

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

Yonggan Fu¹, Lexington Whalen^{1,2†}, Abhinav Garg¹, Chengyue Wu^{1,3†},
Maksim Khadkevich¹, Nicolai Oswald¹, Enze Xie¹, Daniel Egert¹,
Sharath Turuvekere Sreenivas¹, Shizhe Diao¹, Chenhan Yu¹, Ye Yu¹,
Weijia Chen¹, Sajad Norouzi¹, Jingyu Liu^{1,4†}, Shiyi Lan¹, Ligeng Zhu¹,
Jin Wang^{1,3†}, Jindong Jiang¹, Morteza Mardani¹, Mehran Maghoumi¹,
Song Han^{1,5}, Ante Jukić¹, Nima Tajbakhsh¹, Jan Kautz¹, Pavlo Molchanov¹

¹NVIDIA ²Georgia Tech ³HKU ⁴University of Chicago ⁵MIT

Abstract

Diffusion language models (LMs) enable parallel decoding, but still lag behind autoregressive (AR) LMs in accuracy, and behind multi-token prediction (MTP) methods in efficiency. This raises three questions: should diffusion LMs compete with AR LMs or be harmonized with them; can diffusion provide a stronger acceleration mechanism than MTP; and does diffusion decoding have sufficient long-term potential to justify further exploration? We study these questions through Nemotron-Labs-Diffusion, a tri-mode LM that unifies three decoding modes—AR, diffusion, and self-speculation (diffusion drafts, AR verifies)—within a single model to sustain high throughput across all concurrency levels. Our study shows that (1) AR and diffusion objectives are mutually beneficial rather than competing, (2) diffusion-based self-speculation achieves higher acceptance lengths and better system vs. per-user throughput trade-offs than MTP baselines, and (3) diffusion decoding shows strong potential for parallel decoding under an optimal sampler in a speed-of-light analysis. Based on these insights, we develop the Nemotron-Labs-Diffusion family at 3B, 8B, and 14B scales. Our models outperform state-of-the-art open-source AR and diffusion LMs in both accuracy and speed, e.g., achieving $6\times$ tokens per forward over Qwen3-8B with comparable benchmark accuracy, translating to $4\times$ throughput on SPEED-Bench measured with SGLang on an NVIDIA GB200 GPU.

1 Introduction

Diffusion language models (LMs) (Nie et al., 2025; Ye et al., 2025), which decode multiple tokens per forward pass, have emerged as a promising alternative to autoregressive (AR) LMs. Despite this promise, diffusion LMs still lag behind AR LMs in accuracy and learning efficiency (Nie et al., 2024),

because diffusion training treats token permutations uniformly (Xue et al., 2025) without leveraging left-to-right language priors. Moreover, the practical advantages of diffusion LMs over multi-token prediction (MTP) methods remain unclear.

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 broader 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. Specifically, we introduce Nemotron-Labs-Diffusion, a tri-mode LM that jointly optimizes AR and diffusion losses under a unified training framework. Our training scheme uses a global loss-averaging strategy that treats all valid tokens across batches equally and adopts a two-stage training procedure: first strengthening AR capabilities with left-to-right linguistic priors, and then performing joint AR-diffusion training to unlock parallel decoding abilities. The resulting model supports tri-mode decoding, as shown in Fig. 1 (a): (1) AR decoding, (2) diffusion-based decoding, and (3) self-speculation, where diffusion drafts candidate tokens and AR verifies them.

We leverage this training scheme to develop the Nemotron-Labs-Diffusion model family, including base, instruct, and vision-language variants at 3B, 8B, and 14B scales, to answer the three questions.

For *Q1*, we show that AR and diffusion LMs can be harmonized rather than treated as competing alternatives. AR provides strong left-to-right linguistic priors, while diffusion enables parallel

[†]Work done during an internship at NVIDIA.

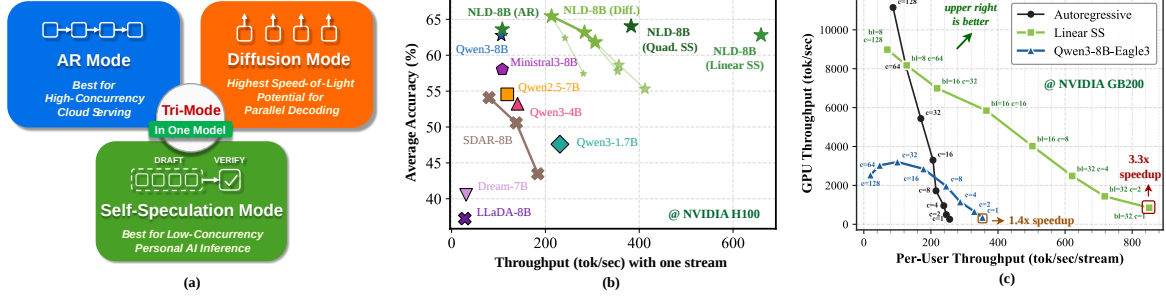


Figure 1: (a) An illustration of three modes in one model. (b) Accuracy–throughput trade-off measured on general benchmarks at batch size 1 on an NVIDIA H100 using PyTorch. NLD denotes our model and Linear/Quad SS denotes linear/quadratic self-speculation. ★ indicates the diffusion mode with different 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 (Abramovich et al., 2026) at different concurrency c on an NVIDIA GB200 GPU using SGLang.

generation without sacrificing accuracy. The joint objective naturally enables self-speculation, which is particularly efficient in low-concurrency settings, while AR remains well suited for high-concurrency scenarios, as shown in Fig. 1 (b) and (c).

For $Q2$, our results show that self-speculation provides a stronger alternative to MTP methods. Prior MTP methods such as Eagle3 (Li et al., 2025b) rely on auxiliary prediction heads to propose future tokens, whereas Nemotron-Labs-Diffusion uses diffusion itself to generate high-quality multi-token drafts. As shown in Fig. 1 (c), this leads to higher acceptance lengths and better system vs. per-user throughput trade-offs, making tri-mode LMs a more effective solution.

For $Q3$, our speed-of-light (SOL) analysis, which estimates the upper bound of diffusion decoding, shows that diffusion decoding has substantial long-term headroom beyond current decoding algorithms. Specifically, the diffusion mode under an optimal sampler can predict over 76.5% more tokens per forward pass than self-speculation, indicating that current samplers still leave a large fraction of the available parallelism unused.

The resulting Nemotron-Labs-Diffusion family outperforms state-of-the-art (SOTA) open-source AR and diffusion LMs in both accuracy and inference speed across a wide range of benchmarks. For example, Nemotron-Labs-Diffusion-8B achieves $6\times$ more tokens per forward than Qwen3-8B with comparable accuracy on general benchmarks, translating to $4\times$ higher throughput on SPEED-Bench (Abramovich et al., 2026) measured with SGLang on an NVIDIA GB200 GPU.

2 Tri-Mode LM Training

To answer $Q1$ in Sec. 1, rather than viewing AR and diffusion language modeling as competing paradigms, our hypothesis is that these two objectives can be complementary and unified within a single model under proper training schemes. The motivation is that AR models inherently learn to plan ahead for future tokens (Samragh et al., 2025), and diffusion training can further unlock this capability; conversely, preserving AR objectives can improve diffusion modeling by injecting strong left-to-right linguistic priors.

2.1 Joint AR and Diffusion Training

We train Nemotron-Labs-Diffusion by jointly optimizing an AR next-token objective and a block-wise diffusion denoising objective.

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

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

Diffusion objective. We adopt the block diffusion formulation (Arriola et al., 2025), 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., $\tilde{x}_t^b \sim q(\cdot | x^b)$, while the prefix $x^{<b}$ remains clean:

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

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:

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

where 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. We set $\alpha = 0.3$ for all experiments.

Two-stage training. We adopt a two-stage training strategy that first trains with the AR objective and then switches to the joint objective, rather than optimizing over all permutations from scratch. 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.

Global loss averaging. Since the diffusion objective involves randomly sampling masked tokens, different training samples may contribute differently 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 (Shao et al., 2024) vs. DAPO (Yu et al., 2025a)) can affect training stability. We consider two loss averaging strategies. Let a batch contain N sequences. For sequence n , let $\mathcal{M}_n$ denote the set of tokens contributing to the diffusion loss, with $M_n = |\mathcal{M}_n|$, and let $\ell_{n,i}$ denote the token-level diffusion loss for $i \in \mathcal{M}_n$. One choice is to first average token losses within each sequence and then average them over sequences:

$$\mathcal{L}_{\text{seq-avg}} = \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{M_n} \sum_{i \in \mathcal{M}_n} \ell_{n,i} \right). \quad (4)$$

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

$$\mathcal{L}_{\text{global}} = \sum_{n=1}^N \sum_{i \in \mathcal{M}_n} \ell_{n,i} / \sum_{n=1}^N M_n. \quad (5)$$

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

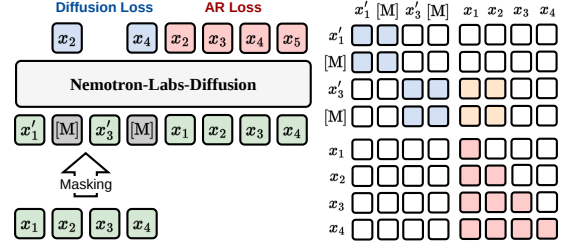


Figure 2: Visualizing the attention pattern of Nemotron-Labs-Diffusion, where ■ denotes attention among noisy tokens, ■ denotes attention from noisy tokens to clean tokens, and ■ denotes attention within the clean context.

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 (Arriola et al., 2025), 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. 2. The *Noisy*→*Noisy* and *Noisy*→*Clean* parts follow the standard block diffusion design (Arriola et al., 2025): 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 (Arriola et al., 2025; Fu et al., 2025; Wu et al., 2025a) 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 (Gat et al., 2025; Samragh et al., 2025). 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 aligns with the pioneering work (Gat et al., 2025), which also achieves joint diffusion and AR training. The key differences in our work lie in

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	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

Table 2: Impact of α during 25B 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

(1) the tri-mode inference strategy, particularly self-speculative decoding, (2) the post-training enhancements, such as samplers and LoRA-enhanced drafters introduced in Sec. 3, and (3) the systematic studies conducted to answer critical questions about the true promise of diffusion LMs.

2.3 Ablation Study on Training Techniques

To save pretraining time, we perform the ablation study starting from the official Ministral3-8B (Liu et al., 2026) base model, which has relatively low benchmark performance and thus mimics a half-trained model without overfitting to benchmarks. We ablate the contribution of each training technique by progressively adding them during continuous pretraining on 25B tokens. Detailed training and evaluation settings are in Appendix A and F.3. All models are evaluated in the diffusion mode.

Observations. As shown in Tab. 1, we observe that (1) *global loss averaging* improves the average accuracy by 2.83%, confirming that treating all tokens equally across the batch reduces gradient variance, as analyzed in Sec. 2.1; (2) *two-stage training* yields a substantial 5.74% gain, implying that stronger AR initialization can enable better future planning and ease AR-to-diffusion conversion; (3) *the addition of AR loss* contributes the largest single improvement of 7.48%, echoing our analysis in Sec. 2. Cumulatively, the full pipeline improves the baseline accuracy by 16.05%, with the AR loss and two-stage training contributing the most.

2.4 Mutual Impact of AR / Diffusion Losses

We further examine whether the AR and diffusion objectives compete for model capacity or reinforce

each other by studying their mutual impact.

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 $\alpha=0.3$. 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.

We also visualize the training loss curves in Fig. 3. We observe that the aforementioned setting $\alpha=0.3$, 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.

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 ($\alpha=0.3$) 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 Appendix A and Sec. F.1.

As shown in Tab. 3, we observe that: 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 (DeepSeek-AI, 2024).

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 at negligible cost to AR-mode accuracy, and the two objectives share a common sweet spot.*

Connecting to the rest of the paper. We deliver the Nemotron-Labs-Diffusion model family across 3B/8B/14B scales, including base, instruct, and

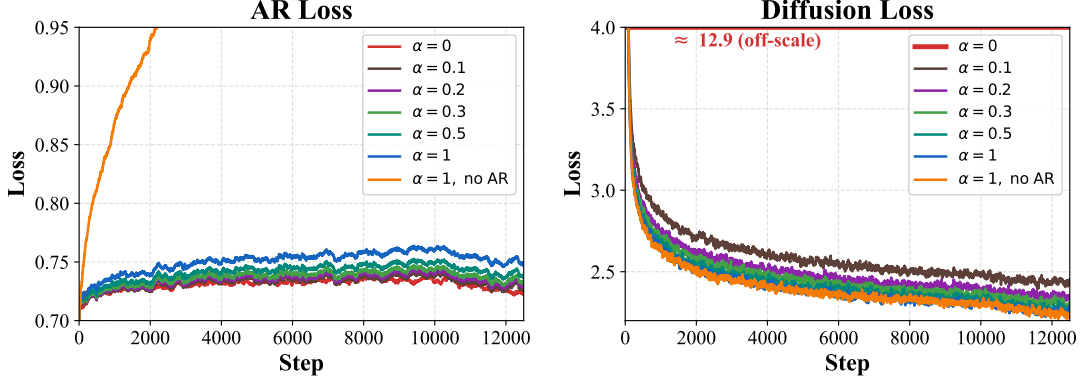


Figure 3: Visualizing the evolution of the AR and diffusion losses across training steps under different diffusion loss coefficients α . The AR loss coefficient is set to 1 by default across all settings, except in the ‘no AR’ setting, where the AR loss is removed and only the diffusion loss is used.

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

<i>Base Model</i>											
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
+ dLM loss	62.80	57.93	65.80	82.54	87.79	66.84	75.99	76.59	79.82	70.32	72.64

<i>Instruct Model</i>											
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
+ dLM loss	44.44	68.65	80.49	85.19	88.00	94.01	33.33	33.33	79.85	28.85	63.61

vision-language variants, with this joint training scheme (hyperparameters detailed in Appendix A). The resulting models are then used to study the tri-mode inference in Sec. 3 and the SOL analysis in Sec. 4. The benchmark performance against SOTA AR and diffusion LMs is reported in Sec. 5.

3 Tri-Mode LM Inference

The joint AR and diffusion training enables decoding in three modes: AR, diffusion, and self-speculation decoding, as visualized in Fig. 4.

3.1 Mode 1: AR Decoding

Tri-mode LMs preserve AR decoding, which samples $x_i \sim p_\theta(\cdot \mid x_{<i})$ at step i with causal attention.

3.2 Mode 2: Block-wise Diffusion Denoising

Confidence-based sampling. Following (Wu et al., 2025a; Fu et al., 2025), the diffusion decoding 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 (Wu et al., 2025b). 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 can optionally 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 when decoding only the highest-confidence token at each step. 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 and feature engineering are detailed in Appendix B.

3.3 Mode 3: Self-Speculation Decoding

Linear self-speculation. This mode uses diffusion to draft and AR to verify.

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

Verification with AR. We then run a second forward pass over the draft tokens $[\hat{x}_{n+1}, \dots, \hat{x}_{n+k}]$ with causal attention, again reusing the prefix KV cache. The AR logits at each position yield next-token predictions $\{x_{n+j}^{\text{AR}}\}_{j=1}^k$. We accept the longest prefix of draft tokens that passes the veri-

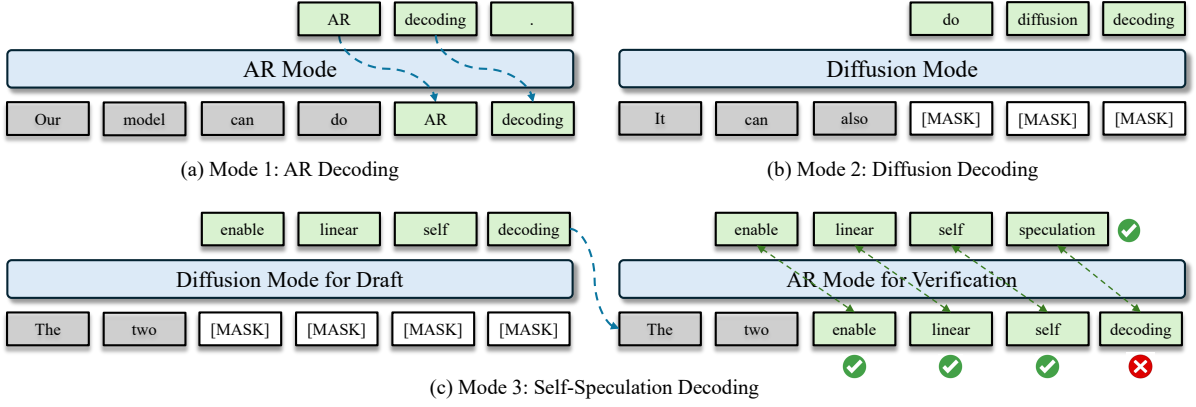


Figure 4: Visualizing tri-mode inference: (a) left-to-right AR decoding, (b) parallel diffusion decoding, and (c) linear self-speculation decoding (quadratic self-speculation is visualized in Appendix D).

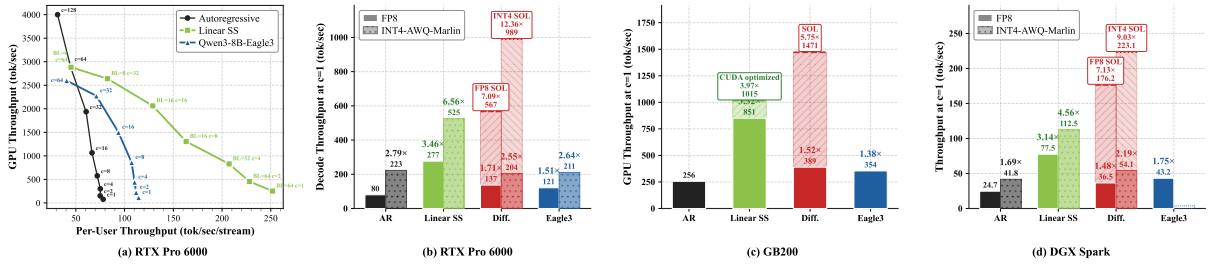


Figure 5: (a) System vs. per-user throughput trade-off on an NVIDIA RTX Pro 6000 GPU; (b)/(c)/(d): The throughput under a concurrency of 1 on NVIDIA RTX Pro 6000, GB200, and DGX Spark, respectively.

fication criterion (e.g., $x_{n+j}^{\text{AR}} = \hat{x}_{n+j}$) and commit the accepted tokens to the verified prefix.

Enhance linear self-speculation w/ LoRA. We further enhance linear self-speculation by tuning a LoRA adapter (Hu et al., 2022) 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, $\alpha=512$, $\sim 36\text{M}$ trainable parameters/ $\sim 0.4\%$ of the backbone), leaving the AR pathway unchanged. The training loss is detailed in Appendix C.

Quadratic self-speculation. A variant of linear self-speculation is quadratic self-speculation, which leverages quadratic decoding (Samragh et al., 2025) with single-forward drafting and verification, following (Liu et al., 2025a). This decoding scheme prepares for the worst case by predicting the next block at all possible acceptance positions, with details provided in Appendix D.

3.4 Benchmark Self-speculation vs. Eagle3

Limitations of Eagle3. Eagle3 (Li et al., 2025b) has become the default choice for efficient LLM deployment at low concurrency. It trains auxiliary draft heads to predict future tokens and uses the original AR model for verification. The two main

bottlenecks are that (1) the draft model has limited capacity and becomes less reliable beyond a short horizon, and (2) the tiny draft model, which is called recursively, can lead to low GPU utilization. In contrast, Nemotron-Labs-Diffusion provides unique advantages through significantly longer acceptance lengths and token-parallel drafting with 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. 5. Evaluations are conducted on SPEED-Bench (Abramovich et al., 2026) across four categories: math, coding, reasoning, and multilingual. We perform a grid search over hyperparameters for Eagle3, while for Nemotron-Labs-Diffusion we vary only the *block length*.

Observations. As shown in Fig. 5, we observe that: (1) Linear self-speculation consistently improves user throughput over the AR mode across all three GPUs, achieving up to a $3.3\times$ speedup over AR on GB200 ($3.97\times$ with an optimized kernel), as shown in Fig. 5 (c). The gains are consistent across platforms and quantization schemes, demonstrating its effectiveness in low-concurrency scenarios.

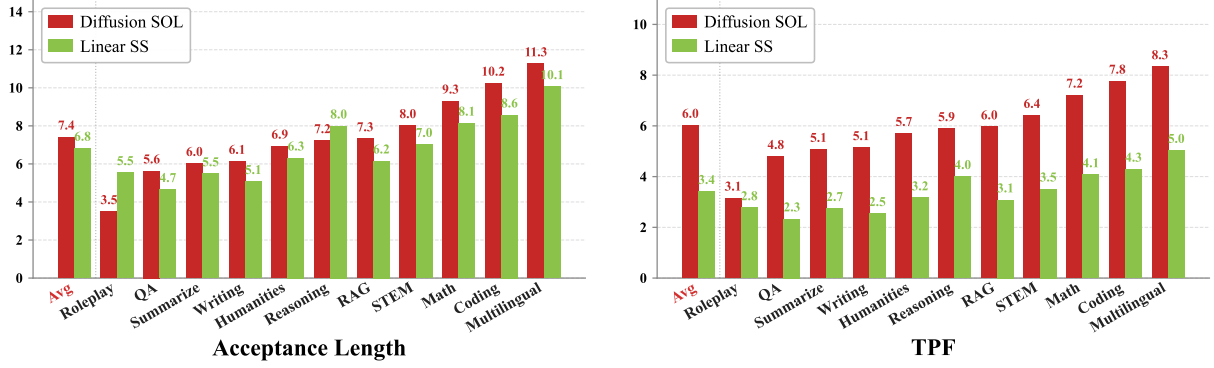


Figure 6: Visualizing the acceptance length and TPF across different SPEED-Bench categories.

Table 4: SPEED-Bench SOL acceptance length.

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 length	7.60	5.68	4.17	2.89

(2) Compared with Eagle3, linear self-speculation delivers a $2.4 \times / 2.3 \times / 1.8 \times$ 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. 5 (a) and Fig. 1 (c). A per-category breakdown in Appendix G shows that our method achieves a 6.82 average acceptance length versus 2.75 for Eagle3. (3) The SOL ceiling (introduced in Sec. 4) reveals substantial remaining headroom. For example, the projected SOL throughput reaches $12.36 \times$ over AR under INT4 quantization on RTX Pro 6000, roughly $2 \times$ higher than linear self-speculation.

4 Speed-of-Light Analysis of Diffusion

We conduct a speed-of-light (SOL) analysis to quantify the maximum acceptance length / token-per-forward achievable by the diffusion mode.

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 and $[M]$ the mask token. Starting from an all-mask input $\mathbf{x}^{(0)} = [M]^B$, we repeatedly select

the masked position with the highest peak probability, commit its argmax token, and re-evaluate f_θ on the partially unmasked sequence until all B positions are filled. This *serial denoising* procedure requires exactly B forward passes and produces a target sequence $\mathbf{t} = (t_1, \dots, t_B)$ representing the model’s converged output without parallel commits. The SOL acceptance ratio is then defined as the average TPF required to reproduce $\mathbf{t}$ from the same all-mask input under an optimal parallel decoding scheme. We use a recursive algorithm to identify the optimal SOL path, detailed in Appendix E.

4.2 SOL Evaluation on SPEED-Bench

We evaluate SOL on 713 SPEED-Bench (Abramovich et al., 2026) samples spanning 11 categories on the diffusion mode of Nemotron-Labs-Diffusion-8B-Instruct.

Observations on diffusion SOL. From Tab. 4, we observe that (1) The diffusion mode exhibits substantial intrinsic parallelism: the SOL acceptance length reaches $7.60 \times$ on average and exceeds $10 \times$ on multilingual and coding content at $B = 32$, and grows nearly linearly with B . (2) Confidence-based sampling, by contrast, achieves only $\sim 2.6 \times$ TPF at comparable accuracy in the diffusion-mode results of Sec. 5, leaving a notable gap to the SOL ceiling. This indicates that confidence-based sampling is far from optimal and that there is substantial headroom for better samplers.

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. 6. Two distinct metrics matter here: The *acceptance length* counts how many tokens are committed per acceptance step. 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

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
<i>Scientific QA & Instruction Following</i>											
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
<i>Coding</i>											
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
<i>Math</i>											
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
<i>Average over All Tasks</i>											
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

top of one forward per acceptance step, so its real TPF is close to its acceptance length; linear self-speculation, however, uses two forwards per cycle (one diffusion draft, one AR verification), so its real TPF is the acceptance length divided by two.

From Fig. 6, we observe that: (1) For the acceptance length, linear self-speculation approaches SOL, achieving $6.82\times$ vs. $7.60\times$ overall, 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: $6.02\times$ for SOL vs. $3.41\times$ 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.

Implications. The SOL analysis highlights two key takeaways: (1) Diffusion 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, which is a promising next step for the diffusion LM field. (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 drafter and the AR verifier are well aligned. Diffusion-based samplers that can safely commit tokens at non-prefix positions are desired.

5 Final Model Evaluation and Analysis

We present the results of our Nemotron-Labs-Diffusion-8B instruct model in Tab. 5 and more detailed **training settings, benchmarks across 3B/14B scales, VLM results, and ablation studies on samplers and LoRA** are in Appendix F.

From 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. (2) The diffusion mode decodes $2.57\times$ 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) Linear self-speculation maintains comparable accuracy while further boosting TPF to $5.99\times$. (4) Quadratic self-speculation can achieve the highest TPF of $6.38\times$. However, due to the use of FlexAttention with less optimized kernels for the dedicated attention mask (Liu et al., 2025a), the real-device efficiency of quadratic self-speculation falls behind the linear one, as shown in Fig. 1 (b). As such, we use the linear variant by

default in our earlier comparisons with MTP/SOL.

6 Related Work

Diffusion LMs, including both continuous (Li et al., 2022; Gong et al., 2022; Han et al., 2022) and discrete variants (Austin et al., 2021; He et al., 2022; Sahoo et al., 2024; Lou et al., 2023; Ou et al., 2024), have been proposed for parallel non-AR generation. Among them, masked dLMs (He et al., 2022; Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025; Ye et al., 2025) have been successfully scaled to large models (Nie et al., 2025; Ye et al., 2025). Follow-up works further explore alternative dLM paradigms (Sahoo et al., 2025b,a; Xue et al., 2025), scaling laws (Bie et al., 2025, 2026; Google DeepMind, 2025), domain-specific applications (Khanna et al., 2025; Song et al., 2025; Gong et al., 2025b; Xie et al., 2025), and reinforcement learning (Zhao et al., 2025; Zhu et al., 2025), and multimodal extensions (You et al., 2025; Yang et al., 2025b).

Despite their acceleration potential (Nie et al., 2025; Ye et al., 2025), dLMs are still limited by bidirectional attention and iterative denoising. Prior works address these issues through dedicated KV caching strategies (Liu et al., 2025b; Ma et al., 2025; Wu et al., 2025b), confidence-based and adaptive decoding (Wu et al., 2025b; Israel et al., 2025; Wei et al., 2025), and AR-initialized or block-wise diffusion training (Gong et al., 2025a; Ye et al., 2025; Arriola et al., 2025). Follow-up studies extend block-wise dLMs by converting pretrained AR models or dLMs into more efficient diffusion decoders (Wu et al., 2025a; Fu et al., 2025; Cheng et al., 2025; Wang et al., 2025; Qian et al., 2026).

7 Conclusion, Insights, and 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. Our analysis reveals the following insights:

1. **Tri-mode generation arises naturally from joint AR-diffusion training.** By enabling both AR and diffusion decoding 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. Adding the AR loss induces linguistic priors, and the diffusion loss preserves/improves AR accuracy through better 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 lengths and better efficiency.
4. **Linear self-speculation is currently the most efficient mode.** Linear self-speculation achieves the best efficiency, while quadratic self-speculation, with higher TPF per step, holds promise with improved infrastructure support.
5. **Diffusion decoding has substantial headroom.** Our SOL analysis shows the potential to correctly predict 76.5% more tokens per forward than the current best strategy, indicating a more promising upper bound for parallel decoding.

Looking forward, these insights shed light on several promising future directions:

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. 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 model training, thereby improving the acceptance length.
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 any-order verification to validate and accept multiple non-contiguous drafted tokens and thus further improve the effective acceptance length.

Limitations

We acknowledge two limitations of this study:

Sampler design for the diffusion mode is still in its early stages. Our trained sampler is an initial step toward narrowing the gap to the SOL ceiling in Sec. 4, but it does not fully close it. Designing effective samplers, which encourage conditionally independent parallel sampling, remains an open problem for fully unlocking the diffusion mode.

Insights are validated only at dense-LM scales and under standard pretraining and SFT pipelines. Our conclusions are drawn from dense models up to 14B trained with continuous pretraining and SFT. Whether the AR-diffusion synergy and tri-mode benefits transfer to larger-scale MoE models and agentic training pipelines requires further exploration.

References

- Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024a. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.
- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024b. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.
- Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, and 1 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.
- Talor Abramovich, Maor Ashkenazi, Benjamin Chislett, Tiyyasa Mitra, Bitu Darvish Rouhani, Ran Zilberstein, Yonatan Geifman, and 1 others. 2026. Speed-bench: A unified and diverse benchmark for speculative decoding. *arXiv preprint arXiv:2604.09557*.
- Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.
- Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, and 1 others. 2025. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*.
- Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, and 1 others. 2026. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*.
- Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, and 1 others. 2025. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*.
- Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, and 1 others. 2026. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*.
- Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and 1 others. 2025. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*.
- DeepSeek-AI. 2024. [Deepseek-v3 technical report](#). Preprint, arXiv:2412.19437.
- 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, and 13 others. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. *arXiv preprint arXiv:2407.11691*.
- Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, and 1 others. 2025. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*.
- Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. 2025. Set block decoding is a language model inference accelerator. *arXiv preprint arXiv:2509.04185*.
- Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.
- 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.

- 2025a. [Scaling diffusion language models via adaptation from autoregressive models](#). In *The Thirteenth International Conference on Learning Representations*.
- Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.
- Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatuo Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025b. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.
- Google DeepMind. 2025. [Gemini diffusion](#). Model page: state-of-the-art, experimental text diffusion model.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*.
- Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.
- Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2025. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*.
- Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*, pages 235–251.
- Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, and 1 others. 2025. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. 2025a. Lavida: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*.
- Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025b. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.
- Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.
- Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. 2025a. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025b. d1lm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2023. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*.
- Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. 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.

- Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.
- Dhruv Nathawani, Shuoyang Ding, Vitaly Lavrukhin, Igor Gitman, Somshubra Majumdar, Evelina Bakhurina, Boris Ginsburg, and Jane Polak Scowcroft. 2025. [Nemotron-Post-Training-Dataset-v2](#).
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.
- NVIDIA Corporation. 2024. Nemo-skills: A toolkit for improving skills of large language models. <https://github.com/NVIDIA-NeMo/Skills>. GitHub repository.
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*.
- Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. 2026. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*.
- Qwen-Team. 2024. [Qwen2.5: A party of foundation models](#).
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. 2025a. The diffusion duality. *arXiv preprint arXiv:2506.10892*.
- Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. 2025b. Esoteric language models. *arXiv preprint arXiv:2506.01928*.
- Aleksei Samarin and 1 others. 2026. Lk losses: Direct acceptance rate optimization for speculative decoding. *arXiv preprint arXiv:2602.23881*.
- Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. 2025. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.
- Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, and 1 others. 2025. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*.
- Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. 2025. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*.
- Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. 2025. Accelerating diffusion large language models with slowfast: The three golden principles. *arXiv preprint arXiv:2506.10848*.
- Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, Leandro von Werra, Aritra Roy Gosthipaty, and Andrés Marafioti. 2025. [Finevision: Open data is all you need](#). *Preprint*, arXiv:2510.17269.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2025a. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025b. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.
- xAI. 2024. [Realworldqa](#).
- Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and 1 others. 2025. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*.
- Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. 2025. Any-order gpt as masked diffusion model: Decoupling formulation and architecture. *arXiv preprint arXiv:2506.19935*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

- Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. 2025b. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.
- Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. 2025. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Lingjun Liu, and 1 others. 2025a. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025b. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruofei 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, and 3 others. 2024. 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.
- Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. dl: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.
- Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

A Model Family Training Pipeline

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

A.1 Base Models

To speed up pretraining, we start from the pre-trained Ministral3 base models (Liu et al., 2026) and apply the two-stage training strategy introduced in Sec. 2.1. Specifically, we adopt the pre-training dataset in (Basant et al., 2025) 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 Warmup-Stable-Decay (WSD) schedule (Hu et al., 2024) 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.

A.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 (Hu et al., 2024) with the AdamW optimizer and a weight decay of 0.1. We train the model on 45B tokens from the SFT dataset of (Chandiramani et al., 2026), with a global batch size of 256 and a sequence length of 16k. Following (Nie et al., 2025), 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.

A.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 pre-trained 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-

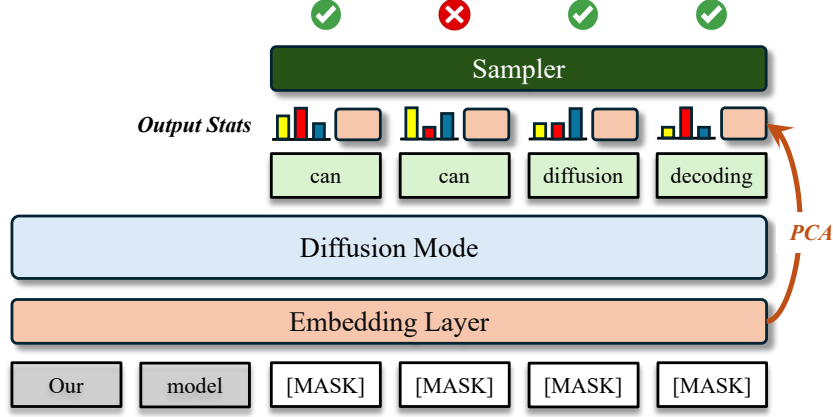


Figure 7: An illustration of the sampler design on top of the diffusion mode.

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 (Minstral3-8B-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 introduced; 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 (Wiedmann et al., 2025).

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:

$$\left[\tilde{x}_t^{(\text{text}, L_{\text{text}})} \mid x^{(L)} \right], \quad L_{\text{text}} = L - N_{\text{vis}}, \quad (6)$$

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 .

A.4 License

The starting-point Minstral3 base models (Liu et al., 2026) are released under the *Apache-2.0* license; the pretraining data (Basant et al., 2025) is released under the *NVIDIA Data Agreement for Model Training*; the post-training data (Chandiramani et al., 2026) is released under the *CC-BY-4.0* license; and the VLM data (Wiedmann et al., 2025) is distributed under heterogeneous licenses (including CC-BY, Apache-2.0, and research-only licenses). As a result, our released VLMs are non-commercial.

B Sampler Architecture and Training

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. 7, the sampler operates on top of the frozen backbone and adds negligible parameter overhead ($\sim 0.06\%$, with 4.8M parameters 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 $\sim 20\text{M}$ denoising trajectories from Nemotron-Labs-Diffusion-8B on (Nathawani et al., 2025) (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 $\mathbf{1}[\text{current top-1} = \text{final ID}]$, 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.

C Linear Self-Speculation with LoRA

This appendix details the training loss used to fine-tune the LoRA-augmented diffusion drafter introduced in Sec. 3.3, which combines an LK-hybrid distribution-matching term with a token-level cross-entropy term, both restricted to the active “accepted + 1” positions illustrated in Fig. 8.

Drafter-verifier setup. For each position $j \in \{1, \dots, k\}$ in the draft block, the LoRA-augmented drafter produces logits $z_j^d \in \mathbb{R}^{|V|}$ over the vocabulary V , while the frozen AR verifier produces target logits z_j^t on the same context. We define the temperature-scaled distributions

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

with $\tau=3.0$. 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. 8, 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 $\mathcal{A} = \{1, \dots, j^*\}$ when there is a rejection in the block, and $\mathcal{A} = \{1, \dots, k\}$ 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 (Samarin et al., 2026) 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 $\text{KL}(\tilde{p}_j \parallel \tilde{q}_j) = \infty$ catastrophe of full-vocabulary KL. The per-position hybrid loss is

$$\begin{aligned} \mathcal{L}_j^{\text{LK}} = & \lambda_j \cdot \text{KL}(\tilde{p}_j \parallel \tilde{q}_j) \\ & + (1 - \lambda_j) \cdot \frac{1}{2} \sum_{v \in \mathcal{U}_j} |\tilde{p}_j(v) - \tilde{q}_j(v)|, \end{aligned} \quad (7)$$

where the right-hand total-variation (TV) term equals $1 - \alpha_j$, with $\alpha_j = \sum_{v \in \mathcal{U}_j} \min(\tilde{p}_j(v), \tilde{q}_j(v))$ the standard speculative-decoding acceptance probability (Leviathan et al., 2023)—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 $\eta=0.5$ 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 ($\alpha_j \rightarrow 1$ and $\lambda_j \rightarrow e^{-\eta} \approx 0.6$, directly minimizing the acceptance length 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:

$$\ell_j = \begin{cases} -\log q_j^{(\mathcal{U}_j)}(y_j) & \text{if } y_j \in \mathcal{U}_j, \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

where $q_j^{(\mathcal{U}_j)}$ is the drafter softmax restricted to $\mathcal{U}_j$ with $\tau=1.0$. The cross-entropy provides a

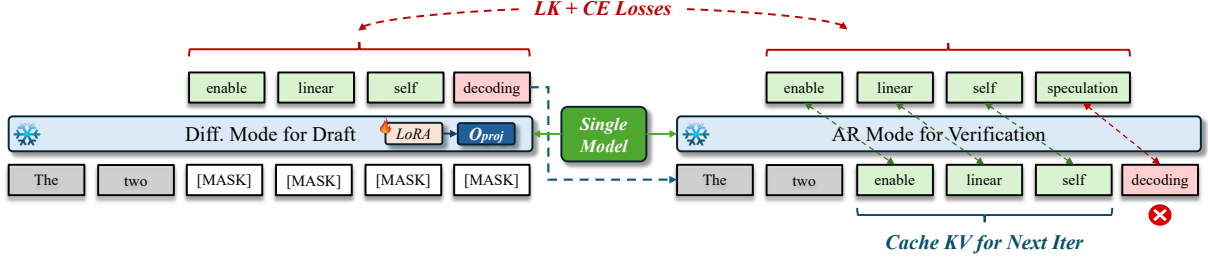


Figure 8: An illustration of LoRA training on the diffusion drafter in linear self-speculation.

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. 9), 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:

$$\mathcal{L}_{\text{LK}} = \frac{1}{|\mathcal{A}|} \sum_{j \in \mathcal{A}} \mathcal{L}_j^{\text{LK}}, \quad \mathcal{L}_{\text{CE}} = \frac{\sum_{j \in \mathcal{A}} \ell_j}{\sum_{j \in \mathcal{A}} \mathbb{1}\{y_j \in \mathcal{U}_j\}}. \quad (9)$$

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

$$\mathcal{L} = \lambda_{\text{KL}} \cdot \mathcal{L}_{\text{LK}} + \lambda_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}, \quad \lambda_{\text{KL}} = \lambda_{\text{CE}} = 1. \quad (10)$$

At training time, 90% of the inner-batch slots draw their drafter tokens by sampling from $\text{softmax}(z_j^d / T_{\text{draft}})$ with $T_{\text{draft}}=1.0$; 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.

D Quadratic Self-Speculation

This section introduces the design details of quadratic self-speculation. Let $[x_1, \dots, x_n]$ 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 $\{x_{n+j}^t\}_{j=2}^{k+1}$, and interleave k fresh mask tokens after each speculative token, yielding the quadratic input:

$$X_m^{t+1} = [x_1, \dots, x_n, x_{n+1}] + [x_{n+2}^t, m_1, \dots, m_k] + \dots + [x_{n+k+1}^t, m_1, \dots, m_k], \quad (11)$$

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

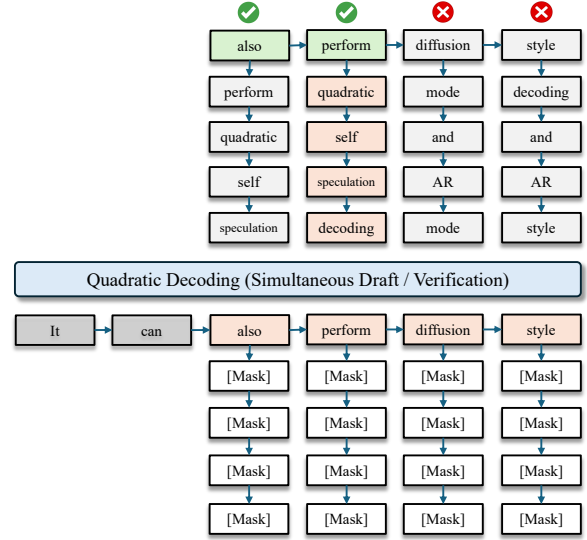


Figure 9: An illustration of quadratic self-speculation with simultaneous drafting and verification. ■ denotes draft tokens that match AR verification, and ■ denotes the block that originates from the last matched token and is to be verified in the next iteration.

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 $\{x_{n+j}^{t+1}\}_{j=2}^{k+1}$. These tokens are used to verify the previous speculative draft through sequential comparison (Samragh et al., 2025): we accept the longest prefix that satisfies a verification criterion (e.g., $x_{n+j}^{t+1} = x_{n+j}^t$, 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 $\{x_{n+j}^{t+1}\}_{j=2}^{k+1}$ 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 (Samragh et al., 2025).

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_{n+j}^t in Eq. 11, we can use the diffusion prediction at the first newly inserted mask position m_1 immediately following it as an alternative verifier for the same token. Concretely, the AR verifier uses the causal logits $p_\theta^{\text{AR}}(\cdot \mid x_{<n+j})$ at position $n+j$, while the diffusion verifier uses the denoising logits $p_\theta^{\text{diff}}(\cdot \mid X_m^{t+1}, t)$ produced for the corresponding m_1 position. Generally, we can form an AR-diffusion ensemble verifier by combining the two distributions:

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

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

E SOL Algorithm

We searched for the SOL path using the following algorithm.

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, $\mathcal{A}^{(k)} = \{j : x_j^{(k)} = [\mathbf{M}] \wedge \hat{t}_j^{(k)} = t_j\}$; 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 with a strategic search for the largest *safe* subset of matching positions, as shown in Fig. 10. At each iteration, we rank the N matched positions by model confidence as $(p_1, \dots, p_N)$ and search for the largest prefix $\{p_1, \dots, p_k\}$ 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.

F More Evaluation and Analysis

F.1 Benchmark Instruct Models

Baselines and benchmarks in Sec. 5. We compare our Nemotron-Labs-Diffusion-8B instruct model against SOTA AR instruct models (Qwen3-8B (Yang et al., 2025a), Qwen2.5-7B (Qwen-Team, 2024), and Ministral3-8B (Liu et al., 2026)) and SOTA diffusion instruct models (LLaDA-8B Instruct (Nie et al., 2025), Dream-7B Instruct (Ye et al., 2025), and SDAR-8B Chat (Cheng et al., 2025)). We evaluate all modes of our model: AR, diffusion, and self-speculation, including both linear and quadratic self-speculation modes. We use LoRA-enhanced linear self-speculation by default. For the diffusion mode evaluation of Nemotron-Labs-Diffusion-8B and SDAR-8B Chat (Cheng et al., 2025), we also report tokens per forward (TPF) by selecting different denoising thresholds (Wu et al., 2025b). 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 (NVIDIA Corporation, 2024) 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 (Nie et al., 2025; Ye et al., 2025; Cheng et al., 2025).

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. 11, the trained sampler shifts the entire Pareto frontier upward, delivering

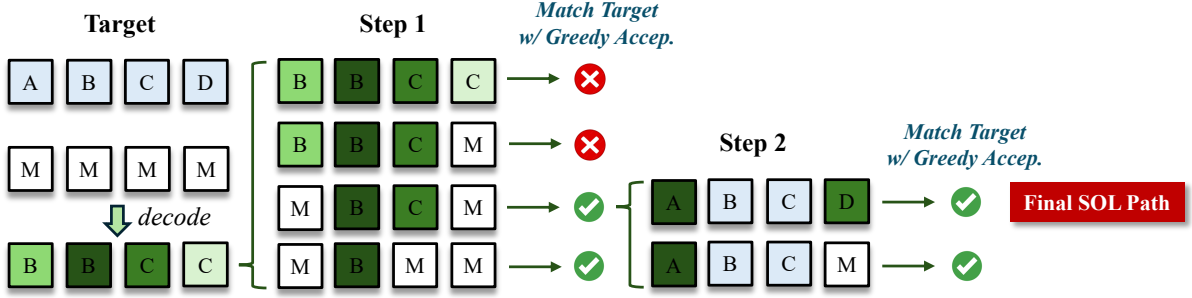


Figure 10: An example of using the recursive dynamic compaction method to identify the SOL path.

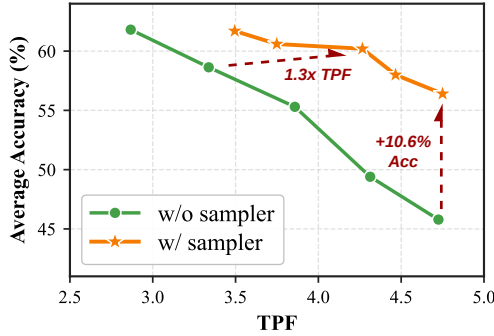


Figure 11: Comparing the accuracy-TPF trade-offs achieved w/ and w/o a sampler.

higher TPF at the same accuracy (e.g., $1.3\times$ 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.

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., $4.67\times$ 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.

F.2 Extend to More Model Scales

Baselines and benchmarks. We extend the evaluation to two additional scales, 3B and 14B, under the same evaluation protocol as Sec. F.1. We benchmark against SOTA AR instruct models at the corresponding scales: Llama-3.2-3B-

Instruct (Grattafiori et al., 2024), Phi-4-mini-Instruct (Abouelenin et al., 2025), and Qwen3-4B (Yang et al., 2025a) at the 3B scale; and Qwen3-14B (Yang et al., 2025a), Ministral3-14B-Instruct-2512 (Liu et al., 2026), Gemma-3-12B-IT (Gemma Team, 2025), Phi-3-Medium-14B (Abdin et al., 2024a), and Phi-4-14B (Abdin et al., 2024b) at the 14B scale.

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 $4.36\times/5.96\times$ TPF, respectively. Since both of our models preserve the AR mode while enabling more efficient generation modes, they are strong drop-in replacement candidates for AR LMs at the 3B/14B scales. (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 $4.36\times$ to $5.96\times$ when scaling from 3B to 14B. We attribute this to the stronger future-prediction abilities of larger models, which yield more reliable draft predictions.

F.3 Benchmark Base Models

Baselines and benchmarks. We compare Nemotron-Labs-Diffusion-8B against SOTA AR base LMs (Llama-3.1-8B (Grattafiori et al., 2024), Ministral3-8B (Liu et al., 2026), and Qwen3-8B (Yang et al., 2025a)) and two representative diffusion LMs (LLaDA-8B (Nie et al., 2025) and Dream-7B (Ye et al., 2025)). We evaluate on coding benchmarks (HumanEval, HumanEval+, MBPP, MBPP+), math reasoning (GSM8K, Minerva Math), knowledge (MMLU), and common-

Table 6: Per-task TPF achieved by linear self-speculation w/ and w/o LoRA tuning across 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

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
<i>3B Scale</i>													
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
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
<i>14B Scale</i>													
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
Minstral3-14B-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

sense 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 Minstral3-8B/Qwen3-8B; (2) the diffusion mode delivers +17.21%/+6.83% higher accuracy than LLaDA-8B/Dream-7B; and (3) self-speculation achieves $4.67\times$ TPF (linear) and $7.04\times$ TPF (quadratic) with over 0.5% higher average accuracy compared to the strongest baseline, Qwen3-8B.

F.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 (Kembhavi et al., 2016), ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), MMMU (Yue et al., 2024), MathVista (Lu et al., 2024), and RealWorldQA (xAI, 2024). Long-answer benchmarks require extended chain-of-thought reasoning: MMMU-Pro-V. We benchmark against existing diffusion VLMs (Yang et al., 2025b; Li et al., 2025a; You et al., 2025; Yu et al., 2025b). All benchmarks are evaluated using VLMEvalKit (Duan et al., 2024) under the same prompts and post-

processing as the AR baseline (Minstral3 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 $2.46\times$ – $3.15\times$ 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 $3.63\times$ – $7.45\times$ TPF, where the higher end is achieved 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.

G Per-Category Acceptance Length

This appendix complements the deployment-efficiency study in Sec. 3.4 with a per-category

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
Minstral3-8B	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

Table 9: Benchmarking discrete diffusion VLMs and Nemotron-Labs-Diffusion-VLM across tasks. The diffusion mode of Nemotron-Labs-Diffusion-VLM uses denoising threshold $\tau=0.9$.

Model	Gen. Mode	AI2D	ChartQA	DocVQA	MMMU	MMMU-Pro-10c	MMMU-Pro-V-CoT	MathVista	RealWorldQA	TPF	Avg
MMaDA	Diffusion	67.4	9.6	9.5	30.2	16.5	8.5	33.4	49.2	1	28.0
LaViDa	Diffusion	70.0	59.0	64.6	43.3	28.7	10.5	44.8	54.5	1	46.9
Dimple	Diffusion	74.4	63.4	37.7	45.2	23.8	12.4	42.3	55.4	1	44.3
LLaDA-V-8B	Diffusion	77.8	78.3	83.9	48.6	35.2	18.6	59.7	63.2	1	58.2
Nemotron-Labs-Diffusion-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

Table 10: Per-category acceptance length on SPEED-Bench for vanilla Linear SS / Linear SS + LoRA of Nemotron-Labs-Diffusion-8B and Qwen3-8B-Eagle3 / Qwen3-9B-MTP, all with draft length 31.

Category	Linear SS	Linear SS + 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 cat. avg	6.75	8.69	2.81	4.73

breakdown of acceptance length on SPEED-Bench (Abramovich et al., 2026). 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 vanilla Linear SS / Linear SS + LoRA 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.

H Potential Risks

Our work develops techniques for efficient language modeling and therefore does not introduce additional risks beyond those already associated with language models. Nevertheless, we advocate for the safe and responsible use of foundation models and encourage adherence to established safety, privacy, and deployment guidelines.