# MARS: Enabling Autoregressive Models Multi-Token Generation

**Ziqi Jin**[1] **Lei Wang**[2] **Ziwei Luo**[3] **Aixin Sun**[1]

[1]Nanyang Technological University [2]Singapore Management University [3]Uppsala University

## Abstract

Autoregressive (AR) language models generate text one token at a time, even when consecutive tokens are highly predictable given earlier context. We introduce **MARS** (**M**ask **A**uto**R**egre**S**sion), a lightweight fine-tuning method that teaches an instruction-tuned AR model to predict multiple tokens per forward pass. MARS adds no architectural modifications and no extra parameters, and produces a single model that can still be called exactly like the original AR model; unlike speculative decoding or multi-head approaches such as Medusa, it requires only continued training on existing instruction data. At the recommended block size, one-token MARS matches or exceeds the AR baseline on the six-benchmark average at both scales we test and on every individual benchmark but one; multi-token mode accepts 1.5–1.9 tokens per forward pass at a small accuracy cost. The ingredient that makes this work is an auxiliary AR loss on the clean input stream: without it, large-block training collapses reasoning (GSM8K at 7B falls from 75.1 to 31.5), and the AR anchor, not the particular two-stream schedule, is what matters. In controlled comparisons on the same 7B checkpoint, MARS delivers more end-to-end speedup than EAGLE-style draft heads and is 1.54× faster than Fast-dLLM parallel decoding applied to a Block Diffusion model trained with the same budget. A deferred-KV batch sampler yields wall-clock speedups over AR with KV cache of 2.16× at batch size 4, 1.70× at batch 16 and 1.17× at batch 32 on Qwen2.5-7B; at batch 64 decoding is compute-bound and the sampler falls back to AR. The confidence threshold and the number of proposed tokens per step are run-time knobs, so one checkpoint serves quality- and latency-sensitive requests without a model swap. [1]

## 1 Introduction

Autoregressive (AR) language models spend the same compute on every token: one forward pass produces exactly one token, whether that token is the inevitable "`the answer is`" following a multiple-choice prompt or a genuinely uncertain next word in an open-ended generation. This uniform cost is wasteful.

Most existing approaches to multi-token generation modify the deployment stack: speculative decoding [4, 22] maintains a separate draft model alongside the target, adding memory footprint and orchestration complexity, and multi-head approaches such as Medusa [3] and EAGLE [24] attach prediction heads that require extra parameters and head-specific training. Both families introduce components beyond the original model, complicating serving pipelines.

In this work we take a different angle: rather than building auxiliary components around an AR model, we ask whether lightweight fine-tuning alone can give the model *an additional capability*

[1]Our code for training and evaluation is available at: `https://github.com/Xalp/MARS`. Our MARS checkpoints are available at: `https://huggingface.co/collections/Xalphinions/mars`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

**Question:** Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?

**MARS-7B (confidence threshold $\tau$=0.95), 66 forwards, 2.55 tokens/forward:**

Let 's break it down step by step: 1 . Seattle has 20 sheep. 2. Charleston has 4 times as many sheep as Seattle , so Charleston has 4 x 20 = 80 sheep. 3. Toulouse has twice as many sheep as Charleston, so T oulouse has 2 x 80 = 16 0 sheep. Now , let 's add up … Total number of sheep : 20 + 80 + 16 0 = 1 00 + 160 = 260 sheep

Shaded regions = multiple tokens accepted in one forward pass. Unshaded tokens are generated one at a time. Darker = more tokens per step. Formulaic phrases and calculations are chunked; novel reasoning proceeds token-by-token.

Figure 1: Example MARS generation on GSM8K. The model adaptively generates 1–4 tokens per forward pass based on confidence: predictable continuations are batched together, while novel content proceeds token-by-token. This achieves 2.55× token per forward over standard AR decoding.

*to generate multiple tokens per forward pass* while preserving its behavior as a standard AR model. The resulting model should be a superset of the original: served exactly like the original, it produces one token at a time without giving up quality, but it can also accept multiple tokens per step when confident.

A natural starting point is block masked diffusion, which trains the model to predict multiple tokens per step within a fixed-size block. However, existing AR-to-block-diffusion conversions degrade quality substantially, particularly on reasoning and coding tasks [1, 44, 16, 11]. In Section 3.1, we identify four gaps between AR and block diffusion that may account for this, and find that only one is inherent to multi-token prediction: the use of mask tokens as placeholders for unknown future tokens. The other three are design choices that unnecessarily depart from the original AR model, and can be fully closed.

From these motivations, we introduce **MARS** (**M**ask **A**uto**R**egre**S**sion). MARS maintains causal intra-block attention, uses right-shifted logits, and generates strictly left-to-right, leaving only the inherent masking gap. As shown in Figure 1, the resulting model adaptively generates multiple tokens per forward pass when confident and falls back to single-token generation for novel content. At the recommended block size $B$=4, one-token MARS matches or exceeds the AR baseline on the six-benchmark average at both 0.5B and 7B and on every individual benchmark except GPQA at 7B ($-1.6$); at larger block sizes the average stays at or above baseline while individual benchmarks can dip, and we state the claim per configuration. Multi-token mode accepts 1.5–1.9 tokens per forward pass for under one point on the average at $B$=4.

**Contributions.**

- We analyze the four gaps between AR and block-masked prediction and show that three are eliminable design choices rather than inherent limitations; closing them recovers baseline quality without architectural changes.
- We propose MARS, a lightweight fine-tuning method that lets an instruction-tuned AR model optionally predict multiple tokens per forward pass with no architectural changes, no additional parameters, and the same SFT data; at the recommended block size it can be used exactly like the original AR model or switched to multi-token mode.
- We show that an auxiliary AR loss on the clean stream is necessary at larger block sizes (without it, $B$=16 training collapses GSM8K at 7B from 75.1 to 31.5), and that the anchor, not the two-stream schedule, is what matters: mixing AR and masked sequences across or within batches works as well.
- We compare MARS with EAGLE-style draft heads and with Fast-dLLM on a Block Diffusion model from the same checkpoint lineage under controlled protocols, and develop a deferred-KV batch sampler reaching 2.16× wall-clock speedup over AR with KV cache at batch 4 on Qwen2.5-7B, 1.70× at batch 16 and 1.17× at batch 32, reporting where the advantage ends.

## 2 Background and Related Work

**Discrete diffusion and masked language models.** Diffusion models for discrete data [2, 23] generate tokens through iterative denoising; early non-autoregressive work in machine translation [17, 14] demonstrated significant speedups at the cost of quality. More recently, masked diffusion language models [33, 35, 27, 13, 42, 28] apply iterative unmasking to text generation and have begun to close the quality gap with AR models, typically with bidirectional attention, a design choice inherited from diffusion models in vision. Several works have converted pretrained AR models into diffusion language models [1, 44, 16, 11], showing the transition is feasible without training from scratch, though often with significant quality degradation that we attribute to the mismatch between the AR model's causal attention and the bidirectional attention adopted during conversion.

**Causality and multi-token generation.** A growing body of work shows that causal attention does not conflict with multi-token parallel generation. Block Diffusion [1] uses causal attention across blocks while keeping bidirectional attention within each block. Building on this, several works further accelerate diffusion LM inference [7, 6], including Fast-dLLM [40] which introduces KV caching for bidirectional diffusion models, and self-speculative decoding [12] which uses the diffusion model itself as both drafter and verifier. CARD [32, 18] find causality actually *improves* over bidirectional alternatives; training with a progressive block size [41] or inference with a flexible block size [38] also helps. ARMD [20] and A3 [9] adopt causal structure for masked diffusion and any-order generation, respectively, and DLMs without causality may face a marginal trap [19]. Eso-LM [34] further unifies AR and masked diffusion under causal attention with KV caching. Together, these results establish that bidirectional attention in diffusion language models was a design choice inherited from vision, not a requirement for parallel token generation.

**Multi-token prediction and parallel decoding.** Multi-token prediction (MTP) [15] trains AR models with $k-1$ auxiliary heads to predict future tokens in parallel; DeepSeek-V3 [25] deploys this at 671B scale. Medusa [3] and EAGLE [24**?** ] add lightweight heads for speculative multi-token proposals. These methods require additional parameters and architectural modifications, and their proposals are lossless. MARS uses the same language model head for all positions within a block; its multi-token mode is lossy, so Section 4.5 compares against lossless methods with quality reported alongside speed. Speculative decoding [22, 4] leaves the target model unmodified but requires a separate draft model. Jacobi decoding [37] and Lookahead decoding [10] achieve parallel generation from a single unmodified model by treating AR generation as a fixed-point iteration, and CLLMs [21] train the model to converge faster under Jacobi iteration.

## 3 Method

The design of MARS is guided by one principle: **the model must remain a fully functional AR model**. Multi-token prediction is an added capability, not a replacement. We begin by analyzing where prior block-masked approaches break this property, then show how MARS closes every eliminable gap at training and inference.

### 3.1 Where Does Block-Masked Prediction Fail?

Block-masked prediction fails when it departs from the autoregressive model along multiple axes. We identify four gaps between AR and block diffusion (Table 1). Of these, one is inherent to multi-token prediction and cannot be avoided; the other three are eliminable design choices that unnecessarily break AR compatibility.

Gap (1), *token masking*, is inherent: to predict multiple tokens in parallel, future positions must be replaced with `[MASK]` placeholders. This is the fundamental cost of multi-token prediction. The remaining three gaps are eliminable design choices in prior systems. Gap (2), *attention pattern*: some block diffusion approaches [1] use bidirectional attention within blocks, but a bidirectional model is no longer a functional AR model; MARS keeps strictly causal attention everywhere. Gap (3), *logits alignment*: AR models predict $x_{t+1}$ from position $t$ (right-shifted logits); changing this convention breaks the output head's AR function, so MARS preserves it. Gap (4), *generation order*: confidence-based diffusion methods unmask tokens out of order within a block; MARS always

Table 1: Gaps between AR and block diffusion. MARS aligns with AR on gaps (2)–(4), leaving only the inherent masking gap.

| | AR | MARS | Block Diffusion |
|---|---|---|---|
| (1) Token Masking | No masking | Masked within block | Masked within block |
| (2) Attention Pattern | Causal | Causal | Bidirectional (intra-block) |
| (3) Logits Alignment | Right-shifted | Right-shifted | Varies |
| (4) Generation Order | Left-to-right | Left-to-right | Confidence-based |

accepts tokens strictly left-to-right. By closing gaps (2)–(4), the model remains a fully functional AR model whose only difference is that it sees `[MASK]` placeholders within the current block.

### 3.2 Training

MARS starts from an AR SFT checkpoint trained on the target instruction-following data with standard next-token prediction, so that MARS training focuses solely on learning the masked prediction paradigm (Section 4.1).

The high-level idea is simple: we run two copies of the sequence through the model in parallel. The *clean stream* keeps the original tokens and trains the model with ordinary AR next-token prediction; the *noisy stream* replaces each block of $B$ tokens with `[MASK]` placeholders and asks the model to predict them from the clean prefix of earlier blocks. Both streams share one forward pass through a structured attention mask, which closes gaps (2) and (3) from Section 3.1 by construction: attention is strictly causal everywhere, and logits are right-shifted to match the AR convention.

Let $\mathbf{c} = (c_1, \ldots, c_P)$ denote the instruction prompt and $\mathbf{x} = (x_1, \ldots, x_L)$ the response sequence. We divide $\mathbf{x}$ into blocks of size $B$ and replace all response tokens in each block with `[MASK]`:

$$\tilde{\mathbf{x}} = (\underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{B}, \underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{B}, \ldots) \tag{1}$$

The full training input is $[\mathbf{c}; \mathbf{x}; \tilde{\mathbf{x}}]$. Both the clean stream $\mathbf{x}$ and the noisy stream $\tilde{\mathbf{x}}$ attend causally to $\mathbf{c}$, which serves as fully-clean prefix context for both; losses are computed only over response positions. For notational compactness we omit $\mathbf{c}$ in the equations below and write $\mathbf{z} = [\mathbf{x}; \tilde{\mathbf{x}}]$ of length $2L$, where the first $L$ positions are the clean stream and the last $L$ are the noisy stream. Let $\beta(t) = \lceil t/B \rceil$ denote the block index of position $t$ within either stream, and let $\mathcal{M}$ denote the set of masked (noisy-stream) positions.

**Objective.** MARS minimizes the sum of two cross-entropy losses computed in the same forward pass, one per stream:

$$\mathcal{L}_{\text{mask}} = -\sum_{t \in \mathcal{M}} \log p_\theta\big(x_t \mid \tilde{\mathbf{x}}, \mathbf{x}_{<\beta(t)}\big)\,, \qquad \mathcal{L}_{\text{AR}} = -\sum_{t=1}^{L} \log p_\theta(x_t \mid \mathbf{x}_{<t})\,, \qquad \mathcal{L} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{AR}}. \tag{2}$$

Here $\mathbf{x}_{<\beta(t)}$ denotes the clean tokens from all blocks preceding block $\beta(t)$; under the right-shifted convention of gap (3), $p_\theta(x_t \mid \cdot)$ in $\mathcal{L}_{\text{mask}}$ is read from the logit at noisy position $t-1$. $\mathcal{L}_{\text{AR}}$ is standard next-token prediction on the clean-stream logits, i.e. the ordinary SFT objective, so we call it the *SFT loss*; there is no weighting hyperparameter. MARS is always trained with $\mathcal{L}$; "MARS w/o SFT loss" in the tables denotes training with $\mathcal{L}_{\text{mask}}$ alone.

**Attention mask.** We define $\mathbf{M} \in \{0, -\infty\}^{2L \times 2L}$ over $\mathbf{z}$. For query position $i$ and key position $j$:

$$M_{ij} = \begin{cases} 0 & \text{if } i, j \le L,\ j \le i \quad \text{(clean causal)} \\ 0 & \text{if } i, j > L,\ \beta(i{-}L) = \beta(j{-}L),\ j \le i \quad \text{(noisy intra-block causal)} \\ 0 & \text{if } i > L,\ j \le L,\ \beta(j) < \beta(i{-}L) \quad \text{(noisy} \to \text{clean cross-stream)} \\ -\infty & \text{otherwise} \end{cases} \tag{3}$$

The *clean causal* case gives the clean stream (positions $1, \ldots, L$) standard causal self-attention, identical to AR training. The *noisy intra-block causal* case allows each noisy position to attend

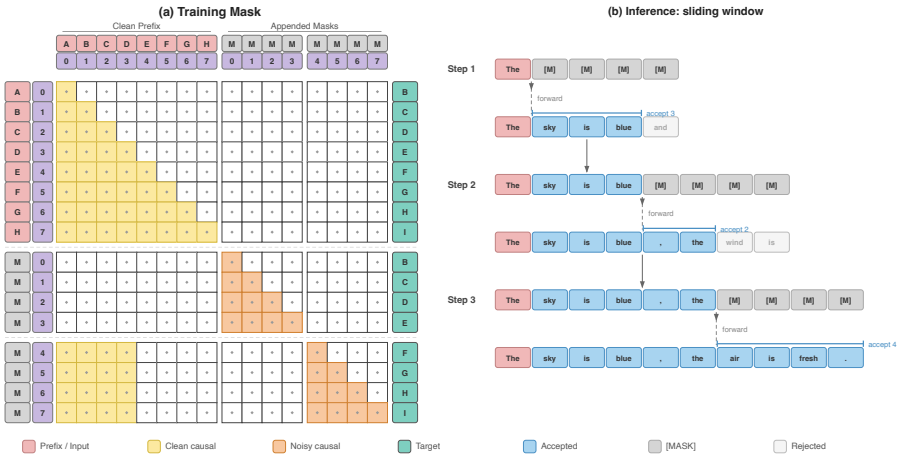


Figure 2: MARS attention mask and inference for $L$=8, $B$=4. **Left:** training mask with $[\mathbf{x}; \tilde{\mathbf{x}}]$ concatenation. The orange cells show that noisy positions attend to each other *causally* within each block, in contrast to Block Diffusion [1] which uses bidirectional attention within blocks. **Right:** sliding-window inference. The dashed line marks the generation cursor; $B$ `[MASK]` tokens are appended and filled via one forward pass. Accepted tokens (blue) slide into the prefix for the next step.

causally within its own block (only `[MASK]` tokens, so only positional information flows). The *cross-stream* case allows each noisy block $k$ to see clean tokens from blocks $1, \ldots, k-1$, providing the prefix context needed for prediction.

### 3.3 Why the Clean Stream Carries a Loss: An Intuition

Closing gaps (2)–(4) ensures that MARS *behaves* like an AR model at inference; training must also keep it one in capability. This section gives the intuition behind $\mathcal{L}_{\text{AR}}$; it is motivation, not a quantitative mechanism, and the evidence for the design is the ablation in Section 4.3.

**The clean stream is not an added component.** Filling a masked block requires the real tokens of the preceding blocks as keys and values in the same forward pass: with $B$=2, predicting $\{x_3, x_4\}$ from placeholders needs $x_1, x_2$ present, or the target $p(x_3, x_4 \mid x_1, x_2)$ is undefined. Training all blocks in one forward therefore yields $[\mathbf{x}; \tilde{\mathbf{x}}]$ with the clean copy first, as in Block Diffusion. Those $L$ clean positions are computed regardless, and since attention among them is causal, their logits are exactly AR next-token logits; the only decision MARS makes is to attach a loss to them, at no extra transformer compute.

**Why it matters more as $B$ grows.** Within a block of size $B$, the $t$-th target is predicted with $t-1$ `[MASK]` placeholders in place of ground truth; only the first sees a fully clean context, the closest analog to AR next-token prediction. Counting, as a coarse proxy for AR-like supervision, the fraction of targets predicted with no `[MASK]` in context, mask-only training gives $r_{\text{AR}}^{(\text{mask only})} = 1/B$, which shrinks with the block; adding $\mathcal{L}_{\text{AR}}$ contributes $L$ clean-context terms and the fraction becomes $r_{\text{AR}}^{(\text{combined})} = (L + L/B)/(2L) = 1/2 + 1/2B$, which no longer vanishes with $B$. We attach no significance to the functional form or to the value $1/2$ (a $1/\sqrt{B}$ count would tell the same story, and late-block positions still see mostly clean context); the one property we rely on is qualitative: without an AR-anchored term the share of AR-like supervision decays with $B$, and with the anchor it is bounded below. Section 4.3 tests this, and shows that other ways of supplying the anchor work as well, so the contribution is the anchor, not the schedule.

### 3.4 Inference: Left-to-Right Sliding Window

The inference procedure closes gap (4): tokens are always accepted strictly left-to-right, matching AR generation order, so MARS at inference is indistinguishable from AR generation when only

one token is accepted per step and extends smoothly to multi-token generation when the model is confident.

At each step, $B$ `[MASK]` tokens are appended after the current prefix and the model runs a single forward pass with pure causal attention. Under the right-shifted convention, the $B$ candidate distributions for new tokens $x_1, \ldots, x_B$ come from $B$ logits: the logit at the *last prefix position* produces $p(x_1)$, and the logits at noisy positions $1, \ldots, B{-}1$ produce $p(x_2), \ldots, p(x_B)$. The logit at the rightmost masked position would produce $p(x_{B+1})$ but is discarded; the next window will overwrite it with fresh prefix context. Starting from $x_1$, tokens are accepted consecutively left-to-right while $\max_v p(x_t{=}v) \geq \tau$, with at least one token always accepted. The $N$ accepted tokens join the prefix, and $N$ new `[MASK]` tokens are appended to keep the window at size $B$. This repeats until generation is complete; since at least one token is always accepted, the model degrades gracefully to standard AR decoding when no prediction is confident enough.

The threshold $\tau$ directly controls throughput: $\tau \to 1.0$ accepts at most one token per step (recovering exact AR behavior), while lower $\tau$ accepts more tokens per step at some quality cost. Crucially, $\tau$ can be adjusted *on-the-fly per request* during serving, without retraining or loading a different model. Section 4.4 characterizes the full tradeoff curve; Section 5 adds a second run-time knob, the number of proposals per step.

# 4 Experiments

We organize the experiments around five claims: MARS preserves AR quality at the recommended block size (Section 4.2); the SFT loss is necessary at larger block sizes, but the two-stream schedule is not (Section 4.3); confidence thresholding gives a smooth speed–quality frontier (Section 4.4); MARS compares favorably with stronger accelerators under controlled protocols (Section 4.5); and the gains translate to wall-clock speedup in batch inference up to a batch size we identify (Section 5).

## 4.1 Setup

We evaluate at two scales: Qwen2.5-0.5B-Instruct and Qwen2.5-7B-Instruct [30], both trained on Dolci-Instruct-SFT [29] ($\sim$2M examples). We first train an AR SFT model for 5 epochs with standard next-token prediction, then continue with MARS training on the same data for another 5 epochs. At 0.5B and 7B, we train with block sizes $B \in \{4, 8, 16\}$.

We compare against the AR SFT starting point and Block Diffusion [1] trained from the same checkpoint with the same recipe, at both scales. Evaluation uses six benchmarks: IFEval (0-shot) [43], BBH (3-shot) [36], MMLU-Pro (0-shot) [39], GPQA (0-shot) [31], GSM8K (0-shot) [8], and HumanEval (0-shot) [5], run with lm-evaluation-harness [**?** ] under greedy decoding. Full hyperparameters for training and evaluation are in Appendix B.

**Statistical protocol.** Table entries are single evaluation runs. Appendix C reports training variance from three independent MARS-0.5B ($B{=}4$) trainings, and evaluation variance from a paired percentile bootstrap over evaluation examples (10,000 resamples on the per-sample logs of the tabulated runs), as 95% confidence intervals on the headline deltas.

**IFEval scoring, and a correction.** IFEval reports four accuracies (prompt- and instruction-level, strict and loose); we report the mean of the two strict ones. The submitted version's collection script selected them by position, and since the harness's output order is not stable across runs, some cells averaged the wrong pair and were about five points too low. Every IFEval number here has been re-scored by name; no other benchmark reports more than one accuracy per task, and we verified that none was affected.

## 4.2 MARS Preserves AR Quality in One-Token Mode

At the recommended block size $B{=}4$, MARS should match or exceed the original model in one-token mode. Table 2 confirms this at both scales on the six-benchmark average (0.5B: 28.7 $\to$ 30.2; 7B: 56.6 $\to$ 58.4) and on every individual benchmark but one. At 7B the GSM8K gain ($+4.5$, 95% CI $[+1.7, +7.3]$) and the IFEval gain ($+3.9$, $[+0.7, +7.0]$) are significant under the paired bootstrap; the HumanEval gain ($+2.4$, $[-3.1, +7.9]$) is not. Three independent 0.5B trainings average 30.0, 30.4

Table 2: One-token mode ($\tau$=1.0): MARS vs. AR SFT, epoch-matched AR SFT (10 epochs; matched in epochs, not FLOPs), and Block Diffusion trained from the same checkpoint with the same recipe. **Bold**: best per column within each scale (gray base rows excluded).

| Model | IFEval | BBH | MMLU-Pro | GPQA | GSM8K | HumanEval | Avg |
|---|---|---|---|---|---|---|---|
| *Qwen2.5-0.5B-Instruct* | | | | | | | |
| *Base (no fine-tune)* | 27.8 | 6.1 | 2.4 | 13.6 | 27.4 | 26.2 | 17.2 |
| AR SFT (5 ep) | 48.4 | 26.3 | 11.9 | 17.9 | 32.0 | 35.4 | 28.7 |
| AR SFT (10 ep) | 47.8 | 26.3 | 9.3 | 14.1 | 28.3 | 32.3 | 26.4 |
| Block Diffusion [1] ($B$=4) | 47.1 | 7.5 | 2.0 | 17.9 | 30.6 | 31.7 | 22.8 |
| MARS-0.5B w/o SFT loss ($B$=4) | 48.5 | **27.4** | 12.3 | 19.0 | 29.5 | 33.5 | 28.4 |
| MARS-0.5B ($B$=4) | **49.9** | 26.6 | **12.4** | **19.4** | **32.8** | **40.2** | **30.2** |
| *Qwen2.5-7B-Instruct* | | | | | | | |
| *Base (no fine-tune)* | 63.4 | 24.0 | 10.5 | 11.6 | 57.3 | 82.9 | 41.6 |
| AR SFT (5 ep) | 65.7 | 53.8 | 43.5 | **29.7** | 68.4 | 78.7 | 56.6 |
| AR SFT (10 ep) | 66.1 | 48.6 | 40.2 | 26.3 | 61.9 | 73.2 | 52.7 |
| Block Diffusion [1] ($B$=4) | 66.9 | 54.2 | 43.0 | 27.9 | 68.2 | 64.0 | 54.1 |
| MARS-7B ($B$=4) | **69.6** | **54.4** | **44.4** | 28.1 | **72.9** | **81.1** | **58.4** |

Table 3: Effect of the SFT loss across block sizes ($\tau$=1.0). Without it, quality erodes monotonically in $B$ at 0.5B and GSM8K collapses at 7B; with it, the average is stable in $B$, although individual benchmarks dip below AR SFT at $B$=8 and 16 (e.g. 7B HumanEval and GPQA at $B$=16), so the per-benchmark superset claim is a $B$=4 claim. $B$=4 rows match Table 2.

| Model | IFEval | BBH | MMLU-Pro | GPQA | GSM8K | HumanEval | Avg |
|---|---|---|---|---|---|---|---|
| **Qwen2.5-0.5B-Instruct** | | | | | | | |
| *AR SFT* | 48.4 | 26.3 | 11.9 | 17.9 | 32.0 | 35.4 | 28.7 |
| *MARS-0.5B w/o SFT loss* | | | | | | | |
| $B$=4 | 48.5 | 27.4 | 12.3 | 19.0 | 29.5 | 33.5 | 28.4 |
| $B$=8 | 48.5 | 24.3 | 11.1 | 20.8 | 24.9 | 28.7 | 26.4 |
| $B$=16 | 42.6 | 21.7 | 10.6 | 16.3 | 21.0 | 20.7 | 22.2 |
| *MARS-0.5B with SFT loss* | | | | | | | |
| $B$=4 | 49.9 | 26.6 | 12.4 | 19.4 | 32.8 | 40.2 | 30.2 |
| $B$=8 | 51.1 | 26.9 | 12.0 | 19.6 | 32.8 | 37.2 | 29.9 |
| $B$=16 | 51.5 | 27.0 | 12.1 | 17.9 | 33.8 | 36.6 | 29.8 |
| **Qwen2.5-7B-Instruct** | | | | | | | |
| *AR SFT* | 65.7 | 53.8 | 43.5 | 29.7 | 68.4 | 78.7 | 56.6 |
| *MARS-7B with SFT loss* | | | | | | | |
| $B$=4 | 69.6 | 54.4 | 44.4 | 28.1 | 72.9 | 81.1 | 58.4 |
| $B$=8 | 63.2 | 55.8 | 45.4 | 27.0 | 71.0 | 79.9 | 57.1 |
| $B$=16 | 64.7 | 55.2 | 44.2 | 25.7 | 75.1 | 76.8 | 57.0 |

and 29.6, each above baseline by more than the spread between runs (Appendix C). The exception is GPQA at 7B, 1.6 points below baseline (28.1 vs. 29.7); we suspect the 512-token training length, since GPQA has the longest chains of thought in our suite and its traces are the most often truncated.

The epoch-matched AR SFT control (10 epochs, the same total as MARS's 5 AR + 5 masked) overfits rather than improves at both scales (0.5B: 28.7 $\rightarrow$ 26.4; 7B: 56.6 $\rightarrow$ 52.7): the same extra epochs help when spent on the masked objective and hurt when spent on more AR SFT. The control is matched in epochs, not FLOPs (Appendix B).

Block Diffusion [1], with bidirectional intra-block attention, collapses on every benchmark at 0.5B and does not recover at 7B, landing 2.5 points below AR SFT and 4.3 below MARS. The shape of the 7B damage is telling: four of the six benchmarks are essentially unchanged, while HumanEval falls 14.7 points (78.7 $\rightarrow$ 64.0) and GPQA 1.8. Bidirectional intra-block attention costs most where left-to-right structure carries the most weight, and code is the extreme case in this suite, consistent with Section 3.1.

Table 4: Multi-token mode. Each cell shows accuracy at $\tau$=0.95 with the absolute change from $\tau$=1.0 (Table 3) in parentheses. Gray italic rows show tokens accepted per forward pass for each task. Full threshold sweep in Appendix Table 10.

| Model | IFEval | BBH | MMLU-Pro | GPQA | GSM8K | HumanEval | Avg |
|---|---|---|---|---|---|---|---|
| *Qwen2.5-0.5B-Instruct* | | | | | | | |
| MARS $B$=4 | 50.1 (+0.2) | 26.6 (0.0) | 11.5 (−0.9) | 19.9 (+0.5) | 31.1 (−1.7) | 39.6 (−0.6) | 29.8 (−0.4) |
| *tok/fwd* | *1.20* | *1.90* | *1.50* | *1.36* | *1.51* | *1.26* | *1.46* |
| MARS $B$=8 | 50.6 (−0.5) | 28.3 (+1.4) | 11.9 (−0.1) | 19.0 (−0.6) | 31.7 (−1.1) | 36.6 (−0.6) | 29.7 (−0.2) |
| *tok/fwd* | *1.23* | *1.96* | *1.49* | *1.46* | *1.54* | *1.27* | *1.49* |
| MARS $B$=16 | 50.9 (−0.6) | 25.9 (−1.1) | 11.9 (−0.2) | 17.9 (0.0) | 31.2 (−2.6) | 36.6 (0.0) | 29.1 (−0.7) |
| *tok/fwd* | *1.21* | *1.86* | *1.51* | *1.43* | *1.57* | *1.25* | *1.47* |
| *Qwen2.5-7B-Instruct* | | | | | | | |
| MARS $B$=4 | 69.0 (−0.6) | 54.3 (−0.1) | 44.2 (−0.2) | 27.7 (−0.4) | 71.0 (−1.9) | 80.5 (−0.6) | 57.8 (−0.6) |
| *tok/fwd* | *1.15* | *2.60* | *1.47* | *1.47* | *1.79* | *1.60* | *1.68* |
| MARS $B$=8 | 62.5 (−0.7) | 55.0 (−0.8) | 45.1 (−0.3) | 26.6 (−0.4) | 69.2 (−1.8) | 78.0 (−1.9) | 56.1 (−1.0) |
| *tok/fwd* | *1.16* | *3.22* | *1.54* | *1.49* | *1.93* | *1.60* | *1.82* |
| MARS $B$=16 | 61.7 (−3.0) | 53.8 (−1.4) | 43.9 (−0.3) | 25.9 (+0.2) | 72.6 (−2.5) | 72.6 (−4.2) | 55.1 (−1.9) |
| *tok/fwd* | *1.16* | *3.29* | *1.63* | *1.54* | *2.02* | *1.60* | *1.87* |

### 4.3 The SFT Loss Is Necessary; the Two-Stream Schedule Is Not

**Block size.** Table 3 is the empirical finding behind Section 3.3. Without the SFT loss, 0.5B quality degrades monotonically in $B$ (average 28.4 → 26.4 → 22.2 for $B$=4/8/16), concentrated in reasoning and coding; with it the sweep is flat (30.2 → 29.9 → 29.8), and the $B$=16 GSM8K gap between the variants is far outside evaluation noise (95% CI $[+9.5, +14.7]$). At 7B the effect is several times larger: training $B$=16 without the SFT loss collapses GSM8K from 75.1 to 31.5 and the five-task average by 8.5 points, while IFEval, BBH and GPQA move within noise. With the SFT loss the 7B average stays at or above AR SFT (56.6) at every block size (58.4 / 57.1 / 57.0), but individual benchmarks dip: at $B$=16, HumanEval is 76.8 against 78.7 and GPQA 25.7 against 29.7. The per-benchmark superset property is therefore a $B$=4 statement, and $B$=4 is our recommended default; larger blocks buy more tokens per forward (Section 4.4) at the price of such dips. Figure 3 (Appendix E) shows the same pattern across the threshold sweep: at every tokens-per-forward rate, MARS with SFT loss is more accurate than without, and the gap widens with $B$.

**Schedule.** The ablation shows that an AR-preserving term is needed, not that the two-stream design is the one. Table 9 (Appendix D) compares simpler ways of supplying the anchor at 0.5B with the same data, epochs and learning rate. *Separate forwards* are not a different scheme: the clean stream in the joint forward has the same visibility and position ids as a standalone causal forward, and in float64 the clean-stream logits agree bit-for-bit while the gradients of $\mathcal{L}$ agree to under $7\times10^{-7}$; it is MARS at twice the forward cost. *Mixing* AR and masked instances, as alternating micro-batches or within every batch, performs on par with or slightly above MARS (six-benchmark averages of 31.0 / 30.3 and 29.9 / 30.3 at $B$=4 / 16 against 30.0 / 30.0 for MARS; alternating at $B$=4 is above all three MARS seeds), and neither collapses at $B$=16, whereas removing the AR signal costs 8.2 points there (21.8). Nothing here shows our schedule is the best one; it shows that an AR-anchored schedule is necessary and that the single-forward version is a convenient instance, since the clean positions are computed anyway. The contribution is the insight, not the architecture.

### 4.4 A Smooth Speed–Quality Frontier via Confidence Thresholding

Table 4 reports accuracy at $\tau$=0.95 with the change from one-token mode; the full sweep is in Appendix E. The cost of multi-token generation is small and predictable: at 0.5B the average drops 0.2–0.7 points at 1.46–1.49 tokens per forward; at 7B, $B$=4 costs 0.6 points at 1.68 tok/fwd, and $B$=8 and 16 reach 1.82–1.87 tok/fwd (BBH peaks at 3.29) for one to two points. The largest drops are on GSM8K and HumanEval (up to −2.6 and −4.2), where a prematurely committed token cannot be undone. IFEval, which the submitted version singled out for ∼5-point drops, moves by 0.2–0.6 points: those drops were the scoring artifact of Section 4.1, not a property of multi-token acceptance (tokens per forward come from the sampler and were unaffected). Appendix G sweeps the acceptance gate on IFEval: at matched speed the top-1 probability gate degrades first and a top-2 margin gate holds up best, consistent with high-confidence format tokens being the ones a probability

Table 5: Controlled comparisons on Qwen2.5-7B (GSM8K, batch size 1, 256 new tokens); within each panel only the acceleration method varies. **(a)** EAGLE-style heads [24**?** ] trained on our frozen AR SFT target with the same SFT data ($d$: head width; 3584 is EAGLE's own 7B width); lossless, so accuracy is the target's; speedup is wall-clock over AR SFT. **(b)** Fast-dLLM [40] on the Block Diffusion model of Table 2 (same checkpoint, data, epochs and block size as MARS); one H100; speedup over its fixed schedule.

(a) EAGLE-style heads vs. MARS (128 GSM8K prompts). Params: added parameters.

| **Method** | **Params** | **Tok/target fwd** | **Speedup** |
|---|---|---|---|
| AR SFT (target) | 0 | 1.00 | 1.00× |
| EAGLE, $d$=896, 1 ep | 19.3M | 1.67 | 1.09× |
| EAGLE, $d$=2048, 1 ep | 72.4M | 1.72 | 1.13× |
| EAGLE, $d$=3584, 5 ep | 192.7M | 1.48 | 0.96× |
| EAGLE, $d$=896, 5 ep | 19.3M | **2.00** | 1.32× |
| MARS, $\tau$=0.95 | 0 | 1.785 | **1.56**× |

(b) Fast-dLLM on Block Diffusion (BD) vs. MARS (128 GSM8K prompts, one H100).

| **Decoder** | **tok/s** | **Speedup** |
|---|---|---|
| BD, fixed schedule | 31.0 | 1.00× |
| BD + Fast-dLLM, $\tau$=0.95 | 45.3 | 1.46× |
| MARS, $\tau$=0.95 | **81.6** | **2.63**× |

gate commits too eagerly. Crucially, the 7B model at $\tau$=0.95 (57.8) still exceeds the AR SFT baseline (56.6), and the frontier has no cliff, so one checkpoint serves quality-sensitive requests at $\tau$=1.0 and latency-sensitive requests at lower $\tau$ with no model swap: the "opt-in" property promised in Section 1.

### 4.5 Controlled Comparisons with Other Accelerators

MARS is lossy in multi-token mode, whereas speculative decoding and draft heads such as Medusa and EAGLE reproduce the target distribution exactly, so throughput alone is not comparable across the two families. Table 5 compares under two controlled protocols at 7B in which only the acceleration method varies, with accuracy next to speed (details in Appendix H). EAGLE-style heads trained on our frozen AR SFT target with the same SFT data propose *more* tokens per target forward than MARS (best head: 2.00 vs. 1.785) but deliver *less* end-to-end speedup (1.32× vs. 1.56×), because every proposal costs an extra forward through the head and its 152k-vocabulary projection, and at EAGLE's own 7B width the loop is slower than plain AR decoding (Table 5a). MARS obtains its proposals inside the target forward it runs anyway; it is not lossless, but on this subset it lands 2.3 points above the model it was trained from. Fast-dLLM [40] applied to the Block Diffusion model of Table 2, trained from the same checkpoint with the same data and budget as MARS, gains 1.71× over that model's fixed schedule; MARS is 1.54× faster still (81.6 vs. 53.1 tokens/s) and 2.3 points more accurate (Table 5b), because under causal attention a committed token's KV is final, whereas a bidirectional block must refresh its cache whenever any token changes. An *uncontrolled* reference point, Fast-dLLM on Dream-7B-Instruct [42] at 53.0 tokens/s against 83.9 for MARS under the same harness and budget, is reported in Appendix H with its caveats (different base model and corpus; throughput only). Jacobi decoding on the same checkpoint reaches 1.07 tokens per forward (Appendix F), and MARS composes with lossless speculative decoding for 1.27–1.40× over standard speculative decoding (Appendix J). All throughput numbers come from a research implementation on HuggingFace Transformers [**?** ]; a production engine such as vLLM [**?** ] or SGLang [**?** ] would change the constant factors on both sides.

## 5 Wall-Clock Speedup in Batch Inference

Tokens per forward measures algorithmic speedup; wall-clock throughput is what matters in serving. AR decoding with a KV cache processes one new token per step, whereas MARS processes several masked positions, so it needs a cache of its own. The sampler of the submitted version (Appendix I) synchronized the batch at block boundaries: the cache advanced at the batch-minimum acceptance and faster samples idled, so it reached 1.71× over AR at batch 4 on an H200 but only 1.34× at batch 16.

**Deferred-KV sampler.** We rebuilt the sampler to remove this barrier. Instead of a separate forward to extend the cache with the tokens committed at a step, it folds their KV computation into the next

Table 6: Batch inference on GSM8K (256 new tokens, $\tau$=0.95), MARS-7B, one NVIDIA H100 (PyTorch SDPA). AR: standard decoding with KV cache. "Block-sync.": the submitted version's sampler on the same hardware; remaining columns: the deferred-KV sampler at $W$ proposals per step, as speedup over AR. **Bold**: best $W$ per batch size. The block size $B$ is 16.

| | tok/s | | Deferred-KV sampler (speedup over AR) | | | | |
|---|---|---|---|---|---|---|---|
| **Batch** | AR | Block-sync. (speedup) | $W$=1 | $W$=2 | $W$=4 | $W$=8 | **Best** |
| 4 | 114 | 187 (1.64×) | 1.00× | 1.61× | 2.03× | **2.16**× | 2.16× |
| 8 | 195 | 303 (1.55×) | 1.00× | 1.54× | 1.90× | **2.06**× | 2.06× |
| 16 | 391 | 497 (1.27×) | 1.00× | 1.41× | **1.70**× | 1.46× | 1.70× |
| 32 | 700 | 708 (1.01×) | 1.00× | **1.17**× | 1.10× | 0.86× | 1.17× |
| 64 | 1289 | 862 (0.67×) | **1.00**× | 0.84× | 0.71× | 0.49× | 1.00× |

proposal forward, with a per-sample cursor and a cache-validity bitmask, removing the extension forward, the cache copy and the batch barrier together. With $W$ proposals per step, the first proposal is read from the last committed token's logit (right-shifted logits; its KV is already cached), so only $W-1$ `[MASK]` slots are appended, and at $W$=1 the implementation dispatches to the plain AR path.

**Results.** Table 6 reports batch sizes 4–64 on an H100. The advantage shrinks with batch size for three reasons: AR throughput scales nearly linearly with batch while MARS's scales sublinearly, so it is the baseline that speeds up; the $W-1$ masked positions are free while decoding is latency-bound but cost real time once it is compute-bound, with every rejected proposal wasted; and the block-boundary synchronization, the one we could remove. The deferred-KV sampler raises the best speedup from 1.64× to 2.16× at batch 4 and from 1.27× to 1.70× at batch 16, still gains 17% at batch 32, and moves the crossover at which no $W$ beats AR from batch 32 to batch 64; the optimal $W$ falls with batch size (8, 8, 4, 2, 1).

**Adapting $W$ to load.** The cache layout does not depend on $W$, so $W$ can change at any step with no rebuild; a controller that tracks throughput per $W$ and takes the argmax with hysteresis, keeping $W$=1 in the candidate set, stops proposing under heavy load and runs as AR. With $\tau$, one checkpoint thus has a per-request quality knob and a per-step throughput knob. The regime is the latency-bound one: MARS pays off up to batch 32 and not at batch 64 in our implementation, and the "high request load" it can exploit is the one in which batches are bounded by latency targets rather than by compute.

## 6 Conclusion

We presented MARS, a lightweight fine-tuning method that gives instruction-tuned AR models the ability to generate multiple tokens per forward pass with no architectural changes, no additional parameters, and a single checkpoint, built on an analysis of which gaps between AR and block-masked prediction must be closed, an AR anchor that keeps competence intact as the block grows, and a cache design for the latency-bound regime. Future directions include a production-engine port of the sampler, other base models and SFT corpora (the speedup grows from 0.5B to 7B, suggesting the tokens a forward can commit grow with capacity), adaptive block sizes during training [41], and longer training lengths and larger models.

**Limitations.** MARS training doubles the per-sample sequence length and roughly doubles training compute relative to SFT (Appendix B); we have not validated it beyond a 512-token per-stream budget, which is also our suspected cause for the one benchmark MARS does not improve (GPQA at 7B), and long-context ($\geq$32k) training would be more expensive still. The per-benchmark superset property holds at the recommended $B$=4; at $B$=8 and 16 the average stays at baseline while individual benchmarks dip. The wall-clock advantage is latency-bound: it persists to batch 32 and disappears at batch 64 in our Transformers implementation. All results use one model family, one SFT corpus, greedy decoding and exact-match or pass@1 benchmarks; open-ended chat quality under multi-token acceptance and models above 7B are untested. Appendix A discusses broader impact.

## References


[1] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

[3] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, pages 5209–5235. PMLR, 2024.

[4] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. Dmax: Aggressive parallel decoding for dllms. *arXiv preprint arXiv:2604.08302*, 2026.

[7] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dLLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

[9] Tianqi Du, Lizhe Fang, Weijie Yang, Chenheng Zhang, Zeming Wei, Yifei Wang, and Yisen Wang. Autoregressive models rival diffusion models at any-order generation. *arXiv preprint arXiv:2601.13228*, 2026.

[10] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *International Conference on Machine Learning*, pages 14060–14079. PMLR, 2024.

[11] Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, et al. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*, 2025.

[12] Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. *arXiv preprint arXiv:2510.04147*, 2025.

[13] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

[14] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 6112–6121, 2019.

[15] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning*, pages 15706–15734. PMLR, 2024.

[16] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[17] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018.

[18] Michael Hersche, Nicolas Menet, Ronan Tanios, and Abbas Rahimi. Locally coherent parallel decoding in diffusion language models, 2026.

[19] Ziqi Jin, Bin Wang, Xiang Lin, Lidong Bing, and Aixin Sun. On the role of discreteness in diffusion llms, 2025.

[20] Mahdi Karami and Ali Ghodsi. Auto-regressive masked diffusion models. *arXiv preprint arXiv:2601.16971*, 2026.

[21] Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[22] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[23] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022.

[24] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, pages 28935–28948. PMLR, 2024.

[25] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[26] Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 2025.

[27] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, pages 32819–32848. PMLR, 2024.

[28] 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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025.

[30] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[31] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[32] Junhao Ruan, Bei Li, Yongjing Yin, Pengcheng Huang, Xin Chen, Jingang Wang, Xunliang Cai, Tong Xiao, and JingBo Zhu. Causal autoregressive diffusion language model. *arXiv preprint arXiv:2601.22031*, 2026.

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

[34] Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models: Bridging autoregressive and masked diffusion llms, 2026.

[35] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[36] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[37] Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. In *The Thirteenth International Conference on Learning Representations*, 2024.

[38] Lipeng Wan, Junjie Ma, Jianhui Gu, Zeyang Liu, Xuyang Lu, and Xuguang Lan. Geoblock: Inferring block granularity from dependency geometry in diffusion language models. *arXiv preprint arXiv:2603.26675*, 2026.

[39] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.

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

[41] Hanchen Xia, Baoyou Chen, Yutang Ge, Guojiang Zhao, and Siyu Zhu. T*: Progressive block scaling for mdm through trajectory aware rl. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

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

[43] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[44] Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. *arXiv preprint arXiv:2602.22661*, 2026.

# A Broader Impact

As a general acceleration method, MARS can reduce the compute and energy cost of serving language models. It adds no capability beyond the underlying model and inherits that model's societal risks, including misuse of generated text; since the checkpoint remains a standard AR model, existing safeguards for the underlying model apply unchanged, and we release only fine-tuned versions of openly available models.

# B Training Details

We include all our hyperparameters for training and evaluation in Table 7. Our method can be directly applied to an instruction model without SFT first, yet we found significant performance improvements from doing so (BBH 6.1 $\rightarrow$ 25.6; MMLU-Pro 2.4 $\rightarrow$ 12.2). That is why we first train for 5 epochs with AR SFT: to ensure the model has seen enough of the data and to eliminate the performance gap due to the superior quality of the dataset. We repurpose an existing placeholder token from the model's vocabulary to serve as the `[MASK]` token, so the vocabulary and the embedding matrices are unchanged.

Table 7: Training hyperparameters. Both stages (AR SFT $\rightarrow$ MARS) use identical settings per model size. Effective batch sizes are matched across scales.

| **Hyperparameter** | **0.5B** | **7B** |
|---|---|---|
| Model | Qwen2.5-0.5B-Instruct | Qwen2.5-7B-Instruct (Under Apache 2.0 License.) |
| Training data | Dolci-Instruct-SFT (Under ODC-BY License.) | |
| Optimizer | AdamW | |
| Learning rate | $5 \times 10^{-6}$ | |
| LR schedule | Cosine decay | |
| Warmup ratio | 0.03 | |
| Epochs | 5 (per stage) | |
| Max sequence length (per stream) | 512 (so the concatenated $[\mathbf{c}; \mathbf{x}; \tilde{\mathbf{x}}]$ input is up to $\sim$1024 tokens) | |
| Per-device batch size | 48 | 24 (AR) / 12 (MARS) |
| Gradient accumulation | 1 | 2 (AR) / 4 (MARS) |
| Effective batch size | 384 | 384 |
| Precision | bfloat16 | |
| Hardware | 8$\times$ NVIDIA H200 | |
| Block sizes tested | 4, 8, 16 | 4, 8, 16 |
| SFT loss | Included (MARS) / Excluded (MARS w/o SFT loss) | |
| *Evaluation* | | |
| Generation length | 256 tokens | |
| Decoding | Greedy (temperature = 0) | |
| Steps (MARS) | 256 (one token per step at $\tau$=1.0) | |
| IFEval metric | Mean of prompt-level strict and instruction-level strict accuracy, selected by name | |

**Training cost.** MARS training concatenates a clean and noisy copy of each sequence, doubling the effective sequence length. This incurs additional training cost: for the 0.5B model, AR SFT takes 15 H200-hours while MARS takes 33 H200-hours (2.2$\times$); for the 7B model, 100 vs 202 H200-hours (2.0$\times$). Peak GPU memory usage is approximately 1.5$\times$ that of AR SFT; if memory is the binding constraint, the $1.5L$ split described in Section 3.2 (mask half of the blocks per forward, two passes, clean stream recomputed once) reduces the peak length at the cost of compute, with unchanged gradients. The overhead is a one-time training cost, while the inference saving recurs on every request; for comparison, the EAGLE-style draft heads of Section 4.5 cost from roughly a third of an SFT run ($d$=896) up to about the cost of MARS training ($d$=3584), and add a module to serve (Appendix H). Both MARS stages use only 5 epochs of SFT data, which is modest compared to continual pretraining.

# C Statistical Evidence

**Training variance.** Three independent MARS-0.5B ($B$=4) trainings with an identical recipe and different seeds give six-benchmark averages at $\tau$=1.0 of 30.0, 30.4 and 29.6, against 28.5 for the AR

SFT baseline evaluated in the same pass. With three runs we do not quote a standard deviation; every run sits above the baseline by more than the 0.8-point spread between the runs themselves.

**Evaluation variance.** Table 8 reports paired percentile bootstrap intervals (10,000 resamples over evaluation examples) for the headline deltas, computed from the per-sample logs of the same runs as the main tables. The 7B GSM8K and IFEval gains of MARS over AR SFT are significant; the HumanEval gain is not, and we report it as such. The ablation carrying the mechanism claim, with versus without the SFT loss at $B{=}16$, is far outside noise.

Table 8: Paired percentile bootstrap over evaluation examples, 10,000 resamples. Deltas are accuracy differences in points.

| **Comparison** | **Benchmark** | $\Delta$ | **95% CI** |
|---|---|---|---|
| MARS-7B ($B{=}4$, $\tau{=}1.0$) vs. AR SFT | GSM8K | $+4.5$ | $[+1.7, +7.3]$ |
| MARS-7B ($B{=}4$, $\tau{=}1.0$) vs. AR SFT | IFEval | $+3.9$ | $[+0.7, +7.0]$ |
| MARS-7B ($B{=}4$, $\tau{=}1.0$) vs. AR SFT | HumanEval | $+2.4$ | $[-3.1, +7.9]$ |
| MARS-0.5B ($B{=}16$) with vs. without SFT loss | GSM8K | $+12.1$ | $[+9.5, +14.7]$ |

# D Schedule Ablation

Table 9 reports the alternatives to the two-stream, single-forward schedule discussed in Section 4.3. *Alternating micro-batches* alternate one AR-only micro-batch (clean sequences, $\mathcal{L}_{\text{AR}}$) with one masked-only micro-batch ($[\mathbf{x}; \tilde{\mathbf{x}}]$ with $\mathcal{L}_{\text{mask}}$ only). *Half of each batch* mixes the two kinds of sequence inside every batch. Both use the same data, epochs and learning rate as MARS. Computing the two losses in separate forwards is numerically identical to MARS: the clean stream in the joint forward has the same visibility and position ids as a standalone causal forward, and in float64 the clean-stream logits agree bit-for-bit (maximum difference $0.00$) while the gradients of $\mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{AR}}$ agree to under $7{\times}10^{-7}$, at twice the forward cost.

Table 9: Alternatives that supply the same AR anchor, trained at 0.5B with the same data, epochs and learning rate; six-benchmark average at $\tau{=}1.0$. Masked-only sequences still carry $[\mathbf{x}; \tilde{\mathbf{x}}]$ (the clean copy is the required context) but receive $\mathcal{L}_{\text{mask}}$ only. All rows were evaluated in one pass, so the AR SFT and MARS rows differ slightly from Table 2.

| **Training schedule (0.5B)** | $B{=}4$ | $B{=}16$ |
|---|---|---|
| AR SFT baseline | 28.5 | 28.5 |
| MARS (two streams, one forward; $\mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{AR}}$) | 30.0 | 30.0 |
| Separate forwards for $\mathcal{L}_{\text{AR}}$ and $\mathcal{L}_{\text{mask}}$ | identical to MARS (see text) | |
| Mixed: alternating AR-only and masked-only micro-batches | 31.0 | 30.3 |
| Mixed: half of every batch AR-only, half masked-only | 29.9 | 30.3 |
| No AR anchor ($\mathcal{L}_{\text{mask}}$ only) | 28.0 | 21.8 |

# E Threshold Sweep Details

Table 10 reports the full speed–quality tradeoff for MARS across block sizes $B \in \{4, 8, 16\}$ at both 0.5B (with and without SFT loss) and 7B (with SFT loss) on GSM8K and HumanEval, sweeping the acceptance threshold $\tau$ from 1.0 (one token per step, equivalent to AR) down to 0.5. At 0.5B, $\tau{=}0.95$ with $B{=}4$ gives 1.51 tokens per forward on GSM8K with only 1.7pp accuracy loss relative to $\tau{=}1.0$; the SFT loss variant consistently dominates the w/o SFT loss variant across all operating points, confirming that the SFT loss stabilizes performance across the entire Pareto frontier.

At 7B, the frontier moves up sharply across all block sizes: on GSM8K, $B{=}16$ reaches 2.02 tok/fwd at $\tau{=}0.95$ with only a 2.5pp drop from $\tau{=}1.0$ (75.1→72.6). Scale and block size compound—the 7B model remains confident over longer spans, so larger $B$ translates directly into higher tok/fwd with modest quality cost in the $\tau \geq 0.9$ regime. HumanEval degrades faster at aggressive $\tau$: at $B{=}16$, $\tau{=}0.6$ already drops pass@1 below 40%, reflecting code's strict correctness requirement—format-critical tokens committed prematurely cannot be undone.

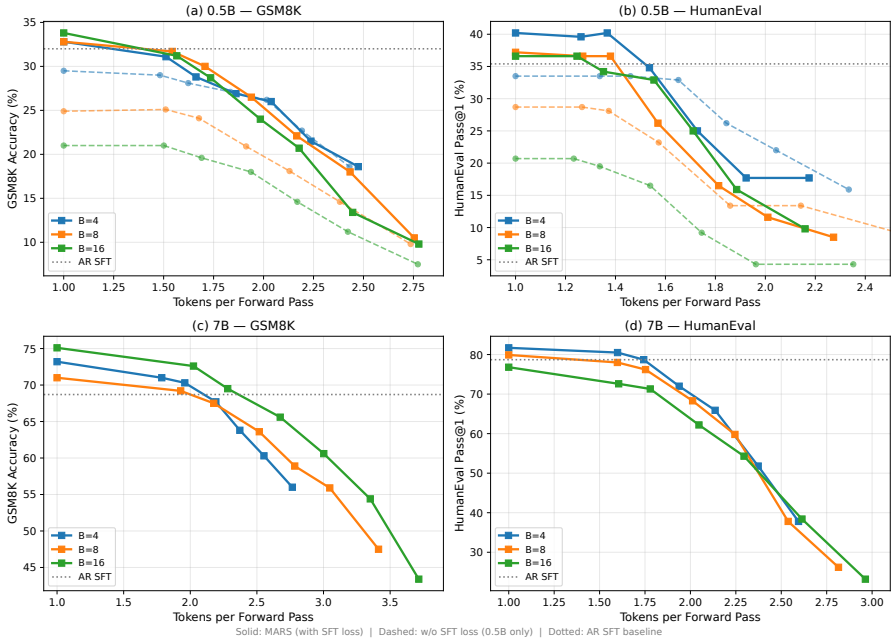


Figure 3: Speed–quality Pareto curves on GSM8K (left) and HumanEval (right). Solid lines: MARS (with SFT loss). Dashed lines: w/o SFT loss. Dotted: AR SFT baseline. With SFT loss, MARS dominates at every operating point on both tasks.

## F Jacobi Decoding Baseline

Table 11 compares Jacobi decoding with MARS on the same AR SFT checkpoint. Jacobi achieves only 1.07× average speedup (tokens per forward), compared to 1.46× for MARS. The limited speedup is expected: in Jacobi, each position sees random tokens as context for earlier positions in the generation region. Since the AR model was never trained to predict from incorrect prefixes, its predictions rarely converge in fewer iterations than sequential generation. MARS addresses this directly by training the model to predict from `[MASK]` placeholders, yielding substantially higher acceptance rates.

Jacobi does have one structural advantage: because it initializes all $N$ output positions at once, the model knows the exact generation length from the start, preventing it from generating beyond the intended boundary. We use a fixed $N$ equal to the evaluation generation length (256) for all instances rather than per-instance ground-truth lengths, so this is not an oracle leak; rather, the same fixed budget that AR SFT uses is presented to the model up-front in Jacobi. This likely explains why Jacobi scores higher than AR SFT on GSM8K (36.5 vs 32.0) and HumanEval (42.1 vs 35.4), where output length control matters for correctness. On format-sensitive and reasoning tasks where this advantage does not apply, Jacobi drops significantly: IFEval ($-7.4$) and BBH ($-7.1$).

Note that under exact greedy fixed-point convergence, Jacobi would by construction reproduce the AR greedy output sequence; the accuracy gap above arises because we use a finite per-step iteration cap (matching the 256 forward-pass budget), so windows that have not fully converged are committed once the budget is exhausted. The IFEval/BBH drops reflect cases where this early-commit behaviour produces predictions that diverge from the AR greedy trajectory.

## G Acceptance Metric Sensitivity

The sliding-window inference by default accepts tokens left-to-right while a confidence score exceeds a threshold $\tau$. In the main experiments we use the probability of the top token, $\max_v p(v \mid \cdot)$, as the confidence score. Here we evaluate two alternatives with MARS ($B{=}4$), on GSM8K at 0.5B and on IFEval at 7B:

Table 10: Speed–quality tradeoff on GSM8K (0-shot) and HumanEval (0-shot) at both scales. Tok/Fwd: average tokens accepted per forward pass. $\tau$=1.0: one token per step, equivalent to AR decoding. Lower $\tau$ accepts more tokens but may reduce accuracy. Each column is a single evaluation sweep of one checkpoint.

| | **GSM8K** | | | | | | **HumanEval** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $B$=4 | | $B$=8 | | $B$=16 | | $B$=4 | | $B$=8 | | $B$=16 | |
| $\tau$ | Acc | T/F | Acc | T/F | Acc | T/F | P@1 | T/F | P@1 | T/F | P@1 | T/F |
| *MARS-0.5B* | | | | | | | | | | | | |
| *AR SFT* | 32.0 | 1.00 | 32.0 | 1.00 | 32.0 | 1.00 | 35.4 | 1.00 | 35.4 | 1.00 | 35.4 | 1.00 |
| 1.0 | 32.8 | 1.00 | 32.8 | 1.00 | 33.8 | 1.00 | 40.2 | 1.00 | 37.2 | 1.00 | 36.6 | 1.00 |
| 0.95 | 31.1 | 1.51 | 31.7 | 1.54 | 31.2 | 1.57 | 39.6 | 1.26 | 36.6 | 1.27 | 36.6 | 1.25 |
| 0.9 | 28.8 | 1.66 | 30.0 | 1.71 | 28.7 | 1.74 | 40.2 | 1.37 | 36.6 | 1.38 | 34.2 | 1.35 |
| 0.8 | 26.9 | 1.86 | 26.5 | 1.94 | 24.0 | 1.98 | 34.8 | 1.54 | 26.2 | 1.57 | 32.9 | 1.55 |
| 0.7 | 26.0 | 2.04 | 22.1 | 2.17 | 20.7 | 2.18 | 25.0 | 1.73 | 16.5 | 1.81 | 25.0 | 1.71 |
| 0.6 | 21.5 | 2.24 | 18.0 | 2.43 | 13.4 | 2.45 | 17.7 | 1.92 | 11.6 | 2.01 | 15.9 | 1.89 |
| 0.5 | 18.6 | 2.47 | 10.5 | 2.75 | 9.8 | 2.78 | 17.7 | 2.18 | 8.5 | 2.27 | 9.8 | 2.16 |
| *MARS-0.5B w/o SFT loss* | | | | | | | | | | | | |
| 1.0 | 29.5 | 1.00 | 24.9 | 1.00 | 21.0 | 1.00 | 33.5 | 1.00 | 28.7 | 1.00 | 20.7 | 1.00 |
| 0.95 | 29.0 | 1.48 | 25.1 | 1.51 | 21.0 | 1.50 | 33.5 | 1.34 | 28.7 | 1.27 | 20.7 | 1.23 |
| 0.9 | 28.1 | 1.62 | 24.1 | 1.68 | 19.6 | 1.69 | 33.5 | 1.46 | 28.1 | 1.37 | 19.5 | 1.34 |
| 0.8 | 27.1 | 1.83 | 20.9 | 1.91 | 18.0 | 1.94 | 32.9 | 1.65 | 23.2 | 1.57 | 16.5 | 1.54 |
| 0.7 | 26.2 | 2.02 | 18.1 | 2.13 | 14.6 | 2.17 | 26.2 | 1.84 | 13.4 | 1.86 | 9.2 | 1.75 |
| 0.6 | 22.7 | 2.19 | 14.6 | 2.38 | 11.2 | 2.42 | 22.0 | 2.04 | 13.4 | 2.14 | 4.3 | 1.96 |
| 0.5 | 18.6 | 2.43 | 9.8 | 2.74 | 7.5 | 2.77 | 15.9 | 2.33 | 9.2 | 2.53 | 4.3 | 2.35 |
| *MARS-7B (with SFT loss)* | | | | | | | | | | | | |
| *AR SFT* | 68.7 | 1.00 | 68.7 | 1.00 | 68.7 | 1.00 | 78.7 | 1.00 | 78.7 | 1.00 | 78.7 | 1.00 |
| 1.0 | 73.2 | 1.00 | 71.0 | 1.00 | 75.1 | 1.00 | 81.7 | 1.00 | 79.9 | 1.00 | 76.8 | 1.00 |
| 0.95 | 71.0 | 1.79 | 69.2 | 1.93 | 72.6 | 2.02 | 80.5 | 1.60 | 78.0 | 1.60 | 72.6 | 1.60 |
| 0.9 | 70.3 | 1.96 | 67.5 | 2.18 | 69.5 | 2.28 | 78.7 | 1.74 | 76.2 | 1.75 | 71.3 | 1.78 |
| 0.8 | 67.7 | 2.19 | 63.6 | 2.52 | 65.6 | 2.68 | 72.0 | 1.94 | 68.3 | 2.01 | 62.2 | 2.05 |
| 0.7 | 63.8 | 2.37 | 58.9 | 2.78 | 60.6 | 3.00 | 65.9 | 2.14 | 59.8 | 2.24 | 54.3 | 2.29 |
| 0.6 | 60.3 | 2.55 | 55.9 | 3.05 | 54.4 | 3.35 | 51.8 | 2.37 | 37.8 | 2.54 | 38.4 | 2.61 |
| 0.5 | 56.0 | 2.77 | 47.5 | 3.41 | 43.4 | 3.72 | 37.8 | 2.60 | 26.2 | 2.82 | 23.2 | 2.96 |

Table 11: Jacobi decoding [37] on the AR SFT checkpoint (0.5B, training-free). Jacobi uses fixed-point iteration: all future positions are initialized with random tokens and iteratively updated via causal forward passes until convergence. Tok/fwd: average tokens accepted per forward pass.

| **Method** | **IFEval** | **BBH** | **MMLU-Pro** | **GPQA** | **GSM8K** | **HumanEval** | **Avg** | **Tok/fwd** |
|---|---|---|---|---|---|---|---|---|
| AR SFT | 48.4 | 26.3 | 11.9 | 17.9 | 32.0 | 35.4 | 28.7 | 1.00 |
| Jacobi (on AR SFT) | 41.0 | 19.2 | 11.7 | 17.6 | 36.5 | 42.1 | 28.0 | 1.07 |
| MARS ($\tau$=0.95, $B$=4) | 50.1 | 26.6 | 11.5 | 19.9 | 31.1 | 39.6 | 29.8 | 1.46 |

- **Entropy**: $H(p) = -\sum_v p_v \log p_v$. Accept while $H \leq \tau$. Lower entropy indicates higher confidence.
- **Top-2 margin**: $p_{\text{top}_1} - p_{\text{top}_2}$. Accept while margin $\geq \tau$. A large gap between the best and second-best token indicates high confidence.

**GSM8K.** Figure 4 shows the speed–quality frontier for each metric. All three trace similar Pareto curves, confirming that the sliding-window acceptance mechanism is robust to the specific confidence measure. Entropy and top-2 margin show marginally smoother degradation at comparable speedups, but the differences are small. We use probability in the main paper for its simplicity.

**IFEval.** GSM8K is not where a format-sensitive failure mode would show, so we repeat the sweep on IFEval with MARS-7B ($B$=4): 49 operating points across the three gates, each an (accuracy,

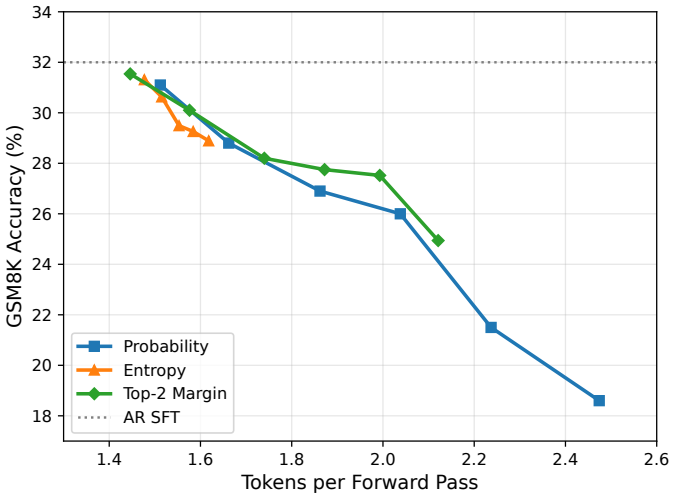


Figure 4: Speed–quality trade-off under three acceptance metrics (MARS $B{=}4$, GSM8K). All three metrics trace similar Pareto frontiers, indicating that the speed–quality trade-off is robust to the choice of acceptance criterion. Entropy and top-2 margin degrade slightly more gracefully than raw probability at comparable tokens per forward pass.

tokens/forward) pair, scored with the strict-metric definition of Section 4.1. Table 12 reports accuracy at matched tokens per forward, linearly interpolated between adjacent measured points. All three gates are within half a point of one-token mode up to 1.15 tok/fwd, which is the IFEval operating point reported in Table 4. Beyond it the gates separate: probability degrades first and entropy fastest, while the top-2 margin, which asks how far ahead the top token is rather than how confident it is, holds up best from 1.25 tok/fwd onward. The shape is what one would expect if high-confidence format tokens are exactly the ones a top-1 probability gate commits too eagerly; a margin gate recovers most of that loss at matched speed. The probability gate remains our default for its simplicity, and the choice matters only past the operating points used in the main tables.

Table 12: IFEval accuracy (strict, mean of prompt- and instruction-level) at matched tokens per forward for three acceptance gates, MARS-7B ($B{=}4$). Values are linearly interpolated between adjacent measured operating points; "–" marks speeds a gate did not reach within the swept thresholds. **Bold**: best gate per row.

| Tokens/forward | Probability | Entropy | Top-2 margin |
|---|---|---|---|
| 1.00 (one-token reference) | 69.4 | – | – |
| 1.10 | 69.5 | **69.6** | – |
| 1.15 | 69.2 | **69.5** | 69.3 |
| 1.20 | **68.7** | 68.6 | 68.6 |
| 1.25 | 67.7 | 66.1 | **68.5** |
| 1.30 | 67.7 | 64.4 | **68.3** |
| 1.35 | 66.3 | 64.8 | **66.9** |
| 1.40 | – | 64.1 | **65.4** |
| 1.45 | – | 63.4 | **63.9** |

## H Details of the Controlled Comparisons

**EAGLE-style draft heads.** We trained draft heads following the EAGLE-1/2 recipe [24**?** ] on our frozen 7B AR SFT model, on the same SFT data, sweeping the head width $d$ and the training length so that neither size nor undertraining can explain the outcome (Table 5a). Evaluation uses 128 GSM8K prompts at batch size 1 on a single GPU, with AR SFT as the baseline. Being lossless, every head reproduces the target's GSM8K accuracy. Every EAGLE proposal costs an extra forward through the head, each ending in a 152k-vocabulary projection, so a higher acceptance rate does

not translate into wall-clock speedup: the 19.3M-parameter head trained for 5 epochs proposes 2.00 tokens per target forward but delivers 1.32×, while a wider head proposes better and slows the loop further, and at EAGLE's own 7B width ($d$=3584, 192.7M parameters) the method ends up slower than plain AR decoding (0.96×). At 0.5B the same 19.3M head reaches 1.06 tokens per target forward and 0.67× wall-clock, against 1.27× for MARS, which suggests draft heads do not suit small targets. On training cost, MARS adds one extra SFT run (0.5B: 15 → 33 GPU-hours; 7B: 100 → 202; Appendix B), whereas a draft head costs from roughly a third of an SFT run at $d$=896 up to about the cost of MARS at $d$=3584, and adds a second module to serve.

**Fast-dLLM on the same lineage.** Fast-dLLM [40] is a training-free decoder for masked diffusion models that unmasks in parallel under a confidence threshold and caches KV at block granularity. To compare decoding methods rather than training corpora, we apply it to the Block Diffusion model of Table 2, which was trained from the same 7B AR SFT checkpoint, on the same SFT corpus, for the same number of epochs at the same block size as MARS, so its training budget matches. The base model, the data, the training budget, the GPU (one H100), the prompts (128 GSM8K prompts) and the generation budget (256 new tokens, batch size 1) are all fixed, and the decoding method is the only variable (Table 5b). Fast-dLLM takes the Block Diffusion checkpoint from 31.0 to 53.1 tokens/s at its best threshold ($\tau$=0.9), a 1.71× gain, while MARS at $\tau$=0.95 reaches 81.6 tokens/s, 1.54× faster, and 2.3 points more accurate (78.9 vs. 76.6). The reason is structural: with bidirectional attention inside a block, changing any token invalidates the whole block's cache and forces a refresh, whereas MARS stays causal, so a committed token's KV is final and is never recomputed.

**Dream-7B (uncontrolled).** For reference only, Fast-dLLM on Dream-7B-Instruct [42], run with the same harness, the same GPU and the same 256-token budget, reaches 53.0 tokens/s against 83.9 tokens/s for MARS-7B. This comparison does not isolate the decoding method: the base model and the training corpus differ, and Dream's accuracy comes from 580B tokens of diffusion pretraining against our ∼1.4B-token SFT corpus. We therefore report throughput only and draw no conclusion about decoding methods from it.

# I Block-Synchronized KV Cache (Submitted Version)

This appendix documents the batch sampler of the submitted version, which Section 5 supersedes. The block-level KV cache (Figure 5) computes the prefix KV once per block; each inner step forwards only $B_{\text{cache}}$ `[MASK]` tokens against it, and the completed block is appended to the shared cache once all batch samples have filled it. Faster samples idle at block boundaries until the slowest finishes. Table 13 reports three batch sizes on GSM8K (256 questions) with Qwen2.5-7B at $\tau$=0.95 on an H200. Speedup is largest at small batches (1.71× at bs=4 → 1.34× at bs=16), where AR's per-token overhead is proportionally higher; the optimal $B_{\text{cache}}$ shifts down as batch grows, balancing prefix amortization against block-boundary synchronization, and several settings fall below AR at bs=16. Accuracy is preserved across all configurations. The deferred-KV sampler of Section 5 removes the synchronization by folding committed tokens' KV computation into the next proposal forward with per-sample cursors and a cache-validity bitmask; Table 6 re-runs this sampler on the same H100 as the new one so that the two are directly comparable (the H200 figures below are therefore not the ones quoted there).

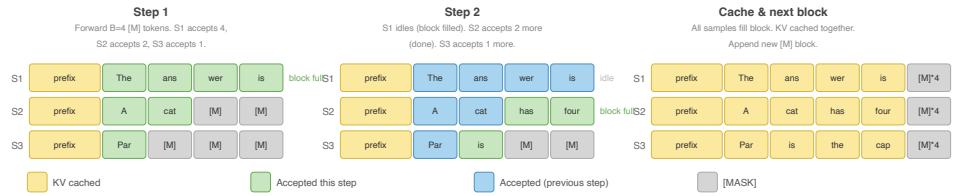


Figure 5: Block-level KV cache for batch inference ($B_{\text{cache}}$=4, batch size 3). Each step forwards $B_{\text{cache}}$ `[MASK]` tokens against the cached prefix. Green tokens are those each sample accepted in the current step (S1 accepts 4, S2 accepts 2, S3 accepts 1). Faster samples then idle at block boundaries until every sample has filled the block; only at that point are the $B_{\text{cache}}$ filled tokens committed to the shared KV cache (yellow), and a fresh group of `[MASK]` tokens is appended for the next block.

Table 13: Submitted version: batch inference on GSM8K (256 questions) with MARS-7B (generation block size $B$=4) at $\tau$=0.95, run on a single NVIDIA H200 with PyTorch SDPA attention. $B_{\text{cache}}$: KV cache synchronization granularity (independent of the generation block size). For each batch size we report tokens/sec, total wall-clock time (seconds), and speedup relative to AR with KV cache (×). **Bold**: best MARS configuration per batch size.

| | | Batch size=4 | | | Batch size=8 | | | Batch size=16 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | $B_{\text{cache}}$ | tok/s | time | × | tok/s | time | × | tok/s | time | × |
| AR (KV cache) | – | 143.4 | 276.2 | 1.00 | 236.7 | 169.1 | 1.00 | 434.3 | 91.8 | 1.00 |
| MARS (no cache) | – | 127.1 | 306.5 | 0.90 | 112.4 | 346.9 | 0.49 | 97.7 | 399.1 | 0.23 |
| MARS + block cache | 4 | 157.2 | 248.8 | 1.11 | 236.4 | 165.4 | 1.02 | 399.3 | 97.2 | 0.94 |
| MARS + block cache | 8 | 196.0 | 199.4 | 1.39 | 289.9 | 134.4 | 1.26 | 484.1 | 80.1 | 1.15 |
| MARS + block cache | 16 | 228.5 | 170.5 | 1.62 | 338.0 | 115.1 | 1.47 | **566.0** | **68.7** | **1.34** |
| MARS + block cache | 32 | **241.9** | **161.2** | **1.71** | **368.4** | **105.6** | **1.60** | 544.9 | 71.5 | 1.28 |
| MARS + block cache | 64 | 239.5 | 162.0 | 1.70 | 342.6 | 113.8 | 1.49 | 383.8 | 101.5 | 0.90 |
| MARS + block cache | 128 | 194.2 | 200.7 | 1.38 | 217.9 | 178.8 | 0.95 | 223.9 | 174.0 | 0.53 |
| MARS + block cache | 256 | 122.7 | 317.6 | 0.87 | 128.7 | 302.7 | 0.56 | 124.1 | 313.9 | 0.29 |

Table 14: Speculative decoding on GSM8K (256 questions) with Qwen2.5-0.5B draft and Qwen2.5-7B target, $K$=4, $\tau$=0.95, run on a single NVIDIA H200. $\Delta$: accuracy change on this 256-question subset relative to the target model's one-token decoding on the same 256 questions. *tok/target*: average tokens produced per target forward; in standard speculative decoding this is upper-bounded by $K$+1=5 (accepted drafts plus one bonus token), and with a MARS-target it is upper-bounded by $K$+$B$=8 since each rejection point can yield up to $B$ confident continuation tokens.

| **Draft** | **Target** | $\Delta$ | **tok/s** | **Time (s)** | **Speedup** | **tok/target** |
|---|---|---|---|---|---|---|
| AR SFT (0.5B) | AR SFT (7B) | +0.1 | 38.1 | 582 | 1.00× | 3.56 |
| AR SFT (0.5B) | MARS (7B, $B$=4) | +1.4 | 53.3 | 457 | 1.27× | **4.14** |
| MARS (0.5B, $B$=4) | MARS (7B, $B$=4) | +1.0 | **58.6** | **417** | **1.40**× | 4.07 |

## J Composing MARS with Speculative Decoding

The multi-token capability of MARS is orthogonal to speculative decoding [22, 4]: the former accelerates each forward pass by generating multiple tokens, the latter reduces the number of expensive target-model forwards by amortizing them across cheap draft-model proposals. We show the two compose naturally for further acceleration.

**Setup.** In standard speculative decoding, a small draft model proposes $K$ tokens and the target model verifies all $K$ in a single forward pass; any prefix that agrees with the target's distribution is accepted, plus one bonus token at the rejection point. We observe that this single bonus token is an unnecessary bottleneck when the target model is MARS-trained: instead of generating one token after the acceptance point, it can generate up to $B$ additional tokens via confidence-based acceptance.

Concretely, inspired by TiDAR [26], we append $K$+1 groups of $B$ mask tokens to the verification input, one group for each possible acceptance outcome $j \in \{0, \ldots, K\}$. A custom attention mask ensures that group $j$ can see the prompt plus the first $j$ draft tokens, and is causal within the group, while remaining invisible to the draft tokens and all other groups (Figure 6). Position IDs for group $j$ continue from position $j$ after the prompt. After a single target forward, standard rejection sampling determines the acceptance count $j$; we then take group $j$'s predictions and apply the same confidence threshold $\tau$ to accept additional tokens left-to-right. The total overhead is $(K+1) \times B$ extra positions in one forward pass, negligible relative to the prompt length.

We test three configurations on GSM8K (256 questions), using Qwen2.5-0.5B as draft and Qwen2.5-7B as target, all with $K$=4, $B$=4, and $\tau$=0.95.

Table 14 shows that MARS provides additive gains at both ends of the pipeline. Replacing the AR target with a MARS target increases tokens per target forward from 3.56 to 4.14, yielding a 1.27× wall-clock speedup. Using MARS for drafting as well reduces draft forwards by 24% (18,959 vs. 24,857) since each draft forward produces multiple tokens, pushing the overall speedup to **1.40**× over standard speculative decoding. The two sources of acceleration are complementary: MARS-enhanced verification produces more tokens per target forward, while MARS drafting reduces the number of

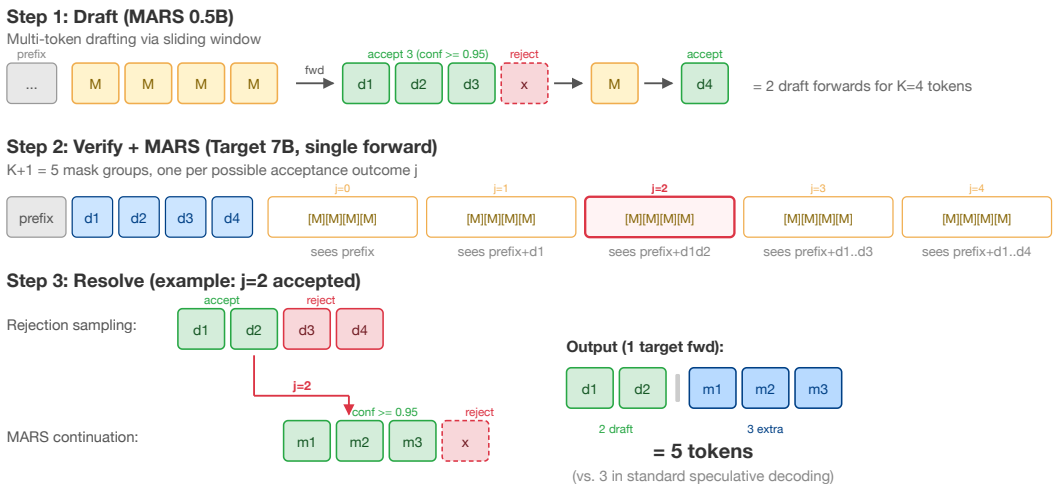


Figure 6: MARS-enhanced speculative decoding ($K$=4, $B$=4). Step 1: the MARS draft model proposes $K$ tokens via multi-token sliding window (2 forwards instead of 4). Step 2: the target model verifies draft tokens and simultaneously predicts $B$ continuation tokens for each possible acceptance outcome $j$, all in a single forward. Step 3: rejection sampling selects $j$=2; the corresponding group yields 3 additional MARS tokens at $\tau$=0.95, for a total of 5 tokens from one target forward (vs. 3 in standard speculative decoding).

(cheap but nonzero) draft forwards. Crucially, accuracy is preserved: the $\Delta$ column shows that all configurations match or slightly exceed the corresponding target model's one-token mode accuracy on GSM8K.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claimed contributions are supported by our experimental results in the Experiment (Section 4).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. We have included a separate Limitations section at the bottom of page 9.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Our paper does not conclude any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included the training details and datasets in Appendix B. And we have a detailed description of our method. Codes for reproducing our results are submitted.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Training codes (including executable scripts) are included in the supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Our training and testing details are included in Appendix B. Our experiments are done once without any hyperparameter tuning.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: All our experiments are done once. Yet, our method consistently demonstrates performance improvements across different block sizes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix B, we give detailed GPU Type and Hours.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our experiment is training open-sourced models on open-sourced datasets. Our work does not involve human subjects or other ethical concerns beyond the model and dataset.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: As a general-purpose acceleration method, MARS could positively reduce computational cost and energy consumption for deployment. It inherits the societal risks of the underlying language models it is applied to.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work poses no pre-trained language models, image generators, or scraped datasets. And it poses no more risks than the underlying language models it is applied to.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the open-sourced dataset and model we applied in our method. Both permit research use. We include their licenses in Appendix B.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide code for details of our proposed method.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used LLM-based coding assistants to assist with code editing and grammar checking.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.