AdaBlock-dLLM: Semantic-Aware Diffusion LLM Inference via Adaptive Block Size

Guanxi Lu[1], Hao (Mark) Chen[1], Yuto Karashima[2], Zhican Wang[1], Daichi Fujiki[2], Hongxiang Fan[1]
[1]Imperial College London, UK [2]Institute of Science Tokyo, Japan

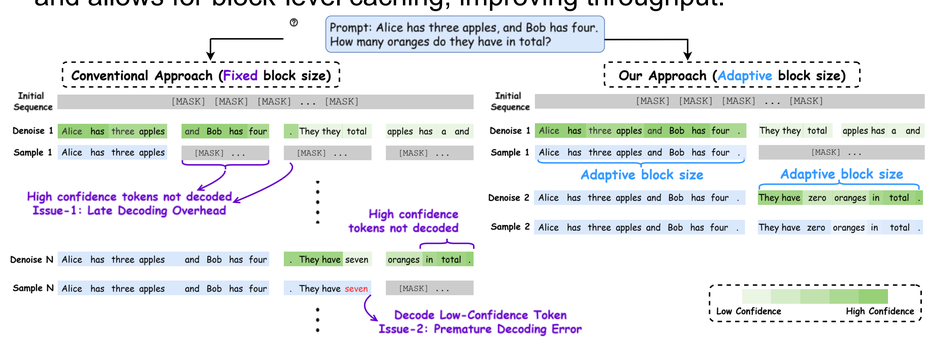
Background & Motivation

➢ Diffusion-Based Large Language Models (dLLMs)

- Iteratively denoise a sequence with <MASK> tokens.
- Emerge as a promising alternative to AR models with parallel decoding and better controllability.

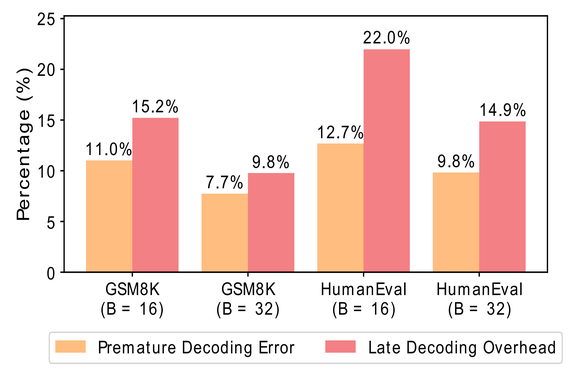
➢ Confidence-Based Semi-AR Decoding

- DLLM inference throughput is typically constrained by the lack of KV caching and the parallelism curse.
- Confidence-based semi-AR decoding balances parallelism and quality, and allows for block-level caching, improving throughput.



➢ Fundamental Issues of Fixed Block Size

- **Late Decoding Overhead:** Delays the unmasking of high-confidence tokens outside the current block (throughput ↓).
- **Premature Decoding Error:** Forces early commitment to low-confidence tokens within each block (quality ↓).



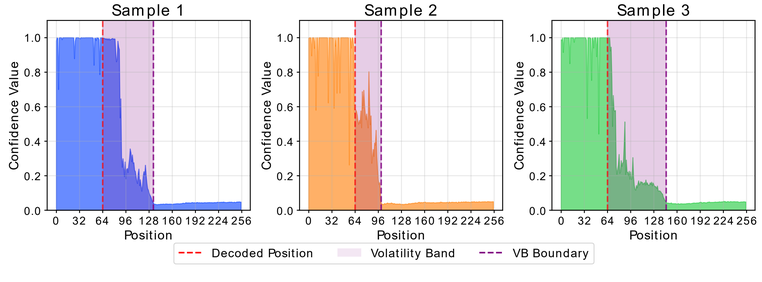
Method

➢ Confidence Dynamics

- A high-confidence region emerges near the decoded token.
- The decoding trace displays a global autoregressive sampling tendency driven by semantic dependencies.
- The high-confidence region extends toward adjacent positions; other positions maintain low confidence.

➢ Volatility Band and Local Stochasticity

- Partitioning the confidence landscape: high-confidence plateau, volatility band (VB), and low-confidence floor.
- Confidence pattern in VB is locally stochastic and dependent on the semantic context.



➢ Semantic-Aware Block Size Scheduling

- A scheduling mechanism that sets the block size guided by the length of the current semantic step
- Use confidence of a set of delimiter tokens to represent the semantic boundary and predict the block size.

Algorithm 1 Semantic-Aware Block Size Determination

Inputs: predicted sequence $\hat{\mathbf{y}}$; confidences $\mathbf{c}$; generation budget L; default block size B_0; delimiter set $\mathcal{D}$; delimiter threshold $\tau_{\mathcal{D}}$; current position g.
Output: block size B

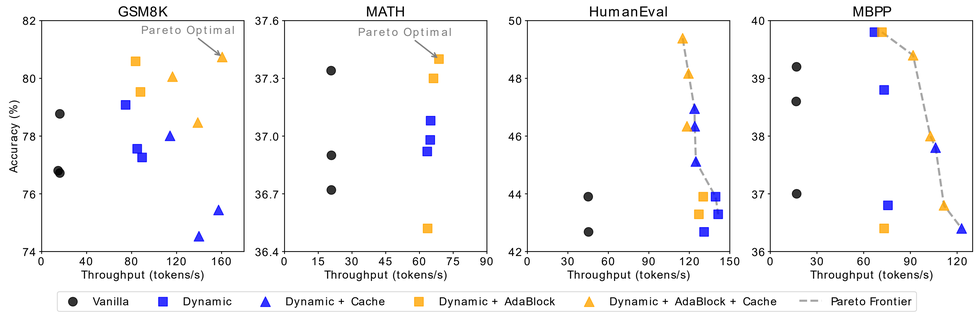
function ComputeBlockLength($\hat{\mathbf{y}}$, $\mathbf{c}$, L, B_0, $\mathcal{D}$, $\tau_{\mathcal{D}}$, g)
  ▷ *Sampling window boundary*
  $start, remaining \leftarrow g, L - g$
  $w \leftarrow \min\big(\max(1, \lfloor 0.25 \cdot g \rfloor), remaining\big)$
  $W \leftarrow \{ start, \ldots, start + w - 1 \}$ ▷ window token indices
  ▷ *Find highest-confidence delimiter*
  $\mathcal{I} \leftarrow \{ i \in W \mid \hat{y}_i \in \mathcal{D} \}$
  if $\mathcal{I} \neq \emptyset$ **then**
    $pos \leftarrow \arg\max_{i \in \mathcal{I}} c_i$ ▷ Select position with max delimiter token confidence
    $c_{\max} \leftarrow c_{pos}$
  else
    $c_{\max} \leftarrow -\infty$
  end if
  ▷ *Determine block size*
  if $c_{\max} \geq \tau_{\mathcal{D}}$ **then**
    $B \leftarrow (pos - start + 1)$ ▷ inclusive length up to the delimiter token
  else
    $B \leftarrow \min(B_0, remaining)$
  end if
  return B
end function

Experiments

➢ Main Results

Method	LLaDA-Instruct			LLaDA-1.5			Dream-Base		
	$B_0 = 16$	$B_0 = 32$	$B_0 = 64$	$B_0 = 16$	$B_0 = 32$	$B_0 = 64$	$B_0 = 16$	$B_0 = 32$	$B_0 = 64$
GSM8K									
Vanilla	78.8	76.7	76.8	82.3	82.3	80.4	76.3	76.4	75.1
Dynamic	79.1	77.6	77.3	82.6	82.2	80.7	75.5	75.5	75.6
+Ada	80.6 +1.5	80.6 +3.0	79.5 +2.2	83.0 +0.4	82.4 +0.2	80.3 -0.4	75.7 +0.2	75.7 +0.2	75.9 +0.3
+Cache	78.0	74.5	75.4	80.7	80.2	80.0	75.6	74.5	74.6
+Ada+Cache	80.0 +2.0	78.5 +4.0	80.7 +5.3	81.3 +0.6	81.7 +1.5	79.7 -0.3	76.5 +0.9	75.1 +0.6	74.6 +0.0
HumanEval									
Vanilla	43.9	43.9	42.7	39.0	36.6	38.4	53.7	52.4	54.3
Dynamic	42.7	43.9	43.3	36.6	37.8	36.6	53.0	51.2	52.4
+Ada	43.3 +0.6	43.3 -0.6	43.9 +0.6	37.8 +1.2	38.4 +0.6	38.4 +1.8	53.7 +0.7	51.2 +0.0	53.7 +1.3
+Cache	45.1	46.3	47.0	33.5	36.0	34.1	50.0	53.0	56.1
+Ada+Cache	49.4 +4.3	46.3 +0.0	48.2 +1.2	36.0 +2.5	39.0 +3.0	36.0 +1.9	52.4 +2.4	53.0 +0.0	57.3 +1.2
MATH									
Vanilla	36.7	36.9	37.3	36.3	37.0	34.4	39.8	40.2	40.1
Dynamic	37.0	36.9	37.1	36.3	36.7	34.4	39.7	39.9	39.9
+Ada	36.5 -0.5	37.3 +0.4	37.4 +0.3	36.8 +0.5	36.7 +0.0	34.1 -0.3	39.6 -0.1	39.9 +0.0	39.9 +0.0
+Cache	35.4	35.8	36.0	34.9	33.2	32.1	38.0	38.5	38.8
+Ada+Cache	35.8 +0.4	35.3 -0.5	35.6 -0.4	35.2 +0.3	33.9 +0.7	32.4 +0.3	37.8 -0.2	38.4 -0.1	38.4 -0.4
MBPP									
Vanilla	39.2	38.6	37.0	38.2	37.0	23.2	12.4	12.4	12.8
Dynamic	39.8	38.8	36.8	38.2	37.0	24.6	12.6	12.4	12.2
+Ada	40.2 +0.4	39.8 +1.0	36.4 -0.4	39.4 +1.2	37.6 +0.6	29.8 +5.2	12.8 +0.2	14.2 +1.8	12.4 +0.2
+Cache	35.6	37.8	36.4	38.0	34.8	19.8	12.8	11.6	9.6
+Ada+Cache	39.4 +3.8	38.0 +0.2	36.8 +0.4	36.6 -1.4	36.4 +1.6	36.6 +6.8	12.8 +0.0	11.6 +0.0	12.4 +2.8

➢ Accuracy (%) VS Throughput (TPS)



➢ Ablation Studies

Model	$\tau_D = 0.3$	$\tau_D = 0.5$	$\tau_D = 0.7$
LLaDA-Instruct	**80.59**	79.08	77.94
Dream-Base	75.66	**75.74**	**75.74**

Ablation on Delimiter Threshold

Method	$B_0 = 16$	$B_0 = 32$	$B_0 = 64$
Vanilla	69.1	66.7	61.3
Dynamic	69.0	66.7	61.2
+Ada	68.4 -0.6	67.5 +0.8	64.4 +3.2
+Cache	67.5	64.6	59.4
+Ada+Cache	68.9 +1.4	66.4 +1.8	62.7 +3.3

Accuracy on IFEval

Delimiter Set	Acc. (%)
None (+Cache)	74.5
{[\n]}	**78.5**
{[,]}	75.1
{[.]}	74.5
{[,], [.]}	75.1
{[\n], [,]}	**78.5**
{[\n], [.]}	**78.3**
{[\n], [,], [.]}	**78.7**

Ablation on Delimiter Set