Nemotron-Labs-Diffusion: A Tri-Mode Language Model Unifying Autoregressive, Diffusion, and Self-Speculation Decoding

Yonggan Fu, Lexington Whalen[1†], Abhinav Garg, Chengyue Wu[2†], Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu[3†], Shiyi Lan, Ligeng Zhu, Jin Wang[2†], Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han[4], Ante Jukić, Nima Tajbakhsh, Jan Kautz, Pavlo Molchanov

We introduce Nemotron-Labs-Diffusion, a tri-mode language model (LM) that unifies AR, diffusion, and self-speculation decoding within a single architecture. Trained with a joint AR-diffusion objective, Nemotron-Labs-Diffusion can switch modes to sustain high throughput across deployment settings and concurrency levels. Our study shows that (1) AR and diffusion objectives are complementary: diffusion improves lookahead planning, while AR provides left-to-right linguistic priors. (2) In self-speculation mode, diffusion drafts while AR verifies, outperforming multi-token prediction (MTP) methods in both acceptance rate and real-device efficiency. (3) A speed-of-light analysis further demonstrates diffusion's long-term potential, with up to 76.5% more tokens per forward pass than self-speculation under an optimal sampler. Scaling to 3B, 8B, and 14B parameters, our Nemotron-Labs-Diffusion family, including base, instruct, and vision-language models, consistently outperforms state-of-the-art open-source AR and diffusion LMs in both accuracy and speed. For example, Nemotron-Labs-Diffusion-8B decodes 6× more tokens per forward than Qwen3-8B with comparable accuracy, translating to 4× higher throughput on SPEED-Bench with SGLang on a GB200 GPU.

Models on Hugging Face: Nemotron-Labs-Diffusion Model Family

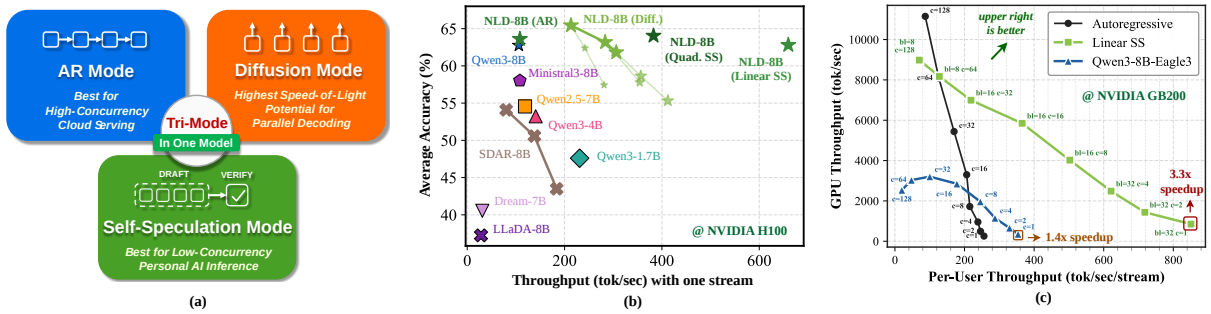


Figure 1 | **(a)** An illustration of three modes in one model. **(b)** Accuracy–throughput trade-off measured on general benchmarks at batch size 1 on an NVIDIA H100 using PyTorch. NLD denotes our model and Linear/Quad SS denotes linear/quadratic self-speculation. ★ indicates the diffusion mode with different denoising block sizes (8, 16, 32), where smaller symbols correspond to smaller block sizes. **(c)** Trade-off between system vs. per-user throughput of the AR and Linear SS modes of our 8B model and Qwen3-8B Eagle3, measured on SPEED-Bench [1] at different concurrency c on an NVIDIA GB200 GPU using SGLang.

1. Introduction

The strictly sequential, token-by-token decoding process of autoregressive (AR) language models (LMs) fundamentally limits their inference parallelism, resulting in resource under-utilization and low throughput, especially in low-batch-size deployment scenarios. Diffusion LMs [2, 3, 4] have recently emerged as a promising alternative, enabling parallel generation by decoding multiple tokens per forward pass. Nevertheless, diffusion LMs often lag behind AR models in accuracy and learning efficiency, requiring substantially more data to reach comparable performance [5]. A key reason is that diffusion training treats all token permutations uniformly [6], rather than leveraging the strong left-to-right prior inherent in natural language. Moreover, existing diffusion LMs still lack clear advantages over multi-token prediction (MTP) methods and often fall behind them in practical efficiency–accuracy trade-offs.

Additional affiliations: [1]Georgia Tech, [2]HKU, [3]University of Chicago, [4]MIT. [†]Work done during internships at NVIDIA.

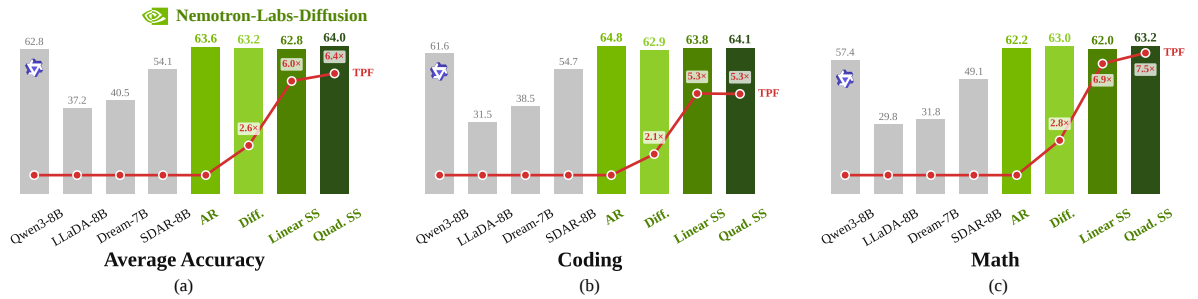


Figure 2 | Benchmarking our Nemotron-Labs-Diffusion-8B (Instruct) against SOTA AR and diffusion instruct LMs across different benchmarks. (a) shows the average accuracy across all 10 tasks (HumanEval, MBPP, LiveCodeBench-CPP, GSM8K, Math500, AIME24, AIME25, GPQA, IFEval, MMLU) and the tokens per forward (TPF) of different models, while (b) and (c) show the average accuracy across coding (HumanEval, MBPP, LiveCodeBench-CPP) and math (GSM8K, Math500, AIME24, AIME25) domains, respectively.

These limitations raise three critical questions for understanding the role of diffusion LMs:

Q1: Should diffusion LMs compete with AR LMs, or can the two paradigms be harmonized?

Q2: Can diffusion LMs provide a stronger acceleration mechanism than MTP methods?

Q3: Does diffusion decoding have enough long-term potential to justify deeper exploration?

Answering these questions is critical for judging the true promise of diffusion LMs and guiding their correct and wide adoption. This work studies these questions by **unifying AR and diffusion modeling within a single model** that preserves the strengths of AR LMs while exploring the benefits and potential of parallel decoding. The motivation is that AR and diffusion LMs might not be competing paradigms in which one should replace the other; instead, they can be mutually beneficial and unified within a single model by switching between causal and bidirectional attention. Specifically, AR models inherently learn to plan ahead for future tokens [7], and diffusion training can further enhance this capability. Conversely, preserving AR objectives during training injects strong left-to-right linguistic priors into diffusion modeling.

Based on this insight, we introduce Nemotron-Labs-Diffusion, a tri-mode LM that jointly optimizes diffusion and AR losses under a unified training framework. Our training scheme employs a global loss-averaging strategy that treats all tokens across batches equally to stabilize optimization. We further adopt a two-stage training procedure: we first strengthen AR capabilities to establish strong left-to-right linguistic priors, and then enable joint diffusion and AR training to fully integrate both objectives. The resulting models support tri-mode decoding as shown in Fig. 1 (a): (1) AR decoding, (2) parallel diffusion-based decoding, which can be paired with a sampler optimized on sampling trajectories for improved parallelism, and (3) self-speculation, where diffusion drafts candidate tokens and AR predictions verify them.

We leverage this training scheme to deliver the Nemotron-Labs-Diffusion model family, including base, instruct, and vision-language variants at 3B/8B/14B scales. As shown in Fig. 2, our models outperform state-of-the-art (SOTA) open-source AR / diffusion LMs in both accuracy and inference speed across a wide range of benchmarks. For example, our Nemotron-Labs-Diffusion-8B delivers 6× tokens per forward over Qwen3-8B while maintaining comparable or better accuracy on general benchmarks, translating to 4× throughput on SPEED-Bench [1] measured with SGLang on an NVIDIA GB200 GPU.

The results and analysis of our tri-mode LMs offer rich insights to answer the above questions. First, AR and diffusion LMs can be harmonized rather than treated as competing alternatives and unifying AR and diffusion objectives is a promising pathway: AR contributes strong next-token modeling and linguistic priors, while diffusion unlocks parallel generation without sacrificing benchmark performance. Beyond accuracy, the joint objective naturally enables self-speculation, allowing tri-mode LMs to adapt to different deployment regimes with different levels of concurrency: self-speculation is especially effective in low-concurrency settings, as shown in Fig. 1 (b), while AR remains well suited for compute-bound high-concurrency scenarios. As such, tri-mode models can serve as drop-in replacements for conventional AR LMs, requiring no architectural or pipeline changes while offering consistently high throughput across deployment scenarios.

Our speed-of-light (SOL) analysis, which estimates the upper bound of diffusion decoding when equipped with an optimal sampler, shows that diffusion decoding has strong potential and substantial headroom beyond current parallel decoding methods. Specifically, diffusion-based decoding with an optimal sampler can

correctly predict over 76.5% more tokens per forward pass than the self-speculation mode, indicating that current samplers still leave a large fraction of the available parallelism unused. These results highlight the long-term promise of diffusion decoding.

Notably, we find that sampling tokens from the diffusion mode to approach the SOL upper bound remains an open challenge, and that the most effective approach is to verify the decoded tokens using the same model in AR mode. This observation motivates the aforementioned self-speculation mode, where diffusion generates high-quality multi-token drafts while AR verifies them within a single model, eliminating the need for the auxiliary prediction heads used by prior MTP methods [8]. As shown in Fig. 1 (c), this yields higher acceptance rates and more favorable trade-offs between system throughput and per-user throughput, making tri-mode LMs a stronger and more flexible alternative to existing MTP approaches.

We hope the above insights can shed light on the proper adoption of diffusion objectives in language modeling and on future directions that fully unlock the potential of diffusion decoding.

Paper structure. The rest of the paper is organized as follows. Sec. 2 and Sec. 3 detail our joint AR-diffusion training framework and tri-mode inference algorithms; Sec. 4 presents the speed-of-light analysis; Sec. 5 and Sec. 6 introduce the Nemotron-Labs-Diffusion model family and present experiments, including comparisons between self-speculation and MTP; Sec. 7 reviews related work and Sec. 8 concludes with insights and future directions.

2. Tri-Mode LM Training

2.1. Training Objectives

Motivation. We hypothesize that AR and diffusion objectives are complementary rather than competing. AR pretraining induces an implicit ability to plan ahead [7], which the diffusion objective further unlocks by forcing the model to reason about future tokens; in turn, the AR objective anchors diffusion training to the left-to-right structure of language and prevents wasted capacity on arbitrary token permutations. Therefore, we train Nemotron-Labs-Diffusion on a weighted combination of an AR next-token loss and a block-wise diffusion denoising loss.

AR objective. For a token sequence x, the AR objective maximizes the likelihood under the left-to-right factorization:

$$\mathcal{L}_{\mathrm{AR}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[-\sum_{i=1}^{|x|}\log p_\theta(x_i \mid x_{<i})\right]. \quad (1)$$

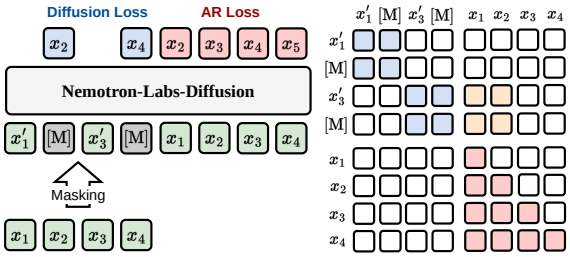


Figure 3 | Visualizing the attention pattern of Nemotron-Labs-Diffusion, where denotes attention among noisy tokens, denotes attention from noisy tokens to clean-context tokens, and denotes attention within the clean context.

Diffusion objective. As shown in Fig. 3, we adopt the block-wise diffusion formulation [9, 10], which partitions the sequence into B contiguous blocks $\{x^b\}_{b=1}^{B}$ and trains the model to denoise one block at a time conditioned on its clean prefix. At noise level $t \sim \mathcal{U}[0,1]$, only the tokens in the current block are corrupted via a forward noising process q, i.e., $\tilde{x}_t^b \sim q(\cdot \mid x^b)$, while the prefix $x^{<b}$ remains clean:

$$\mathcal{L}_{\mathrm{diff}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1]\\ \tilde{x}_t^b\sim q(\cdot|x^b)}}\left[-\frac{1}{t}\sum_{b=1}^{B}\log p_\theta\big(x^b \mid \tilde{x}_t^b,\ x^{<b}\big)\right]. \quad (2)$$

This block-wise design is bidirectional within each block to enable parallel intra-block prediction, and causal across blocks so that previously generated blocks can reuse their KV cache during inference.

Joint objective. We optimize a weighted combination of the two losses:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{AR}}(\theta) + \alpha\,\mathcal{L}_{\mathrm{diff}}(\theta), \quad (3)$$

where the AR loss has coefficient 1 and α controls the strength of diffusion supervision. This design choice is motivated by the observation that the diffusion loss is often larger than the AR loss, and selecting an α that aligns the magnitudes of the two losses yields the best results, i.e., enabling diffusion-style parallel decoding while maintaining AR accuracy. We set $\alpha = 0.3$ across all training stages.

Two-stage training. To strengthen left-to-right priors and improve learning efficiency, we adopt a two-stage training strategy that first trains with the AR objective, which anchors the representation to the language's inherent left-to-right inductive bias, and then switches to the joint objective. In terms of Eq. 3, Stage 1 sets $\alpha = 0$, reducing the optimization to the pure AR objective in Eq. 1. In Stage 2, we turn on diffusion supervision by setting α to align the magnitudes of the two losses, as mentioned above, so that diffusion gradients complement rather than overwrite the AR priors.

Table 1 | Ablation study of each training technique during continuous pretraining on 25B tokens.

Technique	HumanEval	HumanEval+	MBPP	MBPP+	GSM8K	Minerva Math	Avg
Block-wise attention	39.02	37.80	53.40	67.72	82.87	44.58	54.23
+ Global Loss Avg	42.07	39.02	56.20	71.69	83.78	45.36	56.35
+ DP-rank Varying Masking Ratios	45.12	43.29	55.80	70.90	81.58	45.66	57.06
+ Two-stage training	58.54	52.44	53.00	73.81	83.17	55.84	62.80
+ AR loss	64.02	57.93	65.60	80.95	86.73	66.44	70.28

Global loss averaging. Since the diffusion objective involves randomly sampling masked tokens, different training examples may have different numbers of tokens contributing to the diffusion loss. As a result, the strategy for averaging token-wise losses matters, analogous to how the choice of aggregation in on-policy RL objectives (e.g., GRPO [11] vs. DAPO [12]) can affect training stability. We consider two loss averaging strategies. Let a batch contain N sequences, each of length L, and let $\ell_{n,i}$ denote the token-level loss for token i in sequence n based on Eq. 3. One choice is to first average token losses within each sequence and then average them over sequences:

$$\mathcal{L}_{\text{seq-avg}} = \frac{1}{N}\sum_{n=1}^{N}\Big(\frac{1}{L}\sum_{i=1}^{L}\ell_{n,i}\Big). \tag{4}$$

Another choice is to treat all tokens across the batch equally and globally average over the NL token losses:

$$\mathcal{L}_{\text{global}} = \frac{1}{NL}\sum_{n=1}^{N}\sum_{i=1}^{L}\ell_{n,i}. \tag{5}$$

While Eq. 4 and Eq. 5 coincide when every sequence has the same number of loss-contributing tokens, they differ once masking yields variable numbers of noisy tokens across samples, which is common in the diffusion objective. In particular, in Eq. 2, the loss includes a $\frac{1}{t}$ reweighting, and the number of noisy tokens is approximately proportional to t. When t is small, each noisy token tends to carry a larger weight (due to $\frac{1}{t}$), but there are fewer such tokens in the sample. Sequence-wise averaging can therefore amplify the influence of these small-t samples: their per-token losses are larger, yet the per-sequence normalization assigns them the same weight as other samples, increasing batch-to-batch fluctuations and gradient variance. In contrast, global averaging effectively weights each training example in proportion to its number of contributing tokens, preventing samples with only a few highly weighted noisy tokens from disproportionately influencing the batch loss.

2.2. Attention Pattern

Following [9], at training time we use a dual-stream input by concatenating a corrupted/noised view and a clean view of the same sequence, and apply a structured attention pattern, as shown in Fig. 3.

The *Noisy→Noisy* and *Noisy→Clean* parts follow the standard block diffusion design [9]: we partition the sequence into B contiguous blocks $\{x^b\}_{b=1}^{B}$; in the noisy stream, tokens attend bidirectionally within each block and causally across blocks; and for denoising block b, noisy tokens additionally attend to the clean-prefix blocks $x^{<b}$ in the clean stream to achieve clean-context conditioning.

The key difference lies in the *Clean→Clean* mask. Prior designs [9, 10, 13] allow the clean stream to attend to future tokens using block-wise attention. In contrast, we enforce a strictly causal mask within the clean stream [14, 7]. This enables us to compute the AR objective on x in this clean-context part together with the diffusion objective on $\tilde{x}_t$ in the same forward-backward pass, without label leakage.

Relationship with prior works. Our attention pattern follows the pioneering work of [14], which also performs joint diffusion and AR training. The key differences in our work lie in (1) proposing tri-mode inference, particularly self-speculation decoding, along with post-training enhancements for improved parallelism, including the samplers and LoRA-enhanced drafters introduced in Sec. 3; (2) the overall training pipeline used to develop the full model family described in Sec. 5; and (3) the systematic studies and SOL analysis conducted to address critical questions regarding the true potential of diffusion LMs.

2.3. Ablation Study on Training Techniques

We ablate the contribution of each training technique by progressively adding them during continuous pretraining on 25B tokens, starting from the official Ministral3-8B base model. Detailed training/evaluation settings will be elaborated in Sec. 5.1 and Sec. 6.3. All models are evaluated in diffusion mode on coding and math benchmarks.

Observations. As shown in Tab. 1, we progressively add each technique and observe that (1) *block-wise attention* serves as the baseline at 54.23% average accuracy, following the setting of [10, 4] and providing a functional diffusion LM; (2) *global loss averaging* improves the average by 2.12%, confirming that treating all tokens equally across the batch reduces gradient variance from variable masking ratios, as analyzed in Sec. 2.1; (3) *DP-rank varying masking ratios*, which applies different noise levels across data-

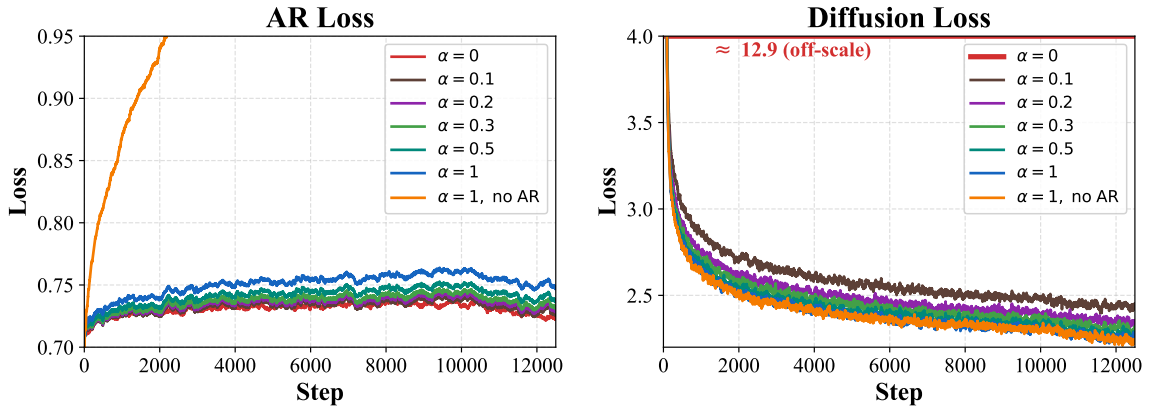


Figure 4 | Visualizing the evolution of the AR and diffusion losses across training steps under different diffusion loss coefficients α. The AR loss coefficient is set to 1 by default across all settings, except in the 'no AR' setting, where the AR loss is removed and only the diffusion loss is used.

Table 2 | Impact of diffusion loss weight α during 25B-token continuous pretraining.

α	Mode	Human Eval	Human Eval+	MBPP	MBPP+	GSM8K	Minerva Math	Avg
0.1	Diff.	56.71	51.83	64.80	81.22	87.64	67.02	68.20
	AR	58.54	53.05	64.60	80.42	87.64	**68.02**	68.71
0.2	Diff.	60.37	54.27	63.40	80.69	86.43	66.58	68.62
	AR	60.98	57.93	**66.40**	**83.60**	87.04	67.04	70.50
0.3	Diff.	61.59	**58.54**	64.60	80.42	87.64	65.86	69.77
	AR	**62.80**	57.93	65.80	82.54	**87.79**	66.84	**70.62**
0.5	Diff.	59.76	54.27	64.40	80.16	87.11	66.98	68.78
	AR	58.54	53.05	65.40	82.80	86.81	67.14	68.96
1.0	Diff.	56.10	48.78	65.00	80.69	84.91	66.12	66.93
	AR	54.27	50.61	64.80	80.69	86.58	66.36	67.22

parallel ranks, further improves the average by 0.71%; (4) *two-stage training*, which provides a better AR initialization with 1T-token AR objective training, yields a substantial 5.74% gain, implying that stronger AR initialization can enable better future planning and ease AR-to-diffusion conversion; (5) *the addition of AR loss* contributes the largest single improvement of 7.48%, significantly boosting diffusion decoding abilities, echoing our analysis in Sec. 2.

Cumulatively, the full pipeline improves the baseline by 16.05% in average accuracy, with the AR loss and two-stage training contributing the most. This validates our core insight that preserving the AR objective during diffusion training anchors the model to linguistically coherent trajectories and is a critical factor for achieving strong diffusion LM accuracy.

2.4. Mutual Impact of AR/Diffusion Losses

In this subsection, we examine whether the AR and diffusion objectives compete for model capacity or reinforce each other: the impact of adding the AR loss on the diffusion mode, and the impact of adding the diffusion loss on AR-mode accuracy.

AR loss boosts diffusion accuracy. As studied in Sec. 2.3 and Tab. 1, AR loss can significantly boost diffusion accuracy. As a complement to this study, we also vary α in Eq. 3 during 25B-token continuous pretraining on top of the two-stage training setting in Tab. 2. We observe that both modes peak at α=0.3. This implies that the two modes do not necessarily compete with each other or achieve the best performance at the two extremes; instead, there exists a sweet spot where both are well harmonized. Similarly, no value of α in $[0.1, 0.5]$ improves one mode at the expense of the other, and the two objectives rise and fall together, indicating that they are complementary rather than competing for model capacity.

We also visualize the training loss curves in Fig. 4. We observe that the aforementioned setting α=0.3, which achieves the best accuracy, provides a good balance between the two losses. Setting α too small or too large leads to increased diffusion or AR loss, respectively. In addition, without the AR or diffusion loss, the corresponding AR or diffusion capabilities are degraded or lost.

Diffusion loss preserves AR accuracy. To study whether diffusion loss can hurt or preserve AR accuracy, we compare models trained w/o and w/ the diffusion loss (α=0.3) under two settings: continuous pretraining on 25B tokens on top of the two-stage training setting in Tab. 1, and further SFT on 45B tokens, following training/evaluation settings in Sec. 5.2 and Sec. 6.1. We ensure that all settings are trained on the same number of tokens.

As shown in Tab. 3, we observe that: (1) In both settings, the average AR accuracy is preserved or slightly boosted, with 0.14% and 0.43% improvements for the base and instruct models, respectively, indicating that diffusion training, when properly integrated, can enhance the future prediction abilities of the AR mode, similar to observations in DeepSeek-V3 [15]; (2) At the per-benchmark level, the instruct model shows gains on coding and math benchmarks, e.g., 4.24% higher on LCB-CPP and 1.59% higher on MBPP, but drops on IFEval (3.01% lower) and HumanEval (2.44% lower),

Table 3 | AR-mode accuracy with and without the diffusion loss (α=0.3). *Base*: 25B-token continuous pretraining from Ministral3-8B. *Instruct*: further SFT on 45B tokens.

Base Model											
Training	**HumanEval**	**HumanEval+**	**MBPP**	**MBPP+**	**GSM8K**	**Minerva Math**	**MMLU**	**Hellaswag**	**PIQA**	**Winogrande**	**Avg**
AR only	60.37	56.10	67.00	81.48	87.64	67.90	76.34	76.54	79.71	71.98	72.50
+ Diff. loss	62.80	57.93	65.80	82.54	87.79	66.84	75.99	76.59	79.82	70.32	72.64
Instruct Model											
Training	**GPQA**	**IFEval**	**HumanEval**	**MBPP**	**Math500**	**GSM8K**	**AIME24**	**AIME25**	**MMLU**	**LCB-CPP**	**Avg**
AR only	44.44	71.66	82.93	83.60	87.80	93.63	36.67	26.67	79.77	24.61	63.18
+ Diff. loss	44.44	68.65	80.49	85.19	88.00	94.01	33.33	33.33	79.85	28.85	63.61

suggesting that strict instruction-following compliance is slightly affected by the diffusion objective.

These results, together with the α sensitivity analysis above, support the conclusion that joint AR–diffusion training is not a zero-sum trade-off: the diffusion loss enables parallel decoding modes (Sec. 3) at negligible cost to AR-mode accuracy, and the two objectives share a common optimal operating point.

3. Tri-Mode LM Inference

The joint AR and diffusion training enables decoding in three modes: AR, diffusion, and self-speculation decoding, as shown in Fig. 5.

3.1. Mode 1: AR Decoding

Tri-mode LMs fully preserve standard left-to-right generation: at step i, they sample $x_i \sim p_\theta(\cdot \mid x_{<i})$ with causal attention. This mode is preferred when serving with high concurrency.

3.2. Mode 2: Block-wise Diffusion Denoising

Confidence-based sampling. Following [13, 10], the diffusion decoding mode proceeds block by block. For the current block, we initialize its positions as mask tokens and iteratively denoise multiple tokens in parallel per step based on a confidence threshold [16]. When a block is completed, its KV cache will be refreshed, and decoding proceeds to the next block.

Sampling with a trained sampler. A fixed confidence threshold is an implicit signal that is not explicitly optimized during training. We therefore train a lightweight sampler that, for every masked position in the current block, predicts whether the top-1 prediction at the current denoising step is *correct*. Here, *correct* means that the decoded token matches the token that will eventually be committed at this position when decoding only the highest-confidence token at each step. At inference, we commit positions whose predicted probability from the sampler exceeds a predefined threshold, which trades off TPF against per-token error rate. The sampler can be viewed as a learned classifier to approach the greedy-acceptance criterion that we later analyze in Sec. 4. The sampler architecture, feature engineering, and training trajectory data collection are detailed in Appendix A.

3.3. Mode 3: Self-Speculation Decoding

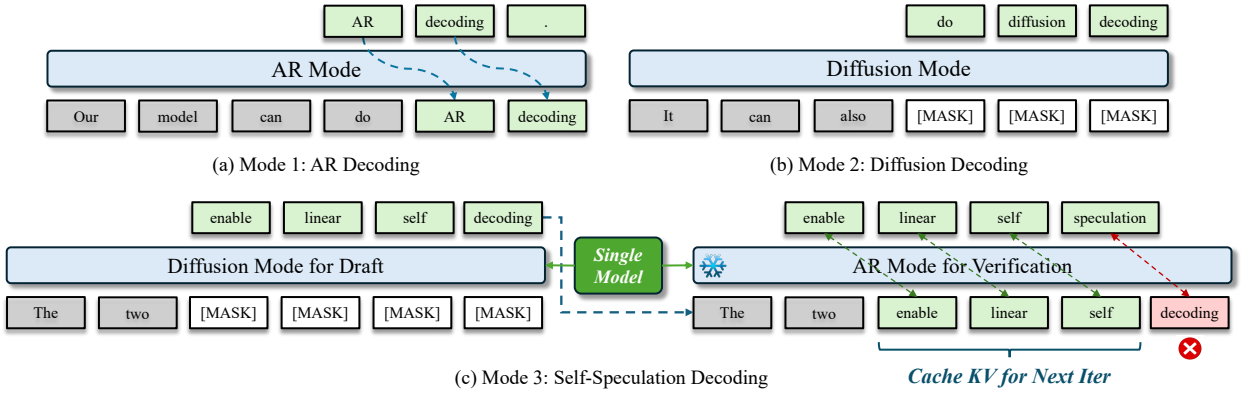
Linear self-speculation. The simplest self-speculative mode separates diffusion-based drafting and AR-based verification into two forward passes. Let $[x_1, \ldots, x_n]$ denote the currently verified prefix and let k be the speculative width.

Drafting with diffusion. We append k mask tokens to the verified prefix, forming the input $[x_1, \ldots, x_n, m_1, \ldots, m_k]$. The model denoises all k mask positions in parallel using the diffusion pathway, producing draft tokens $\{\hat{x}_{n+1}, \ldots, \hat{x}_{n+k}\}$.

Verification with AR. We then run a second forward pass over the draft tokens $[\hat{x}_{n+1}, \ldots, \hat{x}_{n+k}]$ with causal attention, again reusing the prefix KV cache. The AR logits at each position yield next-token predictions $\{x^{\mathrm{AR}}_{n+j}\}_{j=1}^{k}$. We accept the longest prefix of draft tokens that passes the verification criterion (e.g., $x^{\mathrm{AR}}_{n+j} = \hat{x}_{n+j}$) and commit the accepted tokens to the verified prefix. As in standard speculative decoding [17], the AR prediction at the first rejected position provides one additional verified token, so each step produces between 1 and k+1 tokens. Note that both the drafting and verification passes can reuse the cached prefix KVs from prior verified steps.

Enhance linear self-speculation w/ LoRA. We further enhance linear self-speculation by tuning a LoRA adapter [18] on top of the diffusion draft pathway to better align its drafts with the AR verifier, thereby extending the accepted prefix length per step. We apply LoRA only to the o_{proj} layer of the attention module (rank 128, α=512, ∼36M trainable parameters/∼0.4% of the backbone), leaving the AR pathway unchanged. The training loss combines an LK-hybrid distribution-matching term [19] with a token-level cross-entropy term, both applied to the accepted prefix plus the first rejected position of each draft block, as shown in Fig. 11 in Appendix B.

Drafter–verifier setup. For each position $j \in \{1, \ldots, k\}$ in the draft block, the LoRA-augmented drafter produces logits $z^d_j \in \mathbb{R}^{|\mathcal{V}|}$ over the vocabulary



(a) Mode 1: AR Decoding (b) Mode 2: Diffusion Decoding

(c) Mode 3: Self-Speculation Decoding

Figure 5 | Visualizing tri-mode inference: (a) left-to-right AR decoding, (b) parallel diffusion decoding, and (c) linear self-speculation decoding (quadratic self-speculation is visualized in Fig. 12).

$\mathcal{V}$, while the frozen AR verifier produces target logits z_j^t on the same context. We define the temperature-scaled distributions

$$q_j = \text{softmax}(z_j^d/\tau), \qquad p_j = \text{softmax}(z_j^t/\tau),$$

with τ=3.0. The target p_j is treated as fixed (stop-gradient), so only the drafter q_j carries gradient through the LoRA parameters.

Active position mask: "accepted + 1". As shown in Fig. 11, both loss terms are computed only on the *accepted prefix plus the first rejected position*. Letting j^* denote the position of the first mismatch (the smallest j with $\hat{x}_{n+j} \neq x_{n+j}^{\text{AR}}$), the active set is $\mathcal{A} = \{1, \ldots, j^*\}$ when there is a rejection in the block, and $\mathcal{A} = \{1, \ldots, k\}$ otherwise; all other positions are masked out and contribute neither to the loss numerator nor to its denominator. This mask is essential because, at inference, the verifier's KV cache is rebuilt at the rejection point: logits at positions $j > j^*$ are conditioned on a continuation the deployed loop never observes, so training on them would bias the drafter toward a counterfactual distribution.

LK-hybrid distribution-matching loss. The distribution-matching term adapts the LK-hybrid loss of [19] to a truncated top-K support. We retain the union of the top-K token indices, $\mathcal{U}_j = \mathcal{S}_j^t \cup \mathcal{S}_j^d$, where $\mathcal{S}_j^t$ (resp. $\mathcal{S}_j^d$) is the set of K indices with the largest probability under p_j (resp. q_j). Setting K=200 gives $|\mathcal{U}_j| \leq 2K = 400$. We zero both distributions outside $\mathcal{U}_j$ and renormalize to obtain $\tilde{p}_j$ and $\tilde{q}_j$. Truncation to the union avoids the $\text{KL}(\tilde{p}_j \,\|\, \tilde{q}_j) = \infty$ catastrophe of full-vocabulary KL. The per-position hybrid loss is

$$\mathcal{L}_j^{\text{LK}} = \lambda_j \cdot \text{KL}(\tilde{p}_j \,\|\, \tilde{q}_j) + (1-\lambda_j) \cdot \tfrac{1}{2} \sum_{v \in \mathcal{U}_j} \left| \tilde{p}_j(v) - \tilde{q}_j(v) \right|, \tag{6}$$

where the right-hand total-variation (TV) term equals $1 - \alpha_j$, with $\alpha_j = \sum_{v \in \mathcal{U}_j} \min\big(\tilde{p}_j(v), \tilde{q}_j(v)\big)$ the standard speculative-decoding acceptance probability [17]—the probability that a token sampled from $\tilde{q}_j$ is accepted by the speculative-decoding rejection rule against $\tilde{p}_j$. The adaptive coefficient $\lambda_j = \exp(-\eta \cdot \text{sg}[\alpha_j])$ with η=0.5 makes the loss behave like the (forward) KL early in training (when $\alpha_j \approx 0$ and $\lambda_j \approx 1$, providing a stronger distribution-matching gradient) and like TV as the drafter approaches the verifier ($\alpha_j \to 1$ and $\lambda_j \to e^{-\eta} \approx 0.6$, directly minimizing the acceptance-rate gap).

Cross-entropy term. The LK-hybrid term matches the full top-K output distribution. We additionally apply a token-level cross-entropy at each active position against the verifier's argmax target $y_j = \arg\max_v z_j^t(v)$ on the same union support:

$$\ell_j = \begin{cases} -\log q_j^{(\mathcal{U}_j)}(y_j) & \text{if } y_j \in \mathcal{U}_j, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $q_j^{(\mathcal{U}_j)}$ is the drafter softmax restricted to $\mathcal{U}_j$ with τ=1.0. The cross-entropy provides a strong teacher-forcing signal toward the verifier's modal token, complementing the soft distribution-matching of LK. When the truncation excludes y_j (rare, $< 2\%$ at K=200), ℓ_j is set to zero; that position is also dropped from the CE denominator below (Eq. 8), so occasional truncation misses cannot blow up the loss.

Total loss and training-time drafter sampling. The two terms are aggregated as masked means over the active positions of each draft block:

$$\mathcal{L}_{\text{LK}} = \frac{1}{|\mathcal{A}|} \sum_{j \in \mathcal{A}} \mathcal{L}_j^{\text{LK}}, \qquad \mathcal{L}_{\text{CE}} = \frac{\sum_{j \in \mathcal{A}} \ell_j}{\sum_{j \in \mathcal{A}} \mathbb{1}\{y_j \in \mathcal{U}_j\}}. \tag{8}$$

These per-block means are further averaged across the inner training batch. The total loss is

$$\mathcal{L} = \lambda_{\text{KL}} \cdot \mathcal{L}_{\text{LK}} + \lambda_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}, \qquad \lambda_{\text{KL}} = \lambda_{\text{CE}} = 1. \tag{9}$$

At training time, 90% of the inner-batch slots draw their drafter tokens by sampling from

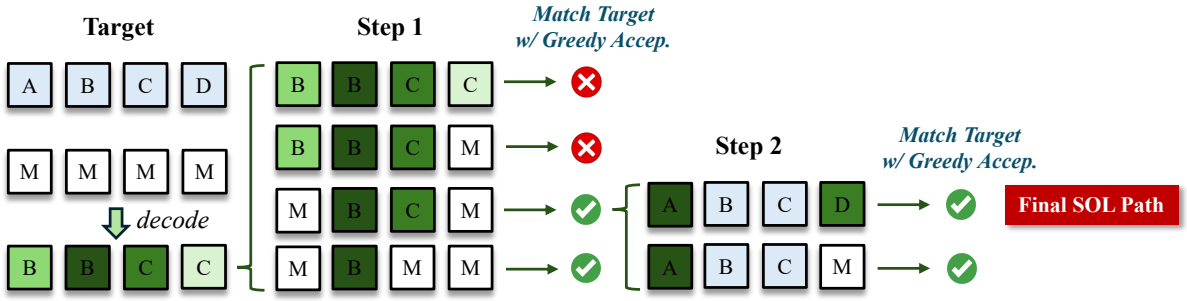


Figure 6 | An example of using the recursive dynamic compaction method to identify the SOL path.

softmax(z_j^d/T_{draft}) with T_{draft}=1.0; the rest stay greedy. Sampling exposes the LK gradient to a broader empirical distribution of drafter outputs and yields adapters that remain robust when the verifier is itself sampled at inference.

3.4. Variant: Quadratic Self-Speculation

A variant of linear self-speculation is quadratic self-speculation, which leverages quadratic decoding [7] with single-forward drafting and verification, following the same process as [20]. This decoding scheme prepares for the worst case by predicting the next block at all possible acceptance positions with a quadratic cost. Specifically, quadratic self-speculation performs speculative drafting and verification simultaneously within a single forward pass by using a structured attention mask, where causal predictions verify previous drafts while parallel diffusion predictions generate new draft tokens for the next iteration. The interleaved quadratic layout ensures that each iteration consistently produces k speculative tokens even when verification terminates early due to mismatches. In addition to standard AR-based verification, our tri-mode model further supports an AR-diffusion ensemble verifier that combines causal and diffusion predictions through weighted interpolation. More details are provided in Appendix C and Fig. 12.

4. Speed-of-Light Analysis

We conduct a speed-of-light (SOL) analysis to quantify the maximum acceptance rate / token-per-forward achievable by the diffusion mode. We apply this analysis to the diffusion mode of Nemotron-Labs-Diffusion-8B delivered in Sec. 3.2. The SOL ceiling tells us how much intrinsic parallelism the current confidence-based sampling is leaving on the table. SOL is computed entirely within the diffusion model, i.e., no AR verifier is involved, so it isolates the diffusion model's own parallel-decoding capability and provides a reference ceiling for any scheme that targets its converged output. Compared with linear self-speculation in Sec. 3, which only commits a contiguous prefix of the draft and is therefore truncated at the first rejection, diffusion-mode decoding can commit *any* subset of masked positions per pass.

4.1. Diffusion SOL Construction

Oracle target via serial denoising. We first define the diffusion model's converged output for each block of length B. Let f_θ denote the diffusion model, which on a partially masked input outputs a categorical distribution over the vocabulary at every masked position; let `[M]` denote the mask token. Starting from an all-mask input $\mathbf{x}^{(0)} = \texttt{[M]}^B$, at each step we identify the masked position whose output distribution has the highest peak probability (across positions and vocabulary), commit its argmax to that position, and re-evaluate f_θ on the resulting partially-unmasked sequence; we repeat until all B positions are filled. This *serial denoising* procedure uses exactly B forward passes—one position committed per pass—and yields a target sequence $\mathbf{t} = (t_1, \ldots, t_B)$ that the model would converge to in the absence of any parallel commits. The SOL acceptance ratio is then the average TPF needed to reproduce $\mathbf{t}$ from the same all-mask input under a parallel scheme.

Greedy parallel acceptance. The simplest parallel scheme is greedy acceptance. At iteration k, the model produces argmax predictions $\hat{\mathbf{t}}^{(k)}$ for every position from the current input $\mathbf{x}^{(k)}$, and we commit every masked position whose prediction matches the serial target, $\mathcal{A}^{(k)} = \{j : x_j^{(k)} = \texttt{[M]} \wedge \hat{t}_j^{(k)} = t_j\}$; if no position matches, we commit the single highest-confidence position as a fallback. After K iterations the block is fully unmasked and the realized TPF is B/K. Greedy is fast but is not always exact: each committed token becomes part of the context for the next forward pass, and committing several context-dependent tokens at once can shift the conditional distribution at the remaining positions away from $\mathbf{t}$. This motivates a second scheme that recovers $\mathbf{t}$ exactly on every block.

Recursive dynamic compaction. To recover $\mathbf{t}$ exactly on every block, we replace greedy acceptance

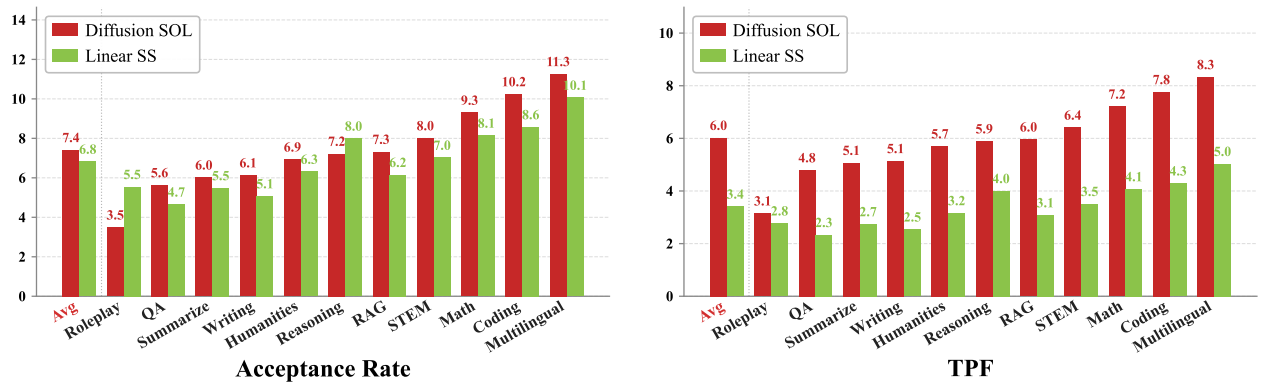


Figure 7 | Visualizing the acceptance rate and TPF across different SPEED-Bench categories for diffusion SOL and linear self-speculation. The average metrics across all categories are highlighted in red.

with a strategic search for the largest *safe* subset of matching positions, as shown in Fig. 6. At each iteration, we rank the N matched positions by model confidence as $(p_1, \ldots, p_N)$ and search for the largest prefix $\{p_1, \ldots, p_k\}$ whose commit is safe—where "safe" means that continuing decoding on the remaining positions still arrives at $\mathbf{t}$. Each safety check itself runs the decoder one level shallower (greedy acceptance) under a simulation budget of 5000 forward passes per block, which is what makes the scheme *recursive*; if the budget is exceeded, the candidate is treated as unsafe and the binary search shrinks the prefix. Because the top-1 match is always safe (committing one position is no different from a serial step), the scheme commits at least one position per iteration and its TPF dominates greedy whenever greedy is exact, while recovering exact-match cases that greedy misses. We use this scheme to report SOL throughout this section; greedy serves as a fast lower-bound proxy.

4.2. SOL Evaluation on SPEED-Bench

We apply recursive dynamic compaction to 713 SPEED-Bench [1] samples spanning 11 categories on the diffusion mode of Nemotron-Labs-Diffusion-8B (instruct), sweeping the block length $B \in \{4, 8, 16, 32\}$. We also report the average accuracy achieved by SOL under different block lengths on the 10 instruct LM benchmarks in Sec. 6.1 to understand their impact.

Observations on diffusion SOL. From Tab. 4, we observe that (1) The diffusion mode exhibits substantial intrinsic parallelism: the SOL acceptance rate reaches 7.60× on average and exceeds 10× on multilingual and coding content at $B = 32$, and grows nearly linearly with B, from 2.89× at $B = 4$ to 7.60× at $B = 32$. (2) Confidence-based sampling, by contrast, achieves only ∼3× TPF at comparable accuracy in the block-32 diffusion-mode results of Sec. 6.1, leaving a notable gap to the 7.60× SOL ceiling. This indicates that confidence-based sampling is far from optimal and that there is substantial headroom for better-designed samplers to capture. (3) Per-category

Table 4 | Per-category SOL acceptance ratio on SPEED-Bench under recursive dynamic compaction. Benchmark accuracy is averaged over 10 instruct LM benchmarks in Sec. 6.1.

Category	BL=32	BL=16	BL=8	BL=4
coding	10.24	7.50	5.32	3.32
humanities	6.93	5.00	3.76	2.76
math	9.30	7.02	4.90	3.20
multilingual	11.26	8.08	5.46	3.37
qa	5.63	4.43	3.46	2.61
rag	7.32	5.67	4.14	2.91
reasoning	7.22	5.06	3.87	2.79
roleplay	3.49	2.76	2.41	2.00
stem	8.01	5.68	4.11	2.98
summarization	6.02	4.48	3.25	2.71
writing	6.13	4.71	3.58	2.62
Acceptance rate	**7.60**	**5.68**	**4.17**	**2.89**
Benchmark Acc	**61.81**	**63.18**	**65.43**	**64.04**

SOL spans a ∼3.2× range, from 3.49× on roleplay to 11.26× on multilingual content. We hypothesize this reflects token-level entropy: templated content has more positions that are confidently determined by partial context, while open-ended generation does not. If this holds, samplers that adapt to the local content type could exploit this variance. (4) A moderately small block length achieves the best accuracy, while larger block lengths degrade it. The trade-off between benchmark accuracy and efficiency, i.e., acceptance rate, indicates that improving diffusion performance under larger block lengths is a critical direction.

Diffusion SOL vs. linear self-speculation. We additionally compare the SOL ceiling with linear self-speculation (Sec. 3) on SPEED-Bench at B=32, visualized in Fig. 7. Two distinct metrics matter here: The *acceptance rate* counts how many tokens are committed per acceptance step. SOL commits multiple positions in a single diffusion forward pass, while linear self-speculation accepts up to k draft tokens per draft+verify cycle. The *real TPF*, in contrast, is the average number of tokens committed per single model

forward pass: SOL incurs only a per-block KV-cache recompute on top of one forward per acceptance step, so its real TPF is close to its acceptance rate; linear self-speculation, however, uses two forwards per cycle (one diffusion draft, one AR verification), so its real TPF is the acceptance rate divided by two. The two settings also target different correctness signals: SOL agrees with the diffusion mode's own serial-denoising output, while linear self-speculation agrees with the AR mode verification. We view this as a fair head-to-head measurement of parallel-decoding potential, considering that the diffusion and AR modes achieve comparable accuracy in Sec. 6.1 and thus both targets are equally credible references for a *correct* token.

From Fig. 7, we observe that: (1) At the acceptance-rate level, linear self-speculation approaches SOL, achieving 6.82× vs. 7.60× overall (approximately 10.3% below the upper bound), with similarly small gaps across categories. Thus, on top of diffusion drafting, applying AR verification is an effective way to approach the SOL ceiling. (2) The real TPF gap, however, is much larger: 6.02× for SOL vs. 3.41× for linear self-speculation, i.e., a 76.5% improvement. Beyond the doubled forward-pass cost, linear self-speculation only commits a contiguous prefix of the draft, discarding confident tokens beyond the first rejection. These two effects together motivate stronger diffusion-mode samplers that can safely commit tokens within a single forward pass, including at non-prefix positions.

Implications for the tri-mode framework. The SOL analysis highlights two key takeaways for the tri-mode framework. *(1)* Diffusion-mode decoding can be a promising approach for highly parallel decoding, provided that an optimized sampler can close the gap between pure confidence-based sampling and the SOL ceiling. *(2)* AR verification is an effective way to sample from diffusion drafts and can approach SOL. However, its additional verification cost and prefix-only acceptance pattern fundamentally cap its real TPF below SOL, even when the diffusion drafter and the AR verifier are well aligned.

5. Nemotron-Labs-Diffusion Family

We deliver the Nemotron-Labs-Diffusion model family in 3B, 8B, and 14B sizes, including base and instruct models as well as VLMs.

5.1. Base Models

To speed up pretraining, we start from the pretrained Ministral3 [21] base models and apply the two-stage training strategy introduced in Sec. 2.1. Specifically, we adopt the pretraining dataset in [22] and perform continuous pretraining for 1T tokens in Stage 1 (pure AR) and 300B tokens in Stage 2 (joint AR and diffusion training with an α of 0.3). The initial learning rate is set to 1e-5 and decayed to 3e-6 using a WSD schedule [23] with the AdamW optimizer and a weight decay of 0.1. We adopt a global batch size of 512 and a sequence length of 4096. The training is performed on 256 NVIDIA H100 GPUs. We release the training and inference pipeline through Megatron Bridge.

5.2. Instruct Models

We perform supervised fine-tuning (SFT) on top of our base models to deliver instruct models. Specifically, we adopt joint AR and diffusion training with an α of 0.3 throughout the SFT process. The initial learning rate is set to 2.5e-6 and decayed to 2.5e-7 using the WSD schedule [23] with the AdamW optimizer and a weight decay of 0.1. We train the model on 45B tokens from the SFT dataset of [24], with a global batch size of 256 and a sequence length of 16k. Following [2], the training pipeline is the same as pretraining except that we do not mask any tokens from the prompt, and the loss is computed only on the answer parts. The training is performed on 256 NVIDIA H100 GPUs.

5.3. Vision-Language Models

We extend Nemotron-Labs-Diffusion to the vision-language setting by adding a vision encoder and a multimodal projector to the diffusion LM backbone. The resulting VLM inherits the joint AR-diffusion training objective, the dual-stream attention pattern, and the tri-mode inference capability. Below, we describe how the VLM is initialized from pretrained components and how image features are integrated into the dual-stream training layout.

Architecture. The VLM augments Nemotron-Labs-Diffusion with a vision encoder and a two-layer MLP projector with 2×2 patch merging. The diffusion training and inference pipeline is shared with the text-only model, with only the vision frontend added.

Weight initialization. We initialize the LM backbone and LM head from the Nemotron-Labs-Diffusion-8B instruct model, which carries the diffusion-aware representations learned during joint AR-diffusion training, and initialize the vision encoder and projector from the corresponding AR VLM (`Ministral3-8B-Instruct-2512`) from the same model family, which provides fully trained visual perception and cross-modal alignment weights.

Because the LM architectures are identical between the two sources, the merge is exact with no parameter mismatch or interpolation. No new parameters are in-

Table 5 | Benchmark our Nemotron-Labs-Diffusion-8B instruct model against SOTA AR and diffusion instruct LMs across scientific QA, instruction following, coding, and math reasoning benchmarks.

Model	**Qwen2.5 7B**	**Qwen3 8B**	**Ministral3-8B Instruct-2512**	**LLaDA-8B Instruct**	**Dream-7B Instruct**	**SDAR-8B Chat**		**Nemotron-Labs-Diffusion-8B (Tri-Mode in One Model)**			
Gen. Mode	AR	AR	AR	Diff.	Diff.	Diff.	Diff.	AR	Diff.	Linear SS	Quad. SS
Scientific QA & Instruction Following											
GPQA	37.12	49.24	42.87	33.30	33.00	40.20	30.80	44.44	43.94	40.40	44.30
IFEval	74.58	87.38	64.31	59.90	62.50	61.40	60.07	68.65	68.32	69.13	71.00
MMLU	74.86	76.66	73.90	65.50	67.00	78.60	78.83	79.85	78.71	79.01	79.95
Coding											
HumanEval	77.44	81.71	71.04	49.40	55.50	78.70	79.27	80.49	78.66	81.71	79.27
MBPP	81.55	81.88	78.97	41.00	58.80	72.00	67.32	85.19	83.86	84.92	85.19
LCB-CPP	12.33	21.09	20.76	4.19	1.25	13.44	11.89	28.85	26.16	24.89	27.70
Math											
Math500	75.10	84.80	83.60	39.20	43.00	78.60	72.40	88.00	85.80	87.60	88.80
GSM8K	91.89	92.42	92.42	79.91	81.00	91.30	88.48	94.01	93.03	93.78	94.16
AIME24	13.75	30.21	27.71	0.00	0.00	16.67	13.33	33.33	46.67	36.67	33.33
AIME25	6.88	22.08	24.58	0.00	3.33	10.00	3.33	33.33	26.67	30.00	36.67
Average over All Tasks											
Accuracy	54.55	62.75	58.02	37.24	40.54	54.09	50.57	63.61	63.18	62.81	64.04
TPF	1.00	1.00	1.00	1.00	1.00	1.00	1.75	1.00	2.57	5.99	6.38

troduced; the vocabulary and embedding dimensions remain unchanged.

Continued SFT. Starting from the merged initialization, we finetune the full model (LM backbone, vision encoder, and projector) with the same joint AR-diffusion objective used for the text-only instruct model, on multimodal instruction-following data [25].

Asymmetric dual stream. A straightforward extension doubles all tokens in the noisy half, including vision tokens. However, vision tokens are never masked; only text response tokens are subject to forward corruption. Carrying vision tokens in the noisy half, therefore, adds FLOPs without contributing to the diffusion loss. For high-resolution images, this overhead is substantial. We address this with an *asymmetric dual-stream* layout that strips all vision token positions from the noisy half:

$$\left[\, \tilde{x}_t^{(\text{text},\, L_{\text{text}})} \;\middle|\; x^{(L)} \,\right], \qquad L_{\text{text}} = L - N_{\text{vis}}, \quad (10)$$

where N_{vis} is the number of vision tokens. The clean half retains the full sequence, including vision tokens, preserving complete visual context for the AR objective and for cross-stream conditioning. The total sequence length becomes $L_{\text{text}} + L$ instead of $2L$, and the reduction of FLOPs of attention scales with the vision tokens N_{vis}/L.

6. Evaluation and Analysis

6.1. Benchmark Instruct Models

Baselines and benchmarks. We compare our Nemotron-Labs-Diffusion-8B instruct model against SOTA AR instruct models (Qwen3-8B, Qwen2.5-7B, and Ministral3-8B Instruct) and SOTA diffusion instruct models (LLaDA-8B Instruct [2], Dream-7B Instruct [3], and SDAR-8B Chat [4]). We evaluate all modes of our model: AR, diffusion, and self-speculation, including both linear and quadratic self-speculation modes (denoted as linear SS and quadratic SS). We use LoRA-enhanced linear self-speculation by default. For the diffusion model evaluation of Nemotron-Labs-Diffusion-8B and SDAR-8B Chat [4], we also report tokens per forward (TPF) by selecting different denoising thresholds [16]. All models are evaluated in the non-thinking mode.

We evaluate across scientific QA and instruction following (GPQA, IFEval, MMLU), coding (HumanEval, MBPP, LiveCodeBench-CPP), and math reasoning (Math500, GSM8K, AIME24, AIME25). We use NeMo-Skills [26] as the evaluation framework for all AR baselines and our Nemotron-Labs-Diffusion-8B, and use the official evaluation pipelines provided in the original papers for the diffusion baselines [2, 3, 4].

Observations. As shown in Tab. 5, we observe that, compared to SOTA AR and diffusion instruct LMs, our Nemotron-Labs-Diffusion-8B achieves both higher accuracy and efficiency across all modes. More specifically, (1) In terms of AR performance, Nemotron-Labs-Diffusion-8B in AR mode delivers +0.86% higher average accuracy than Qwen3-8B and outperforms all other AR baselines, demonstrating that the joint AR–diffusion training objective effectively preserves strong AR accuracy. In fact, the ablation study under a controlled setting in Sec. 2.4 indicates that adding the diffusion objective can maintain or slightly improve AR accuracy, potentially due to an improved

Table 6 | Per-task TPF achieved by linear self-speculation w/ and w/o LoRA tuning across 3B/8B/14B scales.

Model Scale	Setting	GPQA	IFEval	HumanEval	MBPP	Math500	GSM8K	AIME24	AIME25	MMLU	LCB-CPP	Avg TPF	Avg Acc
3B	w/o LoRA	3.07	2.97	4.77	3.74	4.94	4.01	4.33	4.53	2.42	3.34	3.81	55.00
	w/ LoRA	3.57	3.30	5.56	4.23	5.63	4.55	4.99	5.11	2.74	3.95	4.36	55.00
8B	w/o LoRA	5.10	4.32	4.62	3.44	5.43	4.47	5.38	5.05	3.25	4.14	4.52	62.88
	w/ LoRA	6.64	5.52	5.82	4.44	7.36	5.89	7.44	6.92	4.08	5.70	5.99	62.81
14B	w/o LoRA	5.31	3.86	6.75	4.42	5.01	4.54	4.47	3.92	4.95	3.42	4.67	66.35
	w/ LoRA	6.07	4.74	8.11	5.22	6.72	5.79	5.88	5.41	7.22	4.46	5.96	66.36

ability to predict the future. (2) The diffusion mode decodes 2.57× TPF while achieving +0.43% higher average accuracy than Qwen3-8B. Compared to existing diffusion LMs, Nemotron-Labs-Diffusion-8B outperforms SDAR-8B Chat by +9.09% in average accuracy and better maintains accuracy under larger decoding parallelism, as shown in Fig. 1 (b). (3) LoRA-tuned linear self-speculation maintains comparable accuracy to the diffusion mode while further boosting TPF to 5.99×, indicating the effectiveness of aligning the diffusion drafter with the AR target via lightweight LoRA tuning. (4) Quadratic self-speculation can achieve the highest TPF of 6.38×, as it prepares the next block for all possible acceptance positions at a quadratic cost. However, due to the use of FlexAttention with less optimized kernels for the dedicated attention mask [20], the real-device efficiency of quadratic self-speculation falls behind the linear one according to Fig. 1 (b). As such, we use linear self-speculation by default.

Remark. The tri-mode design enables Nemotron-Labs-Diffusion to serve different deployment needs within a single model: (1) The AR mode matches or surpasses SOTA AR LMs in accuracy, meaning that Nemotron-Labs-Diffusion can serve as a drop-in replacement for any application that currently uses an AR model, with no pipeline changes required. (2) The diffusion mode enables one-for-all flexibility: by adjusting the denoising threshold, a single model can achieve a range of accuracy-throughput trade-offs, as illustrated in Fig. 1 (b). (3) Self-speculation is promising for achieving significant inference speedup through the synergy between AR and diffusion. While it sacrifices the flexibility of the diffusion mode by only accepting prefix tokens, it provides a reliable mechanism to verify diffusion drafts, which can lead to substantial inference acceleration, as demonstrated in Sec. 6.5.

Improve diffusion decoding with a better sampler. We evaluate the effectiveness of the proposed sampler in Sec. 3.2. We apply it on top of our instruct model with a block size of 32 and report the average accuracy across all ten tasks under different denoising thresholds to obtain the accuracy–TPF trade-off. As shown in Fig. 8, the trained sampler shifts the entire Pareto frontier upward, delivering higher TPF at the same accuracy (e.g., 1.3× TPF) or higher accuracy at the same TPF (e.g., +10.6% accuracy). The improvements from this simple design suggest that part of the gap between the realized diffusion-mode TPF and the SOL ceiling in Sec. 4 can be closed by learning the acceptance policy itself.

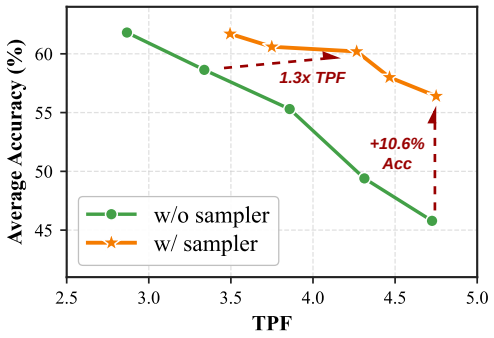


Figure 8 | Comparing the accuracy-TPF trade-offs achieved w/ and w/o a sampler.

The impact of LoRA tuning for linear self-speculation. We perform an ablation study on the impact of LoRA tuning for aligning diffusion drafters and AR verifiers. As shown in Tab. 6, we observe that (1) even without LoRA adapters, linear self-speculation already achieves nontrivial TPF, e.g., 4.67× for our 14B model, with larger model scales generally leading to higher TPF; and (2) adding LoRA tuning consistently improves TPF, yielding 14.4%/32.5%/27.6% relative gains at the 3B/8B/14B scales. We also note that the small accuracy gap between different self-speculation settings and the AR mode is due to kernel mismatches between 1-token decoding and multi-token prefilling.

6.2. Extend to More Model Scales

Baselines and benchmarks. We extend the evaluation to two additional scales, Nemotron-Labs-Diffusion-3B/14B, under the same evaluation protocol as Sec. 6.1, and benchmark against SOTA open-source AR instruct models at the corresponding scales.

Observations. As shown in Tab. 7, we observe that: (1) Nemotron-Labs-Diffusion maintains consistent improvements in accuracy and efficiency across scales and generation modes. For example, using LoRA-tuned linear self-speculation, our Nemotron-Labs-Diffusion-3B/14B outperforms the strongest baselines, Qwen3-4B/14B, by +1.77%/+1.19% in accuracy while achieving 4.36×/5.96× TPF, respectively.

Table 7 | Benchmark our Nemotron-Labs-Diffusion-3B/14B instruct models against SOTA AR instruct models.

Model	Gen. Mode	GPQA	IFEval	HumanEval	MBPP	Math500	GSM8K	AIME24	AIME25	MMLU	LCB-CPP	Avg Acc	Avg TPF
3B Scale													
Llama-3.2-3B-Instruct	AR	27.78	69.69	65.85	66.93	32.60	62.09	0.00	0.00	62.45	9.25	39.90	1.00
Phi-4-mini-Instruct	AR	40.91	71.16	77.44	75.13	70.80	76.57	10.00	6.67	53.37	9.91	49.20	1.00
Ministral3-3B-Instruct-2512	AR	28.79	57.55	64.79	68.39	71.90	87.41	12.50	12.71	63.21	12.50	47.97	1.00
Qwen3-4B	AR	37.18	72.20	75.91	72.49	68.40	92.19	10.63	10.63	77.05	15.58	53.23	1.00
Nemotron-Labs -Diffusion-3B (Tri-Mode)	AR	39.39	69.39	76.22	71.16	77.60	87.87	23.33	16.67	71.70	21.37	55.50	1.00
	Diff.	33.84	68.93	74.39	73.54	74.80	88.40	16.67	10.00	72.06	16.30	52.90	1.91
	Linear SS	35.86	68.96	75.00	70.11	77.40	87.79	26.67	16.67	71.89	20.04	55.00	4.36
	Quad. SS	42.93	71.36	79.27	76.46	78.20	88.25	13.33	16.67	72.12	19.60	55.80	5.42
14B Scale													
Gemma-3-12B-IT	AR	38.32	85.73	58.23	85.45	84.55	90.45	23.75	16.88	76.41	20.54	58.03	1.00
Phi-3-Medium-14B	AR	37.56	85.75	70.43	76.65	43.35	89.69	1.88	0.62	76.67	10.79	49.34	1.00
Phi-4-14B	AR	56.94	68.96	84.60	83.93	79.95	92.27	19.17	15.83	84.76	21.75	60.82	1.00
Ministral3-14B-Instruct-2512	AR	52.02	71.51	72.56	82.47	86.30	92.80	36.25	29.38	79.88	26.54	62.97	1.00
Qwen3-14B	AR	50.51	88.36	83.54	87.30	85.40	94.31	33.33	20.00	81.51	27.42	65.17	1.00
Nemotron-Labs -Diffusion-14B (Tri-Mode)	AR	54.55	68.50	86.59	85.19	88.40	91.36	46.67	43.33	82.51	27.48	67.46	1.00
	Diff.	48.99	69.03	83.54	82.80	85.80	93.71	43.33	50.00	82.17	25.77	66.51	2.74
	Linear SS	47.47	70.06	85.37	84.66	86.60	92.04	50.00	40.00	81.11	26.32	66.36	5.96
	Quad. SS	52.02	72.15	87.20	85.45	88.00	92.12	53.33	40.00	82.45	28.74	68.15	6.92

Table 8 | Benchmark our Nemotron-Labs-Diffusion-8B base model against SOTA AR and diffusion base LMs across coding, math, knowledge, and commonsense reasoning benchmarks.

Model	Gen. Mode	Human Eval	Human Eval+	MBPP	MBPP+	GSM8K	Minerva Math	MMLU	ARC-E	ARC-C	Hella swag	PIQA	Wino grande	Avg Acc	Avg TPF
Llama-3.1-8B	AR	35.37	28.66	48.80	61.90	54.06	18.22	65.15	81.31	53.41	78.93	81.18	77.43	57.04	1.00
Ministral3-8B	AR	42.68	38.41	61.60	76.98	80.21	44.58	76.39	86.15	60.75	79.01	80.74	73.48	66.75	1.00
Qwen3-8B	AR	64.63	56.71	69.40	83.07	86.73	52.94	76.93	81.90	53.16	78.59	79.22	75.69	71.58	1.00
LLaDA-8B	Diff.	32.32	27.44	40.80	51.85	70.96	27.30	65.86	73.78	49.15	71.05	73.88	74.66	54.92	1.00
Dream-7B	Diff.	54.88	49.39	56.80	74.60	77.18	39.60	67.00	82.20	59.13	73.73	75.52	73.56	65.30	1.00
Nemotron-Labs -Diffusion-8B (Tri-Mode)	AR	60.37	53.05	68.20	82.54	88.25	66.00	74.68	83.38	58.11	76.08	80.09	71.98	71.89	1.00
	Diff.	62.80	57.32	67.00	81.75	87.26	65.16	74.68	83.38	58.11	76.08	80.09	71.98	72.13	2.06
	Linear SS	63.41	56.10	67.20	81.75	88.17	67.38	74.68	83.38	58.11	76.08	80.09	71.98	72.36	4.67
	Quad. SS	62.20	54.88	67.60	81.48	88.48	66.24	74.68	83.38	58.11	76.08	80.09	71.98	72.10	7.04

(2) Based on the performance of Nemotron-Labs-Diffusion-3B/8B/14B, larger LMs generally more readily unlock parallel diffusion abilities, as the TPF of diffusion/self-speculation modes grows broadly with scale. For example, the TPF of linear self-speculation increases from 4.36× to 5.96× when scaling from 3B to 14B. We attribute this to the stronger future-prediction abilities of larger models, which yield more reliable draft predictions.

6.3. Benchmark Base Models

Baselines and benchmarks. We compare Nemotron-Labs-Diffusion-8B against SOTA AR base LMs and two representative diffusion LMs (LLaDA-8B [2] and Dream-7B [3]). We evaluate on coding benchmarks (HumanEval, HumanEval+, MBPP, MBPP+), math reasoning (GSM8K, Minerva Math), knowledge (MMLU), and commonsense reasoning (ARC-E, ARC-C, Hellaswag, PIQA, Winogrande).

Observations. As shown in Tab. 8, we observe findings consistent with the instruct model results. Our Nemotron-Labs-Diffusion-8B base model achieves both higher accuracy and efficiency across all modes: (1) the AR mode delivers +5.14%/+0.31% higher average accuracy than Ministral3-8B/Qwen3-8B; (2) the diffusion mode delivers +17.21%/+6.83% higher accuracy than LLaDA-8B/Dream-7B; and (3) self-speculation achieves 4.67× TPF (linear) and 7.04× TPF (quadratic) with over 0.5% higher average accuracy compared to the strongest baseline, Qwen3-8B.

6.4. Benchmark VLMs

Benchmarks and evaluation settings. We evaluate on a diverse set of VLM benchmarks spanning two categories. Short-answer benchmarks require brief, factual responses: AI2D [27], ChartQA [28], DocVQA [29], MMMU [30], MathVista [31], and RealWorldQA [32]. Long-answer benchmarks require extended chain-of-thought reasoning: MMMU-Pro-V [33]. We benchmark against existing diffusion VLMs [34, 35, 36, 37]. All benchmarks are evaluated using VLMEvalKit [38] under the same prompts and post-processing as the AR baseline (Ministral3 VLM). Throughput (tokens per second, TPS) is measured on a single NVIDIA H100 GPU with identical prompt batching for fair comparison.

Observations. As shown in Tab. 9, we compare our Nemotron-Labs-Diffusion-VLM against existing diffusion VLMs in three modes: diffusion, AR, and linear self-speculation decoding. We observe that (1) In terms of AR performance, our model delivers 1.3% higher average accuracy than the strongest baseline, LLaDA-V-8B. (2) The diffusion mode provides 2.46×–3.15× TPF while maintaining competitive accuracy. (3) Linear self-speculation preserves near-AR accuracy, with only a 0.1% average accuracy drop, while further increasing decoding parallelism to 3.63×–7.45× TPF, where the higher end is achieved

Table 9 | Benchmarking discrete diffusion VLMs and Nemotron-Labs-Diffusion-VLM across tasks. The diffusion mode of Nemotron-Labs-Diffusion-VLM uses denoising threshold τ=0.9.

Model	Gen. Mode	AI2D	ChartQA	DocVQA	MMMU	MMMU Pro-10c	MMMU Pro-V-CoT	Math Vista	RealWorld QA	TPF	Acc
MMaDA	Diff.	67.4	9.6	9.5	30.2	16.5	8.5	33.4	49.2	1	28.0
LaViDa	Diff.	70.0	59.0	64.6	43.3	28.7	10.5	44.8	54.5	1	46.9
Dimple	Diff.	74.4	63.4	37.7	45.2	23.8	12.4	42.3	55.4	1	44.3
LLaDA-V-8B	Diff.	**77.8**	78.3	83.9	48.6	**35.2**	18.6	59.7	**63.2**	1	58.2
Nemotron-Labs -Diffusion -VLM-8B	AR	75.0	**81.3**	**89.2**	50.3	32.6	**24.3**	**60.4**	62.6	1	**59.5**
	Diff.	74.7	76.6	88.3	**50.4**	31.7	22.2	58.5	60.3	2.46 all samples 2.80 tok>100 3.15 tok>200	57.9
	Linear SS	74.9	81.2	89.3	50.0	32.8	24.1	60.7	62.4	3.63 all samples 6.03 tok>100 7.45 tok>200	59.4

for responses exceeding 200 tokens. This implies that the advantage of our model is most pronounced on tasks requiring longer reasoning. These results demonstrate that the joint AR–diffusion training framework extends effectively to the vision-language setting, preserving the broad capabilities of the LM backbone while enabling efficient multi-token decoding.

6.5. Inference Efficiency

We analyze and compare the deployment efficiency of Nemotron-Labs-Diffusion against MTP/Eagle3-style speculative decoding.

Self-speculation vs. MTP. MTP methods such as Eagle3 [8] have become the default choice for efficient LLM deployment at low concurrency, where a small model is used to draft multiple future tokens, and then a single forward pass of a larger AR model verifies and accepts some of them. This schedule is more efficient at low concurrency because the memory transfer cost is similar between token-by-token generation and verification passes, while the latter can accept multiple tokens in a single pass. The two main bottlenecks of Eagle3 are: (1) the draft model has limited capacity and is less reliable beyond a short horizon; and (2) proposals are generated recursively, so even if the draft model is tiny, it still incurs the cost of the embedding layer and LM head. In contrast, Nemotron-Labs-Diffusion provides unique advantages through (1) significantly higher acceptance length, and (2) token-parallel drafting that enables better GPU utilization.

Setup. We deploy Nemotron-Labs-Diffusion-8B with the SGLang server and profile it on NVIDIA GB200, RTX Pro 6000, and DGX Spark under different concurrency levels, comparing against Qwen3-8B-Eagle3, with results shown in Fig. 1 (c) and Fig. 9. Evaluations are conducted on SPEED-Bench [1] across four categories (math, coding, reasoning, and multilingual), and limiting generation length to 1024 tokens to avoid repetition/hallucinations. We perform a grid search over hyperparameters for Eagle3, whereas for Nemotron-Labs-Diffusion we only vary the *block length*. We additionally report a SOL throughput estimate mentioned in Sec. 4, providing a reference ceiling for current self-speculation infrastructure.

Observations. As shown in Fig. 1 (c) and Fig. 9, we observe that: (1) Linear self-speculation consistently improves user throughput over the AR mode across all three GPUs, achieving up to 3.3× speedup over AR on GB200 (3.97× speedup and 1015 tok/sec with an optimized kernel), as shown in Fig. 9 (c), and pushing the absolute throughput at batch size 1 to 277/525 tok/sec on RTX Pro 6000 (3.46×/2.35× over AR) and 77.5/112.5 tok/sec on DGX Spark (3.14×/2.69× over AR) under FP8/INT4 quantization, demonstrating its effectiveness as a drop-in low-concurrency accel-

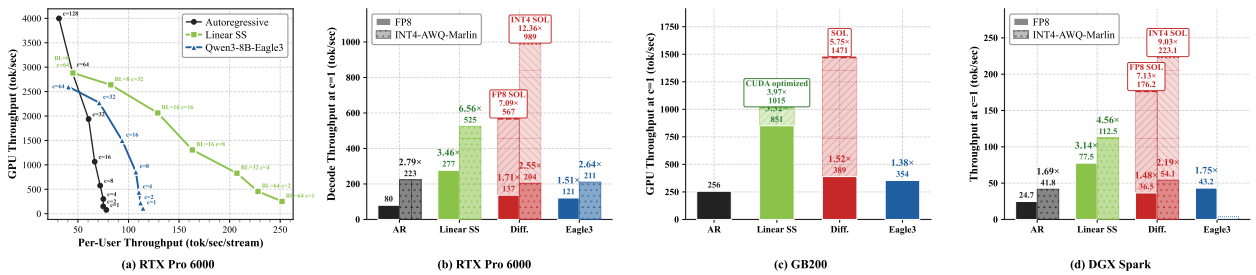


Figure 9 | (a) System vs. per-user throughput trade-off on an NVIDIA RTX Pro 6000 GPU; (b)/(c)/(d): The throughput under a concurrency of 1 on NVIDIA RTX Pro 6000, GB200, and DGX Spark, respectively.

Table 10 | Per-category acceptance length on SPEED-Bench [1]. Comparing Native / LoRA for Nemotron-Labs-Diffusion-8B and Qwen3-8B-Eagle3 / Qwen3-9B-MTP, all with draft length 31.

Category	Native	LoRA	Eagle3	MTP
coding	6.61	8.57	3.14	5.97
math	6.24	8.14	2.79	4.80
reasoning	6.18	7.99	3.40	3.68
multilingual	7.96	10.06	1.91	4.47
humanities	5.01	6.31	3.12	3.76
qa	4.01	4.65	2.63	3.50
rag	5.07	6.15	3.06	4.75
roleplay	4.66	5.54	2.10	2.32
stem	5.55	7.02	2.92	4.45
summarization	4.47	5.48	2.66	3.69
writing	4.28	5.07	2.81	3.21
Average	**5.46**	**6.82**	**2.75**	**4.24**
4 category avg	**6.75**	**8.69**	**2.81**	**4.73**

eration scheme. (2) Compared with Eagle3, linear self-speculation delivers a 2.4×/2.3×/1.8× speedup at batch size 1 on GB200/RTX Pro 6000/DGX Spark and achieves better trade-offs between system throughput and per-user throughput, as shown in Fig. 1 (c) and Fig. 9 (a). This indicates that diffusion drafting paired with AR verification is a more effective acceleration mechanism than auxiliary-head MTP due to its higher acceptance length. (3) The SOL ceiling reveals substantial remaining headroom: on RTX Pro 6000, the projected SOL throughput reaches 7.09×/12.36× over AR under FP8/INT4 quantization, roughly 2× above linear self-speculation.

Acceptance length per category. As shown in Tab. 10, Nemotron-Labs-Diffusion achieves significantly higher acceptance length than both Eagle3 and MTP across all categories, with average acceptance lengths of 5.46/6.82 for Native/LoRA-tuned Nemotron-Labs-Diffusion versus 2.75/4.24 for Eagle3/MTP. The gap further widens to 6.75/8.69 vs. 2.81/4.73 on the four diffusion-friendly categories (coding, math, reasoning, multilingual), implying that diffusion drafting yields more reliable multi-token proposals, especially on structured tasks with strong syntactic or semantic constraints.

7. Related Work

Diffusion language models. To overcome the token-by-token decoding nature of AR LMs, diffusion LMs, both continuous [39, 40, 41] and discrete [42, 43, 44, 45, 46, 47], have been proposed to perform non-AR decoding and thus enable parallel token generation. Among them, masked diffusion LMs [43, 45, 44, 2, 3] have been successfully scaled up (e.g., LLaDA [2] and Dream [3]). Follow-up work has further explored alternative diffusion LM paradigms [48, 49, 6], and scaled them to larger scales [50, 51, 52] or domain-specific specialists such as coding agents [53, 54, 55, 56], explored dedicated reinforcement learning schemes [57, 58], and extended them to more modalities [36, 34]. Compared to AR LMs, diffusion LMs have been demonstrated to be better learners under data-constrained settings [59] and show improved performance in planning [3] and text embedding [60].

Diffusion language model acceleration. Despite the acceleration potential of large diffusion LMs [2, 3], the gap between bidirectional attention and KV caching, along with the one-token-per-step denoising process, limits their achievable speed-up. To address these challenges, dedicated caching strategies [61, 62, 16] have been developed to reuse computations and approximate bidirectional attention. In addition, to realize the potential of parallel token generation, confidence-based sampling [16], guidance from AR models [63], and adaptive decoding with certainty and positional priors [64] have been proposed. Beyond these training-free methods, [65, 3] propose initializing diffusion LMs from AR models with token shifts to accelerate diffusion LM training. Block Diffusion [9] combines AR and diffusion by performing block-wise AR and in-block diffusion to support native KV caching. Follow-up works [13, 10, 4, 66, 67] also convert pretrained AR models or diffusion LMs into block-wise ones. [14, 7] further explore combining AR and diffusion through either joint training or LoRA modules dedicated to diffusion.

8. Insights and Future Directions

We deliver Nemotron-Labs-Diffusion, a tri-mode language model family trained via joint AR-diffusion optimization that unifies AR, diffusion, and self-speculation within a single model. The resulting base, instruct, and vision-language models outperform SOTA open-source AR/diffusion LMs in both accuracy and efficiency. The training and analysis of tri-mode LMs reveal the following insights:

1. **Tri-mode generation arises naturally from joint AR-diffusion training.** By enabling both AR and non-AR parallel token prediction within a single model, the joint training objective simultaneously produces three inference modes without any mode-specific architectural modifications.
2. **AR and diffusion losses are complementary, not competing.** The two objectives mutually benefit each other and peak at the same loss coefficient (α=0.3). Adding the AR loss induces left-to-right linguistic priors for diffusion, and the

diffusion loss preserves or slightly improves AR accuracy through better future planning.

3. **Self-speculation outperforms MTP methods.** Instead of relying on auxiliary prediction heads, self-speculation leverages diffusion to generate high-quality multi-token drafts and uses AR verification to ensure correctness, achieving higher acceptance rates and better efficiency.
4. **Variance reduction is critical for diffusion training.** The diffusion loss introduces intrinsically high variance due to random masking with variable noise levels. More sufficiently trained AR starting points (e.g., via two-stage training) or variance-reduction training techniques (e.g., global loss averaging) can improve training effectiveness.
5. **Linear self-speculation is currently the most efficient mode.** Linear self-speculation achieves the best efficiency in the current infrastructure. Quadratic self-speculation achieves higher TPF per step, making it more promising at batch size 1 with improved infrastructure support.
6. **Diffusion-mode decoding has substantial headroom.** Our SOL analysis shows the potential to correctly predict 76.5% more tokens per forward pass than the current best strategy (linear self-speculation), indicating a more promising upper bound for parallel decoding than speculative decoding based only on prefix decoding.

Looking forward, these insights shed light on several promising directions for further improvements:

1. **Closing the gap between practical diffusion decoding and its SOL upper bound.** Our SOL analysis suggests that diffusion-mode decoding could offer a more attractive path toward parallel decoding than linear decoding, due to its non-prefix acceptance pattern and therefore higher upper bound. However, current confidence-based samplers remain far from this upper bound. Developing optimized samplers that more reliably identify correct tokens, or more advanced training schemes that enable more aggressive parallel sampling of conditionally independent tokens, is a promising direction for closing this gap.
2. **Improving draft-verification alignment for self-speculation.** Given the strong practical speedup achieved by self-speculation, an important future direction is to better align the diffusion draft mode with the AR verification mode during training, thereby improving the acceptance rate. In addition, the cost of drafting can be further reduced by using nested subnets, where weight-shared smaller subnets generate drafts through specialized training techniques [68, 69].
3. **Beyond prefix-only AR verification.** Current AR verification accepts drafted tokens only in a prefix-wise manner, which does not fully exploit the non-AR nature of diffusion drafts. A promising direction is to explore diffusion-mode verification, potentially using another diffusion verifier, to validate multiple non-contiguous drafted tokens and further improve the effective acceptance rate.
4. **Enabling higher-level parallelism in diffusion generation.** Although diffusion decoding enables parallel token prediction, its generation order still exhibits a strong left-to-right tendency and mainly provides token-level parallelism. Future training algorithms that encourage segment-level or paragraph-level parallelism could better unlock the global planning ability and efficiency potential of diffusion-mode generation.

References

[1] Talor Abramovich, Maor Ashkenazi, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, Yonatan Geifman, et al. Speed-bench: A unified and diverse benchmark for speculative decoding. *arXiv preprint arXiv:2604.09557*, 2026.

[2] 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. *arXiv preprint arXiv:2502.09992*, 2025.

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

[4] Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[5] Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

[6] Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order gpt as masked diffusion model: Decoupling formulation and architecture. *arXiv preprint arXiv:2506.19935*, 2025.

[7] Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025.

[8] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

[9] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

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

[11] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[12] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[13] 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. *arXiv preprint arXiv:2509.26328*, 2025.

[14] 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. *arXiv preprint arXiv:2509.04185*, 2025.

[15] DeepSeek-AI. Deepseek-v3 technical report, 2024.

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

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

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[19] Aleksei Samarin et al. Lk losses: Direct acceptance rate optimization for speculative decoding. *arXiv preprint arXiv:2602.23881*, 2026.

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

[21] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

[22] Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025.

[23] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[24] Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, et al. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*, 2026.

[25] Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, Leandro von Werra, Aritra Roy Gosthipaty, and Andrés Marafioti. Finevision: Open data is all you need, 2025.

[26] NVIDIA Corporation. Nemo-skills: A toolkit for improving skills of large language models. https://github.com/NVIDIA-NeMo/Skills, 2024. GitHub repository.

[27] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*, pages 235–251, 2016.

[28] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, 2022.

[29] Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. Docvqa: A dataset for vqa on document images. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209, 2021.

[30] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567, 2024.

[31] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] xAI. Realworldqa, 2024.

[33] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhu Chen, and Graham Neubig. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

[34] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[35] Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*, 2025.

[36] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

[37] Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*, 2025.

[38] Haodong Duan, Xinyu Fang, Junming Yang, Xiangyu Zhao, Yuxuan Qiao, Mo Li, Amit Agarwal, Zhe Chen, Lin Chen, Yuan Liu, Yubo Ma, Hailong Sun, Yifan Zhang, Shiyin Lu, Tack Hwa Wong, Weiyun Wang, Peiheng Zhou, Xiaozhe Li, Chaoyou Fu, Junbo Cui, Jixuan Chen, Enxin Song, Song Mao, Shengyuan Ding, Tianhao Liang, Zicheng Zhang, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, Dahua Lin, and Kai Chen. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. *arXiv preprint arXiv:2407.11691*, 2024.

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

[40] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

[41] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.

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

[43] Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*, 2022.

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

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

[46] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

[47] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

[48] Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025.

[49] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *arXiv preprint arXiv:2506.10892*, 2025.

[50] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu,

Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

[51] Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

[52] Google DeepMind. Gemini diffusion, 2025. Model page: state-of-the-art, experimental text diffusion model.

[53] Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

[54] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

[55] Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

[56] Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

[57] Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.

[58] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

[59] Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*, 2025.

[60] Siyue Zhang, Yilun Zhao, Liyuan Geng, Arman Cohan, Anh Tuan Luu, and Chen Zhao. Diffusion vs. autoregressive language models: A text embedding perspective. *arXiv preprint arXiv:2505.15045*, 2025.

[61] Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

[62] Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

[63] Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025.

[64] Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025.

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

[66] Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*, 2025.

[67] Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*, 2026.

[68] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*, 2024.

[69] Yonggan Fu, Zhongzhi Yu, Junwei Li, Jiayi Qian, Yongan Zhang, Xiangchi Yuan, Dachuan Shi, Roman Yakunin, and Yingyan Celine Lin. Amoeballm: Constructing any-shape large language models for efficient and instant deployment. *Advances in Neural Information Processing Systems*, 37:78299–78319, 2024.

[70] Dhruv Nathawani, Shuoyang Ding, Vitaly Lavrukhin, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Boris Ginsburg, and Jane Polak Scowcroft. Nemotron-Post-Training-Dataset-v2, August 2025.

A. Diffusion Sampler Details

This appendix details the sampler introduced in Sec. 3.2, including its architecture, input features, and training trajectory collection.

Architecture and feature engineering. As shown in Fig. 10, the sampler operates on top of the frozen backbone and adds negligible parameter overhead (∼0.06%, with 4.8M compared to the 8B backbone). It is a 4-layer lightweight Transformer with a hidden dimension of d=384. It attends bidirectionally over the current block, followed by a per-position linear head with a sigmoid output. Each input position is represented by a 144-dimensional feature: PCA-compressed semantic embeddings of the top-3 predictions, as well as statistics summarizing the output distribution (e.g., top-1 probability, margin, top-3 mass, and entropy). The semantic embedding of the model's own top-1 prediction is by far the most informative feature. We also find cross-position attention to be essential as an MLP-only ablation drops accuracy-TPF AUC by 10 percentage points, indicating that the sampler must jointly reason about which positions are mutually safe to commit.

Data collection for sampler training. To train the sampler, we collect ∼20M denoising trajectories from Nemotron-Labs-Diffusion-8B on [70] (math, code, STEM, and chat subsets) at block lengths $B \in 8, 32$. We use two complementary trajectory policies: (i) standard confidence decoding (k=1), where the block is decoded one token at a time in confidence order; and (ii) a hybrid policy that first commits ground-truth tokens whenever the model's top-1 prediction already agrees with them, then falls back to confidence for the remaining positions. At each intermediate step of every trajectory, we store the 144-dimensional per-position features and the binary label $\mathbf{1}$[current top-1 = final ID], where the *final ID* is the token ultimately committed at that position once the block is fully decoded under the same policy. Training uses per-position binary cross-entropy on masked positions, with AUC on a held-out trajectory split as the early stopping criterion. The resulting accuracy–TPF gains over confidence thresholding are empirically reported in Sec. 6.1 and Fig. 8.

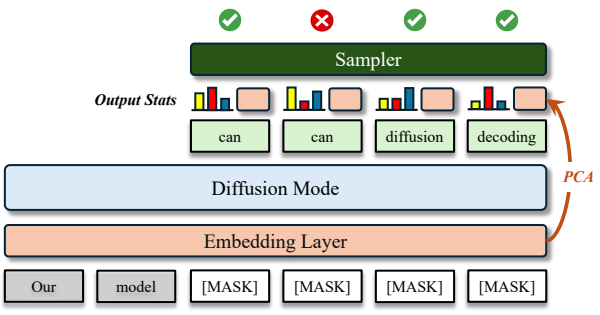


Figure 10 | An illustration of the sampler design on top of the diffusion mode.

B. LoRA-Enhanced Linear SS

We provide a visualization of the enhanced linear self-speculation w/ LoRA in Fig. 11.

C. Quadratic SS Details

We provide more details about quadratic self-speculation, which is visualized in Fig. 12. Specifically, let $[x_1, \ldots, x_n]$ denote the currently verified prefix, and let k be the speculative width. At generation step t+1, we reuse the k speculative tokens from the previous step, denoted $\{x^t_{n+j}\}_{j=2}^{k+1}$, and interleave k fresh mask tokens after each speculative token, yielding the quadratic input:

$$X_m^{t+1} = [x_1, \ldots, x_n, x_{n+1}] + [x^t_{n+2}, m_1, \ldots, m_k] + \cdots + [x^t_{n+k+1}, m_1, \ldots, m_k], \tag{11}$$

where x_{n+1} is the next token generated autoregressively at step t+1 (and is thus immediately verified), and the total number of inserted masks is k^2.

Parallel draft and verification. Feeding X_m^{t+1} via a structured attention mask into our model produces two types of outputs in a single forward pass. First, the model generates next-token predictions for the speculative tokens in a causal manner, yielding $\{x^{t+1}_{n+j}\}_{j=2}^{k+1}$. These tokens are used to verify the previous speculative draft through sequential comparison [7]: we accept the longest prefix that satisfies a verification criterion (e.g., $x^{t+1}_{n+j} = x^t_{n+j}$, as detailed later), commit the accepted tokens to the verified prefix, and stop verification at the first mismatch. Second, in the same forward pass, the model predicts the tokens corresponding to the newly inserted masks $\{m_r\}_{r=1}^{k}$ in parallel; we treat these predictions as the draft tokens that will serve as $\{x^{t+1}_{n+j}\}_{j=2}^{k+1}$ in the next iteration. The interleaved quadratic layout ensures that, even if verification fails early at some position, newly inserted mask positions remain that still yield fresh speculative tokens for the next step, so each iteration consistently produces k tokens to verify [7].

Verification with the AR-diffusion ensemble. For verification, the simplest choice is to use the AR predictions on the speculative tokens. In addition, our tri-mode model provides a complementary verification signal from the diffusion pathway: for each speculative token x^t_{n+j} in Eq. 11, we can use the diffusion prediction at the first newly inserted mask position m_1 immediately following it as an alternative

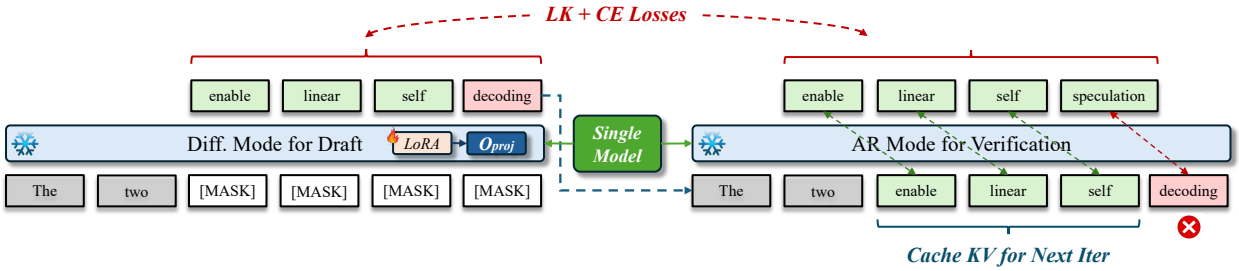


Figure 11 | An illustration of LoRA training on the diffusion drafter of the linear self-speculation mode.

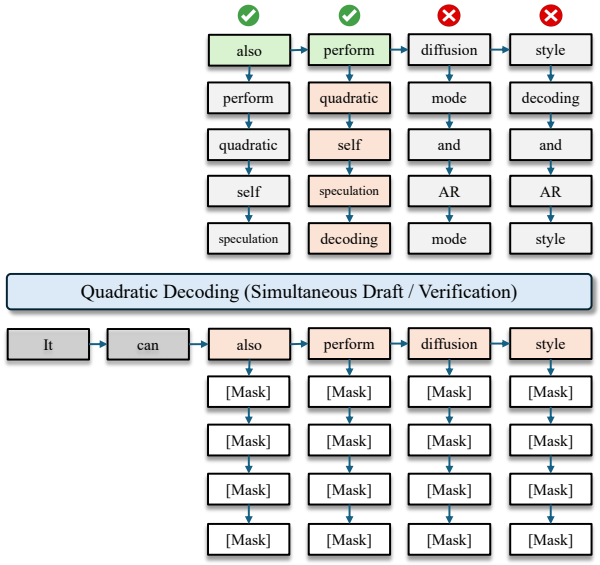


Figure 12 | An illustration of quadratic self-speculation with simultaneous drafting and verification. ■ denotes draft tokens that match AR verification, and ■ denotes the block that originates from the last matched token and is to be verified in the next iteration.

verifier for the same token. Concretely, the AR verifier uses the causal logits $p_\theta^{\mathrm{AR}}(\cdot \mid x_{<n+j})$ at position $n+j$, while the diffusion verifier uses the denoising logits $p_\theta^{\mathrm{diff}}(\cdot \mid X_m^{t+1}, t)$ produced for the corresponding m_1 position. Generally, we can form an AR-diffusion ensemble verifier by combining the two distributions:

$$p_\theta^{\mathrm{ens}}(\cdot) = \lambda\, p_\theta^{\mathrm{AR}}(\cdot) + (1-\lambda)\, p_\theta^{\mathrm{diff}}(\cdot),$$

where $\lambda \in [0, 1]$ controls the interpolation.