LARGE FLOW LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow and diffusion models are leading approaches to generative modeling of continuous signals such as images, videos, audio, and protein structures. Their parallel generation makes them attractive for fast language modeling, but it conflicts with the inherent directionality of text where earlier tokens contextualize the later ones, whereas conventional flows denoise all positions synchronously. As a consequence, flow language models have largely remained at sub-billion scale and have not established competitive quality-speed tradeoffs on realistic tasks. We present **Lafa**, a block-autoregressive flow language model that preserves causal order across blocks while learning an asynchronous progression within each block, and scale it to 8B parameters by converting a pretrained autoregressive model using less than 10B training tokens. Lafa retains both a block denoiser and a next-token predictor, unifying sequential next-token prediction, parallel flow decoding, confidence decoding, and self-speculative decoding, in a single model. Across six benchmarks spanning instruction following, reasoning, and code, these modes trace a broad quality-speed Pareto frontier, and Lafa attains 2.54× the throughput of a strong autoregressive baseline at matched quality. Our findings show that continuous flows can be a practical and scalable foundation for language modeling.

1 INTRODUCTION

Today's large language models are primarily trained as autoregressive (AR) models, learning to predict the next token based on the preceding text (Achiam et al., 2023; Anil et al., 2023; Guo et al., 2025). The parallelizability of the transformer architecture and teacher forcing allows these models to train efficiently with billions of parameters and trillions of training tokens (Vaswani et al., 2017; Williams & Zipser, 1989). However, their generation process is inherently serial, leading to a bottleneck in inference latency. For accelerated inference, recent work has explored methods for parallel generation, namely multi-token prediction, discrete diffusion, and flow language models. The former two have demonstrated competitive performance with AR models at scale (Gloeckle et al., 2024; Cai et al., 2024; DeepSeek-AI, 2024; Khanna et al., 2025; Song et al., 2025b; Taïga et al., 2026).

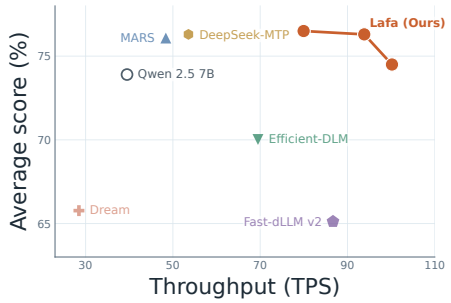


Figure 1: Lafa attains 2.54× the throughput of an AR model at matched or better quality, outperforming the state-of-the-art multi-token prediction and diffusion methods.

Flow language models take a direction distinguished from these two approaches, generating text by integrating a continuous flow from noise to token representations, and they have recently matched discrete diffusion in language modeling (Lee et al., 2026; Chen et al., 2026; Hu et al., 2026; Batzolis et al., 2026; Yang et al., 2026). By operating in a continuous space, they can leverage powerful techniques developed for controllable and accelerated image and video generation, many of which are unavailable for multi-token prediction and discrete diffusion (Lee et al., 2026). Yet, existing flow language models have been demonstrated only at relatively small scales and simple tasks (Lee et al., 2026; Hu et al., 2026), with limited consideration of scalable, hardware-efficient inference primitives such as prefix caching. Critically, prior work has reported that training these models from scratch can be exceedingly inefficient (Gulrajani & Hashimoto, 2023), raising a question on whether flow language models can serve as a practical and scalable approach for language modeling.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we demonstrate that *flow language models can be both practical and scalable*. We introduce ***Lafa* (*La*rge *f*low l*a*nguage model)**, a block-autoregressive flow language model obtained by converting a pretrained AR model, and successfully scale it to 8B parameters, among the largest up to date, using only 9B training tokens, by contrast to similar adaptation of discrete diffusion models that use up to 500B tokens (Ye et al., 2025; Fu et al., 2025). Our design is guided by the sequential nature of natural language, where earlier tokens provide context for later ones. Across blocks, Lafa retains the causal structure of the pretrained model, generating block by block and remaining compatible with prefix caching. Within each block, instead of denoising all positions synchronously, Lafa learns an asynchronous flow that denoises earlier tokens faster, so that they can inform later positions. We further introduce a training objective under which the same weights retain both block denoising and next-token prediction, so that a single model supports next-token decoding, parallel flow decoding, confidence-based decoding, and self-speculative decoding without an external draft model. Together, these modes trace a quality-speed Pareto frontier that outperforms similarly sized AR and discrete diffusion models (Figure 1). Our main contributions are:

- We introduce Lafa, a block-autoregressive flow language model that learns asynchronous denoising while retaining next-token prediction, unifying four decoding modes in a single model.
- We identify key designs for scalable and practical models, including blockwise causal architecture, time conditioning, tokenwise time sampling, the training objective, and the inference strategy.
- We scale Lafa to 8B parameters using only 9B training tokens. Across six benchmarks spanning instruction following, scientific and mathematical reasoning, and coding, Lafa attains $2.54\times$ the throughput of a strong AR baseline at matched quality, outperforming existing parallel generators.

Our work shows that continuous flows offer a practical, scalable foundation for language models.

2 Large flow language models

Lafa adopts a blockwise flow formulation with causal structure across blocks and joint modeling within blocks to enable parallel generation while preserving pretrained autoregressive capabilities and prefix caching, and to align generation with the left-to-right bias of natural language.

In Section 2.1, we present our blockwise formulation via asynchronous flows over token embeddings parameterized with a block denoiser. Sections 2.2 and 2.3 describe how we construct and train the block denoiser by adapting an autoregressive model. Section 2.4 describes flow map distillation for generation with fewer evaluations. Finally, Section 2.5 presents the supported inference schemes, including next-token prediction (NTP), flow, flow-map, confidence, and self-speculative decoding.

2.1 Asynchronous block flows

We formulate the modeled distribution blockwise and represent each block via asynchronous flows.

Blockwise formulation. Let V denote the token vocabulary. We model the conditional distribution $p(\mathbf{y} \mid \mathbf{c})$ of a continuation $\mathbf{y} \in V^M$ given a prefix $\mathbf{c} \in V^N$ through block autoregression (Wang et al., 2018) with block size B, assuming it divides M:

$$p(\mathbf{y} \mid \mathbf{c}) = \prod_{k=0}^{\frac{M}{B}-1} p\big(\mathbf{y}^{kB+1:(k+1)B} \mid \mathbf{c}, \mathbf{y}^{\leq kB}\big). \tag{1}$$

Absorbing previously generated blocks into $\mathbf{c}$, it suffices to model the joint distribution $p(\mathbf{y}^{1:B} \mid \mathbf{c})$ of the next block. To generate a block in parallel while capturing dependencies among its tokens, we model its joint distribution using a continuous flow.

Embedded asynchronous flows. We embed categorical tokens with vocabulary embeddings $\mathbf{E} \in \mathbb{R}^{|V|\times d}$ into a continuous space and model the distribution $p(\mathbf{x} \mid \mathbf{c})$. A clean block is represented as

$$\mathbf{x} = \big(\mathbf{E}_{\mathbf{y}^1}, \ldots, \mathbf{E}_{\mathbf{y}^B}\big)^\top \in \mathbb{R}^{B\times d}. \tag{2}$$

To model this distribution, we use flow matching (Lipman et al., 2023; Albergo et al., 2025), which constructs a probability path between noise and data and a velocity field that transports samples

along the path. We independently draw the data and noise samples as $\mathbf{x}_1 \sim p_1(\cdot \mid \mathbf{c}) \coloneqq p(\cdot \mid \mathbf{c})$ and $\mathbf{x}_0 \sim p_0 \coloneqq \mathsf{N}(\mu, \sigma^2 I)$, respectively, where μ and σ specify the noise shift and scale.

While block autoregression imposes a left-to-right ordering across blocks, interpolation also lets us control the progression within each block. Drawing from prior work on image generation (Baade et al., 2026; Qian & Cheng, 2026) and language modeling (Wu et al., 2023; Bansal & Sanghavi, 2026), we hypothesize that letting positions progress at different rates benefits block generation.

We thus use tokenwise time instead of shared time within each block. Let $\boldsymbol{\tau}(t) \in [0,1]^B$, mapping the global time t to tokenwise time $\tau^j(t)$ of each position j, be differentiable and strictly increasing in each coordinate, with $\boldsymbol{\tau}(0) = \mathbf{0}$, $\boldsymbol{\tau}(1) = \mathbf{1}$. We define the probability path through the interpolant

$$I_t = (1 - \boldsymbol{\tau}(t)) \odot \mathbf{x}_0 + \boldsymbol{\tau}(t) \odot \mathbf{x}_1, \qquad I_t \sim p_t(\cdot \mid \mathbf{c}), \qquad t \in [0,1], \tag{3}$$

where $\boldsymbol{\tau} \odot \mathbf{x} \coloneqq \mathsf{diag}(\boldsymbol{\tau})\mathbf{x}$. Standard linear interpolant is recovered when $\tau^j(t) = t$ at every position.

Block denoiser. (3) admits a velocity field b_t whose ODE transports noise to clean embeddings,

$$\dot{\mathbf{x}}_t = b_t(\mathbf{x}_t; \mathbf{c}), \qquad \mathbf{x}_0 \sim p_0, \qquad \mathbf{x}_t \sim p_t(\cdot \mid \mathbf{c}), \qquad t \in [0,1]. \tag{4}$$

Rather than predicting the velocity directly, we learn a categorical denoiser that predicts clean tokens from the noisy block (Dieleman et al., 2022),

$$D_t(\mathbf{x}; \mathbf{c})^j \coloneqq p(\mathbf{y}^j = \cdot \mid \mathbf{c}, I_t = \mathbf{x}) \in \Delta^{|V|-1}, \qquad j = 1, \ldots, B. \tag{5}$$

Its expected token embeddings recover the asynchronous velocity through

$$b_t(\mathbf{x}; \mathbf{c})^j = \frac{\dot{\tau}^j(t)}{1 - \tau^j(t)} \big(D_t(\mathbf{x}; \mathbf{c})^j \mathbf{E} - \mathbf{x}^j \big). \tag{6}$$

Thus, learning the block flow reduces to learning tokenwise denoising probabilities. The schedules in (3) set how much noise remains at each position, while $\dot{\tau}^j(t)$ in (6) sets how fast its tokenwise time advances. For instance, $\tau^i(t) > \tau^j(t)$ for $i < j$ denoises earlier tokens first, adding a left-to-right bias within each block; we learn these schedules during training (Section 2.3).

2.2 Adapting autoregressive models

We implement the block denoiser by adapting a pretrained autoregressive transformer, modifying only its attention, time conditioning, and embeddings.

Attention. We apply causal attention to the clean prefix and bidirectional attention within the current noisy block, whose tokens can also attend to the entire prefix. Since prefix tokens cannot attend to the noisy block, their keys and values are independent of the block's evolving state and can be cached and reused throughout denoising and when generating subsequent blocks. The clean prefix also retains an autoregressive readout; the output at prefix position i predicts the next token $\mathbf{c}^{i+1}$ given $\mathbf{c}^{\leq i}$ (Tian et al., 2025; Jin et al., 2026).

Time conditioning. Let us write the full transformer output with F_t, including both prefix and block positions. Its inputs are the concatenated clean and noisy embeddings and their tokenwise times,

$$\begin{aligned} F_t(\mathbf{x}; \mathbf{c}) &\coloneqq \mathsf{transformer}_\theta(\mathbf{z}, \mathbf{t}) \in (\Delta^{|V|-1})^{N+B}, \\ \mathbf{z} &\coloneqq \left(\mathbf{E}_{\mathbf{c}^1}, \ldots, \mathbf{E}_{\mathbf{c}^N}, \mathbf{x}^1, \ldots, \mathbf{x}^B\right)^\top, \quad \mathbf{t} \coloneqq (\mathbf{1}_N, \boldsymbol{\tau}(t))^\top. \end{aligned} \tag{7}$$

We write $D_t(\mathbf{x}; \mathbf{c})^j \coloneqq F_t(\mathbf{x}; \mathbf{c})^{N+j}$ for the block denoising output and $A(\mathbf{c})^i \coloneqq F_t(\mathbf{x}; \mathbf{c})^i$ for the autoregressive prefix readout. Hereafter, we write D_t for the learned model of (5).

We inject tokenwise time through adaptive layer normalization (adaLN) with zero-initialized modulation (Peebles & Xie, 2023). The 0.5B models use a separate modulator in each transformer layer, while the 7B and 8B models share one modulator across all layers for parameter efficiency. All prefix positions instead receive the fixed clean time 1; together with causal prefix attention, this makes $A(\mathbf{c})$ and the prefix KV cache independent of both $\mathbf{x}$ and t.

Embeddings. We initialize $\mathbf{E}$ from the pretrained vocabulary embeddings and train it jointly with the denoiser so that its geometry can adapt to flows. Unconstrained embedding updates, however, would allow the model to inflate token norms relative to the fixed noise scale σ, making denoising easier by reducing the effective corruption. We prevent this by fixing each embedding's norm to its pretrained value and learning only its direction, $\|\mathbf{E}_v\|_2 = \|\mathbf{E}_v^{\mathrm{pre}}\|_2$ for all $v \in V$.

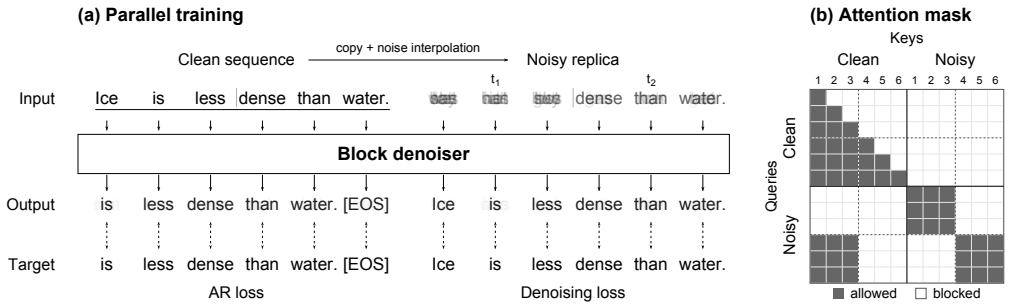


Figure 2: Parallelized training over two blocks of size three.

2.3 Training the block denoiser

In this section, we explain how to adapt the autoregressive model into a block denoiser.

Training objective. For a prefix-block pair $(\mathbf{c}, \mathbf{y})$, draw $t \sim \mathsf{U}[0,1]$ and $\mathbf{x}_0 \sim p_0$, and construct I_t using (3) with $\mathbf{x}_1^j = \mathbf{E}_{\mathbf{y}^j}$. Writing $\mathbf{p}^j := D_t(I_t; \mathbf{c})^j$, we optimize

$$\mathcal{L}(\theta) := \mathbb{E}_{\mathbf{c},\mathbf{y},t,\mathbf{x}_0}\left[\underbrace{\sum_{i=1}^{N-1} -\log A(\mathbf{c})^i_{\mathbf{c}^{i+1}}}_{\text{autoregressive cross entropy}} + \underbrace{\sum_{j=1}^{B} -\log \mathbf{p}^j_{\mathbf{y}^j}}_{\text{denoising cross entropy}} + \underbrace{\sum_{j=1}^{B} \|\mathbf{p}^j\mathbf{E} - \mathbf{E}_{\mathbf{y}^j}\|_2^2}_{\text{embedding squared error}}\right]. \tag{8}$$

Here θ includes the transformer parameters and the trainable embedding directions. The autoregressive term preserves next-token prediction during adaptation and trains the readouts used for self-speculative verification in Section 2.5. The denoising term learns the clean-token posterior, while the embedding squared error supervises the posterior mean embedding that determines the velocity in (6). For fixed embeddings, the latter two terms are minimized by the true denoising posterior, yet we find their combination works best (Table 2).

Parallel training. To train on multiple blocks in parallel, we follow prior work (Arriola et al., 2025; Tian et al., 2025; Jin et al., 2026; Liu et al., 2025) and retain a clean training sequence alongside a second copy partitioned into noisy blocks of size B (Figure 2). The attention mask applies the structure of Section 2.2 to every block at once: clean positions attend causally, and each noisy block attends bidirectionally within itself and to the clean tokens preceding it, but not to later clean tokens or to other noisy blocks. We apply the autoregressive loss to the clean sequence and the two denoising losses to each noisy block, so many prefix–block pairs contribute in a single forward pass.

Learning tokenwise time. We adapt $\boldsymbol{\tau}$ during training so that regions with higher denoising loss occupy a larger interval of global time, placing more steps of a uniform solver grid where each position needs them. This yields a separate monotone reparameterization for each block position; the estimation algorithm is in Appendix C.2. The learned schedules generally satisfy $\tau^i(t) \gtrsim \tau^j(t)$ for $i < j$ (Figure 3), so earlier positions are denoised first.

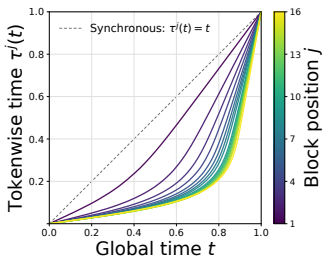


Figure 3: Learned tokenwise time schedules $\tau^j(t)$.

Self-conditioning. We use self-conditioning (Chen et al., 2022) to improve the flow. With a fixed probability during training, we obtain an initial denoising prediction, provide it as an additional conditioning, and compute the loss on a second forward pass. This trains the denoiser to refine an existing prediction (Appendix C.3).

2.4 Flow map distillation

Generating the block from flow requires multiple evaluations per block to reduce the discretization error of Euler integration. We therefore distill the flow into its flow map (Boffi et al., 2025), which transports a trajectory of (4) from source time s to target time t,

$$X_{s,t}(\mathbf{x}_s; \mathbf{c}) := \mathbf{x}_s + \int_s^t b_u(\mathbf{x}_u; \mathbf{c})\,\mathrm{d}u = \mathbf{x}_t, \qquad 0 \le s \le t \le 1. \tag{9}$$

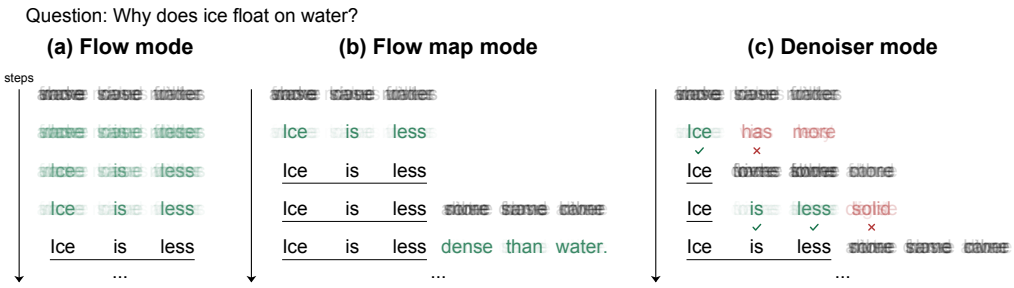


Figure 4: Three parallel inference modes supported by Lafa. Committed tokens are underlined.

Embedded two-time denoiser. We parameterize the learned map $\widehat{X}_{s,t}$ using a two-time categorical denoiser $\delta_{s,t}(\mathbf{x};\mathbf{c})^j \in \Delta^{|V|-1}$ (Lee et al., 2026). Writing $\tau_s^j := \tau^j(s)$, we define

$$\widehat{X}_{s,t}(\mathbf{x};\mathbf{c})^j := \frac{1-\tau_t^j}{1-\tau_s^j}\mathbf{x}^j + \frac{\tau_t^j-\tau_s^j}{1-\tau_s^j}\,\delta_{s,t}(\mathbf{x};\mathbf{c})^j\mathbf{E}, \qquad 0 \le s < t \le 1. \tag{10}$$

The two-time denoiser predicts an embedding-space target for the entire interval $[s,t]$. On the diagonal, we anchor $\delta_{t,t}$ to the teacher denoiser D_t^{T}, recovering its instantaneous velocity. The map itself satisfies $\widehat{X}_{t,t}(\mathbf{x};\mathbf{c}) = \mathbf{x}$. See Appendix D for the derivation.

Architecture. We initialize the student from the frozen teacher denoiser D_t^{T}, keeping its attention structure and its time embedding for the source time s, and add a zero-initialized embedding of the target time t so that the student initially reproduces D_s^{T}. Since the prefix still receives clean time 1, the student retains the prefix KV cache and an autoregressive readout A^{S}.

Distillation objective. We combine semigroup distillation (Lee et al., 2026) with autoregressive distillation, where A^{T} is the teacher's autoregressive readout,

$$\mathcal{L}_{\text{distill}} = \underbrace{\mathbb{E}\,\mathsf{KL}(A^{\mathrm{T}}\|A^{\mathrm{S}})}_{\text{autoregressive distillation}} + \underbrace{\mathbb{E}\,\mathsf{KL}(D_t^{\mathrm{T}}\|\delta_{t,t})}_{\text{diagonal distillation}} + \underbrace{\mathbb{E}\,\mathsf{KL}(\bar{\delta}_{s,t}\|\delta_{s,t})}_{\text{off-diagonal distillation}}. \tag{11}$$

Here, KL sums tokenwise divergences over the $N-1$ prefix positions for the AR term and the B block positions for the denoising terms. We suppress the respective inputs $\mathbf{c}$, $(I_t;\mathbf{c})$, and $(I_s;\mathbf{c})$; the expectations are over $\mathbf{c}$, $(t,\mathbf{c},I_t)$, and $(s,u,t,\mathbf{c},I_s)$, respectively.

The first two terms anchor the student's next-token predictions and instantaneous denoising to the teacher. The off-diagonal target $\bar{\delta}_{s,t}$ composes stop-gradient student transitions through $u \in (s,t)$, enforcing $\widehat{X}_{s,t} \approx \widehat{X}_{u,t} \circ \widehat{X}_{s,u}$ and extending local accuracy to longer transitions (Appendix D.2).

2.5 Inference schemes

With the trained denoiser and its distilled flow map, Lafa can generate each block in several ways that trade off quality against integration step counts (denoted NFE) and number of tokens committed per model forward (denoted TPF) (Figure 4). Every scheme conditions on the cached clean prefix, commits a block or a block prefix, and appends it to the cache; in addition, the prefix readout A also supports standard next-token prediction (NTP).

Flow mode. Starting from $\hat{\mathbf{x}}_0 \sim p_0$, we take K Euler steps on the temporal grid $t_i = i/K$. With $\boldsymbol{\tau}_i := \boldsymbol{\tau}(t_i)$ and $\mathbf{p}_i := D_{t_i}(\hat{\mathbf{x}}_{t_i};\mathbf{c})$, this process is written as

$$\hat{\mathbf{x}}_{t_{i+1}} = \hat{\mathbf{x}}_{t_i} + \left[(\boldsymbol{\tau}_{i+1}-\boldsymbol{\tau}_i) \oslash (1-\boldsymbol{\tau}_i)\right] \odot (\mathbf{p}_i\mathbf{E} - \hat{\mathbf{x}}_{t_i}), \qquad \hat{\mathbf{y}}^j = \arg\max_{v\in V}(\mathbf{p}_{K-1}^j)_v, \tag{12}$$

where $\oslash$ is elementwise division. For self-conditioning (Section 2.2), each step uses previous step's prediction as the self-conditioning input, so it adds no forward passes. We also apply prefix guidance to strengthen prefix conditioning (Appendices C.4 and C.5); this uses one conditional and one prefix-free model evaluation per Euler step, and prefix-free evaluation is cheap as input length is always B.

Flow map mode. The distilled model replaces each numerical update with a learned transition,

$$\hat{\mathbf{x}}_{t_{i+1}} = \widehat{X}_{t_i,t_{i+1}}(\hat{\mathbf{x}}_{t_i};\mathbf{c}), \qquad i = 0,\dots,K-1, \tag{13}$$

and decodes the tokenwise argmax of $\delta_{t_{K-1},1}(\hat{\mathbf{x}}_{t_{K-1}}; \mathbf{c})$.

Denoiser mode. For faster generation, the denoiser drafts a block from a single prediction at $t = 0$,

$$\mathbf{q}^j := D_0(\mathbf{x}_0; \mathbf{c})^j, \qquad \hat{\mathbf{y}}^j = \arg\max_{v \in V} (\mathbf{q}^j)_v. \tag{14}$$

Since $\mathbf{x}_0$ carries no information about the block, each $\mathbf{q}^j$ predicts position j drawing from the prefix, so the most likely token at each position may not be consistent with the others. We therefore accept draft tokens from left until the first one that fails an acceptance rule. *Confidence decoding* (Jin et al., 2026) accepts a token if its confidence $(\mathbf{q}^j)_{\hat{\mathbf{y}}^j}$ is at least γ. *Self-speculative decoding* instead accepts a token if it matches the model's own greedy next-token prediction from A, given the prefix and the preceding draft tokens, so every committed token matches greedy NTP decoding. Since A is read out at prefix positions and D_0 at block positions, verifying and drafting share a single forward pass.

3 Experiments

3.1 Setup

Models and dataset. We build Lafa 0.5B, Lafa 7B, and Lafa 8B by converting the pretrained autoregressive checkpoints Qwen2.5-0.5B-Instruct, Qwen2.5-7B-Instruct (Yang et al., 2024), and Qwen3-8B-Base (Yang et al., 2025), respectively, into blockwise flow language models as described in Section 2. The 0.5B models use a separate adaLN modulator in each transformer layer, whereas the 7B and 8B models share one modulator across layers for efficiency. All three models are fine-tuned on Dolci-Instruct-SFT (Olmo et al., 2025) for five epochs ($\approx$ 9B tokens). Table 3 compares this adaptation budget with large diffusion baselines.

Baselines. At 0.5B we compare against multi-token predictors (Gloeckle et al., 2024; DeepSeek-AI, 2024) and block/parallel decoders (Jin et al., 2026; Gat et al., 2025; Arriola et al., 2025); at 7–8B we additionally compare against masked-diffusion and block-diffusion models (Wu et al., 2026; Nie et al., 2025; Ye et al., 2025; Fu et al., 2025), along with an autoregressive model fine-tuned on identical data. Unless stated otherwise we use each method's native sampling protocol.

Benchmarks. To our knowledge, we are the first to provide evaluation of flow language models on complex generative tasks. We adopt this scheme compared to the usual generative perplexity owing to the flaws stating in Franca & Tong (2026). All methods are evaluated using lm-evaluation-harness (Sutawika et al., 2024) with each benchmark's default prompts, we evaluate on six standard benchmarks spanning instruction following, scientific and mathematical reasoning, and code generation: IFEval (0-shot) (Zhou et al., 2023), BBH (3-shot) (Suzgun et al., 2023), MMLU-Pro (0-shot) (Wang et al., 2024), GPQA (0-shot) (Rein et al., 2023), GSM8K (0-shot) (Cobbe et al., 2021), and HumanEval (0-shot) (Chen et al., 2021). We generate at most 1024 tokens per sample, except 512 for GSM8K and 640 for HumanEval. We report prompt-level strict accuracy for IFEval, pass@1 for HumanEval, and accuracy for the remainder, and summarize with the macro-average across the six.

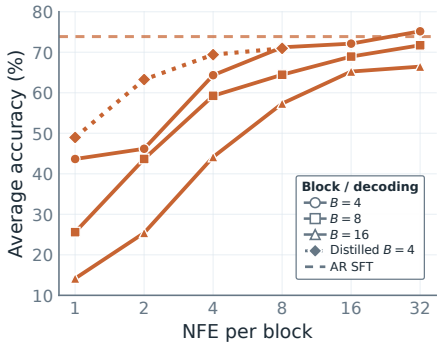


Figure 5: Lafa 7B in flow mode achieves higher average performance as NFEs per block increases. Solid curves denote teachers at different block sizes. The dotted diamond curve denotes the distilled flow map.

3.2 Flow and Flow Map

Figure 5 reports flow mode decoding (12) results for Lafa 7B at block sizes $B \in \{4, 8, 16\}$. Across block sizes, using more number of function evaluations (NFEs) generally improve the average performance, showing that the converted models learn the velocity fields transporting the sample from noise to data. We observe a decline in ceiling performance as we scale block size; larger block

Table 1: Single-token decoding across six benchmarks over Qwen2.5-0.5B, Qwen2.5-7B, and Qwen3-8B. For diffusion models, we decode exactly one highest-probability token per forward. Models marked † use the matched AR model and are trained for five epochs on Dolci-Instruct-SFT.

Method	IFEval	BBH	MMLU-Pro	GPQA	GSM8K	HumanEval	Avg.
0.5B scale							
Qwen-2.5 0.5B SFT†	48.4	26.3	11.9	17.9	32.0	35.4	28.7
MTP† (Gloeckle et al., 2024)	50.6	17.3	9.2	18.8	27.1	32.3	25.9
DS-MTP† (DeepSeek-AI, 2024)	53.2	24.1	14.0	18.5	35.5	36.0	30.2
BD3LM† (Arriola et al., 2025)	47.1	7.5	2.0	17.9	30.6	31.7	22.8
SBD† (Gat et al., 2025)	55.3	20.6	9.2	22.5	27.2	30.8	27.6
MARS† (Jin et al., 2026)	51.3	26.6	12.4	19.4	32.8	**40.2**	30.4
Lafa 0.5B (Ours)†	**55.5**	**27.3**	12.0	**23.4**	**36.8**	35.4	**31.7**
7B scale							
Qwen-2.5 7B SFT†	73.8	59.2	41.6	27.9	69.1	78.7	58.4
MTP† (Gloeckle et al., 2024)	73.2	51.7	37.3	26.8	60.5	73.8	53.9
DS-MTP† (DeepSeek-AI, 2024)	**78.0**	60.7	40.6	**30.1**	70.3	80.5	60.0
Fast-dLLM v2 (Wu et al., 2026)	64.5	58.1	39.7	21.4	80.3	62.2	54.4
Fast-dLLM v2†	70.6	53.5	43.1	28.6	67.2	32.9	49.3
Dream (Ye et al., 2025)	57.1	**66.0**	43.6	26.6	**81.7**	58.5	55.6
MARS† (Jin et al., 2026)	77.8	54.6	**44.4**	26.6	73.2	**81.7**	59.7
Lafa 7B (Ours)†	77.8	63.7	41.9	26.1	74.1	76.8	**60.1**
8B scale							
Qwen-3 8B SFT†	38.1	71.6	56.2	29.5	62.9	**82.3**	56.8
LLaDA (Nie et al., 2025)	58.0	30.9	28.7	25.4	76.0	42.1	43.5
EfficientDLM (Fu et al., 2025)	58.4	70.8	**61.5**	**33.7**	**87.4**	68.9	**63.5**
Lafa 8B (Ours)†	**61.4**	**71.6**	56.4	31.2	77.8	80.5	63.2

sizes provide exponentially larger denoising space and hence we hypothesize that it would require similarly scaled training corpus, which we leave for future work. The corresponding 0.5B frontier is deferred to the appendix (Figure 8), and complete per-task values for both scales are in Table 5. Overall, the 7B frontier is substantially stronger than 0.5B across block sizes and NFEs. For example, at B=4 and NFE=32, average accuracy rises from 38.1 to 75.2, showing that Lafa scales favorably with model capacity.

Table 6 reports the performance of Lafa 8B converted from Qwen3-8B-Base. It achieves a three-task average of 72.3 at NFE=8. We use Qwen3-8B-Base rather than Qwen3-8B-Instruct because further SFT substantially degrades the heavily post-trained Instruct checkpoint.

We further show that flow map distillation improves few-step generation. We distill the B=4 teachers at both scales and report the distilled models as dotted curves in Figures 5 and 8. At both scales, distillation improves accuracy in the one- and two-step regime, e.g., from 43.6 to 49.0 at NFE=1 and from 46.2 to 63.3 at NFE=2 for 7B, while remaining close to the teacher at larger NFE.

3.3 Self-speculation through NTP

As shown in Table 1, Lafa outperforms its matched AR SFT baseline under single-token decoding at all three scales. This observation is consistent with prior work showing that auxiliary multi-token training can improve next-token prediction (NTP) (DeepSeek-AI, 2024; Mahajan et al., 2026). Further, Lafa preserves or improves the matched baseline's single-token decoding performance and remains competitive with the strongest discrete diffusion baseline at 8B. The discrete diffusion uses single-token unmasking per step, which corresponds to its highest-quality decoding regime.

Self-speculative decoding. Since Lafa retains strong NTP performance (Table 1), it can verify its own one-step flow drafts without relying on an external draft model. Figure 6 and Table 7 report the resulting trade-off between tokens per forward (TPF) and macro-average accuracy across GSM8K, HumanEval, and IFEval. For the 0.5B model, Lafa achieves an average accuracy of 44.0 at TPF=3.08, outperforming both the matched Qwen2.5 SFT baseline and all parallel decoding baselines. For Lafa 7B, increasing the block size from B=4 to B=16 raises TPF from 2.45 to 4.49 while maintaining average accuracy at or above the matched Qwen2.5 SFT baseline. More-

over, at larger block sizes, Lafa achieves higher TPF than MARS at comparable accuracy. Finally, Lafa 8B achieves an average accuracy of 73.3 at TPF=2.67, compared with 69.7 at TPF=2.83 and 71.7 at TPF=2.74 for Efficient-DLM's B=4 and B=32, respectively. Across all three scales, self-speculation converts one-step flow drafts into 2.5–4.5 tokens per forward while maintaining at least the accuracy of autoregressive decoding.

Wall-clock throughput. TPF alone does not determine decoding speed because the cost of a forward pass differs across methods. We therefore measure end-to-end generation throughput on a single H100 at batch size 1 using each baseline's native decoder. Full evaluation details are provided in Appendix E.5. At comparable accuracy, Lafa reaches 100.2 tokens/s, yielding a 2.54× speedup over AR decoding (Figure 1). DS-MTP (DeepSeek-AI, 2024) and MARS (Jin et al., 2026) match or exceed AR accuracy, but neither surpasses 53.6 tokens/s. No diffusion baseline matches AR accuracy, and every diffusion configuration above 75 tokens/s trails AR by at least 6.3 accuracy points. These results show that the TPF gains from self-speculative decoding translate into substantial wall-clock speedups.

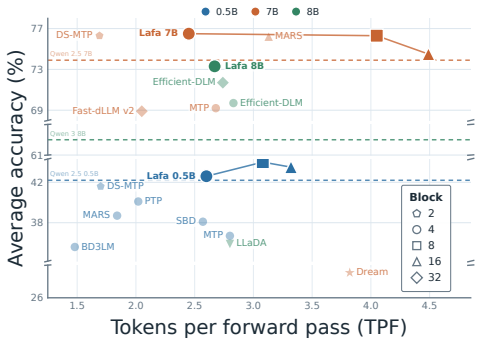


Figure 6: Lafa self-speculation extends the accuracy-TPF frontier across model scales.

3.4 Confidence Decoding

Following MARS (Jin et al., 2026), we apply confidence decoding to Lafa and the causal block-prediction baselines by accepting the longest left-to-right prefix whose token confidence exceeds γ. Dream and LLaDA instead use their native global entropy or confidence ordering over masked positions, whereas Fast-dLLM v2 applies the threshold within the active sub-block. Sweeping γ traces a controlled accuracy–TPF frontier for one-step block prediction. For Lafa, TPF counts raw model forward passes, so this evaluation isolates the calibration of the one-step predictor without speculative verification. At 0.5B, the B=4 model reaches a peak average accuracy of 43.4 at TPF=1.71, exceeding Qwen2.5 SFT and the best high-accuracy operating point of every competing method (Figure 7). Larger blocks offer greater parallelism, but their accuracy degrades more quickly as γ is relaxed. The useful operating region therefore lies near the high-confidence knee of the frontier. The full sweeps, method-specific details, and 7B comparison are reported in Appendix E.3. At the Qwen3-8B scale, Lafa achieves an average accuracy of 72.3 at TPF=1.39, compared with 61.1 at TPF=1.00 for Qwen3 SFT and 71.7 at TPF=2.25 for Efficient-DLM (Tables 9 and 10).

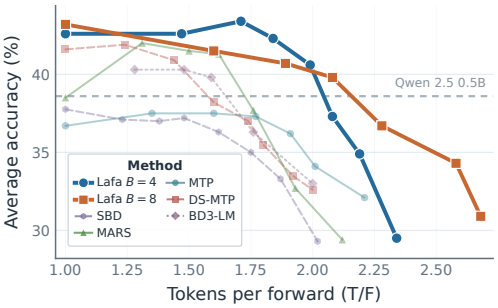


Figure 7: Lafa confidence decoding extends the accuracy-TPF frontier at the 0.5B scale.

3.5 Ablations

We separately ablate the training and inference design choices of Lafa. All training ablations use the 0.5B model with B=4 and a fixed 32-step whole-block decoder (Table 2). The inference ablations vary the initial noise scale, prefix-guidance scale, checkpoint weights, and use of self-conditioning (Appendix E.4).

Training and architecture. Bidirectional attention within the noisy block and self-conditioning are the most important design choices, improving the four-task average by 4.6 and 5.1 points, respectively; the self-conditioning gain is largest on HumanEval (+9.5). The learned EMA-histogram

Table 2: Design ablations for Lafa at 0.5B using a fixed 32-step whole-block decoder. Each section varies one component; bold denotes the best four-task macro-average.

Ablation	IFEval	GPQA	GSM8K	HumanEval	Avg.
Autoregressive references					
Qwen2.5-0.5B-Instruct	25.7	14.7	28.0	26.2	23.7
Qwen2.5-0.5B + Dolci-Instruct SFT	48.4	17.9	32.0	35.4	33.4
Time warper					
Uniform (no warping)	48.2	25.2	26.5	27.4	31.9
ELF (learned Beta CDF)	50.1	21.0	24.3	24.7	30.0
Loss EMA	51.8	24.3	30.4	31.7	**34.5**
Time sampling granularity					
Blockwise	49.4	19.2	30.4	28.7	31.9
Tokenwise	51.8	24.3	30.4	31.7	**34.5**
Embedding treatment					
Frozen	46.6	21.2	27.5	29.0	31.1
Norm preserving	51.8	24.3	30.4	31.7	**34.5**
Training objective					
CE	46.8	15.6	27.0	27.1	29.1
DMSE	45.1	24.8	16.8	27.1	28.4
CE + DMSE	49.7	24.6	28.3	29.3	33.0
CE + DMSE + AR	51.8	24.3	30.4	31.7	**34.5**
Attention masking within the noisy block					
Causal	45.8	21.4	26.1	26.1	29.9
Bidirectional	51.8	24.3	30.4	31.7	**34.5**
Self-conditioning					
Off	47.5	22.5	25.2	22.2	29.4
On	51.8	24.3	30.4	31.7	**34.5**

time sampler, tokenwise time sampling and norm-preserving embedding updates each improve the average by $2-3\%$ over their simpler counterparts. The denoising objectives are complementary: CE or DMSE alone reaches at most 29.1, their combination reaches 33.0, and auxiliary AR objective further raises the average to 34.5 while preserving the NTP capability used for self-speculation.

Inference recipe. Self-conditioning raises the three-task average from 32.3 to 38.1 without incurring any additional inference step (Table 31). Prefix guidance (CFG) contributes another 3.8 points over no guidance; scales from 4 to 8 form a broad plateau (Table 30) and we use CFG$=8$ throughout the paper. EMA checkpoint weights provide a smaller 0.9-point average gain. Finally, low initial-noise scales provide better performance than the training-time noise scale of $4\times$ and 0.1-0.5 provide a good optimal range increasing the performance by 4.3% as shown in (Table 29). We believe that this acts like a robust predictor since denoising a lower noise level is an easier task for the model compared to the original trained higher noise level. We adopt 0.1 throughout the paper.

4 Conclusion

We introduced Lafa, a data-efficient method for converting pretrained autoregressive models into block-autoregressive flow language models while retaining their next-token prediction ability. Across 0.5B, 7B, and 8B parameters, the resulting models support iterative flow generation, few-step flow-map decoding, confidence decoding, and self-speculative decoding with a single shared backbone. At 7B, self-speculative decoding reaches 100.2 tokens/s, $2.54\times$ the throughput of matched AR decoding, at comparable accuracy, while flow map distillation substantially improves one- and two-step generation. Our results indicate that pretrained AR initialization, blockwise causal structure, learned tokenwise schedules, and retained prefix caching as practical ingredients for scaling continuous flow language models. Future work includes stronger post-training at larger scales, more efficient kernels for block verification, and improved distillation for large-block one-step generation.

AI USE STATEMENT

Generative AI tools were used to help with polishing writing, finding related literature, writing and debugging code, and verifying mathematical derivations. The authors are fully responsible for the content of the final manuscript.

ETHICS STATEMENT

Our work does not raise ethical concerns beyond those commonly associated with research on large language models, such as the potential for misuse, generation of inaccurate or biased content, and inherited biases from pretraining data. By releasing our code and checkpoints upon acceptance, we aim to support transparency and enable the community to independently examine and address these limitations.

REPRODUCIBILITY STATEMENT

Implementation details and experimental configurations are provided in Appendix C. Upon acceptance, we will publicly release our code and model checkpoints to enable reproduction of our results.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. URL https://jmlr.org/papers/v26/23-1605.html.

Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Marianne Arriola, Aaron Gokaslan, Justin 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. URL https://openreview.net/forum?id=tyEyYT267x.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. *arXiv preprint arXiv:2602.11401*, 2026.

Parikshit Bansal and Sujay Sanghavi. Token time continuous diffusion for language modeling. *arXiv preprint arXiv:2607.14106*, 2026.

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. Cobit: Language modeling with bitstream diffusion. *arXiv preprint arXiv:2605.07013*, 2026.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 37559–37595, 2025. doi: 10.52202/085713-1121.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

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.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Oscar Davis, Anastasiia Filippova, Pierre Ablin, Victor Turrisi, Amitis Shidani, Marco Cuturi, and Louis Béthune. Scaling categorical flow maps. *arXiv preprint arXiv:2605.07820*, 2026.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

Felix Draxler, Justus Will, Farrin Marouf Sofian, Theofanis Karaletsos, Sameer Singh, and Stephan Mandt. Parallel token prediction for language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AGJomYSrUG.

Antonio Franca and Alexander Tong. Hacking generative perplexity: Why unconditional text evaluation needs distributional metrics. In *ICML 2026 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2026. URL https://openreview.net/forum?id=f1gwcTDwh9.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: From autoregressive to diffusion language models, and beyond in speed, 2025. URL https://arxiv.org/abs/2512.14067.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, pp. 133345–133385, 2024. doi: 10.52202/079017-4239.

Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. Set block decoding is a language model inference accelerator, 2025. URL https://arxiv.org/abs/2509.04185.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=e2MCL6hObn.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous latent diffusion language model, 2026. URL https://arxiv.org/abs/2605.06548.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023. doi: 10.18653/v1/2023.acl-long.647. URL https://aclanthology.org/2023.acl-long.647/.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Jiaxiu Jiang, Jingjing Ren, Wenbo Li, Bo Wang, Haoze Sun, Yijun Yang, Jianhui Liu, Yanbing Zhang, Shenghe Zheng, Yuan Zhang, Haoyang Huang, Nan Duan, and Wangmeng Zuo. Textldm: Language modeling with continuous latent diffusion, 2026. URL https://arxiv.org/abs/2605.07748.

Ziqi Jin, Lei Wang, Ziwei Luo, and Aixin Sun. MARS: Enabling autoregressive models multi-token generation, 2026. URL https://arxiv.org/abs/2604.07023.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025. URL https://arxiv.org/abs/2506.17298.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022. doi: 10.52202/068431-0313.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, Shiqi Yang, B. Yang, Caifeng Shan, Ziwei Liu, and Chenyang Si. Aurora-lm: Autoencoding unified representation for continuous-latent diffusion language modeling, 2026. URL https://arxiv.org/abs/2608.02602.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848. PMLR, 2024. URL https://proceedings.mlr.press/v235/lou24a.html.

Divyat Mahajan, Sachin Goyal, Badr Youbi Idrissi, Mohammad Pezeshki, Ioannis Mitliagkas, David Lopez-Paz, and Kartik Ahuja. Beyond multi-token prediction: Pretraining LLMs with future summaries. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aeYIFVn4vb.

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 *Advances in Neural Information Processing Systems*, volume 38, pp. 56354–56392, 2025. doi: 10.52202/085713-1689.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00387.

Bingshuo Qian and Xiang Cheng. Learning when to denoise: Optimizing asynchronous schedules for latent diffusion. *arXiv preprint arXiv:2606.19662*, 2026.

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. *arXiv preprint arXiv:2311.12022*, 2023.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184, 2024. doi: 10.52202/079017-4135.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander M Rush, Thomas PIERROT, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i5MrJ6g5G1.

Chenze Shao, Darren Li, Fandong Meng, and Jie Zhou. Continuous autoregressive language models. *arXiv preprint arXiv:2510.27688*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 103131–103167, 2024. doi: 10.52202/079017-3277.

Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025a.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference, 2025b. URL https://arxiv.org/abs/2508.02193.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/c4127b9194fe8562c64dc0f5bf2c93bc-Paper.pdf.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, et al. EleutherAI/lm-evaluation-harness: v0.4.3, 07 2024. URL https://zenodo.org/records/12608602.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. Diffusiongemma technical report, 2026. URL https://arxiv.org/abs/2608.00146.

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, Hanting Chen, Xinghao Chen, and Yunhe Wang. From next-token to next-block: A principled adaptation path for diffusion LLMs. *arXiv preprint arXiv:2512.06776*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488, 2018. doi: 10.18653/v1/D18-1044. URL https://aclanthology.org/D18-1044/.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024. doi: 10.52202/079017-3018.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1NZ3DHF9nT.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Yelong Shen, Jian Jiao, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. AR-Diffusion: Auto-regressive diffusion model for text generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39957–39974, 2023. doi: 10.52202/075280-1737.

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, 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. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026.

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

Longxuan Yu, Yunshu Wu, Yu Fu, Siheng Xiong, Rob Brekelmans, Hui Liu, Yue Dong, and Greg Ver Steeg. Dsl-llada: Scaling continuous denoising to 8b masked diffusion lms. *arXiv preprint arXiv:2606.01024*, 2026.

Chi Zhang, Xixi Hu, Bo Liu, and Qiang Liu. Gumbel distillation for parallel text generation. *arXiv preprint arXiv:2603.22216*, 2026.

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.

Shaobin Zhuang, Yuang Ai, Jiaming Han, Xiaohui Li, Huaibo Huang, Xiangyu Yue, Xuefeng Hu, Kun Xu, Yali Wang, and Hao Chen. Bitlm: Unlocking multi-token language generation with bitwise continuous diffusion. *arXiv preprint arXiv:2605.11577*, 2026.

Appendix Contents

A Related work **17**

B Training-data efficiency **17**

C Implementation details **18**

C.1 Training hyperparameters 18

C.2 Per-token time reparameterization 18

C.3 Self-conditioning 19

C.4 Prefix guidance 19

C.5 Training and inference algorithms 19

D Embedded flow maps **21**

D.1 Two-time denoiser 21

D.2 Semigroup distillation 22

E Full results table **23**

E.1 Whole-block flow decoding 23

E.2 Self-speculative decoding 24

E.3 Confidence decoding 24

E.4 Pure-flow inference ablations 29

E.5 Decoding throughput 30

A Related work

Block autoregressive language models. To reduce the serialization of autoregressive inference, prior work has proposed factorizing the data distribution into blocks of tokens, generating each block in parallel. They employ multiple heads to decode each token (Gloeckle et al., 2024; DeepSeek-AI, 2024), an approach called multi-token prediction (MTP), or employ a block of draft tokens as input to predict the corresponding output tokens (Arriola et al., 2025; Gat et al., 2025; Jin et al., 2026). A key challenge is that these methods approximate the joint probability over the block as a product of independent per-token probabilities, which can degrade quality. To sidestep this issue, prior methods use an acceptance criterion to keep a prefix of the block whose generation is deemed accurate (Stern et al., 2018; DeepSeek-AI, 2024; Jin et al., 2026). Some recent methods learn a deterministic map from noise to tokens to avoid this approximation (Draxler et al., 2026; Zhang et al., 2026).

Discrete diffusion language models. Extending blockwise modeling, discrete diffusion language models predict multiple tokens in parallel, possibly over the entire sequence. They employ a discrete noising process such as masking or uniform randomization, and train models to reverse it by modeling the transition probabilities, so generation can be done via iterative ancestral sampling (Sahoo et al., 2024; Shi et al., 2024; Lou et al., 2024; Schiff et al., 2025). However, analogously to block autoregression, they employ a factorized approximation of the transition probabilities (Austin et al., 2021; Gat et al., 2024), limiting their quality in few-step generation. These models have been scaled to billions of parameters (Ye et al., 2025; Fu et al., 2025; Nie et al., 2025; Taïga et al., 2026) and many successful models perform block autoregression with prefix acceptance (Arriola et al., 2025).

Continuous flow language models. In contrast to discrete diffusion models, continuous flow language models perform parallel generation by learning to transport continuous noise to token embeddings (Li et al., 2022; Dieleman et al., 2022; Han et al., 2023; Wu et al., 2023). Since they operate in continuous space, these models can leverage techniques developed for continuous modalities for improved generation. In particular, the transport need not assume independent factorization of tokens, allowing for accurate few-step generation via flow maps. Recent work has shown that flow language models can perform competitively with discrete methods (Lee et al., 2026; Chen et al., 2026; Hu et al., 2026; Batzolis et al., 2026). However, current evidence is limited to relatively small scales (Shao et al., 2025; Guo et al., 2026; Jiang et al., 2026; Liang et al., 2026; Yang et al., 2026; Davis et al., 2026), outside a few recent exceptions (Zhuang et al., 2026; Yu et al., 2026).

B Training-data efficiency

To separate the cost of learning each generative formulation from inherited backbone pretraining, we compare the number of training tokens consumed after each method's stated initialization. LLaDA is trained from scratch, whereas Dream, Efficient-DLM, and Lafa initialize from released autoregressive checkpoints. Consequently, inherited Qwen pretraining tokens are not included for any converted model.

Table 3: **Training-token budgets at 7–8B scale.** LLaDA and Dream are lower bounds because their subsequent supervised fine-tuning data are reported in examples rather than tokens. Lafa is fine-tuned on Dolci-Instruct-SFT for five epochs ($\approx$ 9B tokens).

Method	Initialization	Reported training tokens	Relative to Lafa
	Published baselines		
LLaDA-8B (Nie et al., 2025)	From scratch	$\geq$ 2.3T	$\geq 256\times$
Dream-7B (Ye et al., 2025)	Qwen2.5-7B	$\geq$ 580B	$\geq 64\times$
Efficient-DLM-8B (Fu et al., 2025)	Qwen3-8B	500B	$56\times$
Lafa 7B/8B (Ours)	Qwen2.5-7B / Qwen3-8B	**9B**	$1\times$

C Implementation details

C.1 Training hyperparameters

Table 4 summarizes the block-flow teacher recipe used for the reported models. The $B = 4$ base runs use an effective batch of 384 sequences. Larger-block checkpoints use memory-matched microbatches and, where applicable, continue from the corresponding smaller-block checkpoint. Unless noted otherwise, all scales use the same loss, time-sampler, self-conditioning, and checkpoint-averaging settings.

Table 4: **Training and default flow-inference hyperparameters.** Sequence length and block size list the configurations represented in the reported results. "Per-layer" uses a distinct adaLN modulator in every transformer layer; "shared" reuses one modulator across all layers.

Setting	**Lafa 0.5B**	**Lafa 7B**	**Lafa 8B**
		Model and data	
Initialization	Qwen2.5-0.5B-Instruct	Qwen2.5-7B-Instruct	Qwen3-8B-Base
Training data		Dolci-Instruct-SFT	
Training epochs / tokens		5 / approximately 9B	
Reported block sizes B	$4, 8, 16$	$4, 8, 16$	4
Maximum sequence length	512–1024	512–1024	512
adaLN modulators	Per-layer	Shared across layers	Shared across layers
		Optimization	
Optimizer		AdamW	
Learning rate	1×10^{-4}	1×10^{-5}	1×10^{-5}
Schedule / warm-up		Constant / first 1% of steps	
Adam (β_1, β_2)		$(0.9, 0.999)$	
Weight decay / gradient clipping		0 / 1.0	
Base effective batch size		384 sequences	
Precision		bfloat16	
Distributed strategy	DDP	ZeRO-3	ZeRO-3
Random seed		42	
		Flow objective and architecture	
Training noise scale		4.0	
Loss weights (AR / CE / DMSE)		$1.0/1.0/1.0$	
Self-conditioning probability		0.5	
Embedding-norm projection		Enabled	
adaLN learning-rate multiplier		5	
Model-weight EMA decay		0.999 (enabled from step 500)	
		Learned time sampler	
Estimator		Per-position loss-EMA histogram	
Histogram bins M		200	
Histogram EMA decay β		0.99	
Update interval S / estimation draws		100 steps / 4	
Uniform-mixture weight		0.2	
Warp activation step		500	
		Default whole-block flow inference	
Initial noise scale / CFG scale		$0.1/8$	
Solver / stochastic churn		Euler / none	
Checkpoint weights		EMA	
Self-conditioning		True	

C.2 Per-token time reparameterization

Following the loss-aware time warping of Dieleman et al. (2022), applied separately to each block position, we set each schedule to the inverse cumulative distribution of its denoising loss. Let

$\ell_j(\tau) > 0$ be the expected denoising cross entropy of position j at tokenwise time τ, and define

$$F_j(\tau) := \frac{\int_0^\tau \ell_j(r)\,\mathrm{d}r}{\int_0^1 \ell_j(r)\,\mathrm{d}r}, \qquad \tau^j(t) := F_j^{-1}(t). \tag{15}$$

Then τ^j is strictly increasing from 0 to 1, and $\dot{\tau}^j \propto 1/\ell_j(\tau^j)$ is small where the loss is high, so both a uniform solver grid at inference and uniform t during training concentrate on the hardest noise levels.

Estimation. We keep $\mathbf{H} \in \mathbb{R}^{B\times M}$, an exponential moving average of the per-position cross entropy in M uniform time bins, and set each F_p to the piecewise-linear CDF of the normalized row $\mathbf{H}_{p,:}$ (Algorithm 2). Every S optimization steps, we run four gradient-free estimation passes with independently sampled uniform times for each noisy block. For every visited position–time bin, we update its mean loss with decay β; bins that are not visited retain their previous value. This independent uniform estimation prevents the learned sampling distribution from starving low-probability regions. We initialize $\mathbf{H}$ uniformly and use $M = 200$, $\beta = 0.99$, and $S = 100$. This is the *EMA histogram* estimator in Table 2; the remaining settings are summarized in Table 4.

C.3 Self-conditioning

During training, we apply self-conditioning with a fixed probability by obtaining an initial prediction without gradient and using its expected token embeddings to condition a second forward pass. Following Chen et al. (2026), we incorporate this prediction through additional projection layers,

$$\bar{\mathbf{x}}_1 = \mathsf{sg}\big(D_t(I_t;\mathbf{c})\mathbf{E}\big), \qquad \tilde{I}_t = I_t + I_t\mathbf{W}_a^\top + \bar{\mathbf{x}}_1\mathbf{W}_b^\top. \tag{16}$$

Here $\mathbf{W}_a, \mathbf{W}_b \in \mathbb{R}^{d\times d}$ are trainable matrices initialized to zero. We replace the noisy block input with $\tilde{I}_t$, retain the same time conditioning and clean prefix, and compute the training loss on the second pass. Otherwise, we train with a single pass on I_t. In practice, we use probability of 50%.

At flow inference, we use the denoiser prediction from the previous integration step for self-conditioning. In single-step update of Euler integration of (12), we replace $D_{t_i}(\hat{\mathbf{x}}_{t_i};\mathbf{c})$ with $\mathbf{q}_i$, defined by

$$\tilde{\mathbf{x}}_{t_i} = \hat{\mathbf{x}}_{t_i} + \hat{\mathbf{x}}_{t_i}\mathbf{W}_a^\top + (\mathbf{q}_{i-1}\mathbf{E})\mathbf{W}_b^\top, \qquad \mathbf{q}_i = D_{t_i}(\tilde{\mathbf{x}}_{t_i};\mathbf{c}). \tag{17}$$

The first step uses $\mathbf{q}_0 = D_0(\hat{\mathbf{x}}_0;\mathbf{c})$ without self-conditioning. The integration state remains $\hat{\mathbf{x}}_{t_i}$, and only the denoiser input is modified.

C.4 Prefix guidance

Prefix guidance strengthens conditioning on the clean prefix, analogously to history guidance in video diffusion (Song et al., 2025a). For flow integration, define the guided embedding prediction

$$m_t^g(\mathbf{x};\mathbf{c}) := D_t(\mathbf{x};\varnothing)\mathbf{E} + g\left(D_t(\mathbf{x};\mathbf{c})\mathbf{E} - D_t(\mathbf{x};\varnothing)\mathbf{E}\right), \qquad g \geq 1, \tag{18}$$

and replace $\mathbf{p}_i\mathbf{E}$ in (12) with $m_{t_i}^g(\hat{\mathbf{x}}_{t_i};\mathbf{c})$. The linear combination acts as an embedding prediction; its underlying coefficients need not form a categorical distribution. Guidance changes the velocity and hence the generated distribution. Here $g = 1$ recovers the conditional prediction and the experiments use $g = 8$ for whole-block flow decoding. Guidance requires conditional and prefix-free predictions at each step, increasing the model-forward count from K to $2K$ unless one prediction is otherwise available.

C.5 Training and inference algorithms

The procedures below give the complete block-flow training loop, the loss-histogram update, and Euler sampling. For compactness, define the residual self-conditioning transform

$$\mathrm{SC}(\mathbf{x},\mathbf{q}) = \mathbf{x} + \mathbf{x}W_a^\top + (\mathbf{q}\mathbf{E})W_b^\top, \qquad \mathrm{SC}(\mathbf{x},\varnothing) = \mathbf{x}.$$

The blockwise attention mask is the one illustrated in Figure 2: clean tokens attend causally, while each noisy block attends to its clean prefix and bidirectionally within the block.

Algorithm 1 Parallel block-flow training

Require: denoiser D_θ, AR readout A_θ, embeddings $\mathbf{E}$, schedules $F_{1:B}$, self-conditioning probability p_{sc}
Require: loss weights $\lambda_{\mathrm{AR}}, \lambda_{\mathrm{CE}}, \lambda_{\mathrm{MSE}}$
for each optimization step **do**
  Sample a tokenized sequence and retain a clean causal copy.
  Partition its supervised positions into target blocks $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(R)}$.
  for each target block r **do**
    Sample $u_r \sim \mathsf{U}[0,1]$ and set $\tau_r^j \leftarrow F_j^{-1}(u_r)$ for $j = 1, \dots, B$.
    Sample $\mathbf{x}_0^{(r)} \sim \mathsf{N}(\mu, \sigma^2 \mathbf{I})$.
    $\mathbf{x}_{\boldsymbol{\tau}}^{(r)} \leftarrow (1 - \boldsymbol{\tau}_r) \odot \mathbf{x}_0^{(r)} + \boldsymbol{\tau}_r \odot \mathbf{E}[\mathbf{y}^{(r)}]$.
  end for
  Pack the clean copy, noisy blocks, and their tokenwise times using the blockwise mask.
  if $s \sim \mathrm{Bernoulli}(p_{\mathrm{sc}})$ and $s = 1$ **then**
    Predict all noisy blocks once without gradient: $\mathbf{q}^{(r)} \leftarrow \mathsf{sg}\Big(D_\theta(\mathbf{x}_{\boldsymbol{\tau}}^{(r)}; \mathbf{c}^{(r)})\Big)$.
    Replace each noisy input by $\mathrm{SC}(\mathbf{x}_{\boldsymbol{\tau}}^{(r)}, \mathbf{q}^{(r)})$.
  end if
  Run one packed forward pass to obtain clean AR logits A_θ and block predictions $\mathbf{p}^{(1:R)}$.
  $\mathcal{L}_{\mathrm{AR}} \leftarrow \mathrm{CE}(A_\theta, \text{next clean tokens})$.
  $\mathcal{L} \leftarrow \lambda_{\mathrm{AR}} \mathcal{L}_{\mathrm{AR}} + \sum_r \Big[\lambda_{\mathrm{CE}}\, \mathrm{CE}(\mathbf{p}^{(r)}, \mathbf{y}^{(r)}) + \lambda_{\mathrm{MSE}} \|\mathbf{p}^{(r)}\mathbf{E} - \mathbf{E}[\mathbf{y}^{(r)}]\|_F^2\Big]$.
  Update θ and $\mathbf{E}$; project every $\mathbf{E}_v$ back to its pretrained norm.
end for

Algorithm 2 Updating the tokenwise time sampler

Require: loss table $\mathbf{H} \in \mathbb{R}^{B \times M}$, decay β, update interval S, estimation draws N_{est}
for every S optimization steps **do**
  Initialize empty loss accumulators for every position–time bin.
  for $n = 1, \dots, N_{\mathrm{est}}$ **do**
    Sample a training mini-batch.
    for each supervised block r **do**
      Sample a uniform estimation time $\tilde{t}_r \sim \mathsf{U}[0,1]$ and Gaussian noise $\mathbf{x}_0^{(r)}$.
      Corrupt the block: $\mathbf{x}_{\tilde{t}_r}^{(r)} \leftarrow (1 - \tilde{t}_r)\mathbf{x}_0^{(r)} + \tilde{t}_r \mathbf{E}[\mathbf{y}^{(r)}]$.
      Predict without gradient, $\mathbf{p}^{(r)} \leftarrow \mathsf{sg}(D_{\tilde{t}_r}(\mathbf{x}_{\tilde{t}_r}^{(r)}; \mathbf{c}^{(r)}))$, and record $\mathrm{CE}(\mathbf{p}^{(r),j}, y_j)$ in bin $(j, b(\tilde{t}_r))$.
    end for
  end for
  for each visited bin (j, m) **do**
    $H_{j,m} \leftarrow \beta H_{j,m} + (1 - \beta)\, \mathrm{MeanLoss}_{j,m}$.
  end for
  Normalize each row of $\mathbf{H}$ and update its piecewise-linear CDF F_j.
end for

Algorithm 3 Euler flow sampling with guidance

Require: denoiser D_θ, embeddings $\mathbf{E}$, prefix $\mathbf{c}$, schedules $F_{1:B}$, Euler steps K, CFG scale $g \geq 1$
Sample $\hat{\mathbf{x}}_0 \sim \mathsf{N}(\mu, \sigma^2 \mathbf{I})$.
Set $\tau_i^j \leftarrow F_j^{-1}(i/K)$ for $i = 0, \dots, K$ and $j = 1, \dots, B$.
$\mathbf{q}_{-1}^c, \mathbf{q}_{-1}^u \leftarrow \varnothing$.
for $i = 0, \dots, K-1$ **do**
  $\mathbf{q}_i^c \leftarrow D_{t_i}(\mathrm{SC}(\hat{\mathbf{x}}_i, \mathbf{q}_{i-1}^c); \mathbf{c})$.
  if $g > 1$ **then**
    $\mathbf{q}_i^u \leftarrow D_{t_i}(\mathrm{SC}(\hat{\mathbf{x}}_i, \mathbf{q}_{i-1}^u); \varnothing)$.
    $\mathbf{m}_i \leftarrow \mathbf{q}_i^u \mathbf{E} + g(\mathbf{q}_i^c \mathbf{E} - \mathbf{q}_i^u \mathbf{E})$.
  else
    $\mathbf{m}_i \leftarrow \mathbf{q}_i^c \mathbf{E}$.
  end if
  $\boldsymbol{\delta}_i \leftarrow (\boldsymbol{\tau}_{i+1} - \boldsymbol{\tau}_i) \oslash (1 - \boldsymbol{\tau}_i)$.
  $\hat{\mathbf{x}}_{i+1} \leftarrow \hat{\mathbf{x}}_i + \boldsymbol{\delta}_i \odot (\mathbf{m}_i - \hat{\mathbf{x}}_i)$.
end for
return $\arg\max_V \mathbf{q}_{K-1}^c \in V^B$.

Here K is the number of Euler steps reported as NFE in the experiments. Sampling uses K model forwards when $g = 1$ and $2K$ forwards when guidance uses a separate prefix-free prediction.

D EMBEDDED FLOW MAPS

In this section, we justify the embedded flow map used in Section 2.4: the two-time parameterization in (10) and the diagonal and off-diagonal terms of the distillation objective in (11). Appendix D.1 shows that the teacher's exact flow map admits this parameterization with a categorical two-time denoiser satisfying a semigroup condition, and Appendix D.2 shows that matching the teacher on the diagonal while enforcing this condition recovers the teacher flow.

Setup. Throughout, the prefix $\mathbf{c}$ is fixed, and the vocabulary embeddings $\mathbf{E} \in \mathbb{R}^{|V|\times d}$ and tokenwise schedules are held fixed. A block has positions $j = 1, \dots, B$ with clean embeddings $\mathbf{x}_1^j = \mathbf{E}_{\mathbf{y}^j}$. For each position j, let τ^j be continuously differentiable with $\dot\tau^j(t) > 0$ for $t < 1$, $\tau^j(0) = 0$, and $\tau^j(1) = 1$, and write $\tau_t^j := \tau^j(t)$. Let D_t^{T} be the frozen teacher denoiser. As in (6), it induces the velocity and flow map

$$\begin{aligned} b_t(\mathbf{x};\mathbf{c})^j &= \frac{\dot\tau^j(t)}{1-\tau_t^j}\big(D_t^{\mathrm{T}}(\mathbf{x};\mathbf{c})^j\mathbf{E} - \mathbf{x}^j\big), \\ \partial_t X_{s,t}(\mathbf{x};\mathbf{c}) &= b_t(X_{s,t}(\mathbf{x};\mathbf{c});\mathbf{c}), \qquad X_{s,s}(\mathbf{x};\mathbf{c}) = \mathbf{x}. \end{aligned} \tag{19}$$

We work on $0 \le s \le t < 1$ and assume unique ODE solutions and the continuity and differentiability used below. Expressions at the endpoint $t = 1$, which flow map inference uses, are understood as limits when they exist.

D.1 TWO-TIME DENOISER

Connection to average velocity. For $0 \le s < t < 1$, the average velocity of the exact map is $v_{s,t}(\mathbf{x};\mathbf{c}) := (X_{s,t}(\mathbf{x};\mathbf{c}) - \mathbf{x})/(t-s)$ (Boffi et al., 2025). To express transport between two times through a denoising prediction, we define a two-time denoiser $\delta_{s,t}^*(\mathbf{x};\mathbf{c}) \in (\Delta^{|V|-1})^B$ through its embedding-space mean,

$$\delta_{s,t}^*(\mathbf{x};\mathbf{c})^j\mathbf{E} = \mathbf{x}^j + \frac{(1-\tau_s^j)(t-s)}{\tau_t^j - \tau_s^j} v_{s,t}(\mathbf{x};\mathbf{c})^j. \tag{20}$$

Substituting the definition of $v_{s,t}$ and rearranging gives exactly the form of (10), with X and δ^* in place of their learned counterparts $\widehat{X}$ and δ. The embedding-space mean alone need not determine a unique categorical vector, since distinct categorical distributions can share the same mean embedding. The following proposition establishes that a categorical representative exists and identifies the conditions under which a learned denoiser recovers the flow.

Proposition 1 (Two-time denoiser characterization in embedding space)**.** *There exists a categorical two-time denoiser δ^* satisfying (20) that extends continuously to $\delta_{s,s}^* = D_s^{\mathrm{T}}$ and, for $s < u < t$, satisfies the semigroup condition*

$$\begin{aligned} \delta_{s,t}^*(\mathbf{x};\mathbf{c})^j &= \gamma^j \delta_{s,u}^*(\mathbf{x};\mathbf{c})^j + (1-\gamma^j)\delta_{u,t}^*(X_{s,u}(\mathbf{x};\mathbf{c});\mathbf{c})^j, \\ \gamma^j &= \frac{(1-\tau_t^j)(\tau_u^j - \tau_s^j)}{(1-\tau_u^j)(\tau_t^j - \tau_s^j)}. \end{aligned} \tag{21}$$

Conversely, any sufficiently regular categorical two-time denoiser δ satisfying the diagonal and semigroup conditions with its own induced map $\widehat{X}$ ((10)) generates the flow of b_t.

Proof. For each position j, the velocity in the setup gives

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(\frac{X_{s,r}(\mathbf{x};\mathbf{c})^j}{1-\tau_r^j}\right) = \frac{\dot\tau^j(r)}{(1-\tau_r^j)^2} D_r^{\mathrm{T}}(X_{s,r}(\mathbf{x};\mathbf{c});\mathbf{c})^j\mathbf{E}. \tag{22}$$

Construct

$$\delta^*_{s,t}(\mathbf{x};\mathbf{c})^j = \frac{1}{Z^j_{s,t}} \int_s^t \frac{\dot\tau^j(r)}{(1-\tau^j_r)^2}\, D^{\mathrm{T}}_r(X_{s,r}(\mathbf{x};\mathbf{c});\mathbf{c})^j \,\mathrm{d}r,$$
$$Z^j_{s,t} := \int_s^t \frac{\dot\tau^j(r)}{(1-\tau^j_r)^2}\,\mathrm{d}r = \frac{\tau^j_t - \tau^j_s}{(1-\tau^j_t)(1-\tau^j_s)}. \tag{23}$$

The weights are nonnegative and integrate to one, so $\delta^*_{s,t}$ is a convex average of categorical distributions, and it converges to D^{T}_s as $t \downarrow s$ under continuity of the teacher along the trajectory. Integrating (22) from s to t gives $X^j_{s,t} = \frac{1-\tau^j_t}{1-\tau^j_s}\mathbf{x}^j + \frac{\tau^j_t-\tau^j_s}{1-\tau^j_s}\delta^{*\,j}_{s,t}\mathbf{E}$, which is equivalent to (20). Splitting the integral at u and using the semigroup property of X gives (21), with weights γ^j and $1-\gamma^j$.

Conversely, for the map $\widehat{X}$ induced by a candidate δ through (10),

$$\widehat{X}_{s,s} = \mathrm{Id}, \qquad \partial_t \widehat{X}_{s,t}(\mathbf{x};\mathbf{c})^j\Big|_{t=s} = \frac{\dot\tau^j(s)}{1-\tau^j_s}\big(\delta_{s,s}(\mathbf{x};\mathbf{c})^j\mathbf{E} - \mathbf{x}^j\big) = b_s(\mathbf{x};\mathbf{c})^j. \tag{24}$$

The semigroup condition similarly gives $\widehat{X}_{s,t+h} = \widehat{X}_{t,t+h} \circ \widehat{X}_{s,t}$. Differentiating at $h = 0$ yields $\partial_t \widehat{X}_{s,t} = b_t \circ \widehat{X}_{s,t}$, and ODE uniqueness implies $\widehat{X} = X$. □

Since the embedding-space mean does not determine the categorical vector uniquely, the diagonal condition is imposed on the categorical prediction itself rather than on its mean embedding; this is the role of the diagonal KL term in (11).

D.2 Semigroup distillation

Proposition 1 turns flow map learning into matching the teacher on the diagonal and enforcing the semigroup condition between distinct times. We implement both requirements with categorical KL losses, evaluating the intermediate state with the student's own induced map.

Composition target. For $0 \le s < u < t$, writing $\mathbf{x}_u := \widehat{X}_{s,u}(\mathbf{x}_s;\mathbf{c})$, the off-diagonal target in (11) composes two shorter student transitions (Lee et al., 2026),

$$\bar\delta_{s,t}(\mathbf{x}_s;\mathbf{c})^j := \mathsf{sg}\Big[\gamma^j\,\delta_{s,u}(\mathbf{x}_s;\mathbf{c})^j + (1-\gamma^j)\,\delta_{u,t}(\mathbf{x}_u;\mathbf{c})^j\Big], \tag{25}$$

with γ^j as in (21). Here sg stops gradients through the entire target, including the intermediate state. Monotonicity of the schedules gives $0 \le \gamma^j \le 1$, so the target is categorical. Substituting (10) into the composition gives

$$\widehat{X}_{u,t}(\widehat{X}_{s,u}(\mathbf{x}_s;\mathbf{c});\mathbf{c})^j = \frac{1-\tau^j_t}{1-\tau^j_s}\mathbf{x}^j_s + \frac{\tau^j_t-\tau^j_s}{1-\tau^j_s}\Big[\gamma^j\,\delta_{s,u}(\mathbf{x}_s;\mathbf{c})^j + (1-\gamma^j)\,\delta_{u,t}(\mathbf{x}_u;\mathbf{c})^j\Big]\mathbf{E}, \tag{26}$$

so matching $\delta_{s,t}$ to $\bar\delta_{s,t}$ enforces consistency with the composed transition in embedding space.

Proposition 2 (Semigroup distillation in embedding space)**.** *Let δ be a categorical student with induced map $\widehat{X}$ as in (10). Assume the regularity of Proposition 1 and full support of the joint sampling distributions over the indicated times and states. Over the unrestricted class of such denoisers, the population objective*

$$\mathcal{L}(\delta) = \mathbb{E}_{t,I_t}\sum_{j=1}^{B}\mathrm{KL}\left(D^{\mathrm{T}}_t(I_t;\mathbf{c})^j \,\|\, \delta_{t,t}(I_t;\mathbf{c})^j\right)$$
$$+\,\mathbb{E}_{s,u,t,I_s}\sum_{j=1}^{B}\mathrm{KL}\left(\bar\delta_{s,t}(I_s;\mathbf{c})^j \,\|\, \delta_{s,t}(I_s;\mathbf{c})^j\right), \tag{27}$$

with $0 \le s < u < t < 1$, attains minimum zero, and every zero-loss solution induces X.

Proof. The denoiser δ^* constructed in Proposition 1 makes both KL terms zero. Conversely, non-negativity of KL implies that every zero-loss solution satisfies the diagonal and semigroup conditions almost surely. Full support and continuity extend these conditions to the stated domains. By Proposition 1, its induced map equals X. □

These two terms are the diagonal and off-diagonal terms of (11); the autoregressive term there additionally preserves the teacher's next-token readout and does not affect the flow map. The guarantee is stated for $t < 1$; the target time $t = 1$ used at inference is covered as the limit $t \to 1$ when it exists.

E Full results table

E.1 Whole-block flow decoding

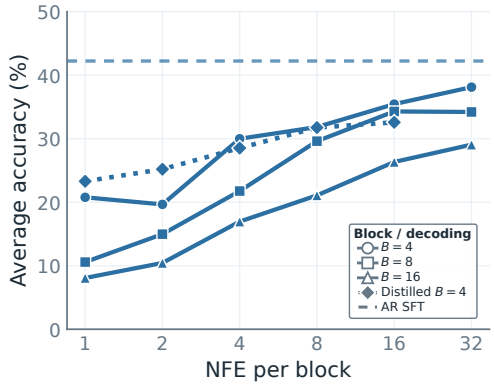


Figure 8: Lafa 0.5B whole-block flow-decoding frontier. Curves average GSM8K, HumanEval, and IFEval; plotting conventions follow Figure 5. Complete per-task values, including NFE 64 and 128, appear below.

Table 5: **Whole-block flow-decoding results.** All rows use whole-block Euler flow decoding with noise scale 0.1, EMA weights, and lagged self-conditioning. Avg. is the macro-average over IFEval, GSM8K, and HumanEval. The best average within each model scale and block size is bold.

	B=4				B=8				B=16			
NFE	**IFEval**	**GSM8K**	**HumanEval**	**Avg.**	**IFEval**	**GSM8K**	**HumanEval**	**Avg.**	**IFEval**	**GSM8K**	**HumanEval**	**Avg.**
Lafa 0.5B												
1	32.5	16.4	13.4	20.8	23.3	2.4	6.1	10.6	20.3	0.8	3.0	8.1
2	35.1	14.7	9.1	19.7	23.5	9.2	12.2	15.0	23.5	1.7	6.1	10.4
4	43.6	25.0	21.3	30.0	29.0	17.4	18.9	21.8	27.0	11.7	12.2	17.0
8	46.6	27.0	22.0	31.8	38.4	24.8	25.6	29.6	30.9	18.3	14.0	21.1
16	50.1	28.8	27.4	35.4	46.4	28.4	28.0	34.3	37.2	24.8	17.1	26.3
32	52.9	30.4	31.1	38.1	45.5	29.1	28.0	34.2	39.9	25.9	21.3	29.1
64	53.0	29.6	32.9	**38.5**	47.1	28.4	28.0	34.5	42.3	28.1	21.3	30.6
128	49.4	30.1	31.7	37.1	49.0	28.7	28.7	**35.5**	44.2	28.1	20.7	**31.0**
Lafa 7B												
1	44.7	53.9	32.3	43.7	34.8	20.1	22.0	25.6	29.8	2.4	10.4	14.2
2	43.8	60.0	34.8	46.2	42.1	47.4	41.5	43.7	36.2	18.7	21.3	25.4
4	60.1	69.5	63.4	64.3	55.8	64.0	57.9	59.2	44.7	47.5	40.2	44.1
8	69.3	73.0	71.3	71.2	64.5	64.8	64.0	64.5	56.9	60.2	54.9	57.3
16	69.5	71.9	75.0	72.1	64.5	71.6	70.7	68.9	64.9	65.6	65.2	65.2
32	73.0	73.8	78.7	75.2	71.0	72.3	72.0	**71.7**	64.1	66.9	68.3	66.5
64	74.3	74.6	78.7	**75.9**	68.8	71.7	72.6	71.0	66.5	67.5	66.5	66.8
128	73.6	75.6	78.0	75.7	69.3	71.0	72.6	70.9	65.2	68.7	67.7	**67.2**

Table 6: **Additional B=4 flow-decoding results.**

NFE	IFEval	GSM8K	HumanEval	Avg.
Lafa 0.5B, distilled				
1	33.6	19.2	17.1	23.3
2	36.2	21.1	18.3	25.2
4	39.4	24.9	21.3	28.5
8	40.3	28.1	26.8	31.8
16	43.3	28.9	25.6	**32.6**
Lafa 7B, distilled				
1	42.7	54.2	50.0	49.0
2	55.1	66.4	68.3	63.3
4	64.7	69.7	73.8	69.4
8	68.2	70.9	73.8	**71.0**
Lafa 8B				
1	34.2	62.6	37.8	44.9
2	47.5	73.7	61.6	60.9
4	56.6	77.3	75.6	69.8
8	59.5	78.2	79.3	**72.3**
16	59.7	77.0	75.6	70.8

E.2 Self-speculative decoding

Tokens per forward. Across evaluation examples i, we report

$$\frac{T}{F} = \frac{\sum_i u_i}{\sum_i f_i}, \tag{28}$$

where u_i is the decoded token count and f_i is the counted forward-pass total. For Lafa, $(u_i, f_i) = (n_i, 1+r_i)$, where n_i is the number of generated completion tokens and r_i counts fused verification-and-redraft calls after the initial flow draft. For MARS, MTP, DS-MTP, BD3LM, and PTP, $u_i = n_i$ and f_i is the measured number of decoding forwards. For LLaDA and Dream, $(u_i, f_i) = (a_i, s_i)$ uses allocated completion positions and denoising forwards. For Efficient-DLM, $(u_i, f_i) = (a_i, 1 + r_i)$ uses allocated completion positions and the landed exact-run acceptance rounds under the same logical fused verification-and-redraft accounting as Lafa; prompt prefill and cache-only refreshes are excluded. For Fast-dLLM-v2, $u_i = n_i$ and f_i includes its top-level denoising, prefill, and cache-update calls. AR SFT has $TPF = 1$. The Avg. TPF entry is the arithmetic mean of the three task-level ratios.

E.3 Confidence decoding

We report confidence-decoding results separately for each method. For Lafa, we report raw model-forward counts. We use the common definition in Equation 28. For Lafa, MARS, MTP, DS-MTP, BD3LM, and PTP, u_i is the number of emitted completion tokens and f_i is the measured number of top-level decoding forwards. One MTP invocation, including its shared trunk and all prediction heads, counts as one forward. LLaDA, Dream, Fast-dLLM-v2 and Efficient-DLM use allocated completion positions for u_i and denoising calls for f_i. The Avg. TPF entry is the arithmetic mean of the GSM8K, HumanEval, and IFEval task-level ratios. In the compact tables below, each task cell reports score/TPF.

Table 7: Self-speculative decoding and matched parallel baselines. **Span** gives the model's native parallel span; "canvas" denotes a full completion canvas, and Fast-dLLM lists global/sub-block span. "Self-spec." denotes draft-and-verify throughout, but the verifier follows each method: causal block predictors use greedy NTP verification, MARS uses its released batched verifier, Efficient-DLM uses its exact verifier, and Fast-dLLM, LLaDA, and Dream verify against their deterministic one-position trajectories. Lafa uses a flow draft with NTP verification. All rows are without SC. TPF is estimated from the acceptance traces under the same logical fused-verifier accounting: one initial draft per prompt and one fused verification-and-redraft forward per round.

Method	**Span**	**Decoder**	**GSM8K**		**HumanEval**		**IFEval**		**Avg.**	
			Acc.	TPF	P@1	TPF	Acc.	TPF	Acc.	TPF
0.5B scale										
Qwen-2.5 0.5B SFT	1	Greedy NTP	32.0	1.00	35.4	1.00	48.4	1.00	42.2	1.00
MARS (Jin et al., 2026)	4	Self-spec.	32.8	2.07	32.3	1.75	51.0	1.71	38.7	1.84
MTP (Gloeckle et al., 2024)	4	Self-spec.	27.1	2.90	32.3	2.80	50.6	2.71	36.7	2.80
DS-MTP (DeepSeek-AI, 2024)	2	Self-spec.	35.5	1.77	36.0	1.74	53.2	1.58	41.6	1.70
BD3LM (Arriola et al., 2025)	4	Self-spec.	31.2	1.46	28.0	1.51	47.5	1.48	35.6	1.48
SBD (Gat et al., 2025)	4	Entropy set decoding	26.7	2.71	31.1	2.75	56.4	2.24	38.1	2.57
PTP (Draxler et al., 2026)	4	Self-spec.	35.3	2.26	**42.7**	2.08	42.3	1.72	40.1	2.02
Lafa-0.5B	4	Flow draft	36.1	2.73	36.6	2.85	55.1	2.22	42.6	2.60
Lafa-0.5B	8	Flow draft	**37.8**	3.57	39.0	2.95	**55.3**	2.74	**44.0**	3.08
Lafa-0.5B	16	Flow draft	36.7	3.59	39.6	3.32	54.2	3.06	43.5	3.32
7B scale										
Qwen-2.5 7B SFT	1	Greedy NTP	69.1	1.00	78.7	1.00	73.8	1.00	73.9	1.00
MTP (Gloeckle et al., 2024)	4	Self-spec.	60.5	2.97	73.8	3.14	73.2	1.92	69.2	2.68
DS-MTP (DeepSeek-AI, 2024)	2	Self-spec.	70.3	1.74	80.5	1.87	78.0	1.45	76.3	1.69
Fast-dLLM v2 (Wu et al., 2026)	32/8	Self-spec.	**80.4**	2.45	63.4	1.84	62.8	1.87	68.9	2.05
Dream (Ye et al., 2025)	Canvas	Self-spec.	38.4	3.46	23.8	3.77	18.9	4.22	27.0	3.82
MARS-7B (Jin et al., 2026)	4	Self-spec.	74.0	2.76	**81.7**	2.88	77.8	1.91	**77.8**	2.51
MARS-7B	8	Self-spec.	71.3	3.49	81.1	3.45	76.3	2.02	76.2	2.99
MARS-7B	16	Self-spec.	75.4	3.71	77.4	3.68	75.8	1.99	76.2	3.13
Lafa-7B	4	Flow draft	73.8	2.77	76.8	2.84	**78.9**	1.73	76.5	2.45
Lafa-7B	8	Flow draft	75.3	4.79	76.2	4.55	77.5	2.79	76.3	4.05
Lafa-7B	16	Flow draft	75.0	5.50	73.8	5.24	74.9	2.74	74.5	4.49
8B scale										
Qwen-3 8B SFT	1	Greedy NTP	62.9	1.00	82.3	1.00	38.1	1.00	61.1	1.00
LLaDA (Nie et al., 2025)	Canvas	Self-spec.	47.2	2.73	23.8	2.47	36.8	3.21	35.9	2.80
Efficient-DLM-8B (Fu et al., 2025)	4	Self-spec.	85.4	2.92	66.5	2.99	57.1	2.60	69.7	2.83
Efficient-DLM-8B	32	Self-spec.	87.4	2.87	68.9	2.98	58.8	2.36	71.7	2.74
Lafa-8B	4	Flow draft	77.8	2.65	79.9	2.90	62.3	2.45	73.3	2.67

E.3.1 QWEN3-8B CONFIDENCE DECODING

Table 8: Efficient-DLM-8B, B=4 (Fu et al., 2025)

Setting	**GSM8K**	**HumanEval**	**IFEval**	**Avg.**
$\gamma = 1.00$	85.4/1.00	66.5/1.00	57.3/1.00	69.7/1.00
$\gamma = 0.99$	85.4/1.66	66.5/1.85	57.3/1.46	69.7/1.66
$\gamma = 0.95$	85.6/1.87	66.5/2.07	57.5/1.56	69.8/1.83
$\gamma = 0.90$	85.7/2.02	66.5/2.20	57.9/1.62	70.0/1.95
$\gamma = 0.80$	86.2/2.22	66.5/2.38	58.4/1.74	70.4/2.11
$\gamma = 0.70$	85.3/2.41	66.5/2.56	58.2/1.86	70.0/2.28
$\gamma = 0.60$	84.7/2.60	61.0/2.75	56.6/2.03	67.4/2.46
$\gamma = 0.50$	80.2/2.84	48.8/2.97	57.9/2.22	62.3/2.68
$\gamma = 0.40$	74.1/3.06	43.9/3.27	55.8/2.62	58.0/2.98
$\gamma = 0.30$	68.5/3.31	32.3/3.47	51.2/3.03	50.7/3.27
$\gamma = 0.25$	64.6/3.45	26.2/3.61	46.4/3.32	45.7/3.46
$\gamma = 0.23$	62.7/3.50	25.6/3.67	43.3/3.39	43.9/3.52
$\gamma = 0.22$	60.6/3.53	25.6/3.69	44.4/3.45	43.5/3.56
$\gamma = 0.20$	59.5/3.58	25.0/3.72	39.0/3.56	41.2/3.62
$\gamma = 0.10$	42.0/3.91	17.1/3.95	32.0/3.93	30.4/3.93
$\gamma = 0.00$	37.9/4.00	12.8/4.00	28.7/4.00	26.5/4.00

Table 9: Efficient-DLM-8B, B=32 (Fu et al., 2025)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 1.00$	87.4/1.00	68.9/1.00	58.4/1.00	71.6/1.00
$\gamma = 0.99$	87.3/2.01	68.9/2.19	58.4/1.50	71.6/1.90
$\gamma = 0.95$	87.3/2.44	68.9/2.66	58.8/1.64	71.7/2.25
$\gamma = 0.90$	87.0/2.78	68.3/3.00	59.3/1.75	71.6/2.51
$\gamma = 0.80$	87.0/3.32	66.5/3.49	58.6/1.91	70.7/2.91
$\gamma = 0.70$	85.2/3.87	62.8/4.17	59.5/2.15	69.2/3.40
$\gamma = 0.60$	82.1/4.47	57.3/4.88	57.3/2.63	65.6/3.99
$\gamma = 0.50$	72.6/5.21	42.1/6.13	54.2/3.27	56.3/4.87
$\gamma = 0.40$	58.0/6.26	29.3/8.01	49.9/4.71	45.7/6.33
$\gamma = 0.30$	39.4/8.93	21.3/10.70	38.8/8.20	33.2/9.28
$\gamma = 0.25$	25.9/11.63	8.5/12.66	33.5/11.25	22.6/11.85
$\gamma = 0.23$	22.0/13.51	10.4/13.85	31.1/13.40	21.1/13.59
$\gamma = 0.22$	18.7/14.35	14.0/14.47	32.9/14.58	21.9/14.47
$\gamma = 0.20$	14.0/16.92	9.8/15.95	29.6/17.01	17.8/16.63
$\gamma = 0.10$	1.2/28.63	2.4/27.01	23.5/28.59	9.0/28.08
$\gamma = 0.00$	0.5/32.00	1.2/32.00	19.2/32.00	7.0/32.00

Table 10: Lafa-8B, B=4

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	77.8/1.00	80.5/1.00	61.4/1.00	73.2/1.00
$\gamma = 0.99$	77.5/1.42	80.5/1.32	58.8/1.43	72.3/1.39
$\gamma = 0.95$	76.4/1.62	79.3/1.49	60.1/1.62	71.9/1.57
$\gamma = 0.90$	76.5/1.73	80.5/1.61	59.5/1.66	72.2/1.66
$\gamma = 0.80$	75.2/1.89	78.7/1.79	61.0/1.76	71.6/1.81
$\gamma = 0.70$	72.2/2.02	77.4/1.94	58.4/1.84	69.3/1.93
$\gamma = 0.60$	68.8/2.16	68.9/2.06	59.9/1.96	65.9/2.06
$\gamma = 0.50$	64.4/2.34	57.9/2.38	59.0/2.13	60.4/2.28
$\gamma = 0.40$	60.2/2.56	48.2/2.73	56.2/2.28	54.9/2.52
$\gamma = 0.30$	54.5/2.90	35.4/2.91	52.9/2.54	47.6/2.78
$\gamma = 0.20$	49.1/3.28	25.6/3.35	48.4/3.01	41.1/3.21
$\gamma = 0.10$	42.1/3.83	22.6/3.81	33.5/3.79	32.7/3.81
$\gamma = 0.00$	42.6/4.00	23.2/4.00	28.1/4.00	31.3/4.00

E.3.2 0.5B CONFIDENCE-DECODING SWEEPS

Table 11: Lafa, B=4

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	36.8/1.00	35.4/1.00	55.5/1.00	42.6/1.00
$\gamma = 0.99$	36.3/1.37	40.9/1.43	50.6/1.62	42.6/1.47
$\gamma = 0.95$	35.3/1.62	43.3/1.65	51.6/1.85	43.4/1.71
$\gamma = 0.90$	34.5/1.77	40.9/1.80	51.6/1.94	42.3/1.84
$\gamma = 0.80$	31.7/1.94	39.6/1.95	50.5/2.07	40.6/1.99
$\gamma = 0.70$	30.8/2.15	31.7/2.10	49.5/2.00	37.3/2.08
$\gamma = 0.60$	25.1/2.34	30.5/2.18	49.2/2.05	34.9/2.19
$\gamma = 0.50$	20.1/2.63	19.5/2.28	49.0/2.11	29.5/2.34
$\gamma = 0.40$	16.6/2.90	16.5/2.68	47.1/2.26	26.7/2.61
$\gamma = 0.30$	11.2/3.33	12.2/3.02	40.7/2.79	21.4/3.04
$\gamma = 0.20$	7.3/3.75	5.5/3.44	35.7/3.37	16.1/3.52
$\gamma = 0.10$	5.3/3.98	4.9/3.95	27.4/3.91	12.5/3.95

Table 12: Lafa, B=8

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	35.98/1.00	40.03/1.00	53.60/1.00	43.2/1.00
$\gamma = 0.99$	37.1/1.49	43.3/1.40	44.0/1.91	41.5/1.60
$\gamma = 0.95$	35.6/1.80	43.3/1.64	43.3/2.22	40.7/1.89
$\gamma = 0.90$	34.7/2.00	42.1/1.89	42.5/2.37	39.8/2.08
$\gamma = 0.80$	30.8/2.30	36.6/2.07	42.7/2.48	36.7/2.28
$\gamma = 0.70$	26.7/2.54	33.5/2.37	42.7/2.82	34.3/2.58
$\gamma = 0.60$	21.8/2.82	29.9/2.46	40.9/2.77	30.9/2.68
$\gamma = 0.50$	16.3/3.07	18.3/2.81	41.4/3.17	25.3/3.02
$\gamma = 0.40$	8.7/4.01	12.8/3.18	37.5/3.55	19.7/3.58
$\gamma = 0.30$	4.1/5.13	6.7/4.14	32.9/4.38	14.6/4.55
$\gamma = 0.20$	1.2/6.95	2.4/6.09	27.9/5.89	10.5/6.31
$\gamma = 0.10$	1.1/7.91	1.2/7.79	22.0/7.56	8.1/7.75
$\gamma = 0.00$	1.1/8.00	1.2/8.00	19.4/8.00	7.2/8.00

Table 13: Lafa, B=16

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	36.1/1.00	38.4/1.00	56.4/1.00	43.6/1.00
$\gamma = 0.99$	28.7/1.56	31.7/1.57	42.0/1.96	34.1/1.70
$\gamma = 0.95$	29.2/2.07	31.7/1.96	41.6/2.36	34.2/2.13
$\gamma = 0.90$	28.5/2.41	32.3/2.10	43.6/2.49	34.8/2.34
$\gamma = 0.80$	26.0/2.66	28.0/2.32	42.5/2.76	32.2/2.58
$\gamma = 0.70$	24.0/3.11	26.2/2.64	40.5/2.80	30.2/2.85
$\gamma = 0.60$	20.1/3.40	22.6/2.88	39.4/2.97	27.3/3.08
$\gamma = 0.50$	11.7/4.09	15.9/3.32	34.9/3.49	20.8/3.64
$\gamma = 0.40$	6.4/4.72	7.9/3.94	34.2/3.88	16.2/4.18
$\gamma = 0.30$	3.1/7.15	4.9/5.40	29.6/5.43	12.5/5.99
$\gamma = 0.20$	1.4/12.28	1.2/7.86	27.4/9.20	10.0/9.78
$\gamma = 0.10$	0.6/15.60	1.2/14.75	20.5/13.80	7.4/14.72
$\gamma = 0.00$	0.5/16.00	1.2/16.00	20.0/16.00	7.2/16.00

Table 14: SBD (Gat et al., 2025)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 1.00$	27.2/1.00	30.8/1.00	55.3/1.00	37.7/1.00
$\gamma = 0.99$	24.3/1.22	34.1/1.20	52.9/1.26	37.1/1.23
$\gamma = 0.95$	24.0/1.40	34.1/1.35	52.9/1.40	37.0/1.38
$\gamma = 0.90$	23.9/1.52	35.4/1.45	52.5/1.47	37.2/1.48
$\gamma = 0.80$	23.1/1.71	34.1/1.62	51.6/1.54	36.3/1.62
$\gamma = 0.70$	21.9/1.87	32.9/1.75	50.1/1.62	35.0/1.75
$\gamma = 0.60$	19.9/2.01	30.5/1.92	49.4/1.67	33.3/1.87
$\gamma = 0.50$	15.3/2.23	23.2/2.11	49.4/1.72	29.3/2.02
$\gamma = 0.40$	12.4/2.53	17.1/2.40	47.9/1.87	25.8/2.26
$\gamma = 0.30$	9.8/2.99	12.8/2.68	46.6/2.10	23.1/2.59
$\gamma = 0.20$	6.4/3.58	7.9/3.35	40.9/2.75	18.4/3.23
$\gamma = 0.10$	5.5/3.96	3.0/3.91	31.8/3.79	13.5/3.89
$\gamma = 0.00$	5.3/3.99	3.0/4.00	28.8/4.00	12.4/3.99

Table 15: MARS (Jin et al., 2026)

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	32.7/1.00	31.7/1.00	51.0/1.00	38.5/1.00
$\gamma = 0.99$	32.2/1.32	41.5/1.16	52.3/1.44	42.0/1.31
$\gamma = 0.95$	31.2/1.55	41.5/1.31	51.9/1.63	41.5/1.50
$\gamma = 0.90$	30.6/1.72	40.9/1.43	52.3/1.71	41.3/1.62
$\gamma = 0.80$	28.4/1.91	34.1/1.58	50.6/1.80	37.7/1.76
$\gamma = 0.70$	25.6/2.10	22.0/1.76	50.6/1.93	32.7/1.93
$\gamma = 0.60$	22.9/2.29	16.5/2.02	48.8/2.06	29.4/2.12
$\gamma = 0.50$	17.9/2.56	12.8/2.30	46.4/2.24	25.7/2.37
$\gamma = 0.40$	13.6/2.91	13.4/2.76	42.1/2.46	23.0/2.71
$\gamma = 0.30$	10.3/3.35	6.7/3.66	41.0/2.95	19.3/3.32
$\gamma = 0.20$	6.1/3.79	6.7/3.66	33.1/3.49	15.3/3.65
$\gamma = 0.10$	4.9/3.99	6.1/3.94	27.0/3.90	12.7/3.94
$\gamma = 0.00$	4.7/4.00	6.1/4.00	25.1/4.00	12.0/4.00

Table 16: Multi-head MTP (Gloeckle et al., 2024)

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	27.1/1.00	32.3/1.00	50.6/1.00	36.7/1.00
$\gamma = 0.99$	27.6/1.39	34.1/1.23	50.8/1.42	37.5/1.35
$\gamma = 0.95$	27.1/1.65	34.1/1.43	51.4/1.72	37.5/1.60
$\gamma = 0.90$	26.5/2.02	34.1/1.43	51.2/1.85	37.3/1.77
$\gamma = 0.80$	25.8/2.02	31.7/1.74	51.0/1.96	36.2/1.91
$\gamma = 0.70$	24.0/2.21	27.4/1.74	51.0/2.07	34.1/2.01
$\gamma = 0.60$	21.3/2.39	25.0/2.05	50.1/2.20	32.1/2.21
$\gamma = 0.50$	18.3/2.63	17.1/2.05	47.1/2.34	27.5/2.34
$\gamma = 0.40$	14.9/2.89	9.8/2.58	47.0/2.58	23.9/2.68
$\gamma = 0.30$	11.1/3.14	9.1/2.69	41.0/2.51	20.4/2.78
$\gamma = 0.20$	8.6/3.53	8.5/3.32	34.8/3.11	17.3/3.32
$\gamma = 0.10$	7.4/3.95	6.1/3.90	33.8/3.90	15.8/3.92
$\gamma = 0.00$	7.7/3.98	5.5/4.00	28.7/4.00	14.0/3.99

Table 17: DeepSeek-style MTP, D=1 (DeepSeek-AI, 2024)

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	35.5/1.00	36.0/1.00	53.2/1.00	41.6/1.00
$\gamma = 0.99$	35.3/1.35	37.8/1.20	52.7/1.17	41.9/1.24
$\gamma = 0.95$	34.6/1.50	37.8/1.33	52.3/1.27	41.6/1.37
$\gamma = 0.90$	34.3/1.58	36.6/1.41	51.9/1.32	40.9/1.44
$\gamma = 0.80$	35.8/1.66	36.6/1.53	51.0/1.38	41.1/1.52
$\gamma = 0.70$	32.8/1.73	31.7/1.62	50.1/1.44	38.2/1.60
$\gamma = 0.60$	33.1/1.80	32.9/1.71	51.0/1.45	39.0/1.65
$\gamma = 0.50$	30.3/1.86	29.9/1.80	51.0/1.56	37.0/1.74
$\gamma = 0.40$	28.4/1.92	28.7/1.86	49.5/1.63	35.5/1.80
$\gamma = 0.30$	27.4/1.95	27.4/1.90	48.6/1.75	34.5/1.87
$\gamma = 0.20$	26.2/1.98	23.2/1.98	51.2/1.80	33.5/1.92
$\gamma = 0.10$	24.8/1.99	24.4/1.99	49.0/1.93	32.7/1.97
$\gamma = 0.00$	26.0/1.99	23.8/2.00	47.9/2.00	32.6/2.00

Table 18: BD3-LM (Arriola et al., 2025)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	35.5/1.23	36.0/1.22	49.5/1.40	40.3/1.28
$\gamma = 0.95$	35.8/1.41	36.0/1.42	49.2/1.60	40.3/1.48
$\gamma = 0.90$	34.6/1.53	36.0/1.56	48.8/1.69	39.8/1.59
$\gamma = 0.80$	32.3/1.69	34.1/1.68	48.4/1.79	38.3/1.72
$\gamma = 0.70$	30.3/1.83	32.3/1.56	46.2/1.88	36.3/1.76
$\gamma = 0.60$	25.9/2.01	26.2/2.03	47.0/1.95	33.0/2.00
$\gamma = 0.50$	21.9/2.19	20.1/1.88	46.2/2.05	29.4/2.04
$\gamma = 0.40$	18.2/2.49	15.9/2.85	44.2/2.26	26.1/2.53
$\gamma = 0.30$	12.9/2.95	10.4/2.85	40.9/2.58	21.4/2.79
$\gamma = 0.20$	7.2/3.59	6.1/3.45	34.2/3.20	15.8/3.41
$\gamma = 0.10$	4.6/3.95	4.9/3.89	30.1/3.83	13.2/3.89
$\gamma = 0.00$	5.1/4.00	3.7/4.00	27.5/4.00	12.1/4.00

Table 19: PTP (Draxler et al., 2026)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.90$	25.9/1.73	28.7/1.46	41.6/1.19	32.1/1.46
$\gamma = 0.80$	25.1/1.94	29.9/1.66	40.9/1.24	31.9/1.61
$\gamma = 0.70$	22.1/2.13	26.8/1.76	37.3/1.31	28.7/1.73
$\gamma = 0.60$	18.3/2.33	23.8/1.97	38.3/1.37	26.8/1.89
$\gamma = 0.50$	13.3/2.57	11.6/2.13	36.4/1.46	20.4/2.05
$\gamma = 0.40$	11.4/2.85	11.6/2.50	34.0/1.66	19.0/2.34
$\gamma = 0.30$	8.2/3.16	9.8/2.77	35.9/1.99	17.9/2.64
$\gamma = 0.20$	6.1/3.58	7.3/3.20	29.4/2.62	14.3/3.13
$\gamma = 0.10$	5.3/3.90	6.1/3.77	26.6/3.54	12.7/3.74
$\gamma = 0.00$	5.4/4.00	6.7/4.00	23.5/4.00	11.9/4.00

E.3.3 7B CONFIDENCE-DECODING SWEEPS

Table 20: Lafa-7B, B=4

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	74.6/1.62	81.7/1.34	56.6/1.29	71.0/1.42
$\gamma = 0.95$	72.6/1.90	78.7/1.52	62.5/1.38	71.3/1.60
$\gamma = 0.90$	72.0/2.07	77.4/1.67	64.0/1.41	71.1/1.71
$\gamma = 0.80$	69.2/2.30	72.6/1.84	65.4/1.47	69.1/1.87

Table 21: Lafa-7B, B=8

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	74.5/1.59	73.8/1.35	77.4/1.09	75.2/1.34
$\gamma = 0.95$	73.8/1.90	73.2/1.55	73.8/1.17	73.6/1.54
$\gamma = 0.90$	71.2/2.11	72.6/1.67	75.2/1.22	73.0/1.67
$\gamma = 0.80$	67.0/2.33	70.1/1.87	68.2/1.38	68.4/1.86

Table 22: Lafa-7B, B=16

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	73.3/1.60	72.6/1.44	76.3/1.12	74.1/1.39
$\gamma = 0.95$	72.6/1.92	74.4/1.69	71.0/1.21	72.6/1.61
$\gamma = 0.90$	71.9/2.11	76.2/1.91	70.6/1.31	72.9/1.78
$\gamma = 0.80$	66.8/2.44	70.1/2.13	67.7/1.38	68.2/1.98

Table 23: Multi-head MTP 7B, n=4 (Gloeckle et al., 2024)

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	60.5/1.00	73.8/1.00	73.2/1.00	69.2/1.00
$\gamma = 0.99$	59.4/1.75	76.2/1.59	74.5/1.13	70.0/1.49
$\gamma = 0.95$	57.1/2.04	75.0/1.88	74.1/1.22	68.7/1.71
$\gamma = 0.90$	56.2/2.22	70.7/2.06	75.6/1.26	67.5/1.85

Table 24: DeepSeek-style MTP 7B, D=1 (DeepSeek-AI, 2024)

Setting	GSM8K	HumanEval	IFEval	Avg.
NTP	70.3/1.00	80.5/1.00	78.0/1.00	76.3/1.00
$\gamma = 0.99$	69.0/1.51	78.0/1.56	75.4/1.16	74.2/1.41
$\gamma = 0.95$	68.5/1.65	77.4/1.67	75.4/1.25	73.8/1.52
$\gamma = 0.90$	67.4/1.71	77.4/1.75	74.5/1.29	73.1/1.58
$\gamma = 0.80$	66.6/1.79	77.4/1.79	75.4/1.36	73.1/1.65
$\gamma = 0.70$	65.0/1.83	75.6/1.84	77.1/1.42	72.6/1.70
$\gamma = 0.60$	61.6/1.88	71.3/1.88	74.7/1.50	69.2/1.76
$\gamma = 0.50$	61.4/1.93	71.3/1.91	76.2/1.58	69.6/1.81
$\gamma = 0.40$	59.1/1.95	71.3/1.96	72.6/1.68	67.7/1.86
$\gamma = 0.30$	59.1/1.97	70.1/1.98	72.3/1.77	67.2/1.91
$\gamma = 0.20$	57.1/1.99	68.9/1.98	73.0/1.88	66.3/1.95
$\gamma = 0.10$	57.1/1.99	68.9/1.99	72.8/1.97	66.3/1.98
$\gamma = 0.00$	57.2/1.99	69.5/1.99	71.5/2.00	66.1/1.99

Table 25: LLaDA 8B (Nie et al., 2025)

Setting	GSM8K	HumanEval	IFEval	Avg.
$k = 1$	76.0/1.00	42.1/1.00	58.0/1.00	58.7/1.00
$k = 2$	72.3/2.00	38.4/2.00	51.8/2.00	54.2/2.00

Table 26: Dream 7B (Ye et al., 2025)

Setting	GSM8K	HumanEval	IFEval	Avg.
mean $k \approx 1$	81.7/1.00	58.5/1.00	57.1/1.00	65.8/1.00
mean $k = 2$	75.0/2.00	35.4/2.00	44.7/2.00	51.7/2.00

Table 27: Fast-dLLM v2 7B (Wu et al., 2026)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	81.2/2.13	60.4/1.22	62.8/1.35	68.1/1.57
$\gamma = 0.95$	81.0/2.66	59.8/1.41	63.6/1.46	68.1/1.84

Table 28: Fast-dLLM v2 7B, Dolci SFT, B=8 (Wu et al., 2026)

Setting	GSM8K	HumanEval	IFEval	Avg.
$\gamma = 0.99$	67.4/0.81	32.9/0.59	70.6/0.93	57.0/0.78
$\gamma = 0.95$	67.2/0.86	32.9/0.64	70.1/0.96	56.7/0.82
$\gamma = 0.90$	66.8/0.89	32.9/0.66	69.9/0.97	56.5/0.84
$\gamma = 0.80$	66.0/0.96	34.1/0.69	69.5/0.99	56.5/0.88
$\gamma = 0.70$	64.1/1.03	33.5/0.74	68.2/1.01	55.3/0.93
$\gamma = 0.60$	60.4/1.10	28.7/0.83	67.8/1.02	52.3/0.98
$\gamma = 0.50$	55.5/1.21	17.7/0.87	64.5/1.07	45.9/1.05
$\gamma = 0.40$	48.8/1.37	14.6/0.96	61.7/1.13	41.7/1.15
$\gamma = 0.30$	41.2/1.55	8.5/1.16	57.5/1.39	35.7/1.37
$\gamma = 0.25$	38.3/1.78	6.1/1.20	54.7/1.76	33.0/1.58
$\gamma = 0.23$	36.5/1.86	5.5/1.31	52.5/1.86	31.5/1.68
$\gamma = 0.22$	34.0/1.93	4.9/1.47	52.1/1.98	30.3/1.79
$\gamma = 0.20$	33.5/2.13	4.9/1.53	49.4/2.14	29.2/1.93

E.4 Pure-flow inference ablations

We isolate four inference-time choices for the selected 0.5B Lafa checkpoint. All rows use the same 0.5B $B = 4$ checkpoint, NFE=32, whole-block flow decoding, the warped time grid, temperature zero, and the same generation seed. Except for the named axis, the base configuration uses noise scale 0.1, cfg=8, EMA weights, and self-conditioning. Avg. is the macro-average over IFEval, GSM8K, and HumanEval.

Self-conditioning is the dominant inference component and is particularly important for GSM8K. EMA also primarily benefits GSM8K, while raw and EMA weights are otherwise similar. Prefix guidance matters most for IFEval, with moderate guidance scales forming a broad plateau. Low noise scales are stable, whereas the checkpoint-native high-noise setting degrades reasoning perfor-

mance. Overall, the base recipe lies in a robust region rather than at a narrow optimum, so we retain it instead of retuning to single-seed fluctuations.

Table 29: **Initial-noise scale at 0.5B.** All rows use cfg=8, EMA weights, and lagged self-conditioning. The base inference scale is 0.1; the checkpoint-native training scale is 4.0.

Noise scale	IFEval	GSM8K	HumanEval	Avg.
Initial-noise sweep				
0.05	50.8	30.0	30.5	37.1
0.10 *(base)*	52.9	**30.4**	31.1	38.1
0.25	48.6	28.5	**31.7**	36.3
0.50	**53.8**	29.3	**31.7**	**38.3**
1.00	50.6	28.3	29.3	36.1
2.00	49.7	26.9	27.4	34.7
4.00	50.8	23.2	27.4	33.8

Table 30: **CFG scale at 0.5B.** All rows use noise scale 0.1, EMA weights, and lagged self-conditioning. cfg=1 disables prefix guidance, while cfg=8 is the base setting.

CFG scale	IFEval	GSM8K	HumanEval	Avg.
Prefix-guidance sweep				
off	44.4	29.8	28.7	34.3
2	50.5	29.3	31.7	37.1
4	51.8	30.3	**32.9**	**38.3**
6	51.6	**30.9**	31.7	38.1
8 *(base)*	**52.9**	30.4	31.1	38.1
12	50.3	29.0	29.9	36.4

Table 31: **EMA and inference self-conditioning at 0.5B.** The weight comparison keeps self-conditioning on, and the self-conditioning comparison uses EMA weights. All rows use noise scale 0.1 and cfg=8.

Inference setting	IFEval	GSM8K	HumanEval	Avg.
Checkpoint weights				
Raw	52.7	27.1	**31.7**	37.2
EMA *(base)*	**52.9**	**30.4**	31.1	**38.1**
Inference self-conditioning				
Off	48.8	21.9	26.2	32.3
On *(base)*	**52.9**	**30.4**	**31.1**	**38.1**

E.5 Decoding throughput

Evaluation protocol. We follow Section 3 for prompts, generation budgets, token counting, and task metrics. Table 32 reports per-H100 end-to-end generation throughput at batch size 1 with bfloat16 inference and a 32-token warm-up. Tok/s is total generated tokens divided by total generation time, and the three-task averages are unweighted. Method-specific settings and score provenance are given below.

AR SFT. The matched Qwen2.5-7B AR SFT checkpoint uses standard cached Hugging Face greedy generation and emits one new token per decoding forward.

Lafa. For each block size $B \in \{4, 8, 16\}$, we use the corresponding trained checkpoint with the self-speculative decoder of Section 3.3. A one-step flow draft of the next block is verified against the model's own greedy NTP predictions. The fused pass packs the clean draft with B noisy successor blocks, one for each possible accepted-prefix length. Clean draft positions attend causally to produce the verification logits, while each successor block attends to the prefix and its candidate accepted draft prefix and bidirectionally within the block. After selecting the longest verified prefix, we

retain its KV states and use the corresponding successor block as the next draft, avoiding a separate redrafting forward.

DS-MTP. DS-MTP is the 7B checkpoint trained with one DeepSeek-style auxiliary multi-token-prediction stage (D=1) (DeepSeek-AI, 2024). Greedy verified self-speculation compares the auxiliary proposal with the base model's next greedy NTP prediction.

MARS. We use the public MARS Qwen2.5-7B checkpoints (Jin et al., 2026) for $B \in \{4, 8, 16\}$. The reported operating point is the paper's probability-threshold decoder at γ=0.95, with its sliding KV cache. Each round commits the leftmost proposal, then commits the longest consecutive continuation whose top-token probabilities remain at least γ.

Fast-dLLM v2. We use the released Fast-dLLM v2 7B checkpoint (Wu et al., 2026) with its native threshold decoder at confidence 0.9. Block caching is disabled, and the global block and sub-block sizes are both set to $B \in \{4, 8, 16\}$, so one checkpoint supplies all three operating points.

Efficient-DLM. We evaluate Efficient-DLM-8B (Fu et al., 2025) with its native confidence-threshold decoder at γ=0.7 for $B \in \{4, 8, 16, 32\}$, together with the B=32 speed–quality sweep at $\gamma \in \{0.5, 0.6\}$. We use the model's native prompt format rather than Qwen chat templating.

Dream. We use Dream-v0-Instruct-7B (Ye et al., 2025) with its native entropy-ordered unmasking, targeting a mean of $k \in \{1, 2, 4\}$ filled canvas positions per denoising forward. For throughput accounting, we follow Efficient-DLM (Fu et al., 2025) and compute tok/s as the total number of generated canvas tokens divided by total generation time.

Table 32: Quality and decoding throughput at 7–8B.

			GSM8K		HumanEval		IFEval		Avg.	
Method	**Setting**	γ	Tok/s	Score	Tok/s	Score	Tok/s	Score	Tok/s	Score
Sequential and auxiliary-head baselines										
AR SFT	–	–	40.6	69.1	37.0	78.7	40.9	73.8	39.5	73.9
DS-MTP	B=2	–	57.2	70.3	58.4	**80.5**	45.1	78.0	53.6	76.3
Adaptive parallel baselines										
MARS	B=4	0.95	59.8	70.7	43.3	79.9	42.0	77.6	48.4	76.1
	B=8	0.95	62.7	69.1	43.1	78.0	40.9	77.3	48.9	74.8
	B=16	0.95	61.3	72.9	44.6	75.0	41.1	75.0	49.0	74.3
Fast-dLLM v2	B=4	0.90	66.2	81.6	56.5	67.1	52.1	65.8	58.3	71.5
	B=8	0.90	98.5	80.1	71.1	57.3	60.9	65.4	76.8	67.6
	B=16	0.90	116.0	77.3	78.9	56.1	65.1	61.9	86.7	65.1
Efficient-DLM	B=4	0.70	37.1	87.8	38.4	70.1	31.8	58.2	35.8	72.0
	B=8	0.70	55.6	87.7	58.5	68.9	40.7	58.2	51.6	71.6
	B=16	0.70	70.5	**88.4**	75.4	67.1	47.2	58.4	64.4	71.3
	B=32	0.70	74.4	87.5	84.8	63.4	49.3	59.1	69.5	70.0
	B=32	0.60	87.1	82.2	96.3	55.5	56.1	58.6	79.9	65.4
	B=32	0.50	102.9	72.0	111.2	42.1	71.9	52.9	95.3	55.7
Dream	k=1	–	36.0	81.7	26.7	58.5	22.7	57.1	28.5	65.8
	k=2	–	72.0	75.0	53.3	35.4	45.6	44.7	57.0	51.7
	k=4	–	143.5	59.3	106.3	20.1	91.2	33.8	113.7	37.7
Lafa	B=4	–	91.7	73.8	87.2	76.8	60.9	**78.9**	79.9	**76.5**
	B=8	–	116.1	75.3	104.3	76.2	61.1	77.5	93.8	76.3
	B=16	–	125.2	75.0	115.7	73.8	59.7	74.9	100.2	74.5