-oriented BAYESER setting achieves the lowest ECE in both scopes. On MATH500, BAYESER improves accuracy from 42.40% to 43.53% and reduces ECE and Brier scores. The accuracy–calibration tradeoff will be further elaborated in Section 5.3.

5.2.2 PLANNING

Table 2 shows contrasting outcomes on the two planning tasks. Sudoku offers little advantage for Bayesian decoding: threshold and Base Entropy reach 97.60% accuracy, and neither posterior setting improves on threshold's Brier score. Countdown exhibits the strongest accuracy–calibration trade-off. The accuracy-oriented BAYESER setting reaches 32.13% accuracy, exceeding threshold (18.90%) and Base Entropy (23.90%), but falling below Info-gain (41.47%). Its generated-content ECE and Brier score are lower than Info-gain's: 0.2898 versus 0.4563 and 0.2671 versus 0.4240, respectively. The calibration-oriented setting further reduces generated-content ECE to 0.2570 at 27.80% accuracy.

5.2.3 MOLECULAR GENERATION

On GenMol V2 scaffold decoration (Table 3), BAYESER achieves the highest mean quality yield at 15.33%, compared with 11.78% for native GenMol and 14.89% for Base Entropy. Its ECE is lower than native GenMol's (35.36 versus 45.08 on the table's scale), but nearly identical to Base Entropy's (35.40). These metrics capture different aspects of generation: ECE concerns chemical validity, whereas quality yield requires both drug-likeness and synthetic accessibility.

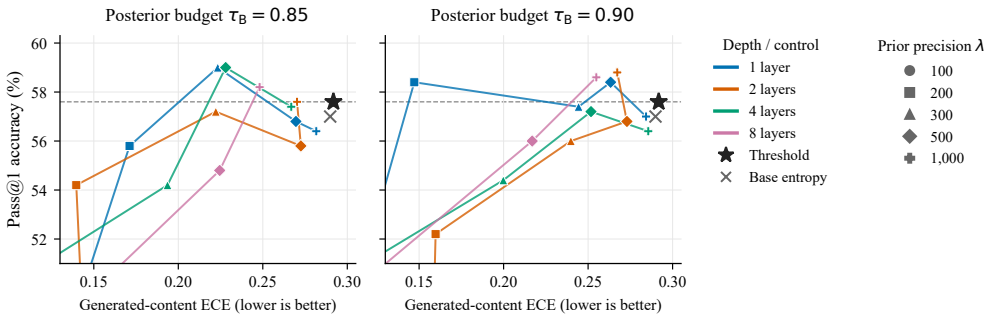

Figure 1: Accuracy–calibration trade-off for MBPP with different hyperparameter choices.

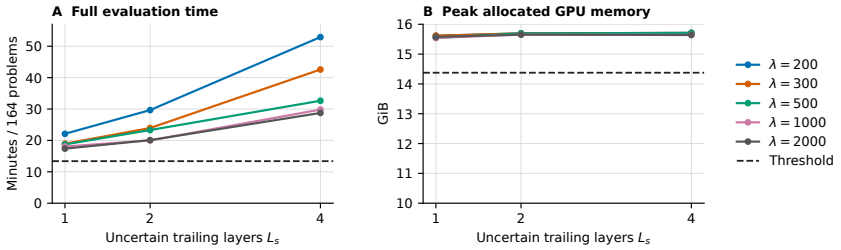

Figure 2: Dream HumanEval cost versus uncertain depth.

5.3 Hyperparameter sensitivity and accuracy–calibration trade-offs

We additionally evaluate how prior precision λ, the number of uncertain trailing layers L_s, and the commitment budget jointly affect the accuracy–calibration tradeoff. In Figure 1, each panel shows five prior precisions ($\lambda \in \{100, 200, 300, 500, 1000\}$) at four depths. Intermediate precision provides operating points with lower ECE and comparable or higher accuracy. Further weakening the prior can reduce ECE while substantially degrading accuracy.

Depth and commitment budget also interact. A single uncertain layer reaches 59.0% at ECE 0.223 for $\tau_B = 0.85$, $\lambda = 300$, and 58.4% at ECE 0.147 for $\tau_B = 0.90$, $\lambda = 200$. These examples show that a shallow posterior can offer a favorable accuracy–calibration trade-off and that additional uncertain layers do not consistently improve it.

5.4 Computational efficiency

Figure 2 shows HumanEval cost on A100 GPUs with $S = 10$ samples and $\tau_B = 0.90$. Across tested priors, increasing uncertain depth from one to four layers raises evaluation time from 17.4–22.1 to 28.8–52.9 minutes, versus 13.4 minutes for threshold. Peak allocated memory remains within 15.54–15.71 GiB, versus 14.38 GiB for threshold. Deeper posteriors require longer sampled forward passes, while weaker priors can prolong decoding. Shallow posteriors therefore limit runtime overhead. Memory varies little with depth.

6 Conclusion and Discussion

We introduced BayesER, a post-hoc Bayesian decoding scheme that uses posterior predictive uncertainty to guide position ordering, token selection, and commitment budgets in masked diffusion language models. Its curvature-only Laplace-LoRA construction preserves the released checkpoint at the posterior mean and requires no adapter fine-tuning. The experiments show that posterior prediction can improve sequence-confidence calibration while preserving or improving task performance at selected operating points. In future work, we plan to extend the single Gaussian posterior to multiple Gaussian posteriors across different adapter-parameter layers to better capture parameter uncertainties and enhance our entropy-based decoding scheme.

Reproducibility Statement

The pretrained checkpoints used in this work (Dream-v0-Instruct-7B, LLaDA-8B-Instruct, and GenMol V2) and the MBPP, HumanEval, GSM8K, and MATH500 benchmarks are publicly available. Planning evaluations use fixed public Sudoku and Countdown test sets; separate synthetic training sets are generated deterministically with test inputs excluded. Molecular-generation experiments use public GenMol fragments and calibration data drawn from the training split of `datamol-io/safe-gpt`. Section 3 and Appendix A describe the method, posterior construction, and predictive computation. Section 5 describes the benchmarks and metrics; Tables 4 and 5 specify the Dream configurations, and Appendix D reports the additional LLaDA evaluation settings. Code, posterior artifacts, data-preparation scripts, and evaluation configurations will be open-sourced upon publication.

Use of AI Statement

Generative AI tools and large language models were used to assist with code implementation, finding related work, manuscript editing, grammar polishing, and typesetting. AI-assisted code was checked by the authors. The authors take responsibility for the final manuscript, including its code, experimental results, and claims.

Ethics Statement

This work studies uncertainty-aware decoding using pretrained models and public benchmark datasets. Improved benchmark calibration does not guarantee reliable predictions under distribution shift or safe deployment in high-stakes settings. The molecular-generation experiments evaluate computational properties; the reported metrics do not establish biological efficacy, toxicity, or safety. Generated molecules would require independent expert assessment and experimental validation before practical use.

References

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.24857.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

Dake Bu, Wei Huang, Andi Han, Si Wu, Hau-San Wong, Qingfu Zhang, Taiji Suzuki, and Atsushi Nitanda. DPRM: A plug-in Doob h-transform-induced token-ordering module for discrete diffusion models, 2026. URL https://arxiv.org/abs/2604.24357.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux — effortless Bayesian deep learning. In *Advances in Neural Information Processing Systems*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011. URL https://arxiv.org/abs/1112.5745.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Metod Jazbec, Eliot Wong-Toi, Guoxuan Xia, Dan Zhang, Eric Nalisnick, and Stephan Mandt. Generative uncertainty in diffusion models. *arXiv preprint arXiv:2502.20946*, 2025. URL https://arxiv.org/abs/2502.20946.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2512.09106.

Siqi Kou, Lei Gan, Dequan Wang, Chongxuan Li, and Zhijie Deng. BayesDiff: Estimating pixel-wise uncertainty in diffusion via Bayesian inference. *arXiv preprint arXiv:2310.11142*, 2023. URL https://arxiv.org/abs/2310.11142.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models, 2025. URL https://arxiv.org/abs/2511.05563.

Yansi Li, Gongshen Liu, and Zhuosheng Zhang. The confidence paradox: Unveiling the latent discriminative power of diffusion large language models in mathematical reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43179–43196, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.2142. URL https://aclanthology.org/2026.findings-acl.2142/.

David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2901–2907, 2015. doi: 10.1609/aaai.v29i1.9602.

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. URL https://arxiv.org/abs/2502.09992.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

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

Haizhou Shi, Yibin Wang, Ligong Han, Huan Zhang, and Hao Wang. Training-free Bayesianization for low-rank adapters of large language models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2412.05723.

Yingte Shu, Yuchuan Tian, Chao Xu, Yunhe Wang, and Hanting Chen. Deferred commitment decoding for diffusion language models with confidence-aware sliding windows. *arXiv preprint arXiv:2601.02076*, 2026. URL https://arxiv.org/abs/2601.02076.

Yibin Wang, Haizhou Shi, Ligong Han, Dimitris Metaxas, and Hao Wang. BLoB: Bayesian low-rank adaptation by backpropagation for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 67758–67794, 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. URL https://arxiv.org/abs/2505.22618.

Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FJiUyzOF1m.

Kaisen Yang, Jayden Teoh, Kaicheng Yang, Yitong Zhang, and Alex Lamb. Improving sampling for masked diffusion models via information gain, 2026. URL https://arxiv.org/abs/2602.18176.

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

A Posterior and implementation details

A.1 Posterior fitting, sampling and computation

We use the notation of Section 3.1: $\boldsymbol{\theta}_0$ denotes the fixed pretrained parameters, $\boldsymbol{\phi}$ collects the LoRA adapter parameters, j indexes selected transformer layers, and m indexes individual adapter-matrix blocks. We fit model-Fisher curvature at $\boldsymbol{\phi}_0$, where the LoRA update is zero, using task-matched calibration prompts and suppressing the mask-token logit before softmax. Algorithm 2 summarizes the procedure. For posterior draws we use the exact factorization implied by Eq. 4. Let $\boldsymbol{\phi}_m$ be the vectorized adapter matrix for block m, with mean $\boldsymbol{\phi}_{0,m}$; here vec stacks rows, consistent with the ordering $G_m \otimes C_m$. Draw a matrix $z_m^{(s)}$ of independent standard normal entries in the shape of that adapter matrix and expand it in the retained basis, $z_m^{(s)} = \sum_{a,b} \xi_{m,ab}^{(s)} u_{m,a}^G (u_{m,b}^C)^\top + z_{m,\perp}^{(s)}$. The adapter draw is

$$\boldsymbol{\phi}_m^{(s)} = \boldsymbol{\phi}_{0,m} + \mathrm{vec}\left[\sum_{a,b} \frac{\xi_{m,ab}^{(s)}}{\sqrt{\lambda + N_m g_{m,a} c_{m,b}}} u_{m,a}^G (u_{m,b}^C)^\top + \frac{1}{\sqrt{\lambda}} z_{m,\perp}^{(s)}\right], \tag{8}$$

where $c_{m,b}$ and $g_{m,a}$ are the retained eigenvalues of C_m and G_m and N_m is the represented source-position observation count defined in the main text. The coefficients $\xi_{m,ab}^{(s)}$ are independent standard normal draws. The retained-basis perturbation is the data- and prior-informed component; the last term is the prior-only complement corresponding to truncated directions. Each block draws its own independent $z_m^{(s)}$, realizing the block-independence assumption above. Substituting Eq. 8 into Eq. 2 gives the nonlinear weight-space predictive $p^{(l)}_{\boldsymbol{\theta}_0,\boldsymbol{\phi}^{(s)}}(\cdot \mid \boldsymbol{x}_t^{(1:L)})$ at position l. Normalizing the sampled logits on $\mathcal{V}_t^{(l)}$ gives $p_{t,s}^{(l)}$ from Section 3.2.

Algorithm 2 Curvature-only KFAC Laplace-LoRA posterior

Require: Released model $f_{\boldsymbol{\theta}_0}$; calibration set $\mathcal{D}_{\mathrm{cal}}$; selected projections; LoRA rank r and scale α; KFAC rank k; prior precision λ; per-block row-sampling cap N_s

Wrap each selected projection as $\boldsymbol{W}_j(\boldsymbol{\phi}_j) = \boldsymbol{W}_{j,0} + \frac{\alpha}{r}\boldsymbol{B}_j\boldsymbol{A}_j$ with $\boldsymbol{B}_{j,0} = 0$ (Eq. 2); let $\boldsymbol{\phi}_0$ collect $\{\boldsymbol{A}_{j,0}, \boldsymbol{B}_{j,0}\}_j$, with projection indices omitted as in the main text
Initialize weighted sums $S_m^C = 0$, $S_m^G = 0$ and counts $N_m = 0$ for each block m
for each calibration batch of masked sequences $\widetilde{\mathbf{x}}_n$ and target masks **do**
  Forward the masked sequence; suppress the mask-token logit before softmax and sample one categorical target from the model distribution at each masked target position
  Backpropagate the summed cross-entropy against these sampled targets
  for each adapter-matrix block m ($\boldsymbol{A}_j$ and $\boldsymbol{B}_j$ separately) **do**
    Collect paired rows $(h_{m,n}, g_{m,n})$ at positions with nonzero gradient; skip blocks with no eligible rows
    Uniformly subsample up to N_s rows; set each retained row's weight w_n to the number of eligible rows divided by the number retained
    $S_m^C \leftarrow S_m^C + \sum_n w_n h_{m,n} h_{m,n}^\top$, $S_m^G \leftarrow S_m^G + \sum_n w_n g_{m,n} g_{m,n}^\top$, and $N_m \leftarrow N_m + \sum_n w_n$
  end for
end for
for each block m **do**
  if $N_m = 0$ **then**
    Set $F_m = 0$ (prior-only block)
  else
    Normalize $C_m = S_m^C / N_m$ and $G_m = S_m^G / N_m$
    Eigendecompose C_m and G_m; retain the top-k directions of large factors
    Store $(\{u_{m,b}^C\}, \{c_{m,b}\}, \{u_{m,a}^G\}, \{g_{m,a}\}, N_m)$ with curvature $F_m = N_m(G_m \otimes C_m)$
  end if
end for
Form $F_{\mathrm{KFAC}} = \mathrm{blockdiag}(F_1, \ldots, F_R)$ and $q_\lambda(\boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\phi}_0, (F_{\mathrm{KFAC}} + \lambda I)^{-1})$
return q_λ and the per-block eigenbasis; λ is applied at inference

All reported evaluations are cache-free. Within each decoding step, activations before the first uncertain layer are computed once and reused across posterior draws; each draw evaluates the remaining layers with its sampled weights. This reuse is exact because the preceding layers are unchanged across draws. Activations are recomputed at every decoding step, and draws are batched where memory permits.

A.2 EXECUTION PROTOCOLS

All rows in Tables 1 and 2 use exact cache-free forward passes. Planning evaluations use the task geometry described in Section 5.

Task-specific decoding geometry. For the Dream experiments, MBPP, HumanEval, GSM8K, and MATH500 use sliding-window decoding with an initial window width of 16 tokens and a maximum width of 128 tokens. Sudoku uses in-place infilling, whereas Countdown uses a 48-token block. These planning-specific settings replace the sliding-window configuration used for code generation and mathematical reasoning.

Table 4: Main experimental configuration for the six-benchmark Dream study. HumanEval uses the MBPP-trained posterior without refitting; all other posteriors are fitted on task-specific or synthetic training data.

Component	Setting
Mean model	Dream-v0-Instruct-7B
Benchmarks	MBPP, HumanEval, GSM8K, MATH500, Sudoku, Countdown
Prompt protocol	task-matched Dream instruction prompts
Posterior mean	released checkpoint; zero LoRA update
Curvature	categorical model Fisher, low-rank KFAC ($k = 10$)
Target modules	trailing-layer query/value projections
Posterior draws	$S = 10$
Vocabulary support	$K_v = 256$
Predictive backend	nonlinear posterior weight sampling
Commit threshold	$\tau = 0.9$; posterior budget gate $\tau_{\mathrm{B}} = 0.85$ or 0.9
Ranking	posterior predictive entropy
Commit budget	posterior confidence gate for all selected BAYESER rows
Window (code/math)	Sliding, initial 16 tokens, maximum 128 tokens
Geometry (Sudoku)	In-place infilling
Geometry (Countdown)	48-token block
Token proposal	posterior argmax for all selected BAYESER rows
Sequence confidence	geometric mean of commit-time probabilities, generated and answer scopes
ECE	15 equal-width bins

Table 5: Hyperparameters for the evaluation in the main results. HumanEval reuses the MBPP-trained posterior; the remaining rows are task-fitted.

Task	Target	L_s	Rank	Alpha	Prior	τ_{B}
MBPP	Accuracy	1	8	16	300	0.85
MBPP	Calibration	1	8	16	200	0.90
HumanEval	Accuracy	1	8	16	300	0.85
HumanEval	Calibration	1	8	16	200	0.90
GSM8K	Accuracy	4	8	16	2,000	0.90
GSM8K	Calibration	1	8	16	500	0.85
MATH500	Accuracy / calibration	2	8	16	2,000	0.90
Sudoku	Accuracy	2	8	16	500	0.90
Sudoku	Calibration	1	8	16	1,000	0.90
Countdown	Accuracy	4	8	16	200	0.85
Countdown	Calibration	1	8	16	100	0.90

B ADDITIONAL DREAM ACCURACY–CALIBRATION SWEEPS

Figures 3–6 extend Figure 1 to four additional benchmarks.

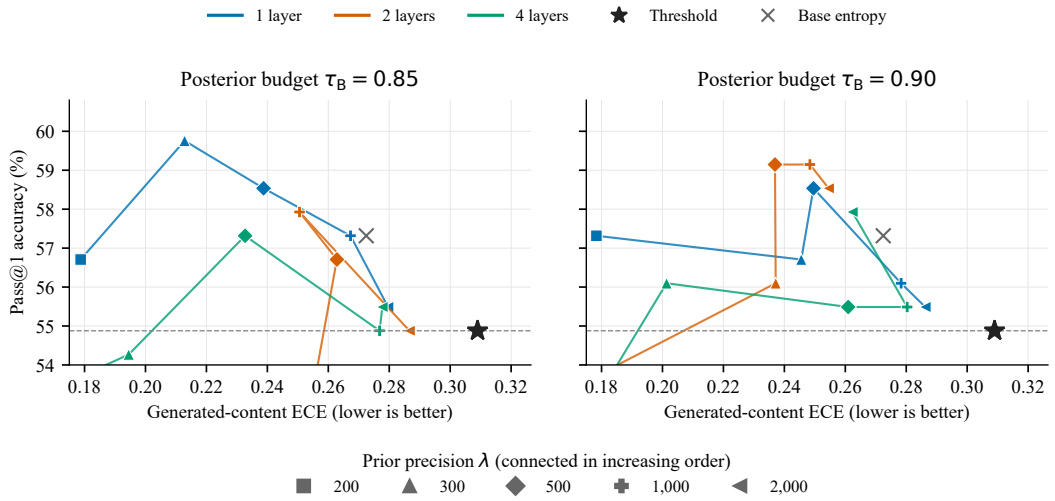

Figure 3: HumanEval accuracy versus calibration.

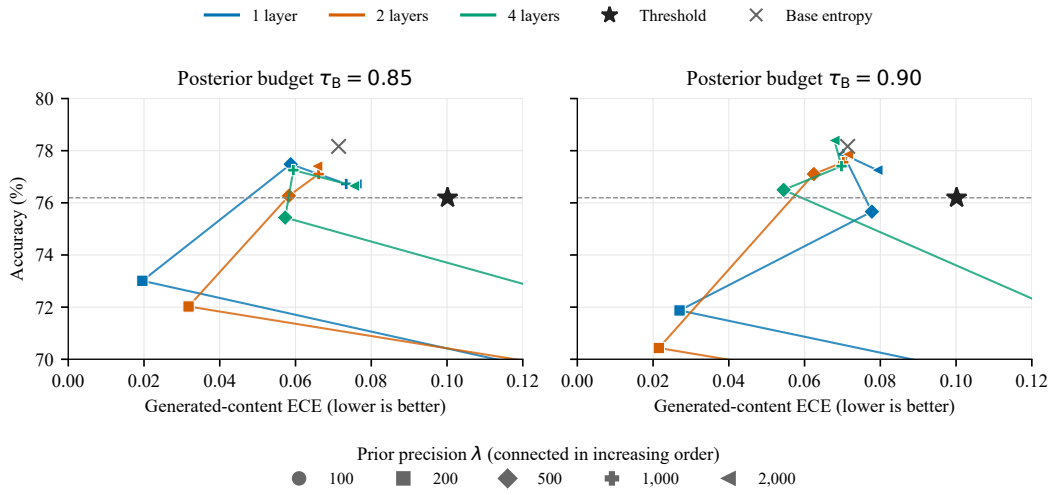

Figure 4: GSM8K accuracy versus calibration.

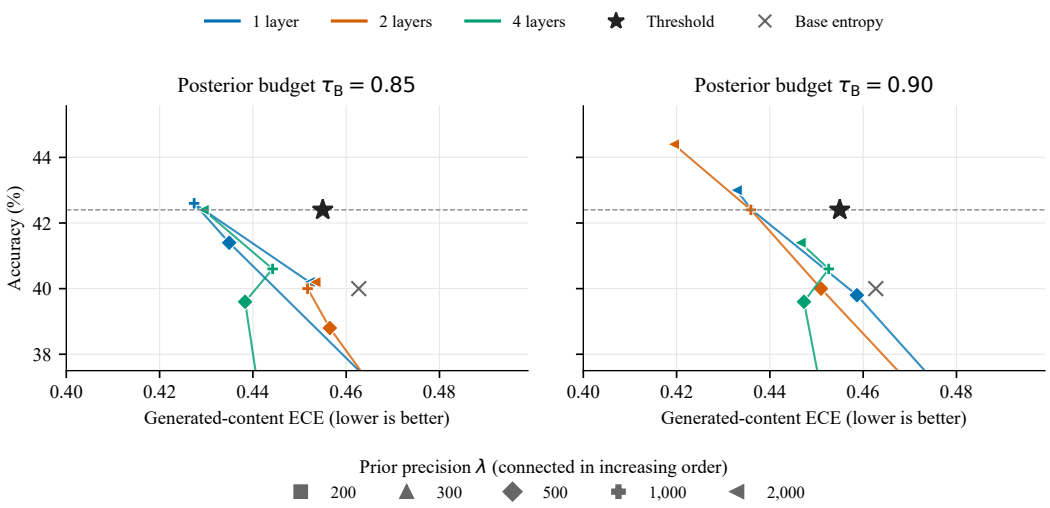

Figure 5: MATH500 accuracy versus calibration.

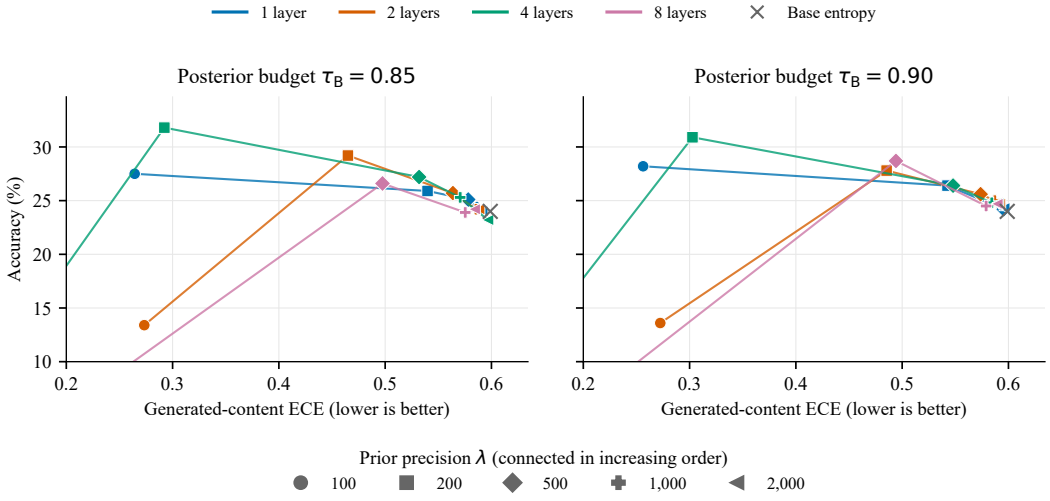

Figure 6: Countdown accuracy versus calibration.

C ADDITIONAL MEASURED COST

Figure 7 compares BAYESER costs with Confidence-thresholding baseline on H100 GPU.

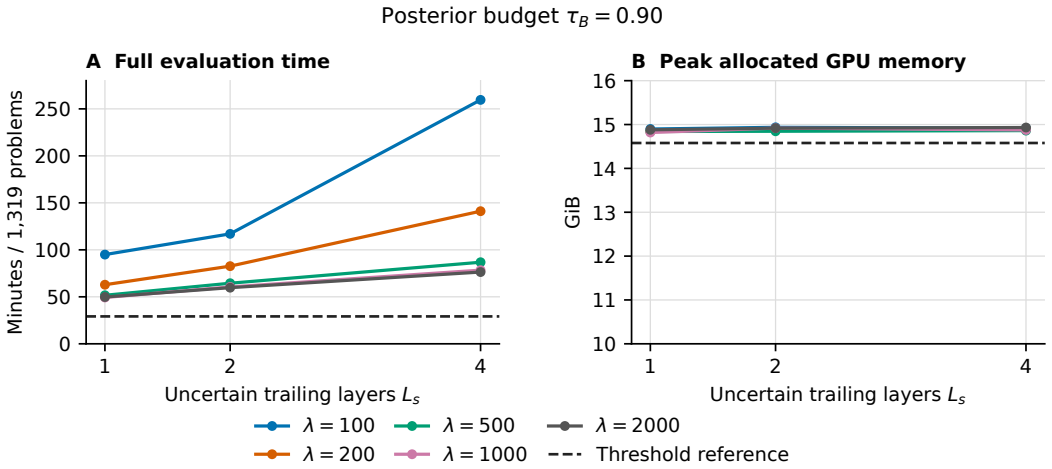

Figure 7: **Measured Dream GSM8K cost versus uncertain depth.** Full test set ($n = 1{,}319$), GSM8K-fitted rank-8 q/v Laplace-LoRA with $S = 10$, on H100 GPUs. Color denotes prior precision λ. Dashed lines show the confidence-threshold run. Wall time includes loading and scoring; the loose prior increases decoding steps, especially at four layers.

D LLaDA RESULTS

Table 6 reports full-test results for `GSAI-ML/LLaDA-8B-Instruct`. The recorded MBPP, GSM8K and MATH500 curvature-only sweeps use rank-8 q/v adapters ($\alpha = 16$), KFAC rank 10, and vocabulary support $K_v = 256$. HumanEval retains the MBPP transfer configuration described above. Info-gain uses $N = 8$, fixed $K = 2$.

Table 6: LLaDA full-test results. Accuracy is reported in percent; ECE and generated-content Brier values and their standard deviations are multiplied by 100. Subscripts are the standard deviations of three seeds on the corresponding column scale. Numbers in parentheses are relative percentage improvements over confidence-thresholding decoding. Bold values indicate the best result and underlined values indicate the second-best result per metric within each benchmark, including ties.

Method	Accuracy (%) ↑	ECE_{gen} (×100) ↓	ECE_{ans} (×100) ↓	$\text{Brier}_{\text{gen}}$ (×100) ↓
MBPP (500)				
Threshold (Fast-dllm)	39.40	47.63	50.01	46.14
Base Entropy	41.20 (+4.57)	<u>44.51</u> (+6.53)	<u>46.84</u> (+6.33)	<u>43.48</u> (+5.76)
Info-gain	<u>41.27</u>$_{0.61}$ (+4.74)	44.96$_{0.50}$ (+5.59)	48.01$_{0.52}$ (+4.00)	43.52$_{0.26}$ (+5.67)
BAYESER	**41.33**$_{0.58}$ (+4.91)	**43.57**$_{0.67}$ (+8.52)	**45.89**$_{0.66}$ (+8.23)	**42.70**$_{0.45}$ (+7.46)
HumanEval (164)				
Threshold (Fast-dllm)	18.29	69.05	66.50	62.26
Base Entropy	<u>22.56</u> (+23.33)	<u>63.34</u> (+8.26)	<u>60.48</u> (+9.04)	<u>57.46</u> (+7.70)
Info-gain	15.65$_{2.31}$ (−14.44)	70.51$_{2.11}$ (−2.11)	68.38$_{1.89}$ (−2.84)	62.95$_{1.39}$ (−1.11)
BAYESER	**23.17**$_{0.00}$ (+26.67)	**62.76**$_{0.02}$ (+9.11)	**59.81**$_{0.03}$ (+10.06)	**56.99**$_{0.07}$ (+8.46)
GSM8K (1,319)				
Threshold (Fast-dllm)	<u>76.12</u>	17.49	<u>20.59</u>	20.63
Base Entropy	75.51 (−0.80)	17.11 (+2.21)	20.98 (−1.89)	20.64 (−0.05)
Info-gain	**78.09**$_{0.53}$ (+2.59)	**15.87**$_{0.50}$ (+9.27)	**20.21**$_{0.58}$ (+1.87)	**18.88**$_{0.45}$ (+8.53)
BAYESER	75.92$_{0.37}$ (−0.27)	<u>16.67</u>$_{0.37}$ (+4.68)	20.64$_{0.40}$ (−0.21)	<u>20.24</u>$_{0.30}$ (+1.89)
MATH500 (500)				
Threshold (Fast-dllm)	<u>38.00</u>	<u>53.68</u>	<u>55.23</u>	51.44
Base Entropy	**38.20** (+0.53)	**52.13** (+2.90)	**55.01** (+0.41)	**49.67** (+3.43)
Info-gain	37.73$_{0.42}$ (−0.70)	53.81$_{0.43}$ (−0.23)	57.35$_{0.60}$ (−3.84)	<u>51.08</u>$_{0.41}$ (+0.69)
BAYESER	35.40$_{0.72}$ (−6.84)	54.60$_{0.75}$ (−1.71)	58.35$_{0.57}$ (−5.64)	51.48$_{0.64}$ (−0.08)
Sudoku (500)				
Threshold (Fast-dllm)	**85.60**	**9.57**	**9.57**	<u>10.42</u>
Base Entropy	81.80 (−4.44)	12.85 (−34.35)	12.85 (−34.35)	11.99 (−15.05)
Info-gain	78.27$_{1.10}$ (−8.57)	10.10$_{1.14}$ (−5.54)	10.10$_{1.14}$ (−5.54)	14.85$_{0.66}$ (−42.48)
BAYESER	<u>85.47</u>$_{1.47}$ (−0.16)	<u>9.81</u>$_{0.24}$ (−2.49)	<u>9.81</u>$_{0.24}$ (−2.49)	**10.26**$_{0.82}$ (+1.57)
Countdown (1,000)				
Threshold (Fast-dllm)	<u>35.10</u>	<u>53.65</u>	52.14	48.39
Base Entropy	29.70 (−15.38)	54.08 (−0.80)	<u>50.65</u> (+2.85)	47.92 (+0.97)
Info-gain	**39.57**$_{1.55}$ (+12.73)	**49.28**$_{1.23}$ (+8.14)	**46.99**$_{1.05}$ (+9.88)	**45.38**$_{1.04}$ (+6.22)
BAYESER	28.83$_{0.38}$ (−17.85)	54.07$_{0.33}$ (−0.78)	51.02$_{0.36}$ (+2.14)	<u>47.68</u>$_{0.20}$ (+1.46)

E MUTUAL-INFORMATION RANKING AND BAYESIAN ENTROPY DECODING

We compare MIRANK with BAYESER on the first 128 examples of MBPP and GSM8K. The grid uses $L_s \in \{1, 2, 4\}$, $\lambda \in \{100, 200, 500, 1000, 2000\}$, and $\tau_{\mathrm{B}} \in \{0.85, 0.90\}$ for BAYESER.

Table 7: Comparison between MIRANK and BAYESER on 128 examples of MBPP. Accuracy is in percent; both ECE columns are multiplied by 100.

		MIRANK				BAYESER, $\tau_{\mathrm{B}} = 0.85$				BAYESER, $\tau_{\mathrm{B}} = 0.90$			
L_s	λ	Acc.	ECE$_g$	ECE$_a$	Steps	Acc.	ECE$_g$	ECE$_a$	Steps	Acc.	ECE$_g$	ECE$_a$	Steps
1	100	28.91	10.89	10.77	25.74	34.38	8.40	6.61	100.70	**40.63**	12.62	10.44	93.91
1	200	48.44	19.63	21.55	38.02	**57.03**	16.42	18.14	59.38	55.47	19.27	20.04	67.27
1	500	54.69	27.78	30.58	39.11	**57.81**	25.77	27.89	43.26	**57.81**	27.02	29.19	45.13
1	1000	53.13	30.62	33.09	43.25	57.81	27.18	29.16	37.36	**58.59**	27.30	29.60	42.54
1	2000	**57.81**	26.34	29.27	40.44	**57.81**	27.56	29.50	37.62	57.03	29.37	31.60	42.23
2	100	**28.13**	16.87	17.83	20.91	9.38	14.20	16.24	123.17	5.47	18.04	22.09	123.62
2	200	45.31	15.05	16.11	33.48	**55.47**	13.03	14.74	72.17	53.13	15.10	17.64	76.32
2	500	52.34	29.39	32.08	42.05	**60.94**	22.64	24.62	42.09	56.25	27.86	29.85	46.04
2	1000	54.69	29.13	32.01	39.08	**57.81**	27.10	29.00	37.65	57.03	28.62	30.97	43.27
2	2000	53.13	30.67	32.96	40.84	**59.38**	25.93	27.90	39.11	57.81	28.49	30.61	42.89
4	100	**28.91**	26.44	27.72	21.88	0.00	38.72	38.58	224.40	0.00	37.81	37.89	222.68
4	200	42.19	12.85	11.29	27.82	**46.09**	5.04	10.20	112.18	**46.09**	6.27	10.23	119.48
4	500	53.13	26.61	28.95	40.52	**58.59**	23.36	25.51	46.13	56.25	26.04	28.29	48.66
4	1000	58.59	24.67	27.11	41.99	**59.38**	25.22	27.17	39.10	57.81	27.50	29.84	44.98
4	2000	57.81	26.09	28.48	40.41	**58.59**	26.87	28.74	37.05	**58.59**	27.65	29.80	42.36

Table 8: Comparison between MIRANK and BAYESER on 128 examples of GSM8K. Accuracy is in percent; both ECE columns are multiplied by 100.

		MIRANK				BAYESER, $\tau_{\mathrm{B}} = 0.85$				BAYESER, $\tau_{\mathrm{B}} = 0.90$			
L_s	λ	Acc.	ECE$_g$	ECE$_a$	Steps	Acc.	ECE$_g$	ECE$_a$	Steps	Acc.	ECE$_g$	ECE$_a$	Steps
1	100	59.38	31.56	12.84	30.39	58.59	31.50	16.28	102.63	**63.28**	38.08	15.94	118.50
1	200	64.84	10.14	19.34	41.83	**77.34**	5.56	15.76	70.57	75.78	6.65	16.95	77.88
1	500	78.91	4.61	18.37	49.34	**79.69**	8.37	13.51	56.29	75.78	9.10	17.81	63.16
1	1000	77.34	8.21	19.78	52.70	76.56	10.08	18.80	53.72	**78.13**	7.45	16.38	60.42
1	2000	77.34	9.39	17.38	51.38	76.56	9.09	18.56	53.38	**78.13**	7.63	17.32	61.18
2	100	44.53	25.79	12.27	27.08	**67.19**	37.97	12.24	110.55	64.84	38.51	14.77	122.64
2	200	64.06	5.68	17.23	37.99	73.44	8.06	12.95	76.51	**75.00**	5.96	18.17	88.73
2	500	75.78	6.84	16.15	49.79	80.47	9.69	14.06	57.14	**82.03**	4.96	14.46	65.91
2	1000	78.13	6.43	16.87	52.21	79.69	11.50	15.85	53.67	**81.25**	4.04	15.12	61.98
2	2000	72.66	10.91	22.94	54.52	79.69	10.08	14.79	54.41	**80.47**	5.94	14.97	60.62
4	100	**48.44**	44.56	33.95	26.02	17.19	25.23	31.09	206.01	19.53	28.30	23.09	206.45
4	200	63.28	25.31	15.01	34.52	67.97	19.21	17.75	103.86	**72.66**	22.09	12.85	109.77
4	500	71.88	6.91	20.19	52.28	76.56	10.89	17.49	60.30	**79.69**	9.86	15.90	69.25
4	1000	**79.69**	10.22	14.45	52.49	**79.69**	7.53	16.26	55.61	**79.69**	6.29	17.09	62.16
4	2000	78.13	5.62	17.96	54.41	75.78	10.38	17.55	54.13	**78.91**	7.69	16.13	62.65