LAD: LoRA-Adapted Diffusion

Ruurd J. A. Kuiper[1], Lars de Groot[2], Bram van Es[2], Maarten van Smeden[1], Ayoub Bagheri[2]

[1]University Medical Center Utrecht. [2]Utrecht University.
r.j.a.kuiper@umcutrecht.nl

Abstract

Autoregressive models dominate text generation but suffer from left-to-right decoding constraints that limit efficiency and bidirectional reasoning. Diffusion-based models offer a flexible alternative but face challenges in adapting to discrete text efficiently. We propose LAD (LoRA-Adapted Diffusion), a framework for non-autoregressive generation that adapts LLaMA models for iterative, bidirectional sequence refinement using LoRA adapters. LAD employs a structural denoising objective combining masking with text perturbations (swaps, duplications and span shifts), enabling full sequence editing during generation. We aim to demonstrate that LAD could be a viable and efficient alternative to training diffusion models from scratch, by providing both validation results as well as two interactive demos directly available online:
https://ruurdkuiper.github.io/tini-lad/
https://huggingface.co/spaces/Ruurd/tini-lad
Inference and training code:
https://github.com/RuurdKuiper/lad-code

1 Introduction

In recent years, the field of natural language generation (NLG) has been transformed by the introduction of large language models (LLMs), predominantly based on transformer models generating text using the autoregressive (AR) paradigm (Vaswani et al. 2017). While these models excel at sequential prediction, their inherent left-to-right generation process presents limitations in efficiency, flexibility, and bidirectional reasoning (Zhu and Zhao 2023; Yi et al. 2024; Zou, Kim, and Kang 2023; Nie et al. 2025). Recently, diffusion models (Sohl-Dickstein et al. 2015), initially achieving state-of-the-art performance in continuous domains like image and audio synthesis, have emerged as a promising alternative to AR models. Diffusion models operate via iterative denoising, progressively refining a noisy initial sample towards a novel sample that mimics the original distribution, offering potential advantages in parallel generation and bidirectional information transfer. Applying diffusion principles to the discrete nature of text, however, requires significant adaptation, which has led to an emerging field of active exploration within the NLP community (Zhu and Zhao 2023; Y. Li et al. 2023).

Research in text diffusion has primarily followed two trajectories: modifying the diffusion process to operate directly on discrete token spaces, often involving masking or absorbing states (He et al. 2022; Nie et al. 2024; Sahoo et al. 2024; Ye et al. 2023; Shi et al. 2024; von Rütte et al. 2025; Austin et al. 2021; Gong et al. 2024; Schiff et al. 2024; Cardei et al. 2025; Lou, Meng, and Ermon 2023; Yuan et al. 2022), or embedding discrete text into continuous latent spaces where standard Gaussian

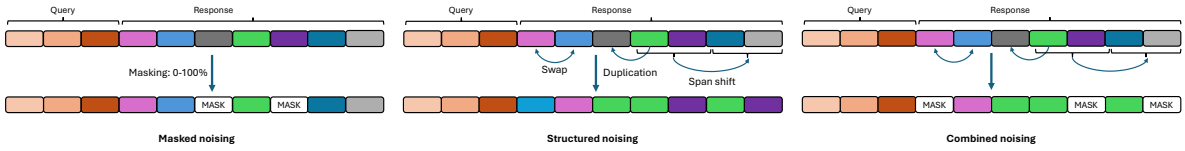

Figure 1: **The noising process applied during training.** The left illustration shows **masked** noising, where noising is applied by masking a random percentage of tokens. The middle shows **structured** noising, where tokens are randomly swapped, duplicated or spans of tokens are shifted. On the right **combined** noising is shown, where both masking and structured noising is applied, which is the noising strategy used to train all models in this study. Noising is applied only on the response tokens.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 97–110

diffusion can be applied (Zhang et al. 2023; Lovelace et al. 2022; X. L. Li et al. 2022).

Recent innovations that have improved diffusion model performance include linguistically informed masking strategies (Chen et al. 2023), simplified continuous-time objectives (Shi et al. 2024) and novel loss functions like score entropy (Lou, Meng, and Ermon 2023). Furthermore, recognizing the computational cost of training large models from scratch, recent studies have often employed existing pretrained AR models, adapting them for diffusion-based generation (Gong et al. 2024; Nie et al. 2024), demonstrating competitive performance (Nie et al. 2025). Post-training techniques like reinforcement learning are also being explored to enhance reasoning in these models (Zhao et al. 2025).

Despite recent progress, several challenges limit the adoption of text diffusion models. Training large models, either from scratch or via full-parameter tuning, is computationally expensive, making efficient adaptation strategies attractive. Inference methods like those in Masked Diffusion Models (MDMs) often fix tokens once unmasked (Sahoo et al. 2024; Schiff et al. 2024; Nie et al. 2025, 2024; Shi et al. 2024), limiting the model's ability to perform holistic refinement or recover from early errors. Efficiently using the knowledge encoded within large pretrained AR models for diffusive generation remains an open goal(Han et al. 2024).

We introduce LAD (LoRA-Adapted Diffusion), a method for non-autoregressive generation that adapts pretrained AR models—Llama 3.2 1B/3B and Llama 3.1 8B (Grattafiori et al. 2024) using bidirectional attention and lightweight Low-Rank Adaptation (Hu et al. 2021), trained using a novel structural noising scheme. LoRA enables parameter-efficient finetuning, preserving pretrained knowledge while avoiding 'catastrophic forgetting' (Luo et al. 2023; Goodfellow et al. 2013).

Additionally, instead of relying solely on mask tokens such as typical MDMs, we also apply a structured corruption process using token swaps, duplications, and span shifts. These perturbations mimic common diffusive generation errors, helping the model learn corrections and enabling updates even when no mask tokens are present. Finally, LAD is trained directly on instruction data, unifying diffusion adaptation and instruction-tuning without requiring a separate pretraining phase.

Our main contributions can thus be summarized as follows:

- A method for adapting AR LLMs to diffusion using LoRA and bidirectional attention.
- A combined structural-masking noising scheme enabling full-sequence editing.
- Unified diffusion adaptation and instruction tuning on instruction data.
- Two real-time demo interfaces exposing key generation parameters like re-noising and max steps.

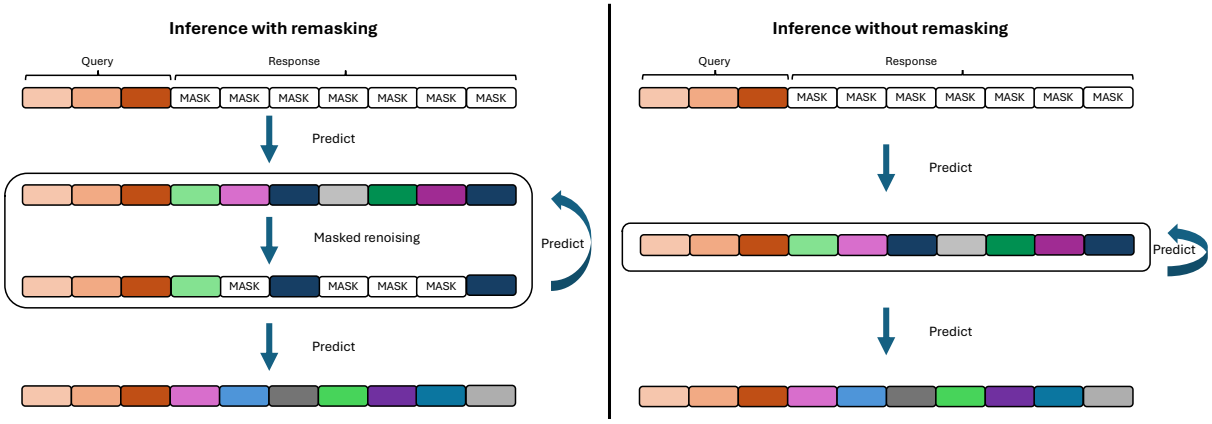

Figure 2: **The inference process.** The left column illustrates inference with intermediate remasking: tokens are predicted for each position in every iteration. After each prediction, part of the generated text is remasked. For inference without remasking, tokens are never remasked. However, the model still refines the output because it has been trained to correct structurally corrupted text as well.

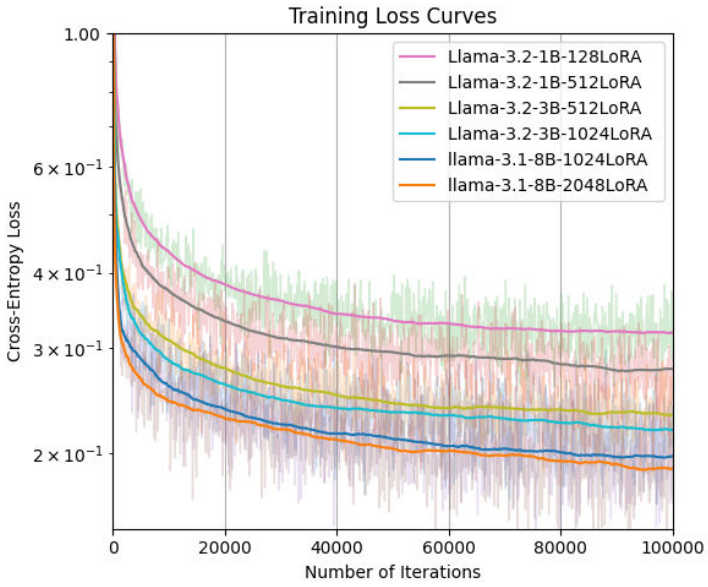

Figure 3: **Training cross-entropy loss.** The lightly colored lines show raw loss values, while the bold lines represent smoothed losses using a progressive moving average.

2 Methods

2.1 Models

All LAD models are based on Llama 3.2 1B/3B-Instruct and Llama 3.1 8B-Instruct (Grattafiori et al. 2024). To enable non-autoregressive generation, we replaced causal attention with bidirectional masks and applied LoRA fine-tuning, freezing all weights except the output layer. Following (Hu et al. 2021) we limited LoRA to the query and value projections, which capture most of its performance benefits, and set $\alpha = r$ across models. We trained six LAD variants: two 1B models (r=128, 512), two 3B models (r=512, 1024), and two 8B models (r=1024, 2048), to study scaling effects of model size and LoRA rank. Model naming and details can be found in Table 1, while detailed hyperparameters are in Appendix A.

2.2 Data

Because we used pre-trained models, our training corpus consisted solely of instruction finetuning datasets. Three general question-answering datasets were used: tatsu-lab/alpaca (Taori et al. 2023) , vicgalle/alpaca-gpt4 (Peng et al. 2023) and a filtered subset of crumb/Clean-Instruct-3M (Crumb 2023). We found that 'catastrophic forgetting' (Goodfellow et al. 2013) would occur when the model was not also trained on more specialized tasks, such as multiple-choice questions, coding, and math. Therefore, task-specific corpora were included. Only the training partition of each of these datasets was used for training. For multiple choice question answering, we used MMLU (Hendrycks et al. 2020), ARC-Easy (Clark et al. 2018) and HellaSwag (Zellers et al. 2019). For math, the GSM8K (Cobbe et al. 2021) and ORCA (Mitra et al. 2024) datasets were used. For coding, the OpenCoder (Huang et al. 2024) dataset was used. All datasets were filtered so the prompt and output together were less than 896 characters. This resulted in a combined dataset of 951k examples, which was subsequently split into training (98%), validation (1%), and test (1%) sets. All strings were tokenized, either truncated or padded with 'end-of-sequence' tokens to a length of 256, and formatted using standard Llama instruction templates. Splits were saved for consistent comparison of loss curves.

Table 1: Comparison of the trained models. The model naming reflects both the size of the base model and LoRA rank, which determines the number of trainable parameters.

Model (LAD-)	Total params	Trainable params	Trainable (%)	LoRA Rank	Training (hours)
1B-r128	1.2B	13.6M	1.1	128	2.5
1B-r512	1.2B	54.4M	4.2	512	2.8
3B-r512	3.4B	146.8M	4.4	512	7.5
3B-r1024	3.5B	293.6M	8.4	1024	8.1
8B-r1024	8.5B	436.2M	5.1	1024	16.0
8B-r2048	8.9B	872.4M	9.8	2048	16.5

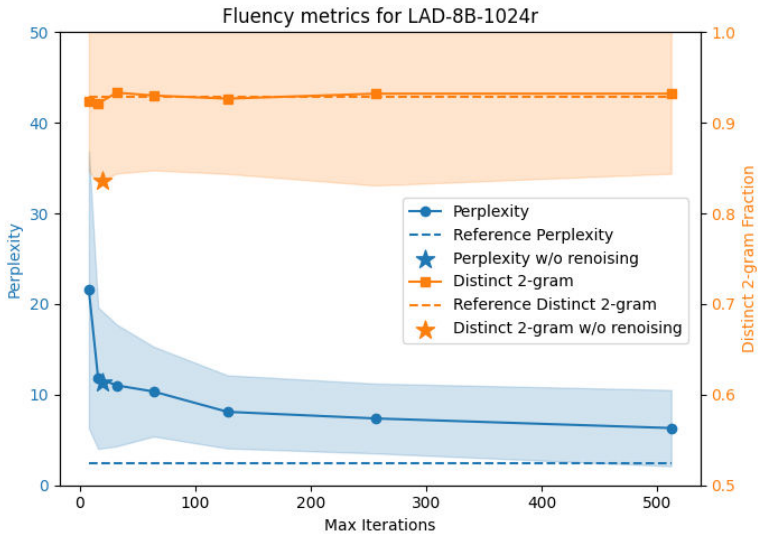

Figure 4**: Median perplexity and distinct 2-gram fraction** of LAD-8B-1024 model outputs across varying max iterations. Shaded areas show interquartile ranges. Dashed lines mark reference answer metrics. Stars show results without renoising.

2.3 Noising

When applying diffusion models to text, we found that intermediate outputs often retain useful lexical content yet suffer from structural issues, such as repeated words, misordered phrases, and fragmented sentences, that require correction. To train a model capable of identifying and correcting such errors, we introduced similar perturbations into the training data through structured noising. However, using only structured noising, there is no well-defined end-point at which a sample would be 'fully noised'. For MDMs, this is commonly represented by a fully masked sequence (Austin et al. 2021; Lou, Meng, and Ermon 2023; Nie et al. 2025; Ou et al. 2024). Therefore, we also incorporated the more regularly employed strategy of incrementally adding 'mask' tokens. The combination enables the model to denoise from an entirely masked input while iteratively refining partially coherent sequences, as illustrated in Figure 1.

Unlike formal MDMs, which use an invertible, probabilistic corruption process and derive a variational bound on the data likelihood, our process, based on local swaps, duplications, and span shifts, is non-invertible. Therefore, we do not model the reverse process in a likelihood-based framework and instead train the model directly using a standard cross-entropy loss.. More details about the training and noising schedule are provided in Appendix B.

2.4 Training

For computational efficiency, all models were trained using float16 mixed precision. The AdamW optimizer (Loshchilov and Hutter 2017) was used with an initial learning rate of 1×10^{-5}, a cosine learning rate schedule with 1000 warmup steps, and a weight decay of 0.01. The batch size was set to 8. Gradient clipping was used with a maximum norm of 0.5. All models were trained for 100k iterations using a batch size of 8 and a context window of 256 tokens to adhere to memory constraints. This means that a total of 205 million tokens were used to train each model. Training was performed on a single NVIDIA A100 GPU (40GB). Full model checkpoints were saved every 10,000 steps to inspect the training process and estimate convergence. All models were trained to minimize the cross-entropy loss between the output logits and the original, uncorrupted token sequence.

2.5 Inference

At inference time, the models can generate text using two modes: a scheduled denoising approach analogous to traditional MDMs, and a self-refining approach that relies on the model's ability to correct sequences without explicit renoising. Both generation strategies are illustrated in Figure 2.

The first method follows a conventional iterative denoising schedule. Given a prompt, the process for generating a response sequence, $\mathbf{x}$, begins with

a fully masked initial state, $\mathbf{x}^{(0)}$, of a predefined length. The model then enters a refinement loop for a maximum of N steps. At each iteration i, the model takes the current sequence $\mathbf{x}^{(i-1)}$ as input and produces a new, complete sequence prediction, $\hat{\mathbf{x}}^{(i)}$.

Following this prediction, a fraction $\eta(i)$ of the tokens in $\hat{\mathbf{x}}^{(i)}$ are randomly re-masked to produce the input for the next iteration, $\mathbf{x}^{(i)}$. This fraction is determined by an exponentially decreasing noise-schedule, $\eta(i)$. The noise schedule is defined as:

$$\eta(i) = \eta_0 \frac{e^{-s\cdot(i/N)} - e^{-s}}{1 - e^{-s}}$$

Here, s is a hyperparameter controlling the sharpness of the decay, and η_0 is the initial noise level at $i = 0$. This process continues until the final iteration N, at which point no tokens are re-masked.

The second method used the model's capability to correct text directly without masking tokens. Similar to the scheduled approach, generation begins with an initial prediction $\hat{\mathbf{x}}^{(1)}$ from a fully masked response sequence. However, in all subsequent iterations ($i > 1$), the explicit renoising step is omitted. Instead, the full, unmasked output from the previous step, $\hat{\mathbf{x}}^{(i-1)}$, is fed directly back into the model to produce the next refinement, $\hat{\mathbf{x}}^{(i)}$. In this mode, the model both identifies misplaced or incorrect tokens, and replaces these by a better fitting alternative.

It should be noted that both methods are not equivalent to standard MDM generation. Firstly, any token can be re-masked when the first method is used. Moreover, for both methods the model can amend any token in the sequence, not just those that were explicitly masked.

Lastly, both methods employ an early stopping criterion. The generation process is considered to have converged and is terminated before the preset maximum number of iterations if the output sequence remains identical for three consecutive iterations.

2.6 Evaluation

We evaluated all models on five benchmarks: ARC-Easy and ARC-Challenge (Clark et al. 2018), MMLU (Hendrycks et al. 2020), HellaSwag (Zellers et al. 2019), and GSM8K (Cobbe et al. 2021), using GPT-4o to judge the correctness of generated answers. For most benchmarks, purely diffusive generation was used, except for GSM8K, where semi-autoregressive generation was used. Inference details can be found in Appendix C. For each benchmark, 100 samples were tested per model. To assess text fluency, we computed perplexity using the larger and more recent Phi-4 14B model (Abdin et al., 2024). Output diversity was quantified using the distinct 2-gram fraction, which measures the proportion of unique consecutive word pairs in the generated text. We evaluated six diffusion-based models and three autoregressive base models, all limited to a maximum output length of 256 tokens.

Finally, we also measured the impact of the maximum number of iterations on output fluency for the LAD-8B-1024r model.

2.7 User interface

To demonstrate the test time compute flexibility of the LAD models, we developed two interfaces for

Table 2: Summary of benchmark scores (% correct) for varying model and LoRA sizes. Generative perplexity and distinct-2 gram fractions are also included. Results for the three Llama basemodels, evaluated using our own protocol, are shown for comparison.
Underline = best diffusion model; * = best model for its size; **bold** = best overall model

	Model	**LAD**						**Llama**		
	- Size	1B		3B		8B		1B	3B	8B
	- LoRA rank	128	512	512	1024	1024	2048	-	-	-
Benchmarks	ARC-Easy	56	*58	91	*93	93	93	57	91	***94**
	MMLU	29	30	43	*48	54	53	*48	*48	***65**
	ARC-Challenge	*42	35	68	74	79	***80**	40	*77	***80**
	HellaSwag	27	26	51	*58	***80**	76	*38	55	62
	GSM8K	5	4	37	43	44	45	*49	*72	***80**
Intrinsic metrics	Perplexity	17.1	14.7	11.0	10.1	9.1	9.5	*2.9	*2.8	***2.7**
	Distinct 2-grams	0.88	0.90	0.92	0.94	0.94	0.94	0.92	0.98	0.94

interactive experimentation (Figure 5). The simple interface allows a user to provide a prompt, set the maximum number of generation iterations, and toggle between the two inference methods. A pause function is included to visualize the step-by-step generation process.

The advanced interface provides more detailed control over the generation process. It allows tuning of hyperparameters for the noise schedule (initial fraction, decay sharpness), sampling strategy (top-k, top-p), and bias towards end-of-sequence tokens. This interface also enables alternative methods such as confidence-guided noising, which preferentially re-masks low-confidence tokens, and semi-autoregressive generation, where text is produced in smaller, sequential blocks.

3 Results

Exact model sizes and training durations for the diffusion models are shown in Table 1. The 1B, 3B, and 8B models were trained in approximately 3, 8, and 16 hours, respectively.

Figure 3 shows the training loss curves for all models. Lower final cross-entropy loss values are observed for models with larger parameter sizes and higher LoRA ranks.

The fluency metrics for the LAD-8B-1024r model across different numbers of diffusion iterations are shown in Figure 4. Perplexity decreases as the number of iterations increases, while the distinct 2-gram fraction remains stable and similar to that of the reference text. The figure also includes results for the same model run without intermediate re-noising. In this setting, perplexity remains comparable to the model with re-noising for the same number of iterations, while the distinct 2-gram fraction is slightly lower.

Table 2 presents benchmark scores for all models across five datasets, along with perplexity and distinct 2-gram fraction. Results are reported for diffusion models of varying size and LoRA rank, as well as for the original Llama foundation models. Across most benchmarks—ARC-Easy, MMLU, ARC-Challenge, and HellaSwag—the diffusion models perform comparable to the base models, also for similar sized smaller models. On GSM8K, however, the foundation models show clearly higher accuracy. Perplexity values are higher for all diffusion models compared to the base models, while the distinct 2-gram fractions are similar across all models.

Finally, a short quantitative and qualitative study of diffusive generation without renoising can be seen in Appendix D and E.

4 Discussion

The results demonstrate that LAD effectively adapts pretrained autoregressive LLMs into diffusion-style models with competitive performance and efficient training.

The LAD models were fine-tuned using only 100k iterations, which amounted to 200 million training tokens. For comparison, the base model LLaMA 3.1-8B (Grattafiori et al. 2024) was trained on 15 trillion tokens, and comparable diffusion models such as LlaDa (Nie et al. 2025), used 2.3 trillion tokens. This means LAD used just 0.001% and 0.008% of their respective token counts.

Model performance scales with both model size and LoRA rank, although diminishing returns were seen for large LoRA ranks. This was apparent from both the training loss curves (Figure 3) as well as the benchmark and text fluency results (Table 2). Base model size plays a dominant role in performance, validating the choice to use pre-trained frozen autoregressive backbones with lightweight bidirectional LoRA adapters.

Increasing the number of diffusion refinement iterations reduced perplexity (Figure 4), meaning that increasing ‘test time compute’ increased the performance of the model. This behavior could enable the user to make choices on whether speed or quality is more important, which is a unique feature that is not available through autoregressive generation. The diminishing returns observed beyond a certain number of iterations do suggest practical limits for speed-quality trade-off. We also showed that the model can be used without intermediate re-noising (Figure 6 and Figure 7). This strategy ensures no intermediate information generated by the model is lost, as the model has access to all information generated in the previous iteration.

Benchmark (Table 2) results further showed that LAD models achieve comparable accuracy to base autoregressive models on most datasets despite their non-autoregressive nature, although performance lags on complex reasoning tasks like GSM8K.

Finally, the interactive user interfaces (Figure 5) concretely demonstrate the flexibility enabled by LAD’s diffusion framework. By exposing control over generation iterations, noise scheduling,

sampling strategies, and novel techniques like confidence-guided noising and semi-autoregressive generation, users can tailor compute and output quality trade-offs in real time.

5 Conclusion

LAD provides an efficient and flexible framework to adapt pretrained autoregressive models for diffusion-based generation, demonstrating for the first time that LoRA finetuning alone can enable this transformation. Our unique structured noising strategy makes diffusion without denoising possible, though it does not yet outperform diffusion with renoising. While LAD matches base models on some tasks, complex reasoning challenges like GSM8K remain areas for improvement. Overall, LAD lays a foundation for scalable, controllable diffusion models, with further evaluation needed on diffusion without renoising.

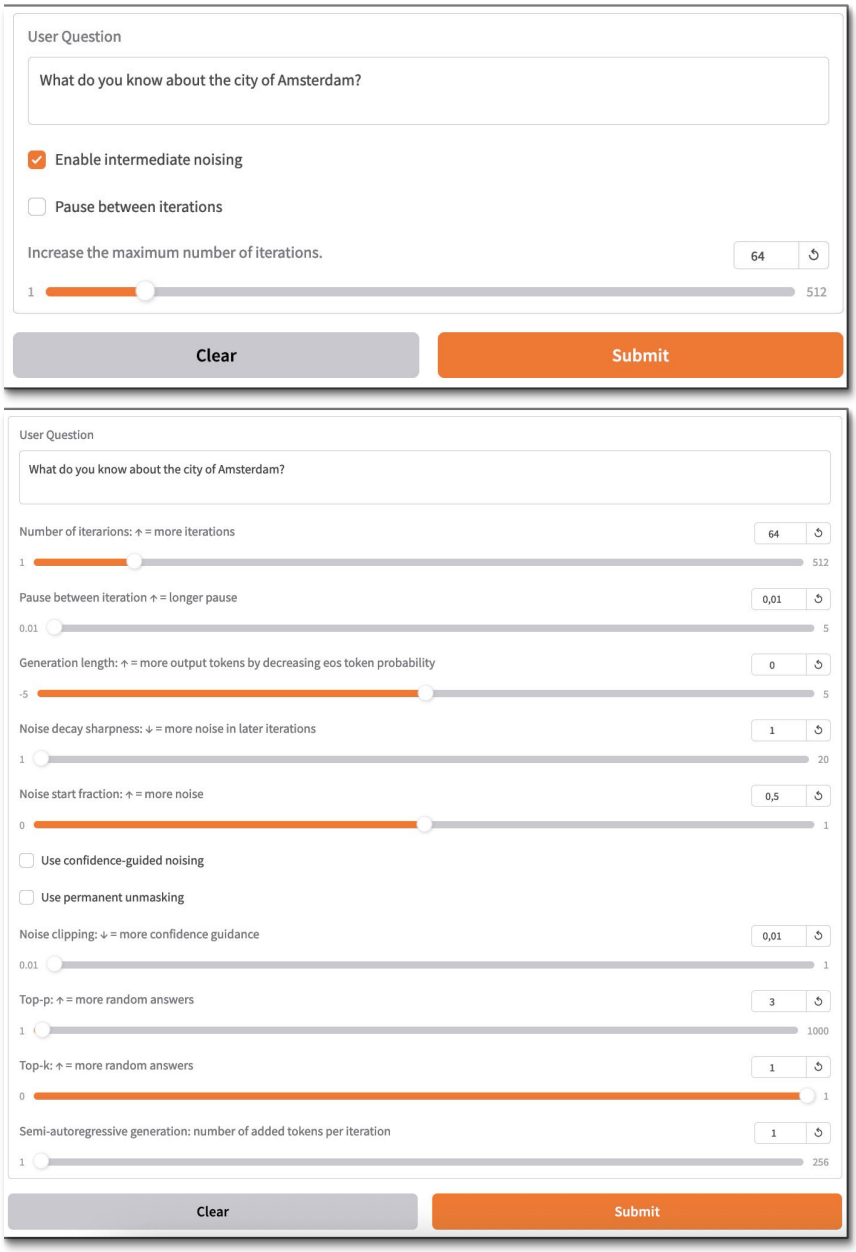

Figure 5: **Interfaces of the LAD model used for inference.** Top shows the interface of the simple demo and decide the inference method. Bottom shows the extended demo which includes a larger number of customization options

Limitations

Our work shares part of its motivation with (von Rütte et al. 2025) who enabled sequence-level correction in diffusion models by augmenting masking with discrete uniform noise (random token replacement). They noted this method, while promising, often degraded benchmark performance. They attributed this to the increased task complexity requiring larger models. We observed similar performance drops using our models, reinforcing the idea that larger models might be necessary to achieve similar results as autoregressive models. However, using the LAD training paradigm might provide a solution to this problem.

Firstly, by applying LoRA to large, pre-existing models, we enable the use of larger networks with only a fraction of the compute required for training from scratch. This parameter-efficient approach training approach could thus offer a viable approach to further test the scalability hypothesis posed by von Rütte et al. Second, we see that fewer iterations are necessary to achieve similar results when using larger models. This could offset the increased compute cost of using the larger models.

Comprehensive evaluation of large language models requires substantial effort, as there are many dimensions to consider—ranging from reasoning ability and factual accuracy to multilingual performance and robustness. While we have attempted to test key aspects of the LAD framework, a full exploration of all capabilities was beyond the scope and length constraints of this study. Nonetheless, the following limitations highlight important areas for future research.

Firstly, diffusion models uniquely enable bidirectional attention and editing, which we did not explicitly study here. Measuring bidirectional reasoning behavior is non-trivial and should be included when validating these models further, as it is a desirable feature of diffusion models.

We were also not able to compare perplexity scores between LAD and other diffusion-based text models. This was primarily due to mismatched generation settings: most diffusion models in prior work are evaluated under unconditional generation, while LAD was trained for instruction-following. Direct comparison would thus be misleading, as conditional perplexity is typically lower.

Furthermore, evaluation was limited to five benchmarks and two intrinsic metrics (perplexity and distinct-2). While these provide a useful first indication of model quality, future work should expand to a broader range of tasks, particularly ones that stress reasoning, factuality, or multilingual capabilities. This will help to better judge the strengths and limitations of diffusion-based text generation.

Likewise, we have so far tested LAD on models up to 8B parameters. While this is already on par with the largest open-source diffusion LLMs to date (Nie et al. 2024, 2025), scaling further should be possible using LoRA finetuning. However, one of our goals was to show that AR models could be adapted for diffusive generation using minimal resources. Even with our parameter-efficeint finetuning methods, models larger than 8B will require access to GPUs with more than 40GB VRAM or multi-GPU setups.

Finally, while our structural noising strategy was designed to simulate typical generation errors, it remains hand-engineered and not learned. This meant that the training samples do not necessarily reflect the intermediate samples generated during inference.

Finally, we provide interactive interfaces for tuning inference hyperparameters, which enables users to intuitively find settings that work best for them. However, a deeper exploration of optimal settings (e.g. for top-k, sharpness, or re-noising schedule) was outside the scope of this paper. These choices may have a strong effect on output quality, and merit further systematic study.

Ethics Statement

This work follows the general principles of the ACM Code of Ethics. No human or sensitive data was used, and we relied solely on publicly available datasets. Ethical risks were considered in line with standard practice for language model research.

Acknowledgments

This research was supported by the Utrecht University Focus Area Applied Data Science. We thank the ADS steering committee for their support.

References

Austin, Jacob, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. "Structured Denoising Diffusion Models in Discrete State-Spaces." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2107.03006.

Cardei, Michael, Jacob K. Christopher, Thomas Hartvigsen, Brian R. Bartoldson, Bhavya Kailkhura, and Ferdinando Fioretto. 2025. "Constrained Language Generation with Discrete Diffusion Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2503.09790.

Chen, Jiaao, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. 2023. "A Cheaper and Better Diffusion Language Model with Soft-Masked Noise." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2304.04746.

Clark, Peter, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. "Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge." *ArXiv [Cs.AI]*. arXiv. http://arxiv.org/abs/1803.05457.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, et al. 2021. "Training Verifiers to Solve Math Word Problems." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2110.14168.

Crumb, A. I. 2023. "Clean-Instruct-3M." https://huggingface.co/datasets/crumb/Clean-Instruct-3M.

Gong, Shansan, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, et al. 2024. "Scaling Diffusion Language Models via Adaptation from Autoregressive Models." *ArXiv [Cs.CL]*. arXiv. https://github.com/HKUNLP/DiffuLLaMA.

Goodfellow, Ian J., Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. "An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks." *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/1312.6211.

Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. 2024. "The Llama 3 Herd of Models." *ArXiv [Cs.AI]*. arXiv. http://arxiv.org/abs/2407.21783.

Han, Kehang, Kathleen Kenealy, Aditya Barua, Noah Fiedel, and Noah Constant. 2024. "Transfer Learning for Text Diffusion Models." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2401.17181.

He, Zhengfu, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. "DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2211.15029.

Hendrycks, Dan, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. "Measuring Massive Multitask Language Understanding." *ArXiv [Cs.CY]*. arXiv. http://arxiv.org/abs/2009.03300.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. "LoRA: Low-Rank Adaptation of Large Language Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2106.09685.

Huang, Siming, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, et al. 2024. "OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2411.04905.

Li, Xiang Lisa, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. "Diffusion-LM Improves Controllable Text Generation." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arXiv.2205.14217.

Li, Yifan, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. "Diffusion Models for Non-Autoregressive Text Generation: A Survey." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2303.06574.

Loshchilov, Ilya, and Frank Hutter. 2017. "Decoupled Weight Decay Regularization." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/1711.05101.

Lou, Aaron, Chenlin Meng, and Stefano Ermon. 2023. "Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution." *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/2310.16834.

Lovelace, Justin, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. 2022. "Latent Diffusion for Language Generation." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2212.09462.

Luo, Yun, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. "An Empirical Study of Catastrophic Forgetting in Large Language Models during Continual Fine-Tuning." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2308.08747.

Mitra, Arindam, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. "Orca-Math: Unlocking the Potential of SLMs in Grade School Math." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2402.14830.

Nie, Shen, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. "Scaling up Masked Diffusion Models on Text." *ArXiv [Cs.AI]*. arXiv. http://arxiv.org/abs/2410.18514.

Nie, Shen, 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 [Cs.CL]*. arXiv. http://arxiv.org/abs/2502.09992.

Ou, Jingyang, 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 [Cs.LG]*. arXiv. http://arxiv.org/abs/2406.03736.

Peng, Baolin, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. “Instruction Tuning with GPT-4.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2304.03277.

Rütte, Dimitri von, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. 2025. “Generalized Interpolating Discrete Diffusion.” *ArXiv [Cs.CL]*. arXiv. https://github.com/dvruette/gidd/.

Sahoo, Subham Sekhar, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. “Simple and Effective Masked Diffusion Language Models.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2406.07524.

Schiff, Yair, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. 2024. “Simple Guidance Mechanisms for Discrete Diffusion Models.” *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2412.10193.

Shi, Jiaxin, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. “Simplified and Generalized Masked Diffusion for Discrete Data.” *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2406.04329.

Sohl-Dickstein, Jascha Narain, Eric A. Weiss, Niru Maheswaranathan, and S. Ganguli. 2015. “Deep Unsupervised Learning Using Nonequilibrium Thermodynamics.” Edited by Francis Bach and David Blei. *International Conference on Machine Learning*, Proceedings of Machine Learning Research, abs/1503.03585 (March): 2256–65.

Taori, Rohan, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. 2023. “Stanford Alpaca: An Instruction-Following LLaMA Model.” https://github.com/tatsu-lab/stanford_alpaca.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/1706.03762.

Ye, Jiasheng, Zaixiang Zheng, Yu Bao, Lihua Qian, and Quanquan Gu. 2023. “Diffusion Language Models Can Perform Many Tasks with Scaling and Instruction-Finetuning.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2308.12219.

Yi, Qiuhua, Xiangfan Chen, Chenwei Zhang, Zehai Zhou, Linan Zhu, and Xiangjie Kong. 2024. “Diffusion Models in Text Generation: A Survey.” *PeerJ. Computer Science* 10 (e1905): e1905.

Yuan, Hongyi, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. “SeqDiffuSeq: Text Diffusion with Encoder-Decoder Transformers.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2212.10325.

Zellers, Rowan, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. “HellaSwag: Can a Machine Really Finish Your Sentence?” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/1905.07830.

Zhang, Yizhe, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. 2023. “PLANNER: Generating Diversified Paragraph via Latent Language Diffusion Model.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2306.02531.

Zhao, Siyan, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. “D1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2504.12216.

Zhu, Yuansong, and Yu Zhao. 2023. “Diffusion Models in NLP: A Survey.” *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2303.07576.

Zou, Hao, Zae Myung Kim, and Dongyeop Kang. 2023. “A Survey of Diffusion Models in Natural Language Processing.” *ArXiv [Cs.CL]*. arXiv. https://www.semanticscholar.org/paper/4ca824e792f3a2c777ffa5896a2e7cdf11b9518d.

A Model hyperparameters

All models were trained using the same training framework. The foundation models used were the Llama 3.2 1B- and 3B-Instruct and Llama 3.1 8B-Instruct models. For computational efficiency, all training was performed using 16-bit floating-point precision. The `AdamW` optimizer was used with a `cosine` learning rate scheduler, starting with a learning rate of `1e-5` after 100 warmup steps. We used a weight decay of 0.01 and a maximum gradient norm of 0.5. The models were trained for a single epoch with a per-device batch size of 8. All input sequences were tokenized and then padded or truncated to a fixed length of 256 tokens.

To analyze the impact of parameter-efficient fine-tuning, we evaluated several model variants by applying Low-Rank Adaptation (LoRA) to the base model. As summarized in the main text, we varied the LoRA rank (`r`) across values of 128, 512, 1024 and 2048, also depending on the model size. For all LoRA configurations, the scaling parameter `lora_alpha` was set equal to the rank, and LoRA was applied to the query (`q_proj`) and value (`v_proj`) matrices of the attention mechanism. No dropout was used in the LoRA layers. This approach allowed us to create a range of models with varying parameter counts while keeping the core architecture and training hyperparameters consistent.

B Training noising schedule

Our approach is motivated by the empirical observation that when applying diffusion-style models to text, intermediate outputs often contain relevant lexical content but exhibit structural issues, such as repeated words, incorrect word order, or sentence fragments. To train a model capable of correcting such errors, we introduce a hybrid corruption process that combines token masking with structured, non-local perturbations. Let $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ represent a clean sequence of tokens from the data distribution $p_{\text{data}}(\mathbf{x})$. Our corruption process, $C(\mathbf{x})$, is a stochastic function that produces a corrupted sequence $\tilde{\mathbf{x}}$. This process is a composition of two distinct noising strategies:

- **Token masking (C_{mask}):** We employ a masking operator that replaces a fraction of tokens with a special [MASK] token. This is analogous to the forward process in Masked Diffusion Models (MDMs). Let $\mathbf{m} \in \{0,1\}^L$ be a binary mask vector sampled from a uniform random distribution, where $m_i = 1$ indicates masking. The masking process can be defined as:
$$C_{\text{mask}}(\mathbf{x}, \mathbf{m})_i = \begin{cases} [\text{MASK}] & \text{if } m_i = 1 \\ x_i & \text{if } m_i = 0 \end{cases}$$
- **Structural perturbation (C_{struct}):** We apply a set of non-invertible, structure-altering

Algorithm 1: Masked and Structured Noising for Denoising Training

Require: token sequence x = [x_1, ..., x_n], masking token MASK, noise probability p ∈ [0, 0.5], RNG

1: With 50% probability: # Masking
2: Sample mask fraction f ~ Uniform(0, 1)
3: Let m ← floor(f × n)
4: Randomly select m indices I_mask ⊆ {1, ..., n} without replacement
5: For each i ∈ I_mask, set x_i ← MASK
6: For each position i ∈ {1, ..., n−1}: # Swapping
7: With probability p/4, swap $\hat{x}_i$ and $\hat{x}$_{i+1}
8: For each position i ∈ {1, ..., n}: # Duplication
9: With probability p/4, do:
10: Sample direction d ∈ {−1, +1}
11: If i + d ∈ [1, n], set $\hat{x}_i$ ← $\hat{x}$_{i+d}
12: With probability p/4: # Span shift
13: Sample span length s ∈ {1, 2, 3}
14: Sample shift distance δ ∈ {1, ..., 4} and direction d ∈ {−1, +1}
15: Let start ∈ {1, ..., n − s + 1}
16: Let span ← x_{start : start + s − 1}
17: Let target ← clamp(start + d × δ, 1, n − s + 1)
18: Overwrite x_{target : target + s − 1} ← span
19: Return x

transformations, including local token swaps, duplications, and random shifts of token spans. These operators are designed to mimic the specific structural artifacts we observe during iterative generation.

The full corruption $\tilde{\mathbf{x}} = C(\mathbf{x})$ is a probabilistic application of these operators. This hybrid approach enables us to create a distribution of corrupted samples that trains the model to correct both content-level (via masking) and structure-level errors.

Our methodology departs from the usual formal probabilistic framework of Masked Diffusion Models (MDMs) as presented by, among others, (Shi et al. 2024; Ou et al. 2024; Nie et al. 2025; Sahoo et al. 2024). A typical MDM defines a time-indexed forward process $q(\mathbf{x}_t|\mathbf{x}_0)$ that gradually masks tokens, where $t \in [0,1]$ represents the noise level, and each token is independently masked with probability t. This process allows for the derivation of a reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and an objective function $\mathcal{L}(\theta)$ that serves as a variational upper bound on the negative log-likelihood of the data:

$$-\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log p_\theta(\mathbf{x})] \leq \mathcal{L}(\theta)$$

This provides a principled, likelihood-based training framework. In contrast, our corruption process $\mathcal{C}(\mathbf{x})$ is not defined as a time-indexed, reversible Markov chain. The structural perturbations in C_{struct} are deterministic operations applied stochastically which are not directly invertible. Consequently, we cannot define a corresponding probabilistic forward model or derive a tractable variational bound on the data likelihood.

Given the nature of our corruption process, we frame the training as a direct denoising auto-encoding task rather than likelihood maximization. The model parameterized by θ, is trained to reconstruct the original sequence $\mathbf{x}$ from its corrupted version $\tilde{\mathbf{x}}$. The objective is to minimize the standard cross-entropy loss between the model's output distribution and the original clean sequence:

$$\mathcal{L}_{\text{CE}}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\tilde{\mathbf{x}}\sim\mathcal{C}(\mathbf{x})}\left[-\sum_{i=1}^{L}\log p_\theta(x_i|\tilde{\mathbf{x}})\right]$$

While this prevents tractable likelihood evaluation, it allows the model to learn a practical denoising function. The masking component serves two critical roles within this framework. First, a fully masked input ($\tilde{\mathbf{x}}$ generated with a 100% mask rate) provides a well-defined starting point for generation from pure noise, analogous to $\mathbf{x}_1$ in formal diffusion. Second, partial masking provides stochasticity during the inference process that enables more diverse and natural responses, which complements the convergence to grammatically sound sentences offered by the structured corruptions. This combined approach trains a single model capable of both iterative refinement of flawed text and conditional generation from a corrupted prompt.

To simulate intermediate inference steps, we apply a structured noising function to each sequence of tokens. For each sequence, a noise probability $p\in[0.0,0.5]$ is sampled uniformly and used to parameterize three types of corruption. First, with 50% probability, a random fraction (between 0% and 100%) of the tokens is replaced with a special `MASK` token. Second, adjacent tokens are randomly swapped with probability $p/4$. Third, tokens are duplicated either from the previous or next position, also with probability $p/4$. Finally, with probability $p/4$, a short span of 1 to 3 tokens is copied and shifted to a new location within the sequence, moved by 1 to 4 positions in either direction. These noising operations are applied during preprocessing via a batched mapping function and are only performed on sequences of at least two tokens. The resulting corrupted input serves as the model input, while the original sequence is used as the target for supervised training. An algorithmic notation of the noising process is shown in Algorithm 1.

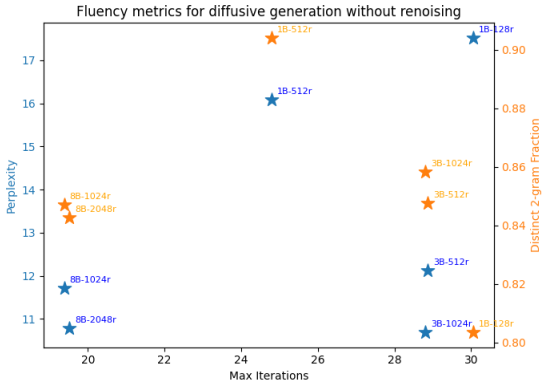

Figure 6: **Perplexity and distinct 2-gram fraction for multiple LAD models without intermediate re-noising.** Each star represents a model's performance at its average convergence iteration, shown on the x-axis.

C Evaluation details

We evaluated model performance on five standard benchmarks: ARC-Easy, MMLU, ARC-Challenge, HellaSwag, and GSM8K. For each benchmark, we sampled 100 representative examples per model. Answers were generated using a custom diffusion-based decoding process, with settings tailored to each benchmark.

For ARC-Easy, MMLU, ARC-Challenge, and HellaSwag, model responses were generated using 16 diffusion steps, with a noise schedule starting at 1.0, a maximum generation length of 256 tokens, top-k sampling set to 1, and top-p sampling set to 0.1. These settings encourage the model to refine highly noised inputs into coherent answers over a fixed number of iterations.

For GSM8K, which involves multi-step mathematical reasoning, we used a different configuration: the generation process was allowed up to 256 diffusion steps, with no initial noise (i.e., noise start = 0.0), a maximum generation length of 256 tokens, top-k set to 1, and top-p set to 1.0. In addition, we used semi-autoregressive generation, allowing the model to generate and refine 4 tokens at a time. A noising sharpness value of 1.0 was used to guide the refinement process. We followed the semi-autoregressive implementation described by (Nie et al. 2025).

Answer correctness was evaluated using GPT-4o, prompted in a zero-shot setting to determine whether each model's response correctly answered the original question. All diffusion-based models were capped at a maximum of 256 generated tokens. In addition to the six diffusion-tuned models, we also evaluated three autoregressive base models under identical constraints.

To assess fluency, we calculated perplexity using the Phi-4 14B model (Abdin et al., 2024), a large autoregressive language model trained for strong generalization and language modeling quality. We selected Phi-4 for this purpose under the assumption that stronger models produce more reliable and meaningful evaluations of fluency. Evaluating weaker or mid-sized models with a more capable evaluator helps prevent underestimation of fluency due to evaluator limitations.

To measure lexical diversity, we computed the distinct 2-gram fraction for each model output. This metric represents the proportion of unique consecutive word pairs (bigrams) in the generated

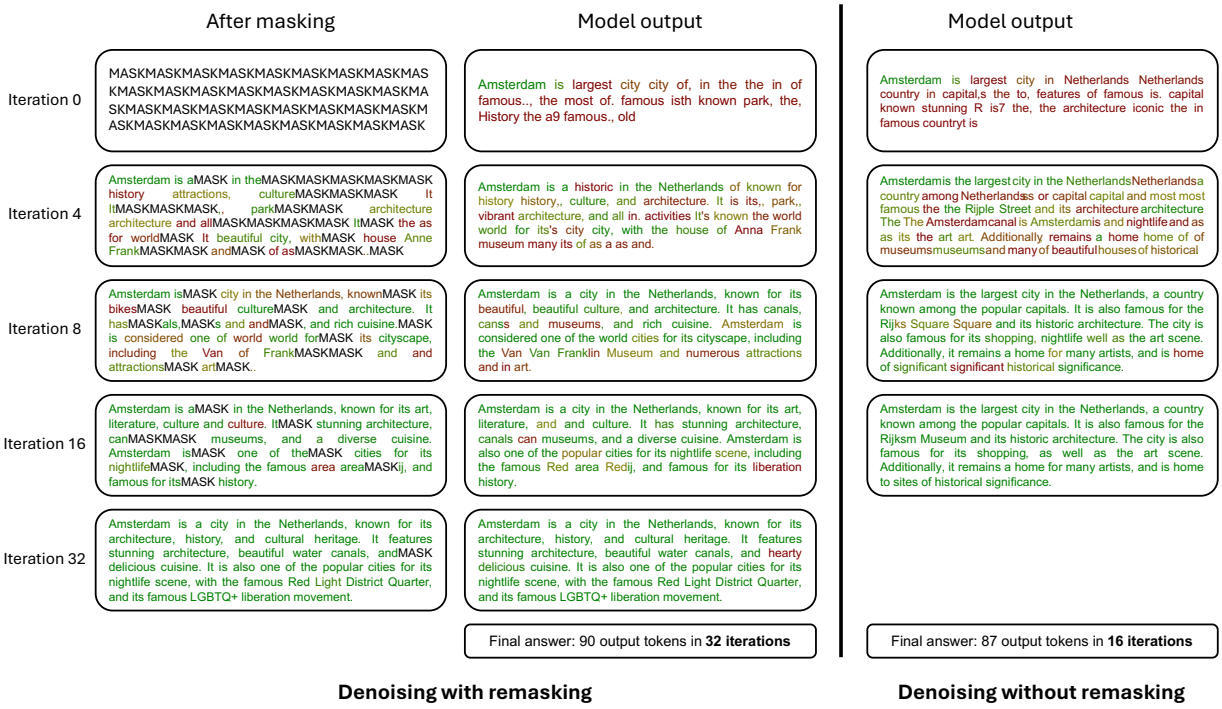

Figure 7: **Examples of model output with and without renoising after various iterations.** Left: Denoising with renoising after each step. Tokens are renoising with decreasing frequency to allow iterative correction. Right: Denoising without renoising. Tokens are refined only once, reducing flexibility but speeding up inference. For both versions, token shading reflects model confidence, ranging from red (low certainty) to green (high certainty). "MASK" refers to a masking token.

text, offering a simple yet informative measure of repetition and variation. A higher distinct-2 score indicates more varied, less repetitive output.
Perplexity and distinct-2 scores were computed for each model using 64 diffusion iterations. In addition, we performed an ablation study on the LAD-8B-1024r model to examine the effect of varying the number of diffusion steps. This model was tested at 8, 16, 32, 64, 128, 256 and 512 iterations. We also evaluated model performance without re-noising, allowing the model to iteratively refine its output until convergence or a hard limit of 256 iterations. All tests were conducted with a maximum context window of 256 tokens.

D Diffusion without renoising

The results in Figure 6 indicate that increasing model size leads to lower perplexity when performing diffusion without intermediate renoising, reflecting improved fluency, particularly when scaling from 1B to 3B parameters. Further scaling to 8B not only reduces perplexity but also decreases the average number of iterations required for convergence, suggesting faster refinement. A similar trend is observed with higher LoRA ranks, which contribute to both improved perplexity and efficiency. Across all models, distinct 2-gram fractions remain consistently high between 0.8 and 0.9, indicating relatively stable lexical diversity regardless of model size or LoRA rank.

E Inference Examples

Figure 6 illustrates a side-by-side comparison of an answer to a single prompt, denoised using LAD with and without intermediate renoising. Diffusion with renoising allows for flexible sequence-level corrections over time, often leading to more coherent and confident completions. In contrast, the right panel shows inference without renoising, where each token is refined only once. While this approach is faster, it limits the model's ability to revisit and revise earlier decisions, resulting in less polished outputs in some cases.