LEARNING LENGTH PRIORS FOR CANVAS ALLOCATION IN MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models generate responses by progressively denoising a preallocated canvas of masked tokens. A globally fixed canvas can constrain responses that need more space or waste computation on shorter ones. Recent methods address this issue by dynamically adjusting the response canvas during denoising, but the required canvas length is still unknown before denoising begins. We find that the information predictive of reference response length could be extracted from prompt representations in dLLMs before denoising begins. We propose CALP, an approach to **C**anvas **A**llocation with **L**earned **P**riors derived from frozen dLLM prompt representations. A lightweight predictor determines the canvas length before denoising, leaving the backbone and denoising procedure unchanged. We evaluate our approach on LLaDA-8B-Instruct, LLaDA-1.5, and Dream-7B-Instruct across five benchmarks covering mathematical reasoning, coding, and instruction following, against baselines with a fixed canvas and adaptive methods including DAEDAL and ρ-EOS. Compared with DAEDAL, our method improves the average task score by 1.43 percentage points, reduces the average number of allocated response positions by 55.5%, and achieves an observed inference speedup of 3.75×. Relative to ρ-EOS, it improves the average task score by 2.21 percentage points and achieves an observed inference speedup of 1.83×.

1 INTRODUCTION

Masked diffusion large language models (dLLMs) have emerged as an alternative to autoregressive language models. Recent work in discrete diffusion objectives, model scaling, and systems design has improved their generation quality and efficiency (Austin et al., 2021a; Lou et al., 2023; Sahoo et al., 2024; Nie et al., 2024; Gong et al., 2024; Inception Labs et al., 2025; Song et al., 2025). Current large dLLMs such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) achieve competitive performance in reasoning, code generation, and tasks that require following instructions. To generate a response, masked dLLMs initialize a set of masked token positions and progressively denoise them into output tokens. We refer to these preallocated positions as the *response canvas*. In standard denoising (Nie et al., 2025; Ye et al., 2025), the canvas length is set by a global hyperparameter before denoising and shared across inputs.

The amount of response space needed for generation varies across prompts, creating an inherent mismatch between a globally fixed canvas and allocation needs that vary with the input. An undersized canvas may limit the response space and degrade task performance, whereas an oversized one wastes computation by repeatedly processing excess positions beyond what the response actually needs. Existing approaches enable variable length denoising through online canvas adjustment or changes in the denoising phase (Cheng et al., 2026; Bouhedja et al., 2026). DAEDAL (Li et al., 2025) expands the allocated canvas using EOS confidence and mask insertion, while ρ-EOS (Yang et al., 2026) uses an implicit signal based on EOS density to adjust the canvas during denoising. Other approaches support variable length generation through retraining the dLLM (Yang et al., 2025; Wu et al., 2026; Liu et al., 2026a). Online canvas adjustment uses signals that become available during denoising, while a prompt-conditioned allocation can make the length decision before denoising starts. This motivates a complementary question: can the prompt support a one-shot canvas allocation that preserves standard denoising while avoiding unnecessarily large canvases? We study this question by predicting response length from the frozen model's prompt representation.

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.

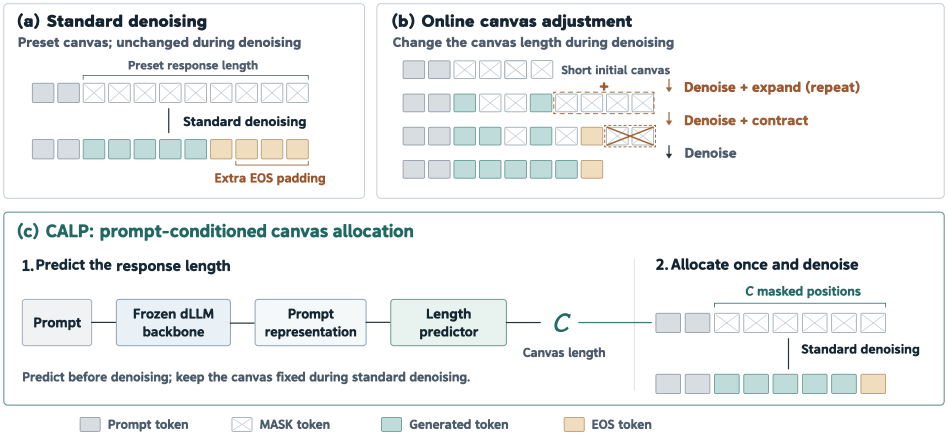


Figure 1: Comparison of response canvas allocation strategies in masked dLLMs. **(a)** Standard denoising uses the same canvas length for all prompts in a given configuration and keeps this length fixed during denoising. **(b)** Methods with online canvas adjustment can change the canvas length during denoising. **(c)** Our method predicts response length using prompt representations from the frozen backbone, allocates a canvas tailored to each prompt once before denoising, and keeps its length fixed during standard denoising.

We find that the prompt representations from masked dLLM inference contain information predictive of response length before denoising begins. Our evaluation (c.f. Section 4.3) confirms that prompt representations from the frozen dLLM backbones could provide predictive signals that serve as a response length prior conditioned on the prompt for initial canvas allocation.

In this paper, we introduce CALP, an approach to **C**anvas **A**llocation with **L**earned **P**riors derived from frozen dLLM prompt representations. As illustrated in Figure 1, a lightweight predictor determines the canvas length once before denoising. The backbone and denoising procedure remain unchanged, and the canvas stays fixed throughout generation. We evaluate the proposed method on LLaDA-8B-Instruct (Nie et al., 2025), LLaDA-1.5 (Zhu et al., 2025), and Dream-7B-Instruct (Ye et al., 2025) across five benchmarks covering mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021; Lightman et al., 2023), code generation (Austin et al., 2021b; Chen et al., 2021) and instruction following (Zhou et al., 2023). We compare our approach with multiple fixed canvas lengths and adaptive methods including DAEDAL and ρ-EOS. Our method improves the average task score over Fixed Avg , DAEDAL, and ρ-EOS by 1.79, 1.43, and 2.21 percentage points, respectively. Relative to Fixed Avg, which averages results at lengths 128, 256, and 512, it reduces the average allocated canvas length by 20.7%. Compared with DAEDAL and ρ-EOS, the reductions are 55.5% and 4.5%, and the observed inference speedups are 3.75$\times$ and 1.83$\times$, respectively. These aggregate results show average improvements in both generation quality and computational efficiency across the three backbones and five benchmarks. The main contributions of this paper are as follows:

- We identify a response length prior in masked dLLMs: prompt representations from the frozen backbone contain substantial predictive information about response length, providing a useful signal for canvas allocation before denoising.
- We use this response length prior to allocate a response canvas tailored to each input before denoising, without online canvas adjustment or retraining the dLLM backbone.
- We evaluate our approach on three dLLMs across five diverse benchmarks, showing gains in average task score and reductions in average inference time while documenting how canvas use varies across tasks and baselines.

2 Problem Statement and Motivation

Masked diffusion denoising. Masked diffusion language models generate text through iterative denoising over a set of masked response positions. Let F_θ denote the model backbone and p_θ the token distribution it induces. Given a tokenized prompt x and a response canvas of length C, we initialize the response canvas by appending C mask tokens (i.e., $[\mathrm{MASK}]^C$) to the prompt. At each denoising step, the model predicts tokens for the currently masked positions conditioned on the prompt and the partially denoised response. A denoising schedule then progressively reveals or refines these positions until denoising is complete.

Let $\tilde{\mathbf{y}}_t$ denote the state of the response canvas with length C at step $t \in \{1, \ldots, T\}$, and $\mathcal{M}_t = \{j : \tilde{y}_{t,j} = \mathrm{MASK}\}$ be the set of masked positions. Denoising starts from the fully masked canvas $\tilde{\mathbf{y}}_T = [\mathrm{MASK}]^C$. At each step, the model predicts a token distribution for currently masked positions,

$$p_\theta(y_j \mid x, \tilde{\mathbf{y}}_t, t), \qquad j \in \mathcal{M}_t, \tag{1}$$

and the denoising schedule uses these predictions to progressively update the canvas.

Under standard denoising with a fixed canvas length, all C response positions are allocated before the first denoising step. This differs from autoregressive generation, where the response grows token by token and its length emerges during decoding. During masked diffusion denoising, the canvas length determines both the response capacity available to the model and the sequence length processed during denoising.

Response canvas allocation problem. The allocation rule determines the canvas length $C(x)$ for each prompt x. The appropriate response space usually varies substantially across inputs: a short factual response may require a small canvas, whereas mathematical reasoning or code generation may require considerably more response positions. Nevertheless, standard masked dLLM denoising typically uses a globally fixed canvas length $C(x) = C_0$ for every prompt.

This uniform policy creates a mismatch between generation quality and computational cost that varies with the input. If C_0 is too small for a given prompt, the limited allocated response space may constrain generation and degrade task performance. If C_0 is substantially larger than necessary, the excess positions are repeatedly processed during iterative denoising, increasing computation without contributing useful response capacity. Thus, canvas length is not merely a denoising hyperparameter, but an allocation decision that depends on the input.

Existing adaptive denoising approaches demonstrate that the response canvas could be adjusted using signals obtained during denoising (Li et al., 2025; Yang et al., 2026). These signals provide useful feedback once the response canvas has been initialized and denoising is underway. However, denoising must still begin with an initially allocated response space, whose length must be chosen before any signals from denoising are available. Determining this initial canvas length therefore remains a challenge. This motivates an earlier allocation step: using information available from the prompt to estimate the response space likely to be needed before denoising begins.

We formulate this task as *the response canvas allocation problem*. Given a prompt x, the objective is to determine a canvas length $C(x)$ tailored to the input before the denoising. The allocated canvas length $C(x)$ may vary across prompts but remains fixed throughout denoising for each prompt. The feasibility of this formulation depends on whether response length estimates based on the prompt can provide useful guidance for canvas allocation.

3 Method

In this section, we present our approach for canvas allocation. We first formalize the length prediction problem, then describe the design choices for the length predictor, prompt representation, and response length target, followed by predictor training and canvas allocation at inference.

3.1 Method overview

Our approach treats canvas allocation before denoising as a prediction problem, and uses the information available in the prompt to estimate response length for allocating the canvas before denoising

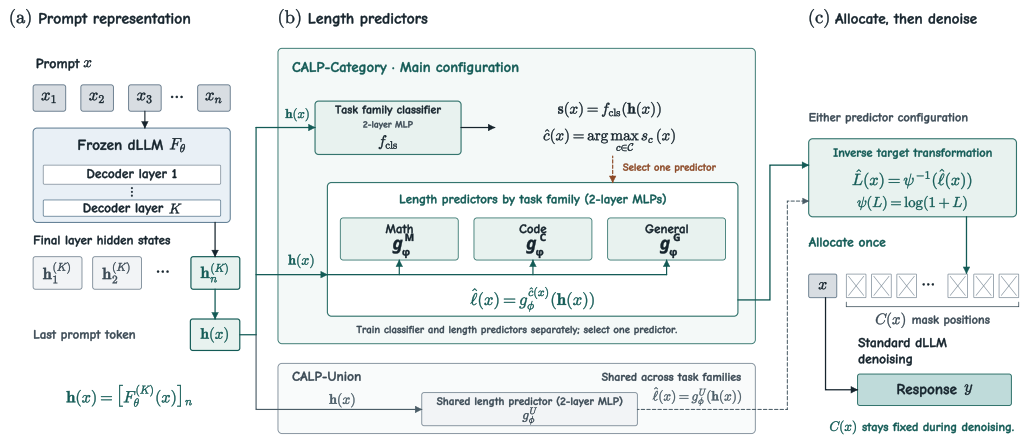


Figure 2: Overview of the proposed method. A prompt representation is extracted from the frozen dLLM and passed to a length predictor. The inverse target transformation yields a response length estimate for canvas allocation. The canvas length remains fixed throughout denoising.

begins. The core design is to read the length signal from the backbone's prompt representations using a lightweight predictor, while leaving the backbone and its denoising procedure unchanged.

As shown in Figure 2, given a frozen dLLM backbone F_θ and a dataset of N tokenized pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where x_i is the prompt and y_i is the reference response, we use the reference response length as a proxy target for canvas allocation. For a tokenized prompt $x = (x_1, \ldots, x_n)$, a forward pass of the dLLM F_θ produces the matrix of the prompt's hidden states $F_\theta^{(k)}(x) \in \mathbb{R}^{n \times d}$ at layer k, where d is the hidden dimension. A readout function S selects or aggregates these states into a prompt representation

$$\mathbf{h}(x) = S\Big(F_\theta^{(k)}(x)\Big) \in \mathbb{R}^d. \tag{2}$$

Then, an invertible target transformation ψ maps the response length L to the prediction target $\ell = \psi(L)$. Given the prompt representation $\mathbf{h}(x)$, a lightweight length predictor $g_\phi : \mathbb{R}^d \to \mathbb{R}$ with trainable parameters ϕ produces a scalar estimate $\hat{\ell}(x) = g_\phi(\mathbf{h}(x))$ on the scale defined by ψ. The inverse target transformation ψ^{-1} converts $\hat{\ell}(x)$ into a response length estimate $\hat{L}(x)$ in token units:

$$\hat{L}(x) = \psi^{-1}\Big(\hat{\ell}(x)\Big) = \psi^{-1}(g_\phi(\mathbf{h}(x))). \tag{3}$$

This design requires specifying how the length predictor is configured, which prompt representation it uses, the scale on which length is represented, and how the predictor is trained and its output is converted into a canvas allocation. We discuss these components in the following subsections.

3.2 Length Predictor

The length predictor g_ϕ is the central component for length estimation in canvas allocation. We consider two configurations: a single length predictor shared across all task families (e.g., math, code, and general instruction following), and the extended predictor using a dedicated length predictor for each task family to capture differences in their response length patterns. We refer to these configurations as *CALP-Union* and *CALP-Category*, respectively. CALP-Union uses one length predictor g_ϕ^U for all task families, with $\hat{\ell}(x) = g_\phi^U(\mathbf{h}(x))$. CALP-Category assigns an independent length predictor g_ϕ^c to each task family $c \in \mathcal{C}$, where $\mathcal{C}$ denotes the set of task families. A task family classifier f_{cls} is used for selecting the proper $g_\phi^{\hat{c}(x)}$, which is based on the representation $\mathbf{h}(x)$ as input and produces a score vector $\mathbf{s}(x)$:

$$\mathbf{s}(x) = f_{\text{cls}}(\mathbf{h}(x)), \qquad \hat{c}(x) = \arg\max_{c \in \mathcal{C}} s_c(x), \qquad \hat{\ell}(x) = g_\phi^{\hat{c}(x)}(\mathbf{h}(x)). \tag{4}$$

Here, $s_c(x)$ is the classifier score for task family c. To limit the additional computation introduced by length prediction, we use a 2-layer MLP for all length predictors.

3.3 PROMPT REPRESENTATION AND RESPONSE LENGTH TARGET

Constructing the prompt representation $\mathbf{h}(x)$ involves choosing which layer supplies the hidden states and how the readout function S selects or aggregates states across token positions. The target transformation ψ determines the scale on which response length is predicted. We examine these choices through controlled probes, varying one component while fixing the others, to guide the representation and target used by the length predictor (see results in Sections 4.3 and 4.4 for details).

Layer selection. Layer selection determines which stage of the backbone supplies the hidden states for length prediction. Early, intermediate, and final layers provide different representations of the same prompt, and useful length information may not be equally accessible at each layer. With the last token readout and the logarithmic target fixed, ridge probes attain the highest test R^2 at intermediate layers, while the final layer retains substantial predictive information across all three backbones (Figure 4(a)). We therefore use the final layer as a common readout.

Token readout. Token readout determines how the readout function S converts states at individual prompt positions into a single prompt representation $\mathbf{h}(x)$. It can select one contextualized state or aggregate states across positions. At the final layer, we compare the last token's hidden state with mean pooling over all prompt states. The former yields higher test R^2 across all three backbones under the logarithmic target (Figure 4(b)), so we use the last token's hidden state.

Response length target. The response length target determines the scale on which the predictor is trained. Response lengths typically have a distribution skewed to the right: most responses are relatively short, while a small fraction extends substantially farther into the long tail. Raw length $\psi_{\text{raw}}(L) = L$ retains the original scale of token counts, whereas logarithmic length $\psi_{\log}(L) = \log(1 + L)$ compresses the long tail, reducing the influence of large numerical errors under squared error training. The diagnostic probes yield higher R^2 for the logarithmic target in its own target space and smaller normalized prediction shifts after excluding long training responses (Figure 4(c) and Appendix C.2). We therefore adopt $\psi_{\log}(L_i)$ as the response length target.

3.4 PREDICTOR TRAINING AND CANVAS ALLOCATION

Training the length predictors. Let $\mathcal{I}$ index the training examples for the configuration under consideration, with targets $\ell_i = \psi(L_i)$ and prompt representations $\mathbf{h}_i = \mathbf{h}(x_i)$. CALP-Union trains the shared predictor g_ϕ^U on all examples; CALP-Category trains each predictor g_ϕ^c on its task family subset $\mathcal{I}_c = \{i \in \mathcal{I} : c_i = c\}$. Both minimize mean squared error (MSE):

$$\mathcal{L}_{\text{Union}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} [g_\phi^U(\mathbf{h}_i) - \ell_i]^2, \qquad \mathcal{L}^c_{\text{Category}} = \frac{1}{|\mathcal{I}_c|} \sum_{i \in \mathcal{I}_c} [g_\phi^c(\mathbf{h}_i) - \ell_i]^2. \tag{5}$$

For CALP-Category, the classifier f_{cls} is trained via cross entropy with class weights and focal modulation (Lin et al., 2017); Appendix A.2 gives details.

Canvas allocation before denoising. After obtaining the predicted response length $\hat{L}(x)$, we set $C(x) = \mathcal{A}(\hat{L}(x))$, where the allocation rule $\mathcal{A}$ first rounds the prediction to the nearest integer. The allocation rule $\mathcal{A}$ first rounds the prediction to the nearest integer. In LLaDA's block diffusion sampling (Nie et al., 2025), the canvas length must be a multiple of the block length b. When this alignment is required, we round the integer length up to the nearest multiple of b. Then we append $C(x)$ mask tokens to the prompt and apply standard denoising, keeping the canvas length fixed throughout generation. Allocation adds only one backbone pass over the prompt and a lightweight prediction step, with overhead expected to be negligible compared with iterative denoising.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation details. We evaluate three representative dLLMs to assess canvas allocation across backbone families: Dream-7B-Instruct (Ye et al., 2025), LLaDA-8B-Instruct (Nie et al., 2025), and LLaDA-1.5 (Zhu et al., 2025). All backbones remain frozen to evaluate allocation without backbone adaptation. CALP-Union uses a shared length predictor implemented as a 2-layer MLP with

Table 1: Main results across three backbones and five benchmarks. The proposed method CALP-Category achieves the best performance compared with the baselines, with the highest accuracy marked by blue shading mask. Note that Fixed Avg averages task scores and token statistics over denoising runs with fixed canvas lengths of 128, 256, and 512.

		Dream-7B				**LLaDA-8B**				**LLaDA-1.5**			
Benchmark	Metric	Fixed Avg	DAEDAL	ρ-EOS	CALP-Category	Fixed Avg	DAEDAL	ρ-EOS	CALP-Category	Fixed Avg	DAEDAL	ρ-EOS	CALP-Category
GSM8K	Acc. ↑	80.09	**80.36**	80.21	80.14	81.43	82.49	**82.87**	82.26	82.89	84.15	83.32	**84.31**
	E_{token}	—	104.2	108.3	105.6	—	261.2	153.0	190.3	—	265.7	159.6	209.4
	N_{token}	—	240.3	134.3	149.5	—	458.1	197.8	215.5	—	463.7	208.4	233.1
	E_{ratio} ↑	34.0%	43.4%	**80.6%**	70.7%	60.1%	57.0%	77.3%	**88.3%**	61.1%	57.3%	76.6%	**89.8%**
MATH500	Acc. ↑	46.53	47.40	48.60	**49.20**	39.00	39.80	35.20	**41.40**	40.00	**42.00**	38.60	40.40
	E_{token}	—	274.1	209.3	192.5	—	521.3	330.2	322.6	—	549.0	362.4	327.7
	N_{token}	—	697.1	251.9	234.3	—	802.8	379.2	369.6	—	834.3	423.3	363.7
	E_{ratio} ↑	58.0%	39.3%	**83.1%**	82.2%	81.8%	64.9%	87.1%	**87.3%**	84.3%	65.8%	85.6%	**90.1%**
MBPP	Acc. ↑	55.00	54.80	**56.20**	56.00	39.53	**40.00**	38.80	**40.00**	40.53	41.20	39.60	**42.80**
	E_{token}	—	52.6	54.4	48.4	—	70.7	109.1	65.0	—	83.8	124.1	78.7
	N_{token}	—	178.7	73.6	147.1	—	231.5	132.1	255.7	—	249.8	152.4	286.9
	E_{ratio} ↑	16.0%	29.4%	**73.9%**	32.9%	21.5%	30.5%	**82.6%**	25.4%	25.1%	33.5%	**81.5%**	27.4%
HumanEval	Acc. ↑	57.93	55.49	55.49	**61.59**	42.07	43.29	43.90	**44.51**	42.28	42.07	42.68	**45.12**
	E_{token}	—	157.5	91.8	114.7	—	739.3	416.9	280.1	—	743.7	482.3	258.8
	N_{token}	—	441.7	126.6	194.7	—	834.7	429.8	295.2	—	891.0	498.6	269.7
	E_{ratio} ↑	38.0%	35.6%	**72.5%**	58.9%	93.4%	88.6%	**97.0%**	94.9%	92.2%	83.5%	**96.7%**	96.0%
IFEval	Acc. ↑	50.34	51.02	52.50	**53.79**	56.32	54.34	55.27	**57.49**	59.09	59.89	53.42	**60.81**
	E_{token}	—	326.7	200.6	147.7	—	393.3	172.7	154.7	—	399.3	213.6	158.7
	N_{token}	—	532.9	256.0	183.4	—	553.2	208.9	175.5	—	566.5	248.8	179.1
	E_{ratio} ↑	54.2%	61.3%	78.4%	**80.5%**	63.6%	71.1%	82.7%	**88.2%**	64.0%	70.5%	85.9%	**88.6%**
Avg.	Acc. ↑	57.98	57.81	58.60	**60.14**	51.67	51.98	51.21	**53.13**	52.96	53.86	51.52	**54.69**
	E_{token}	—	183.0	132.9	121.8	—	397.2	236.4	202.5	—	408.3	268.4	206.6
	N_{token}	—	418.2	168.5	181.8	—	576.1	269.6	262.3	—	601.1	306.3	266.5
	E_{ratio} ↑	40.1%	43.8%	**78.9%**	67.0%	64.1%	68.9%	**87.7%**	77.2%	65.3%	67.9%	**87.6%**	77.5%

dimensions $d \to d/2 \to 1$ and a GELU activation, where d is the backbone hidden dimension. CALP-Category assigns an independent length predictor with the same architecture to each of three task families: mathematics (Math), code generation (Code), and general instruction following (General). Its task family classifier is a 2-layer MLP. Further implementation and training details are given in Appendix A.2. The length predictor training data cover these three task families. Appendix A.1 details the data composition, sources, and exact duplicate checks. Our code is available at `https://anonymous.4open.science/r/CALP-F8A3`.

Baselines. We compare with denoising on fixed canvases and online adjustment methods DAEDAL (Li et al., 2025) and ρ-EOS (Yang et al., 2026), using their official implementations with initial length 64 and recommended hyperparameters (Appendix A.3). Fixed Avg averages task scores and token statistics at lengths 128, 256, and 512 to avoid choosing a length for each benchmark in hindsight. It is a reporting reference, not an inference policy. CALP uses a learned length proxy to allocate each canvas before denoising; we assess its value through downstream scores and measured inference time. Appendix B.1 reports results at each fixed length and discusses the comparison and proxy limitations.

Benchmarks and metrics. We evaluate five benchmarks covering mathematical reasoning, code generation, and general instruction following. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and MATH500 (Hendrycks et al., 2021; Lightman et al., 2023) to cover grade school word problems and competition mathematics, respectively. For code generation, we use MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021) to assess functional correctness through Python test cases. For general instruction following, we use IFEval (Zhou et al., 2023) to assess compliance with verifiable instruction constraints. We report accuracy for mathematics, pass@1 for code, and strict prompt accuracy for IFEval, denoted as Acc. We report average allocated response

Table 2: Canvas allocation of CALP-Category relative to Fixed Avg, DAEDAL, and ρ-EOS. Overall, CALP-Category reduces canvas allocation. Each entry reports $N_{\text{category}}/N_{\text{baseline}}$, with values below one indicating lower allocation. Fixed Avg uses $(128 + 256 + 512)/3$ allocated positions.

	Dream-7B			LLaDA-8B			LLaDA-1.5		
Benchmark	Fixed Avg	DAEDAL	ρ-EOS	Fixed Avg	DAEDAL	ρ-EOS	Fixed Avg	DAEDAL	ρ-EOS
GSM8K	0.50	0.62	1.11	0.72	0.47	1.09	0.78	0.50	1.12
MATH500	0.78	0.34	0.93	1.24	0.46	0.97	1.22	0.44	0.86
MBPP	0.49	0.82	2.00	0.86	1.10	1.94	0.96	1.15	1.88
HumanEval	0.65	0.44	1.54	0.99	0.35	0.69	0.90	0.30	0.54
IFEval	0.61	0.34	0.72	0.59	0.32	0.84	0.60	0.32	0.72
Avg.	0.61	0.43	1.08	0.88	0.46	0.97	0.89	0.44	0.87

Table 3: Inference speedup of CALP-Category relative to DAEDAL and ρ-EOS. CALP-Category achieves faster inference than both baselines across all three backbones and five benchmarks. Speedup is computed as $T_{\text{baseline}}/T_{\text{category}}$, with values above one indicating faster inference.

	Dream-7B		LLaDA-8B		LLaDA-1.5	
Benchmark	DAEDAL	ρ-EOS	DAEDAL	ρ-EOS	DAEDAL	ρ-EOS
GSM8K	2.17	1.56	2.96	1.35	2.79	1.30
MATH500	3.36	1.64	3.22	1.42	3.53	1.63
MBPP	2.13	1.56	2.01	1.59	1.95	1.68
HumanEval	3.34	1.34	4.78	2.22	5.60	2.90
IFEval	4.72	2.71	6.07	2.20	6.43	2.79
Avg.	3.30	1.80	3.76	1.70	4.01	1.98

Table 4: Comparison of CALP-Category and CALP-Union. CALP-Category achieves higher average task scores on all three backbones and outperforms CALP-Union in most benchmark evaluations. Scores are reported in percentages; Δ denotes CALP-Category minus CALP-Union in percentage points, computed before rounding. Blue shading and bold mark the higher displayed score, including ties.

	Dream-7B			LLaDA-8B			LLaDA-1.5		
Benchmark	CALP-Category	CALP-Union	Δ	CALP-Category	CALP-Union	Δ	CALP-Category	CALP-Union	Δ
GSM8K	**80.14**	79.00	+1.14	**82.26**	81.58	+0.68	**84.31**	84.00	+0.30
MATH500	**49.20**	49.00	+0.20	**41.40**	38.60	+2.80	40.40	**41.60**	-1.20
MBPP	**56.00**	55.40	+0.60	**40.00**	39.00	+1.00	42.80	**43.20**	-0.40
HumanEval	**61.59**	60.37	+1.22	44.51	**46.34**	-1.83	45.12	**45.73**	-0.61
IFEval	53.79	**55.08**	-1.29	**57.49**	56.93	+0.55	**60.81**	58.41	+2.40
Avg.	**60.14**	59.77	+0.37	**53.13**	52.49	+0.64	**54.69**	54.59	+0.10

positions (N_{token}), average generated tokens excluding special tokens (E_{token}), and average canvas utilization ($E_{\text{ratio}} = E_{\text{token}}/N_{\text{token}}$). All inference runs use a single H800 GPU with batch size 1.

4.2 Main results

Task performance. To test whether the response length prior reduces canvas allocation while preserving generation quality, Table 1 compares CALP-Category with Fixed Avg and the dynamic methods DAEDAL and ρ-EOS across three backbones and five benchmarks. The comparison considers task performance (Acc.) and canvas allocation (N_{token}, E_{token}, and E_{ratio}). Appendix B.4 reports the predicted response length distributions. Table 1 shows that CALP-Category achieves the highest performance on average across all three backbones while maintaining competitive E_{ratio}. Especially, CALP-Category achieves the highest accuracy in 11 of the 15 benchmark evaluations and the lowest average N_{token} on both LLaDA backbones.

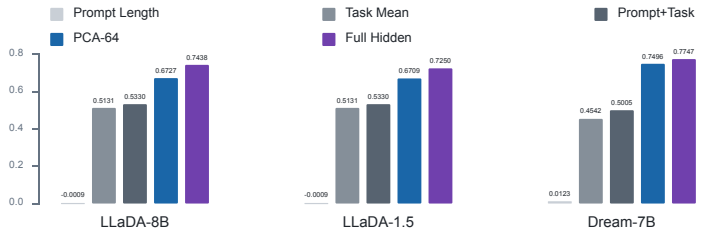


Figure 3: Response length prediction from prompt representations before denoising. Full Hidden achieves the highest test R^2, followed by PCA-64, both substantially outperforming the simple baselines across all three backbones.

Inference efficiency. To quantify the improvement in inference efficiency achieved by our method, we compare CALP-Category's N_{token} with each baseline and report the corresponding measured speedups over DAEDAL and ρ-EOS. Table 2 divides CALP-Category's N_{token} by that of the baseline named in each column, while Table 3 divides that baseline's average inference time per request by CALP-Category's. Thus, lower allocation ratios and higher speedups favor CALP-Category. The inference times used to calculate these speedups are reported in Appendix B.3.

Comparison of predictor configurations. To assess whether selecting a separate length predictor for each task family offers a practical benefit, we compare CALP-Category with CALP-Union, which uses one shared predictor. Table 4 shows that CALP-Category achieves higher average task scores on all three backbones and performs better in most benchmark evaluations. This overall advantage supports using CALP-Category as our main configuration.

4.3 Predictive information about response length

We use controlled probes to test whether prompt representations from frozen dLLM backbones contain information predictive of response length before denoising begins. The diagnostic dataset is a balanced mixture constructed in the style of Tulu3 (Lambert et al., 2024), with 512 examples each from code, mathematics, general instruction, and instruction in the style of FLAN (Longpre et al., 2023). It is separate from the length predictor training data. The prediction target is $\ell = \log(1 + L)$. Probe protocol details are provided in Appendix C.1.

Full Hidden applies ridge regression to the hidden state of the last prompt token at the final layer. PCA-64 applies ridge regression after projecting the same representation onto 64 principal components. To distinguish predictive information in these representations from prompt length and coarse differences between diagnostic groups, we compare three baselines: "Prompt Length" uses prompt length after the logarithmic transformation; "Task Mean" uses each diagnostic group's average training target; and "Prompt+Task" combines the two.

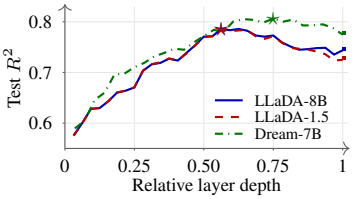
Figure 3 shows that Full Hidden ($R^2 = 0.7250$–0.7747) and PCA-64 ($R^2 = 0.671$–0.750) achieve higher test R^2 than all three baselines across all three backbones. Prompt Length provides little predictive information, whereas Task Mean captures coarse variation between groups. The stronger prediction from prompt representations therefore supports learning a response length prior from the prompt for initial canvas allocation. This evidence concerns statistical predictability, not a causal explanation of model behavior or a guarantee of optimal canvas allocation.

4.4 Sensitivity to Design Choices

Using the dataset and ridge probing protocol in Section 4.3, we vary one component while fixing the others, following the design choices in Section 3.3.

Layer selection. Figure 4(a) varies the readout layer with the last token readout and logarithmic target fixed. Test R^2 peaks at intermediate layers, reaching 0.7862, 0.7848, and 0.8063 on LLaDA-8B, LLaDA-1.5, and Dream-7B, respectively. The final layer retains substantial predictive information (0.7438, 0.7250, and 0.7747), supporting its use as a common readout across backbones.

(a) Layer selection



(b) Token readout

Backbone	Last token	Mean
LLaDA-8B	**0.744**	0.654
LLaDA-1.5	**0.725**	0.655
Dream-7B	**0.775**	0.679

(c) Response length target

Backbone	L	$\log(1+L)$
LLaDA-8B	0.625	**0.744**
LLaDA-1.5	0.600	**0.725**
Dream-7B	0.651	**0.775**

Figure 4: Sensitivity of response length prediction to layer selection, token readout, and target transformation. The final layer retains substantial predictive information, while the last token readout and logarithmic target yield higher test R^2 than their respective alternatives across all three backbones.

Token readout. Figure 4(b) compares the last token's hidden state with mean pooling at the final layer under the logarithmic target. The last token yields test R^2 of 0.744, 0.725, and 0.775, compared with 0.654, 0.655, and 0.679 for mean pooling. Its consistent advantage supports using the last token's hidden state as the prompt representation.

Response length target. Figure 4(c) compares raw length with logarithmic length using the same readout. Logarithmic length achieves higher test R^2 (0.725–0.775 versus 0.600–0.651) in the respective target spaces. Together with its lower sensitivity to long training responses (Appendix C.2), these results support our choice of the logarithmic target.

5 Related Work

Masked diffusion language models. Discrete and masked diffusion models generate text through iterative denoising with bidirectional, parallel token prediction (Austin et al., 2021a; Lou et al., 2023; Sahoo et al., 2024; Nie et al., 2024). LLaDA and Dream extend this paradigm to large language models, demonstrating competitive performance across diverse tasks (Nie et al., 2025; Ye et al., 2025). Denoising operates on a preallocated response canvas whose length determines response capacity and computational cost. Unlike autoregressive generation, which extends a response token by token, this setup requires positions to be allocated in advance.

Canvas allocation and adjustment in dLLMs. One line of work enables length control by adapting the generative model through additional states, modified denoising transitions, or learned EOS and padding behavior (Wu et al., 2026; Kim et al., 2025; Yang et al., 2025; Liu et al., 2026a). Other methods avoid additional training by adjusting canvases using confidence, EOS density, or prediction divergence during denoising (Li et al., 2025; Yang et al., 2026; Cheng et al., 2026; Bouhedja et al., 2026); Complementary work models EOS occurrence as a discrete time survival process to reduce unnecessary denoising (Kobyzev et al., 2026). CAL and Entropy-Valley select infilling and translation lengths, respectively, by scoring canvases with calibrated confidence or predictive entropy (Liu et al., 2026b; Zhan et al., 2026).

Predicting output length before generation. Length prediction supports autoregressive serving (Qiu et al., 2024; Cheng et al., 2024; Xie et al., 2026; Wang et al., 2026; Messenger, 2026), while probing studies identify length signals in frozen representations (Merzouk et al., 2026). We explore using length prediction to guide canvas allocation in masked dLLMs.

6 Conclusion

We present CALP, which learns a response length prior from frozen dLLM prompt representations to guide canvas allocation. Our analysis shows that these representations contain information predictive of reference response length before denoising begins. A lightweight predictor uses this information to allocate a canvas tailored to each prompt, while leaving the backbone and standard denoising procedure unchanged. The allocated length remains fixed throughout generation. Across three backbones and five benchmarks, CALP improves average task scores and reduces inference time relative to DAEDAL and ρ-EOS. These results demonstrate that learned length priors can improve generation quality and computational efficiency without online canvas adjustment.

AI USE STATEMENT

Generative AI assistants were used for feedback on experimental design, code and data analysis, interpretation of results, and polishing the manuscript. The empirical claims reported here are based on recorded experiments, not on values invented by a language model. Human authors remain responsible for the final text, claims, code, and artifacts.

REPRODUCIBILITY STATEMENT

Section 3 specifies the model components, training objectives, and procedure for allocating a response canvas before denoising. Section 4.1 summarizes the experimental setup. Appendix A documents the datasets, training configurations, and evaluation protocol. Appendix B reports the canvas length sweep, task family classification results, and inference times. Appendix C describes the diagnostic probes and sensitivity analysis. Total wall-clock inference times include the additional backbone forward pass that uses only the prompt and is required by the length predictor. An anonymous code repository is linked in Section 4.1.

REFERENCES

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. OpenCodeInstruct: A Large-scale Instruction Tuning Dataset for Code LLMs, 2025. URL https://arxiv.org/abs/2504.04030.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *Proceedings of NAACL-HLT*, pp. 2357–2367, 2019. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245/.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces, 2021a. URL https://arxiv.org/abs/2107.03006.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021b. URL https://arxiv.org/abs/2108.07732.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, 2025. URL https://arxiv.org/abs/2502.02737.

Wail Bouhedja, Amr Mohamed, and Guokan Shang. Carve: Verified expansion for variable-length generation in diffusion language models, 2026. URL https://arxiv.org/abs/2608.30922.

Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, Yulia Tsvetkov, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. The Art of Saying No: Contextual Noncompliance in Language Models, 2024. URL https://arxiv.org/abs/2407.12043.

Sahil Chaudhary. Code Alpaca: An instruction-following LLaMA model for code generation. GitHub repository, 2023. URL https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian,

Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374.

Ke Cheng, Wen Hu, Zhi Wang, Peng Du, Jianguo Li, and Sheng Zhang. Enabling Efficient Batch Serving for LMaaS via Generation Length Prediction, 2024. URL https://arxiv.org/abs/2406.04785.

Zicong Cheng, Ruixuan Jia, Jia Li, Guo-Wei Yang, Meng-Hao Guo, and Shi-Min Hu. Improving variable-length generation in diffusion language models via length regularization, 2026. URL https://arxiv.org/abs/2602.07546.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling Synthetic Data Creation with 1,000,000,000 Personas, 2024. URL https://arxiv.org/abs/2406.20094.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling Diffusion Language Models via Adaptation from Autoregressive Models, 2024. URL https://arxiv.org/abs/2410.17891.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search, 2019. URL https://arxiv.org/abs/1909.09436.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-Fast Language Models Based on Diffusion, 2025. URL https://arxiv.org/abs/2506.17298.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-Order Flexible Length Masked Diffusion, 2025. URL https://arxiv.org/abs/2509.01025.

Ivan Kobyzev, Abbas Ghaddar, and Yufei Cui. Survival-Guided Length Control for Efficient Diffusion Language Models, 2026. URL https://arxiv.org/abs/2608.26374.

Hynek Kydlíček. Math-Verify: Math Verification Library. Software, 2025. URL https://github.com/huggingface/Math-Verify.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing Frontiers in Open Language Model Post-Training, 2024. URL https://arxiv.org/abs/2411.15124.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath TIR. Dataset, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-TIR.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond Fixed: Training-Free Variable-Length Denoising for Diffusion Large Language Models, 2025. URL https://arxiv.org/abs/2508.00819.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023. URL https://arxiv.org/abs/2305.20050.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection, 2017. URL https://arxiv.org/abs/1708.02002.

Chunyu Liu, Zhengyang Fan, Kaisen Yang, and Alex Lamb. Voidpadding: Let [void] handle padding in masked diffusion language models so that [eos] can focus on semantic termination, 2026a. URL https://arxiv.org/abs/2606.17999.

Hengchang Liu, Zhao Yang, and Bing Su. Diffusion lms can approximate optimal infilling lengths implicitly, 2026b. URL https://arxiv.org/abs/2602.00476.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22631–22648, 2023. URL https://proceedings.mlr.press/v202/longpre23a.html.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, 2017. URL https://arxiv.org/abs/1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution, 2023. URL https://arxiv.org/abs/2310.16834.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering Code Large Language Models with Evol-Instruct, 2023. URL https://arxiv.org/abs/2306.08568.

Mohamed Amine Merzouk, Dmitri Carpov, Mirko Bronzi, Damiano Fornasiere, and Adam Oberman. How Much is Left? LLMs Linearly Encode Their Remaining Output Length, 2026. URL https://arxiv.org/abs/2607.05316.

Glen Messenger. ALPS: Activation-Based Length Prediction for Intelligent LLM Inference Scheduling. Zenodo preprint, 2026. URL https://doi.org/10.5281/zenodo.19078431.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-Math: Unlocking the potential of SLMs in Grade School Math, 2024. URL https://arxiv.org/abs/2402.14830.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up Masked Diffusion Models on Text, 2024. URL https://arxiv.org/abs/2410.18514.

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, 2025. URL https://arxiv.org/abs/2502.09992.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. Efficient Interactive LLM Serving with Proxy Model-based Sequence Length Prediction, 2024. URL https://arxiv.org/abs/2404.08509.

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, 2024. URL https://arxiv.org/abs/2406.07524.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed Diffusion: A Large-Scale Diffusion Language Model with High-Speed Inference, 2025. URL https://arxiv.org/abs/2508.02193.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Zdeněk Kasner, Khalid, LSinev, Jeffrey Hsu, Anjor Kanekar, KonradSzafer, and AndyZwei. EleutherAI/lm-evaluation-harness: v0.4.3. Software release v0.4.3, 2024. URL https://doi.org/10.5281/zenodo.12608602.

theblackcat102. Evol-CodeAlpaca-v1. Dataset, 2023. URL https://huggingface.co/datasets/theblackcat102/evol-codealpaca-v1.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating AI for Math with Massive Open-Source Instruction Data, 2024. URL https://arxiv.org/abs/2410.01560.

Jing Wang, Yu-Yang Qian, Ke Xue, Chao Qian, Peng Zhao, and Zhi-Hua Zhou. Robust Length Prediction: A Perspective from Heavy-Tailed Prompt-Conditioned Distributions, 2026. URL https://arxiv.org/abs/2604.07931.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering Code Generation with OSS-Instruct, 2023. URL https://arxiv.org/abs/2312.02120.

Yuxiang Wei, Federico Cassano, Jiawei Liu, Yifeng Ding, Naman Jain, Zachary Mueller, Harm de Vries, Leandro von Werra, Arjun Guha, and Lingming Zhang. SelfCodeAlign: Self-Alignment for Code Generation, 2024. URL https://arxiv.org/abs/2410.24198.

Zirui Wu, Lin Zheng, Zhihui Xie, Jiacheng Ye, Jiahui Gao, Shansan Gong, Yansong Feng, Zhenguo Li, Wei Bi, Guorui Zhou, and Lingpeng Kong. DreamOn: Diffusion Language Models For Code Infilling Beyond Fixed-size Canvas, 2026. URL https://arxiv.org/abs/2602.01326.

Huanyi Xie, Yubin Chen, Liangyu Wang, Lijie Hu, and Di Wang. Predicting LLM Output Length via Entropy-Guided Representations, 2026. URL https://arxiv.org/abs/2602.11812.

Jingyi Yang, Yuxian Jiang, and Jing Shao. ρ-EOS: Training-free Bidirectional Variable-Length Control for Masked Diffusion LLMs, 2026. URL https://arxiv.org/abs/2601.22527.

Yicun Yang, Cong Wang, Shaobo Wang, Zichen Wen, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Diffusion LLM with Native Variable Generation Lengths: Let [EOS] Lead the Way, 2025. URL https://arxiv.org/abs/2510.24605.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion Large Language Models, 2025. URL https://arxiv.org/abs/2508.15487.

Yan Zhan, Mengkai Hou, Wanting Zhang, and Zhijun Gao. Length-adaptive decoding for masked diffusion machine translation, 2026. URL https://arxiv.org/abs/2608.22274.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models, 2023. URL https://arxiv.org/abs/2311.07911.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-Reduced Preference Optimization for Large Language Diffusion Models, 2025. URL https://arxiv.org/abs/2505.19223.

A Experimental Details

This section records the data sources, training settings, and evaluation protocol underlying Section 4.

A.1 Training Data and Sources

Table 5 summarizes the length predictor training and validation data: 107,900 training and 13,418 validation examples, totaling 121,318. Mathematics spans grade school problems, competition mathematics, and reasoning framed as instruction following, drawing on GSM8K and MATH training material, OpenMathInstruct-2, Orca-Math, and Tulu3 data (Cobbe et al., 2021; Hendrycks et al., 2021; Toshniwal et al., 2024; Mitra et al., 2024; Lambert et al., 2024). Code covers functions, program completion, and programming requests in natural language, using Self-OSS, code based on personas, Magicoder, OpenCodeInstruct, and additional public sources (Wei et al., 2024; 2023; Ahmad et al., 2025; Ge et al., 2024). General instruction following combines Tulu3 persona instructions, CoCoNot, and SmolTalk (Lambert et al., 2024; Brahman et al., 2024; Ben Allal et al., 2025). Table 6 gives the complete source composition, including collections released with synthetic reference responses.

Table 5: Training and validation examples by task family for length predictor and classifier training.

Task family	Training	Validation	Total
Math	35,992	4,474	40,466
Code	43,078	5,407	48,485
General	28,830	3,537	32,367
Total	107,900	13,418	121,318

Table 6: Source composition of the training and validation data, with related source aliases grouped together.

Task family	Source group	Examples
Math	OpenMathInstruct-2 (Toshniwal et al., 2024)	10,719
	Tulu/persona mathematics and algebra (Lambert et al., 2024; Ge et al., 2024)	11,938
	NuminaMath-TIR adapted partitions (Li et al., 2024)	2,578
	MATH training material (Hendrycks et al., 2021)	5,241
	GSM8K training material (Cobbe et al., 2021)	5,646
	MathQA (Amini et al., 2019)	1,116
	Orca-Math (Mitra et al., 2024)	2,230
	native records in the style of FLAN (Longpre et al., 2023)	998
Code	Self-OSS Python (Wei et al., 2024)	15,142
	Tulu 3 persona Python (Lambert et al., 2024)	11,534
	PersonaHub Code v2 (Ge et al., 2024)	4,503
	CodeAlpaca-20k (Chaudhary, 2023)	4,489
	Evol CodeAlpaca (decontaminated subset) (theblackcat102, 2023; Luo et al., 2023)	4,457
	Magicoder OSS (Wei et al., 2023)	4,262
	OpenCodeInstruct (Ahmad et al., 2025)	3,973
	CodeSearchNet Python (Husain et al., 2019)	125
General	Tulu 3 persona instruction following (Lambert et al., 2024)	21,591
	CoCoNot adapted records (Brahman et al., 2024)	9,665
	SmolTalk (Ben Allal et al., 2025)	1,111

Exact duplicate check. We audited all 107,900 training prompts and 13,418 validation prompts against the 3,024 evaluation questions from GSM8K, MATH500, HumanEval, MBPP, and IFEval (Section 4.1). The audited prompt membership and splits are shared across all three backbones. We checked full-prompt equality after Unicode NFKC normalization, case folding, and whitespace normalization, and compared ASCII letter-and-number token sequences while ignoring punctuation. Both checks found zero matches for every benchmark in both splits, confirming the absence of exact whole-prompt duplicates under either normalization scheme.

A.2 Predictor and Classifier Training

Table 7: Training configurations for the separately optimized length predictor and task family classifier.

Setting	Length predictor	Task family classifier
Architecture	$d \to d/2 \to 1$ MLP	LayerNorm, $d \to 512 \to 3$ MLP
Activation	GELU	GELU
Objective	Mean squared error	Weighted CE with focal modulation
Learning rate	10^{-4}	3×10^{-3}
Weight decay	0.01	0.01
Batch size	512	2048
Epochs	30	30

Length predictor. The length predictors use the MLP with two layers described in Section 3 and the mean squared error objective in Equation 5. The backbone remains frozen, and only the lightweight prediction head is optimized. CALP-Category trains one predictor per task family, whereas CALP-Union trains one predictor on their union.

Task family classifier. The classifier is optimized separately from the length predictors. We use the prompt representations $\mathbf{h}_i$ and true task families c_i defined in Section 3.4, and the classifier score vector $\mathbf{s}(x_i)$ defined in Equation 4. Feature standardization is included in f_{cls}. Let ω denote the classifier parameters and $\mathcal{I}_{\text{cls}}$ its training index set, with $N_{\text{cls}} = |\mathcal{I}_{\text{cls}}|$. Define

$$p_\omega(c \mid \mathbf{h}_i) = [\text{softmax}(\mathbf{s}(x_i))]_c . \tag{6}$$

With n_c training examples in task family c, inverse frequency weights and weighted cross entropy are

$$w_c = \frac{N_{\text{cls}}}{|\mathcal{C}|\, n_c}, \qquad a_i = -w_{c_i} \log p_\omega(c_i \mid \mathbf{h}_i). \tag{7}$$

The implementation applies a variant of focal modulation (Lin et al., 2017) to this weighted loss. For a minibatch $\mathcal{B} \subseteq \mathcal{I}_{\text{cls}}$,

$$\mathcal{L}_{\text{cls}}(\omega; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} [\max\{1 - \exp(-a_i), \epsilon\}]^\gamma \, a_i, \tag{8}$$

where $\gamma = 1$ and $\epsilon = 10^{-6}$. The modulation uses the weighted loss, so $\exp(-a_i) = p_\omega(c_i \mid \mathbf{h}_i)^{w_{c_i}}$, not the unweighted class probability. Class weights are computed from $\mathcal{I}_{\text{cls}}$ only, never from validation data or generation benchmarks. The classifier uses AdamW (Loshchilov & Hutter, 2017) and the datasets in Table 5. Classification results are reported in Appendix B.2.

A.3 Evaluation Protocol and Baseline Settings

Benchmarks and scoring. We evaluate GSM8K, MATH500, MBPP, HumanEval, and IFEval. We evaluate the benchmarks using LM Evaluation Harness (Sutawika et al., 2024), with 8, 4, 3, 0, and 0 demonstrations, respectively. The metrics are flexible numeric exact match for GSM8K, Math-Verify (Kydlíček, 2025) accuracy for MATH500, pass@1 for both code benchmarks, and strict accuracy at the prompt level for IFEval.

Canvas and denoising settings. The predicted response length is converted once into the response canvas before denoising and remains fixed within the sample. Our runs use a maximum canvas length of 1024, block size 32, transfer confidence 0.9, and temperature zero. The backbone and the original fixed canvas denoising procedure are unchanged.

Dynamic baseline settings. We use the official implementations of DAEDAL and ρ-EOS with the hyperparameters recommended in their papers. DAEDAL (Li et al., 2025) starts with 64 positions and permits expansion to a maximum canvas length of 1024. For ρ-EOS, we use an initial length of 64, density thresholds of 0.4 and 0.8, an exponential scheduler, expansion factor 8, and maximum length 1024.

B Additional Experimental Results

This section provides the full fixed canvas sweep, task family classification results, detailed inference time measurements, and predicted response length distributions supporting Section 4.

B.1 Fixed Canvas Length Sweep

To examine how fixed canvas length affects task performance and canvas utilization, we sweep lengths $C \in \{32, 64, 128, 256, 512, 1024\}$ across three backbones and five benchmarks. Tables 8–10 report Acc., E_{token}, and E_{ratio} at each length; the allocated length N_{token} equals the corresponding column heading. These individual results allow readers to inspect how quality varies with canvas allocation.

Comparisons with fixed canvases require care because the best fixed length for a test benchmark is known only after observing its outcomes. We therefore use Fixed Avg, the average of task scores and token statistics at canvas lengths 128, 256, and 512, as a transparent summary across fixed lengths, rather than selecting the length with the highest test score for each benchmark. Fixed Avg is a reporting reference, not an inference policy. A length chosen from this sweep using evaluation scores is a reference selected in hindsight, not a deployable oracle for new prompts.

In contrast, CALP selects a canvas for each prompt using only information available before denoising. The reference response length used to train its predictor is a proxy for useful response capacity, not an oracle for the optimal canvas length. We therefore assess the value of this proxy through downstream task scores and measured inference time, without claiming that accurate length prediction guarantees optimal allocation.

The sweep illustrates why utilization and quality must be considered together. In Table 8, GSM8K at length 32 has 96.9% utilization but a score of 34.19, whereas length 256 raises the score to 81.12 despite lower utilization. Nor is more capacity always better: Table 9 reports a MATH500 score of 42.20 at length 512 and 37.00 at length 1024. These observations are consistent with both inadequate capacity and excess allocation being relevant, without identifying a universal best length.

Table 8: Fixed canvas length sweep on Dream-7B. Each length is evaluated using the native benchmark score (Acc., %), N_{token}, E_{token}, and canvas utilization E_{ratio}. Lengths 128, 256, and 512 contribute to Fixed Avg in the main results.

Benchmark	Metric	Canvas length					
		32	64	128	256	512	1024
GSM8K	Acc.↑	34.19	58.61	78.70	81.12	80.44	80.89
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	31.0	62.5	100.9	102.3	101.8	106.7
	E_{ratio} ↑	96.9%	97.7%	78.8%	39.9%	19.9%	10.4%
MATH500	Acc.↑	21.80	30.60	44.20	47.60	47.80	47.20
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	28.7	61.3	115.4	178.9	225.4	283.1
	E_{ratio} ↑	89.7%	95.8%	90.2%	69.9%	44.0%	27.7%
MBPP	Acc.↑	36.80	51.80	55.80	55.80	53.40	55.20
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	29.5	42.6	47.1	47.6	48.6	50.6
	E_{ratio} ↑	92.1%	66.6%	36.8%	18.6%	9.5%	4.9%
HumanEval	Acc.↑	19.51	45.73	59.15	59.15	55.49	56.71
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	29.9	49.7	77.9	128.4	134.2	114.4
	E_{ratio} ↑	93.3%	77.7%	60.8%	50.2%	26.2%	11.2%
IFEval	Acc.↑	31.98	47.69	49.35	50.28	51.39	50.83
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	15.1	45.6	86.9	153.4	245.7	310.1
	E_{ratio} ↑	47.1%	71.3%	67.9%	59.9%	48.0%	30.3%

Table 9: Fixed canvas length sweep on LLaDA-8B. Each length is evaluated using the native benchmark score (Acc., %), N_{token}, E_{token}, and canvas utilization E_{ratio}. Lengths 128, 256, and 512 contribute to Fixed Avg in the main results.

Benchmark	Metric	Canvas length					
		32	64	128	256	512	1024
GSM8K	Acc.↑	40.18	60.88	79.61	81.65	83.02	83.70
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	31.0	62.8	116.5	215.5	206.6	163.4
	E_{ratio} ↑	96.8%	98.1%	91.0%	84.2%	40.4%	16.0%
MATH500	Acc.↑	21.80	27.40	36.00	38.80	42.20	37.00
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	30.9	62.6	121.9	228.7	382.4	470.0
	E_{ratio} ↑	96.5%	97.8%	95.2%	89.3%	74.7%	45.9%
MBPP	Acc.↑	28.40	36.80	38.80	39.00	40.80	40.20
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	30.6	57.1	68.2	61.6	63.2	81.6
	E_{ratio} ↑	95.7%	89.3%	53.3%	24.0%	12.4%	8.0%
HumanEval	Acc.↑	19.51	33.54	41.46	41.46	43.29	40.24
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	30.0	61.6	122.6	238.1	476.5	840.3
	E_{ratio} ↑	93.8%	96.3%	95.8%	93.0%	93.1%	82.1%
IFEval	Acc.↑	48.24	53.97	55.27	56.38	57.30	56.38
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	22.7	48.7	96.8	172.1	300.7	421.8
	E_{ratio} ↑	71.1%	76.1%	75.6%	67.2%	58.7%	41.2%

Table 10: Fixed canvas length sweep on LLaDA-1.5. Each length is evaluated using the native benchmark score (Acc., %), N_{token}, E_{token}, and canvas utilization E_{ratio}. Lengths 128, 256, and 512 contribute to Fixed Avg in the main results.

Benchmark	Metric	Canvas length					
		32	64	128	256	512	1024
GSM8K	Acc.↑	42.76	65.05	80.06	84.23	84.38	84.08
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	31.0	62.7	118.0	219.1	209.9	168.2
	E_{ratio} ↑	96.9%	98.0%	92.2%	85.6%	41.0%	16.4%
MATH500	Acc.↑	22.40	30.60	36.00	39.40	44.60	40.40
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	31.0	62.8	123.1	237.7	394.7	507.1
	E_{ratio} ↑	96.9%	98.1%	96.2%	92.8%	77.1%	49.5%
MBPP	Acc.↑	27.60	37.40	39.40	40.60	41.60	40.80
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	30.7	57.3	75.0	80.4	69.6	85.8
	E_{ratio} ↑	96.0%	89.5%	58.6%	31.4%	13.6%	8.4%
HumanEval	Acc.↑	21.34	31.10	40.85	43.29	42.68	40.24
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	E_{token}	30.0	62.1	125.2	238.7	461.9	774.4
	E_{ratio} ↑	93.8%	97.0%	97.8%	93.2%	90.2%	75.6%
IFEval	Acc.↑	51.57	58.04	58.78	59.33	59.15	60.07
	N_{token} ↓	32.0	64.0	128.0	256.0	512.0	1024.0
	$E_{\text{token}}E_{\text{token}}$	24.6	50.3	97.9	176.5	299.4	422.2
	E_{ratio} ↑	76.9%	78.5%	76.5%	68.9%	58.5%	41.2%

B.2 Task Family Classification Accuracy

We evaluate the classifier before length predictor selection to distinguish routing errors from errors in length prediction or response generation. The architecture and optimization are specified in Appendix A.2. Table 11 reports task family classification accuracy on the five generation benchmarks.

Table 11: Task family classification accuracy on the five generation benchmarks before length predictor selection. Accuracy exceeds 99% for every backbone and benchmark, with routing errors confined to IFEval. Entries report percentages, with correct/total counts in parentheses. Micro pools all 3,024 benchmark examples for each backbone.

Benchmark	Family	Dream-7B	LLaDA-8B	LLaDA-1.5
GSM8K	Math	100.00 (1319/1319)	100.00 (1319/1319)	100.00 (1319/1319)
MATH500	Math	100.00 (500/500)	100.00 (500/500)	100.00 (500/500)
MBPP	Code	100.00 (500/500)	100.00 (500/500)	100.00 (500/500)
HumanEval	Code	100.00 (164/164)	100.00 (164/164)	100.00 (164/164)
IFEval	General	99.26 (537/541)	99.63 (539/541)	99.63 (539/541)
Micro	All	99.87 (3020/3024)	99.93 (3022/3024)	99.93 (3022/3024)

Every evaluated mathematics and code example is assigned to the expected task family. Errors are confined to IFEval: 4/541 on Dream-7B and 2/541 on each LLaDA backbone. Pooling the 3,024 benchmark examples within each backbone gives classification accuracies of 99.87%, 99.93%, and 99.93%, respectively. These results validate the routing mechanism on the evaluated benchmarks; they measure coarse task family recognition rather than response correctness.

B.3 Inference Time Results

Table 12 reports the measured inference times for each configuration and benchmark. To investigate the source of the observed speedup, we record the number of backbone forward passes during denoising. For example, on LLaDA-8B/MBPP, CALP-Category uses an average of 39.898 backbone evaluations per request, compared with 53.660 for DAEDAL and 44.786 for ρ-EOS, measured over 500 requests per method. CALP-Category requires fewer backbone evaluations. This lower evaluation count helps explain its faster inference.

Table 12: Average inference time per request in seconds for CALP-Category, DAEDAL, ρ-EOS, and CALP-Union. Both CALP configurations achieve faster inference than DAEDAL and ρ-EOS across all three backbones and five benchmarks.

Backbone	Benchmark	CALP-Category	DAEDAL	ρ-EOS	CALP-Union
Dream-7B	GSM8K	3.195	6.941	5.001	3.014
	MATH500	5.541	18.639	9.101	5.218
	MBPP	1.742	3.706	2.726	1.662
	HumanEval	2.747	9.177	3.684	2.789
	IFEval	3.679	17.361	9.963	3.877
LLaDA-8B	GSM8K	5.175	15.335	7.011	4.430
	MATH500	10.096	32.506	14.343	8.773
	MBPP	2.786	5.605	4.420	2.552
	HumanEval	6.235	29.804	13.851	6.429
	IFEval	3.485	21.154	7.658	3.744
LLaDA-1.5	GSM8K	5.497	15.335	7.158	4.815
	MATH500	9.956	35.116	16.220	9.378
	MBPP	3.037	5.924	5.110	2.891
	HumanEval	5.656	31.656	16.400	5.662
	IFEval	3.673	23.604	10.250	4.099

B.4 Predicted Response Length Distributions

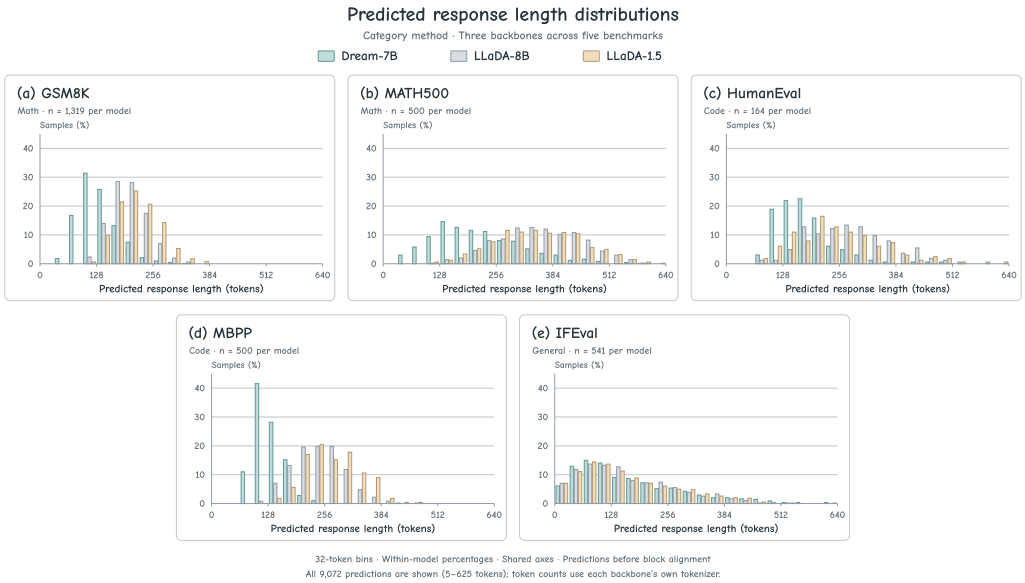


Figure 5: Predicted response length distributions of CALP-Category across five benchmarks and three backbones. Bars show percentages in 32-token bins, normalized separately for each backbone within each benchmark. Each example is counted once, regardless of generation correctness. Lengths use each backbone's tokenizer and correspond to recorded predictions before block alignment, rather than generated response lengths or allocated canvas lengths.

Figure 5 shows that CALP-Category predicts different response lengths for different prompts, even within the same benchmark. This enables the method to tailor the response canvas to each prompt's estimated length requirement, rather than assigning a uniform length to all inputs.

C Diagnostic Analyses

This section documents the diagnostic data and probing protocol used to assess statistical readability of response length, followed by the sensitivity analysis for long training responses. These analyses support the method design but do not imply a causal length variable in the backbone.

C.1 Diagnostic Data and Probing Protocol

Section 4.3 describes the diagnostic data and its four groups: code, mathematics, general instruction, and instruction in the style of FLAN. The length predictor training data instead use the three task families in Table 5.

Probe fitting and evaluation. Ridge probes use the hidden state of the last prompt token at the final layer of the frozen backbone. Full Hidden uses the complete representation vector; PCA-64 uses its projection onto 64 principal components. We select the regularization coefficient α by cross validation with five folds on the training split and report R^2 on the test set. The comparisons in Section 4.4 follow this protocol, varying the readout layer, token readout, or target scale while fixing the other choices. The limited probe capacity tests whether length information is accessible without modifying the backbone.

Response length may correlate with prompt length and vary systematically across diagnostic groups. Prompt Length, Task Mean, and Prompt+Task test whether hidden representations provide predictive information beyond these associations. Their main results are reported in Section 4.3; Prompt Length achieves $R^2 = -0.001$–0.012, while Task Mean reaches 0.454–0.513, and adding prompt length provides only a small further improvement.

C.2 Sensitivity to Long Training Responses

To test robustness to long training responses, we fit one ridge probe on the full training set and another on the subset obtained by excluding responses with $L > 1000$. Both are evaluated on the same unchanged test set $\mathcal{T}$. We repeat this comparison for $\psi_{\mathrm{raw}}(L) = L$ and $\psi_{\log}(L) = \log(1+L)$.

For each target transformation ψ, let $u_i = \psi(L_i)$ be the target of test example i. Let $\hat{u}_i^{\mathrm{full}}$ and $\hat{u}_i^{\mathrm{trim}}$ be the predictions from the two training versions, expressed on that same target scale. We define

$$\Delta_u = \frac{1}{\sigma_{u,\mathcal{T}}} \sqrt{\frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \left(\hat{u}_i^{\mathrm{trim}} - \hat{u}_i^{\mathrm{full}}\right)^2}, \tag{9}$$

where $\sigma_{u,\mathcal{T}}$ is the standard deviation of the test targets $\{u_i\}_{i \in \mathcal{T}}$. This measures the change between two sets of predictions relative to target variability, not their error against the target values.

Across the three backbones, the normalized prediction shift averages 0.331 for raw length and 0.103 for logarithmic length (Table 13). The smaller shift indicates lower sensitivity to excluding long training responses on this diagnostic dataset, supporting the logarithmic target used by the length predictor.

Normalized prediction shift ↓	
$\psi_{\mathrm{raw}}(L) = L$	$\psi_{\log}(L) = \log(1 + L)$
0.331	**0.103**

Table 13: Normalized prediction shift Δ_u of ridge probes after excluding training responses with $L > 1000$. The logarithmic target yields a smaller shift than raw length, indicating lower sensitivity to excluding long training responses. Results are evaluated on the unchanged test set and averaged across three backbones. Δ_u measures prediction change rather than prediction error.