Predict-then-Diffuse: Adaptive Response Length for Compute-Budgeted Inference in Diffusion LLMs

Michael Rottoli, Subhankar Roy, Stefano Paraboschi
Università degli Studi di Bergamo, Italy

***Abstract*—Diffusion-based Large Language Models (D-LLMs) represent a promising frontier in generative AI, offering fully parallel token generation that can lead to significant throughput advantages and superior GPU utilization over the traditional autoregressive paradigm. However, this parallelism is constrained by the requirement of a fixed-size response length prior to generation. This architectural limitation imposes a severe trade-off: oversized response length results in computational waste on semantically meaningless padding tokens, while undersized response length cause output truncation requiring costly re-computations that introduce unpredictable latency spikes. To tackle this issue, we propose Predict-then-Diffuse, a simple and model-agnostic framework that enables compute-budgeted inference per input query by first estimating the response length and then using it to run inference with D-LLM. At its core lies an Adaptive Response Length Predictor (AdaRLP), which estimates the optimal response length given an input query. As a measure against under-estimating the response length and re-running inference with a higher value, we introduce a data-driven safety mechanism based on a small increase of the predicted length. As a whole, our framework avoids that computation is wasted on padding tokens, at the same time preserving output quality. Experimental validation on multiple datasets demonstrates that Predict-then-Diffuse significantly reduces computational costs (FLOP) compared to the default D-LLM inference mechanism, while being robust to skewed data distributions. Our code is available at: https://github.com/mchl-labs/predict-then-diffuse**

***Index Terms*—Diffusion Language Models, Efficient Inference**

I. Introduction

The landscape of Natural Language Processing (NLP) is dominated by highly-scalable Autoregressive Large Language Models (AR-LLMs) such as GPT [1] and LLaMA [2]. These models generate text sequentially, a process that inherently limits inference speed and hardware utilization due to memory-bandwidth bottlenecks and especially due to its sequential nature. Recently, Diffusion-based Language Models (D-LLMs) like LLaDA [3] and Dream [4] have emerged as a promising alternative. By adapting diffusion principles from image generation [5], D-LLMs generate all tokens in an output simultaneously using bidirectional attention, thus departing from sequential generation. A crucial advantage of the diffusion paradigm is that it unlocks the potential for high-throughput generation with superior GPU utilization.

Despite the parallelization promises offered by D-LLMs, they face an operational hurdle of *static sequence length* during inference. Unlike AR models that for a given input query generate output until an `<EOS>` token is produced, a D-LLM must be initialized with a fixed sequence (or response) length, defining the length of the output sequence. As illustrated in Fig. 1 (a), a D-LLM performs computations over both meaningful tokens and semantically meaningless `<PAD>` tokens because the model cannot know *a priori* at the start of the diffusion process which tokens will be meaningful. The waste becomes substantial as the true response length and maximum response length diverge, since the computational complexity in Transformer-based models scales quadratically with the sequence length. Moreover, manually setting the response length to a conservatively low value causes truncation of the output, degrading quality and user experience.

Since queries vary significantly in their complexity, existing mitigations, such as Block Diffusion [6], introduce a semi-autoregressive approach, which combines the AR and diffusion paradigms to generate arbitrary-length outputs. While this enables intra-block parallel sampling, it calls for modified training objective and sampling algorithm. Distinguishing ourselves from previous approaches, we propose the **Predict-then-Diffuse** framework, which fulfills the same goal without making any architectural modifications or introducing any new training objectives. Our proposed framework is based on the simple idea of predicting the optimal response length conditioned on the input prompt, prior to generation, such that computation on `<PAD>` tokens can be minimized. At its core is an **Ada**ptive **R**esponse **L**ength **P**redictor (**AdaRLP**) module (see Fig. 1 (b)) that implements a lightweight predictive model, which takes as input the raw input prompt and predicts the response length. Its lightweight design adds negligible computational overhead to the actual output generation process and its model agnostic nature allows it to be plugged into any D-LLM. In addition, we equip Predict-then-Diffuse with a data-driven risk-aware Safety Margin mechanism that mitigates the issue of under-predicting the response length. This reduces the degradation of the output quality and provides a reliable estimate of latency, which is crucial in production environments.

In summary, our **contributions** are:

- We highlight and study the computational bottleneck of standard Diffusion LLMs during inference. To this end we derive an analytical cost model for D-LLM inference and validate it empirically on modern hardware.
- We propose **Predict-then-Diffuse** framework that estimates the response length prior to generation, thus reducing the computational waste on `<PAD>` tokens. In addition, we introduce a data-driven Safety Margin that trades a negligible padding overhead for preventing

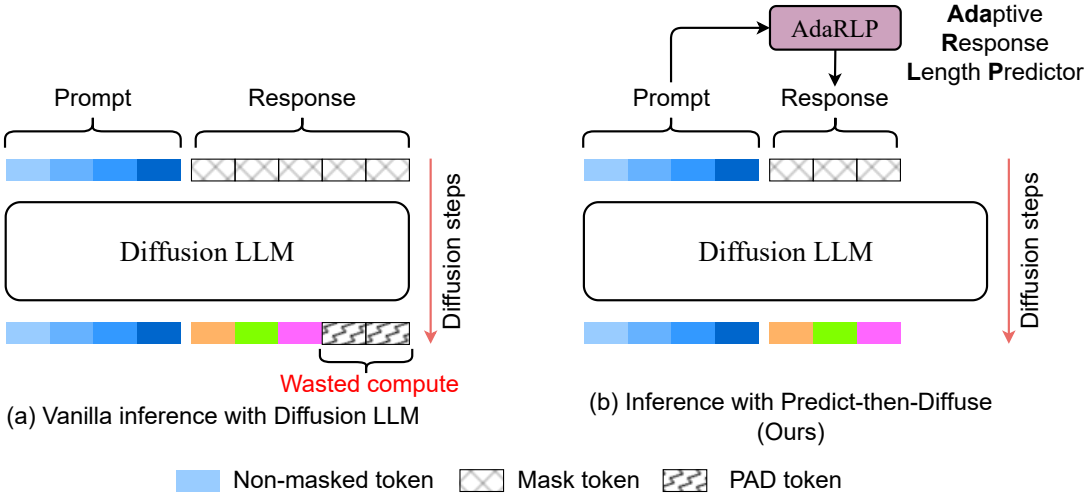

Fig. 1: **Comparison of inference strategies with Diffusion LLMs.** (a) Vanilla inference with a fixed response length results in wasted compute for the <PAD> tokens. (b) In our proposed Predict-then-Diffuse system, the **Ada**ptive **R**esponse **L**ength **P**redictor (**AdaRLP**) auxiliary module predicts the response length conditioned on the input prompt, circumventing wasted compute on processing <PAD> tokens, without affecting the quality of the response

output truncation.

- We benchmark our proposed framework and a series of baselines on a collection of multiple datasets. Experimental results demonstrate that Predict-then-Diffuse reduces FLOP by 99.34% with respect to default D-LLM inference, while maintaining quality of output.

II. Related Work

Autoregressive (AR) LLMs are a family of generative models that model the conditional probability of each token given all the previous tokens, as exemplified by GPT and LLaMA [1], [2]. Albeit highly-scalable and effective, AR generation is inherently sequential (i.e., predict one token at a time), creating a bottleneck during inference. To improve throughput and efficiency in AR, inference engines (eg, vLLM [7]) have been developed that primarily improve KV-cache memory management. In order to make LLMs inherently parallel, an alternative diffusion-based language model has been studied, which is the focus of this work.

Diffusion LLMs (D-LLMs) [8] are built upon the idea of diffusion, where clean text is systematically corrupted (through noise injection or masking) and then the model learns to remove the noise through iterative denoising (eg, LLaDA [3]). By design, D-LLMs are parallelizable on modern hardware at inference, as the bidirectional attention allows all token positions to be simultaneously updated at each denoising step. However, D-LLMs suffer from two main drawbacks: (i) D-LLMs are less performant than AR models, and (ii) Unlike AR LLMs, D-LLMs must be initialized with a fixed response length *a priori* during inference. In particular, fixed response length can be problematic, as it can lead to computation being wasted on processing <PAD> tokens, and output being truncated if the true response is longer than the maximum response length. This has led to further studies aiming to address these two challenges.

Budgeted-inference of LLMs. To combine the strengths of both approaches, BlockDiffusion [6], [9] unifies the two paradigms by modelling an autoregressive probability distribution over blocks of discrete random variables. This allows the model to generate arbitrary-length high-quality output and also enables intra-block parallel sampling. In contrast, our proposed compute-budgeted strategy makes no modifications to the architecture or the training objective of a D-LLM. We train a lightweight AdaRLP module that predicts the expected response length for a given input query, thus saving computation on processing <PAD> tokens. A related work, TimeBill [10], aims to achieve an optimal KV cache eviction ratio through the estimation of end-to-end execution time, but unlike our work, TimeBill employs it for AR LLMs that are time-budgeted during inference.

III. Methods

In this section we present Predict-then-Diffuse framework designed for compute-budgeted inference. We first discuss some preliminaries on D-LLMs (Sec. III-A), and the analytical cost model of D-LLMs (Sec. III-B). Finally, we present the Predict-then-Diffuse framework (Sec. III-C) in detail.

A. Preliminaries: Diffusion LLMs

Diffusion models, pioneered for image generation tasks [5], have also been used for language modelling tasks. At its core, D-LLMs learn to recover data, i.e., text, from increasingly noised versions by reverse diffusion process, and generate new samples by iteratively reversing the stochastic corruption. Based on the noising process, which can be performed either in the continuous embedding space or in the discrete token space, D-LLMs can be categorized as Continuous D-LLMs

and Discrete D-LLMs, respectively. Our proposed framework can be seamlessly applied to any D-LLM, but for convenience we focus on the Discrete D-LLM LLaDA [3].

In LLaDA, the **forward diffusion** process masks a clean sequence x_0 with some probability $t \sim \mathcal{U}(0,1)$ with <MASK> tokens to obtain a partially masked sequence x_t. The **reverse diffusion** process learns to reverse masking by predicting the tokens in masked positions. In detail, a parametric model $p_\theta(\cdot \mid x_t)$ takes the masked sequence x_t as input, and predicts all the masked tokens, denoted by $\mathbf{M}$, simultaneously. The model is trained with a cross-entropy loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{t,x_0,x_t}\Big[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = \mathbf{M}]\log p_\theta(x_0^i \mid x_t)\Big], \quad (1)$$

where $\mathbf{1}[\cdot]$ is an indicator function that computes loss only over the masked tokens, and L is the sequence length. During inference, starting from a completely masked sequence ($t = 1$), the model predicts a complete sequence of tokens. Inference is iterative, as it involves re-masking some tokens and then gradually refining its prediction as t moves from 1 to 0. Notably, because of fixed response length L, the tokens appearing after the <EOS> token, designated as <PAD> tokens, are discarded after the final output is generated.

B. Analytical Cost Model

In this work, we highlight that in LLaDA the <PAD> tokens, which have no impact on the final generated output, waste computation during inference, a phenomenon not present in AR LLMs. To better understand the wasted computation in D-LLM, we first quantify the computation during inference by deriving its analytical cost model. We use a standard FLOP (FLoating point OPeration) accounting for Transformer models [11], [12] (see Tab. I for per-component computation) to calculate the FLOP of a forward pass for a block l:

$$\text{FLOP}_l = \underbrace{(6DF + 8D^2)L}_{\text{Terms linear in } L} + \underbrace{(4D)L^2}_{\text{Term quadratic in } L} \quad . \quad (2)$$

For a D-LLM with N Transformer blocks and T diffusion steps, the total FLOP during inference are:

$$\text{FLOP}_{\text{total}} = T \cdot N \cdot D \cdot (\alpha L + \beta L^2), \quad (3)$$

where we use $\alpha = 6F + 8D$ and $\beta = 4$ for brevity. For LLaDA-8B ($D = 4096, F \approx 3D$), the quadratic term dominates when $L > \alpha/\beta \approx 26,000$. Ideally, for short sequences the cost grows linearly, whereas for longer sequences, the quadratic attention term dominates the cost. In the case of D-LLM, the sequence length L needs to be fixed upfront, with the aim of catering to variable-length responses. As a consequence, when the actual response length is short, significant computation is wasted on processing the <PAD> tokens that constitute most of the response.

TABLE I: Per-block FLOP for major components of **Gated Transformer with Multi Head Attention** during a single **forward pass** (inference-only) as a function of sequence length (L), hidden dimension (D), and MLP width (F)

Component	**Computational Cost**
Head	***FLOP***
MLP block	$6LDF$
Q, K, V, O projections	$8LD^2$
Dot-product attention	$4DL^2$

C. Predict-then-Diffuse Framework

We propose **Predict-then-Diffuse**, an intuitively simple framework designed for compute-budgeted inference with a D-LLM. The main idea of Predict-then-Diffuse lies in predicting the expected response length given an input prompt. This most of the times produces a significant shortening of the sequence length, and therefore a reduction in computation on processing the <PAD> tokens. It operates in the pre-inference stage and incurs negligible computational overhead with respect to the FLOP incurred by the actual inference with D-LLM.

As shown in Fig. 1(b), at the core of Predict-then-Diffuse is the **Ada**ptive **R**esponse **L**ength **P**redictor (**AdaRLP**) module that takes as input a prompt and outputs the expected response length. We cast the response length prediction as a regression task. Formally, we assume access to a paired dataset $\mathcal{D} = \{(s_i, k_i)\}_{i=1}^n$, where $s_i \in \mathcal{S}$ is raw text or the $i-$th input prompt in the dataset, and $k_i \in \mathcal{Z}$ is the response length of its corresponding output. Using $\mathcal{D}$, we train the AdaRLP module, or a function $f_{\text{AdaRLP}} : \mathcal{S} \mapsto \mathcal{Z}$, to predict the response length given a prompt. Note that $\mathcal{D}$ is not any specialized dataset, but any supervised fine-tuning (SFT) dataset that is typically used for instruction after the pre-training phase.

AdaRLP design. We design AdaRLP with the following considerations. *First*, we do not want to introduce any significant computational overhead in predicting response length. *Second*, we want to keep the design fairly simple, with no need for an extensive ad-hoc feature engineering. With these goals in mind, we employ, as the predictor f_{AdaRLP}, CatBoost [13], a widely used machine learning algorithm based on gradient boosting over decision trees. CatBoost offers two distinct advantages that satisfy our design desiderata: (i) its extremely fast inference speed ($<$ 1ms), (ii) its ability to natively handle mixed data types, including raw text, in a robust way, and (iii) its consistently high performance on tabular and heterogeneous data without extensive tuning; all these features make it an extremely good response-length predictor.

To test this hypothesis, we evaluated two approaches: (i) $\textbf{AdaRLP}_{\text{text-only}}$: the raw prompt string is passed directly to CatBoost, utilizing its internal quantization and support for text features. (ii) $\textbf{AdaRLP}_{\text{engineered}}$: a more engineered model that in addition to the raw prompt string, uses metadata such as token counts, specific keywords (e.g., "summarize", "list"), punctuation patterns, and Shannon entropy. As we will show later in Sec. IV-C, $\textbf{AdaRLP}_{\text{text-only}}$ significantly outper-

formed **AdaRLP**$_{\text{engineered}}$, suggesting that the native CatBoost algorithm is sufficient to handle a simple task of predicting response length. In theory, one can adapt a small language model to predict response length, but at the cost of increased inference time; hence, we did not explore this option.

Introducing Safety Margin. Contrary to over-predicting the response length, one potential drawback of under-predicting the expected response length is that it will lead to output truncation, i.e., incomplete response from D-LLM, which in turn impacts the output quality. A straightforward solution to this problem consists in increasing the response length upon failure, and re-invoking the D-LLM for inference, but this approach introduces additional FLOP and wastes computation.

As a more adaptive safeguard, we introduce a data-driven safety mechanism specifically designed to prevent degradation of the quality of output. Let us denote the predicted response length from f_{AdaRLP} by $\hat{L}$. We introduce a safety margin δ that is added to the initial prediction $\hat{L}$. The margin is determined empirically, set by calculating a high percentile (e.g., the 95th, denoted p_{safe}) of the positive prediction errors, i.e., instances where true length is greater than the predicted length, $\hat{L} < k$:

$$\delta = \text{Quantile}_{p_{\text{safe}}}[\{k_i - \hat{L}_i \mid k_i > \hat{L}_i\}]. \tag{4}$$

The final effective response length L^* passed to the D-LLM:

$$L^* = \min(\hat{L} + \delta, L_{\text{max}}), \tag{5}$$

where L_{max} is the maximum response length D-LLM was trained with. This statistical buffer minimizes fallbacks while maintaining a tight and close-to-optimal response length.

Inference with Predict-then-Diffuse. Our proposed framework is agnostic to the implementation of the D-LLM, as it operates in the pre-inference phase. While in this work we experiment only with LLaDA, the proposed framework can be seamlessly integrated into the inference pipeline of any D-LLM. In detail, once L^* is determined for a given test sample, LLaDA inference is invoked with the response length L^*, without needing to alter its native inference mechanism. LLaDA then proceeds with its standard T denoising steps, but only processes a sequence of effective length L^*. The experiments show that this drastically reduces the computational load for shorter output responses. In very extreme cases of the model failing to generate a `<EOS>` token even with utilizing L^* as the effective length, we adopt a conservative fallback strategy of re-running inference with maximum response length L_{max}.

IV. Experiments

A. Empirical Validation of Analytical Cost Model

We first empirically validate the analytical cost model introduced in Sec. III-B, where we theoretically showed that FLOP scale quadratically with the response length L of a D-LLM, especially for long sequence lengths. To validate this, we run the LLaDA-8B model with varying response length on NVIDIA H100 (80 GB VRAM) GPUs. In detail, we measured the total FLOP using DeepSpeed [14] and the VRAM usage for a single generation (diffusion steps $T = 128$) with response length ranging from 64 to 30k tokens.

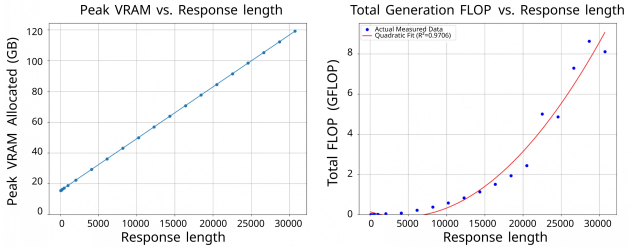

Fig. 2: Empirical validation of D-LLM cost scaling on NVIDIA H100 GPUs. **Left:** VRAM usage scales linearly. **Right:** Total FLOP scales quadratically with canvas size ($R^2 = 0.9706$), strongly supporting the theoretical structure ($O(\alpha L + \beta L^2)$) of our analytical cost model.

Results. We observe from Fig. 2 (**left**) that VRAM usage is linear with the response length. In Fig. 2 (**right**) we observe that FLOP follow a linear fit when the response length is short, whereas for a longer response length, FLOP exhibit a quadratic fit. Overall, the curve follows a quadratic non-linear trend (variance explained, $R^2 = 0.9706$). This experiment confirms that the growth term is not just a theoretical construct, but a dominant factor in real-world deployments where the response length can vary widely.

B. Experimental Setup

Benchmarks. We aggregated 39,994 prompt-response pairs from 5 diverse public sources: ShareGPT, Alpaca, Dolly-15k, OpenOrca, and ELI5 to benchmark our framework [15]–[19]. We curated this composite benchmark to ensure a high variance in output response lengths ($mean \approx 96$, $std \approx 120$, $kurtosis \approx 107$), with an emphasis on long-form generation. This selection strategy was critical to mitigate the bias toward short responses typical of standard open-source instruction-tuning datasets, thereby providing a robust testbed for adaptive response length. Responses were tokenized using the GPT2TokenizerFast tokenizer before calculating the length. The dataset was split 80/20 for training and testing.

Models and baselines. As discussed in Sec. III-C, we evaluated two variants of CatBoost models for implementing AdaRLP: (i) **AdaRLP**$_{\text{text-only}}$ that uses CatBoost's native support for text features, and (ii) **AdaRLP**$_{\text{engineered}}$ that additionally uses hand-engineered features. The predicted real-valued output length in tokens is rounded to the closest integer, since response length is an integer value.

Due to the lack of baselines and competitor methods, we construct the following baselines for quantitative comparison: (i) **Max Response Length**, where we set $L = L_{\text{max}} = 4096$, default setting used by LLaDA. (ii) **Static Response Length** ($L = 200$), a static-length conservative baseline that doubles on fallback if output truncation occurs. (iii) **Mean Doubling Heuristic**, a strategy that initializes the response length to the mean dataset length (≈ 95 tokens) and doubles the size

TABLE II: Length-prediction metrics on the held-out test set. **AdaRLP**$_{\text{text-only}}$ outperforms the **AdaRLP**$_{\text{engineered}}$ variant with hand-engineered features

Model	Metrics		
	RMSE	*MAE*	*% ≤ 10% error*
AdaRLP$_{\text{engineered}}$	81.6	51.6	10.5
AdaRLP$_{\text{text-only}}$	**11.4**	**1.7**	**97.5**

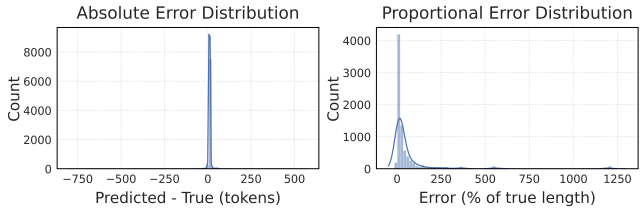

Fig. 3: Absolute and Proportional error distributions of **AdaRLP**$_{\text{text-only}}$

$(95 \rightarrow 190 \rightarrow \dots)$ if truncation occurs. (iv) **Oracle**, where we assume knowledge of the true response length, ie, $L = k$. Unless otherwise stated, all proposed models and baselines use LLaDA-8B for output generation.

Metrics. We compare different methods mainly using Total Tera Floating-point Operations (**TFLOP**) on the full test set, including penalties for fallback re-runs.

C. Predictor Performance

We compare the response length predictive performance of two variants of AdaRLP by comparing their Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). As evident from Tab. II, the simpler **AdaRLP**$_{\text{text-only}}$ outperformed the feature-engineered **AdaRLP**$_{\text{engineered}}$ model by a large margin. In detail, **AdaRLP**$_{\text{text-only}}$ achieved a MAE of 1.7 tokens and an RMSE of 11.4 tokens, with 97.5% of predictions falling within a 10% error margin (see Fig. 3). This indicates that CatBoost effectively learns length-correlated features (e.g., "write a poem" vs. "write a novel") directly from raw text (see Fig. 4). Given its clear superiority, the remainder of our analysis focuses on **AdaRLP**$_{\text{text-only}}$.

D. Computational Efficiency

Tab. III summarizes the computational costs of **Predict-then-Diffuse** and the baselines introduced in Sec. IV-B. The **Max Response Length** baseline is prohibitively expensive, as evident from the highest computational cost of 4.3 TFLOP, due to a long response length, with almost every alternative showing significant improvements. This again underscores the need for adaptive response length during D-LLM inference.

Interestingly, the **Mean Doubling Heuristic** baseline is competitive (99.31% savings) with **Predict-then-Diffuse**. This is due to the specific distribution of our dataset (mean length ≈ 95, median ≈ 63). Since most answers are short, starting at the mean covers the majority of cases without fallback. However, **Predict-then-Diffuse** still slightly outperforms the

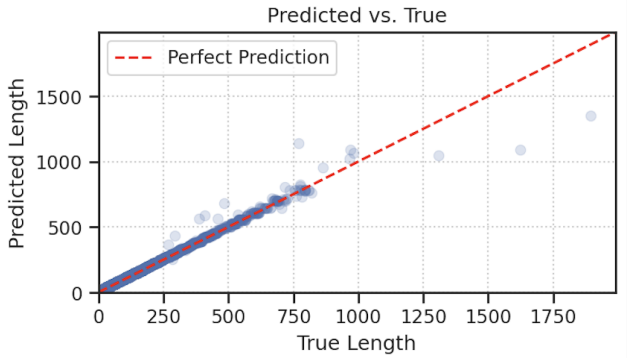

Fig. 4: Deviations from the perfect prediction of **AdaRLP**$_{\text{text-only}}$

TABLE III: Comparison of Computational Cost (in TFLOP), Savings and Fallback rate

Methods	TFLOP	Savings (%)	Fallback Rate (%)
Max Response Length (4096)	4.03	0.0%	0.0%
Static Response Length (200)	0.054	98.6%	22.1%
Doubling Heuristic	0.027	99.31%	-
Predict-then-Diffuse (Ours)	**0.026**	**99.34%**	**0.1%**
Oracle	0.024	99.4%	0.0%

heuristic-based baseline (99.34% savings) while offering superior robustness. The heuristic is brittle, as on a bimodal dataset (e.g., 50% short chats, 50% long reports) the mean-based initialization would fail frequently for long prompts, incurring higher costs. To verify this, we simulated a dataset with responses following a bimodal distribution (60% short queries, $mean = 50$ tokens; 40% long reports, $mean = 3000$ tokens). We observe that the heuristic-based baseline frequently fails multiple times before converging on the required length (e.g., $95 \rightarrow 190 \rightarrow \dots \rightarrow 4096$), incurring cumulative retry costs. In contrast, **Predict-then-Diffuse** retains a 19% computational advantage over the heuristic in this bimodal regime, demonstrating its necessity for general-purpose deployment where output length variance is high.

E. Latency Determinism

Beyond aggregate TFLOP reduction, reliable inference entails another important requirement: *latency determinism*. While heuristic strategies may achieve high average savings on skewed distributions, they also introduce stochastic latency behavior and exhibit a high variance. A long-tail query (e.g., a lengthy report) processed by the **Mean Doubling Heuristic** baseline triggers multiple inferences (e.g., $L = 95 \rightarrow 190 \rightarrow \dots$), causing the user-perceived latency to frequently spike unpredictably. In contrast, **Predict-then-Diffuse** aims to yield a consistent, single-shot generation profile for 99.9% of requests. By integrating the data-driven Safety Margin ($\delta = 5$ tokens), as discussed in Sec. III-C, we suppressed the frequency of fallback retries to negligible levels (down from 1.43% to 0.1%). This ensures that, while the theoretical worst-case latency exists, it is statistically contained, yielding

TABLE IV: LLaDA Generation Quality across different response lengths. Results taken from [3]

Canvas Size	GSM8K Accuracy	HumanEval Pass@1
1024	70.3%	35.4%
512	70.8%	32.9%
256	70.0%	32.9%

Input Prompt:
`"Lily runs 12 km/h for 4 hours, then 6 km/h. How far does she run in 8 hours?"`

$L = 24$ tokens	$L = 64$ tokens
LLaDA Output: *Lily can run 48 + 24 = 72 kilometers in 8 hours.*	**LLaDA Output:** *In the first 4 hours, Lily runs 12 x 4 = 48 kilometers. Then, she runs 6 x 4 = 24 kilometers. In total, Lily runs 48 + 24 = 72 kilometers.*

Fig. 5: Adaptability to response length (L) constraint. For the same prompt, LLaDA generates outputs of varying verbosity and token count while maintaining answer correctness

a stable latency profile. Furthermore, the overhead of the AdaRLP (< 0.04ms) is negligible compared to the generation time. Overall, a stable end-to-end execution time is often more critical than raw throughput for meeting strict Service Level Agreements in production environments.

F. Impact on Generation Quality

We rely on prior results on whether adaptive sizing affects generation quality. We reference findings from LLaDA [3], where the model was evaluated with different response lengths. As reported in Tab. IV, performance remains stable across response lengths. This supports our premise that as long as the canvas L^* is sufficient to contain the answer ($L^* \geq k$), the D-LLM performs optimally. The degradation comes only from significant truncation, which our system actively prevents via the Safety Margin and fallback mechanisms.

We observe that D-LLMs exhibit the ability to adapt verbosity based on the available response length. As illustrated in Fig. 5, when constrained to a short response length, the model tends to produce more concise yet correct answers, whereas a longer response length results in more detailed reasoning.

V. Conclusions and Future Work

In this work, we studied the computational costs associated with Diffusion LLMs (eg, LLaDA) that operate with a fixed response length during inference. Due to its implicit design, when the true response length is shorter than the fixed response length, compute is wasted processing semantically meaningless `<PAD>` tokens. As a remedy, we introduced Predict-then-Diffuse, a framework designed for compute-budgeted inference with D-LLM. The core idea lies in predicting the response length conditioned on the input prompt using a lightweight and fast AdaRLP module. We also introduced a data-driven Safety Margin that circumvents output truncation, and therefore minimizing frequent fallbacks and ensuring predictable inference latency. Experiments on a collection of several datasets show that the proposed framework brings significant savings in computation when compared with default D-LLM inference mechanisms and other baselines based on heuristics. As D-LLMs with massive context window (e.g., 32k or 128k tokens) become more prevalent, the benefits of the Predict-then-Diffuse will become more significant.

A limitation of departing from fixed-size response length is that it prevents batching during inference. Although a straightforward solution to this problem involves grouping samples with similar estimated response lengths, more sophisticated and optimized solutions [7] are left for future work.

Acknowledgments

The authors acknowledge the ISCRA initiative of CINECA and project EVFTEG funded by MASE "Mission Innovation 2.0 - D.M. 386/2023".

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan *et al.*, "Language models are few-shot learners," *NeurIPS*, 2020.
[2] H. Touvron, T. Lavril, G. Izacard *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
[3] S. Nie, F. Zhu, Z. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and C. Li, "Large language diffusion models," *arXiv preprint arXiv:2502.09992*, 2025.
[4] J. Ye, Z. Xie, L. Zheng, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong, "Dream 7b: Diffusion large language models," *arXiv preprint arXiv:2508.15487*, 2025.
[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.
[6] M. Arriola, A. K. Gokaslan, N. Shazeer, and O. Firat, "Block diffusion: Interpolating between autoregressive and diffusion language models," in *ICLR*, 2025.
[7] W. Kwon, Z. Li, S. Zhuang *et al.*, "Efficient memory management for large language model serving with pagedattention," in *SOSP*, 2023.
[8] T. Li, M. Chen, B. Guo, and Z. Shen, "A survey on diffusion language models," *arXiv preprint arXiv:2508.10875*, 2025.
[9] S. Sahoo, M. Arriola, Y. Schiff *et al.*, "Simple and effective masked diffusion language models," *NeurIPS*, 2024.
[10] Q. Fan, A. Zou, and Y. Ma, "Timebill: Time-budgeted inference for large language models," *arXiv preprint arXiv:2512.21859*, 2025.
[11] J. Austin *et al.*, "How to scale your model," 2025, retrieved from https://jax-ml.github.io/scaling-book/, Google Deepmind.
[12] J. Kaplan *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.
[13] L. Prokhorenkova, G. Gusev *et al.*, "Catboost: unbiased boosting with categorical features," in *NeurIPS*, 2018.
[14] R. Y. Aminabadi *et al.*, "Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale," *arXiv preprint arXiv:2207.00032*, 2022.
[15] S. Xia *et al.*, "Ground every sentence: Improving retrieval-augmented LLMs with interleaved reference-claim generation," in *NAACL*, 2025.
[16] L. Chen *et al.*, "Alpagasus: Training a better alpaca with fewer data," in *ICLR*, 2024.
[17] S. Zhang *et al.*, "Instruction tuning for large language models: A survey," *arXiv preprint arXiv:2308.10792*, 2025.
[18] G. Wang *et al.*, "Openchat: Advancing open-source language models with mixed-quality data," in *ICLR*, 2024.
[19] A. Mitra *et al.*, "Orca 2: Teaching small language models how to reason," *arXiv preprint arXiv:2311.11045*, 2023.