Less Uniform Discrete Diffusion is More Powerful and Scalable

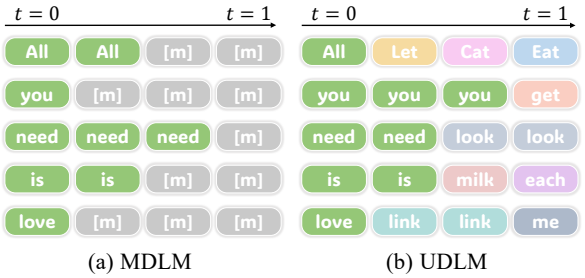
Anonymous ACL submission

Abstract

Although uniform diffusion language models (UDLMs) represent a promising diffusion paradigm, scaling them remains challenging. We identify the core obstacle as an over-uniform training objective and condition-target confusion during sampling. To address these, we propose Less Uniform Diffusion (LUDI), a novel UDLM framework. Specifically, we (i) introduce a less uniform loss that directs each reverse transition toward the clean token, and (ii) equip the model with per-token time embeddings that supply token-level corruption hints, enabling confidence-based few-step sampling. Experiments across scales show that LUDI yields cleaner supervision and improves few-step generation. We further continue-train a 7B autoregressive model into LUDI-7B, resulting in the first UDLM capable of complex reasoning. It achieves 3× speedup over AR decoding and a competitive performance compared with masked diffusion baselines, revealing that the full potential of UDLMs for complex generation remains to be unlocked.

1 Introduction

Diffusion large language models (dLLMs) have emerged as competitive alternatives to autoregressive (AR) models, driven by their potential for parallel decoding and bidirectional context modeling (Sahoo et al., 2024; Ou et al., 2025). Among them, masked diffusion language models (MDLMs, Figure 1(a)), which use a mask token as the noise prior, dominate current research, with models at 8B and even 100B scales delivering strong results on tasks like mathematics and programming (Nie et al., 2025b; Bie et al., 2025). Uniform diffusion language models (UDLMs, Figure 1(b)), which instead adopt a uniform prior over the vocabulary, represent another branch that has shown greater potential for few-step generation and self-correction (Sahoo et al., 2025; Schiff et al., 2025). Yet their development lags significantly behind. To date, the advantages of UDLMs have only been demonstrated at small scales, and their viability on complex reasoning tasks remains unestablished.

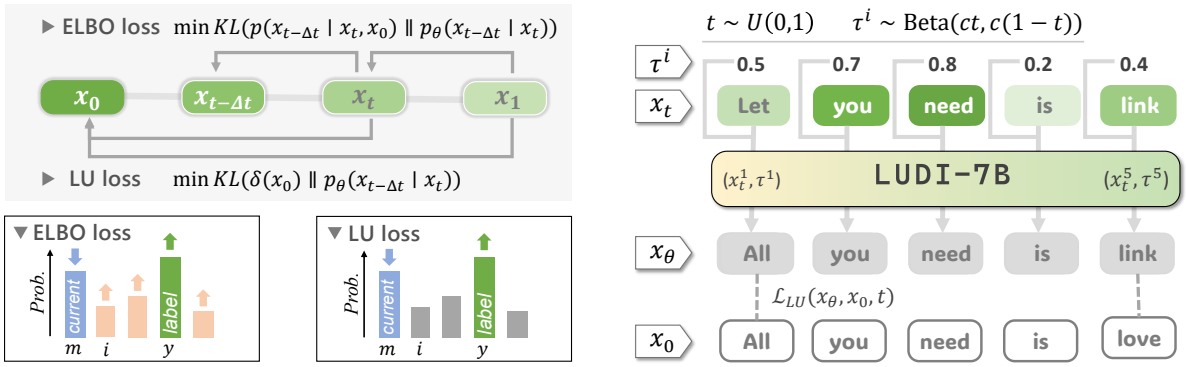


(a) MDLM (b) UDLM

Figure 1: Schematic comparison between masked diffusion (a) and uniform diffusion (b).

Prior efforts mainly focused on conceptually validating the promise of UDLMs. UDLMs can be distilled through connections to continuous Gaussian diffusion (Sahoo et al., 2025), trained stably with a simplified loss (Zhu et al., 2025c), and exhibit more favorable scaling behavior under data-constrained conditions (von Rütte et al., 2025b). Nevertheless, existing efforts remain confined to small datasets or scaling-law investigations, leaving a pressing question: *can UDLMs scale to a larger regime and handle complex reasoning tasks?*

We find that UDLMs do not scale up as naturally as MDLMs. The central difficulty arises from an *over-uniform training objective* and *condition-target confusion* during sampling. During training, the KL-based objective provides an excessively label-smoothed learning signal: the model's predictions become more uniform and less confident in the clean token. During sampling, for complex tasks, simpler tokens should be generated first to serve as conditions, after which the remaining tokens are denoised. In UDLMs, however, all tokens are denoised at a uniform rate. While this works for simple tasks, it leads to condition-target confusion on complex tasks, eventually causing the model to collapse to repeated tokens. These two



(a) Less uniform loss

(b) LUDI with per token time

Figure 2: Illustration of Less Uniform Diffusion (LUDI). (a) **LU Loss.** The ELBO loss (*left*) matches the reverse transition to $p(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_0)$, leading to over-uniform by encouraging all token's probabilities ($\sum_i$, ↑). Our LU loss (*right*) directly aligns the transition with $\delta(\mathbf{x}_0)$, removing the smoothing term and retaining only the suppression of the erroneous token (m ↓) and promotion of the target token (y ↑). (b) **Per-token time embedding.** The model receives token-level corruption hints via per-token time embeddings. Combined with the LU loss, this allows scaling LUDI to 7B parameters, enabling confidence-based decoding for complex reasoning tasks.

issues jointly limit the performance of UDLMs on complex tasks: over-uniformity reduces training efficiency and weakens few-step generation capability, while uniformly updating all tokens across many steps leads to condition-target confusion.

To address these issues and make UDLMs practically scalable, we propose **L**ess **U**niform **Di**ffusion (LUDI). LUDI introduces two key modifications. First, by simplifying and approximating the original UDLM training loss, we decompose it into three terms and remove the term that encourages uniform predictions, yielding a less uniform (LU) loss (Figure 2(a)). We prove theoretically that LU loss steers each reverse transition toward the clean token, supporting few-step sampling. Second, to mitigate the confusion between conditions and targets, we provide each token with an individual time embedding during training, which indicates the probability that the token has been randomly replaced (Figure 2(b)). This allows the model to better distinguish condition tokens from target tokens and naturally enables confidence-based sampling strategies akin to those in MDLMs, facilitating an easy-to-hard generation paradigm.

We conduct extensive experiments to validate LUDI. Pretraining at 170M and 1B scales demonstrates that the LU loss offers cleaner supervision and improves both few-step generation and downstream task performance. We further obtain LUDI-7B by continuing training from an autoregressive checkpoint, which, to the best of our knowledge, is the first UDLM capable of complex reasoning tasks. Compared with the AR model, LUDI-7B achieves approximately 3× speedup while maintaining comparable performance. Compared with MDLMs, LUDI-7B also achieves a competitive performance, revealing that the potential of UDLMs for complex generation remains largely untapped. Our contributions are threefold:

1. We attribute the difficulty of scaling UDLMs to over-uniform training objectives and condition-target confusion during sampling.

2. We propose the LUDI loss and per-token time embeddings, enabling UDLMs to perform confidence-based few-step generation.

3. Our experiments with LUDI-7B establish that UDLMs scale effectively for complex reasoning, delivering a competitive balance between generation quality and speed.

2 Background

In this section, we review UDLM. A clean token is corrupted by the forward process into the uniform distribution. By parameterizing and learning the reverse process, samples can be generated by gradually denoising from the uniform prior.

2.1 Forward Process

Continuous-time discrete diffusion models corrupt the data distribution $p_0(\mathbf{x})$ into a noise prior through a forward Markov process. In masked diffusion models, the terminal prior is a point mass on the mask token. In uniform-state diffusion models, as t evolves from 0 to 1, a clean token $\mathbf{x}_0$ is gradually perturbed toward the uniform distribution over

the vocabulary $\mathcal{V}$, where $|\mathcal{V}| = V$. Over an infinitesimal interval from t to $t + \Delta t$, the transition probability of $\mathbf{x}_t$ (ignoring $o(\Delta t)$ terms) is

$$p(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t) = \delta(\mathbf{x}_t, \mathbf{x}_{t+\Delta t}) + Q_t(\mathbf{x}_t, \mathbf{x}_{t+\Delta t})\Delta t.$$

Here, $\mathbf{x}_0$ and $\mathbf{x}_t$ are one-hot vectors over $\mathcal{V}$, $\delta(\mathbf{x}_t, \mathbf{y}) = \mathbb{I}_{\mathbf{y}=\mathbf{x}_t}$, and $Q_t \in \mathbb{R}^{V\times V}$ denotes the transition-rate matrix. For uniform-state diffusion, the rate matrix is given by $Q_t(\mathbf{e}_i, \mathbf{e}_j) = -\frac{\alpha_t'}{\alpha_t}\left(\frac{1}{V} - \mathbb{I}_{i=j}\right)$, where α_t is a predefined noise schedule satisfying $\alpha_0 = 1$ and $\alpha_1 = 0$. A common choice is $\alpha_t = 1 - t$. This forward process yields the marginal distribution

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathrm{Cat}\left(\cdot;\, \alpha_t \mathbf{x}_0 + (1-\alpha_t)\frac{\mathbf{1}}{V}\right).$$

The exact reverse posterior $p(\mathbf{x}_s \mid \mathbf{x}_t, \mathbf{x}_0) = \mathrm{Cat}\left(\cdot;\, \boldsymbol{\pi}_{t\to s}\right)$ can be written in closed form. Let $\alpha_{t|s} := \alpha_t/\alpha_s$ and define $\bar{\mathbf{x}} := V\alpha_t \mathbf{x} + (1-\alpha_t)\mathbf{1}$:

$$\boldsymbol{\pi}_{t\to s} = \frac{1}{\langle \bar{\mathbf{x}}_t, \mathbf{x}_0\rangle}\Big[V\alpha_t\, \mathbf{x}_t \odot \mathbf{x}_0 + (\alpha_{t|s} - \alpha_t)\mathbf{x}_t + (\alpha_s - \alpha_t)\mathbf{x}_0 + (1-\alpha_{t|s})(1-\alpha_s)\frac{\mathbf{1}}{V}\Big]. \quad (1)$$

We provide a more detailed derivation in Appendix A.1.1.

2.2 Reverse Process and Training Objective

To sample from $p_0(\mathbf{x})$, one learns a reverse denoising process from the noise prior. This requires parameterizing the reverse transition $p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$ and matching it to the exact posterior $p(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_0)$. The standard diffusion objective minimizes the local KL divergence

$$\mathrm{KL}\left(p(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)\right), \quad (2)$$

which forms a local term in the negative evidence lower bound (NELBO).

SEDD (Lou et al., 2023) parameterizes $p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$ through rate ratios. Duo (Sahoo et al., 2025) instead parameterizes the reverse transition as $p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t) = p(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_\theta(\mathbf{x}_t, t))$, that is, the model directly predicts $\mathbf{x}_0$. Both approaches optimize the KL objective in Eq. (2). Let $m = \arg\max_j (\mathbf{x}_t)_j$ denote the current noisy token and $y = \arg\max_j (\mathbf{x}_0)_j$ denote the clean token. Under the Duo parameterization, the token-level loss $\mathcal{L}_{\mathrm{Duo}}^{\ell}$ at position ℓ is

$$\mathbb{E}_{t,\mathbf{x}_t}\frac{-\alpha_t'}{V\alpha_t}\left[\frac{V}{\bar{x}_{\theta,m}} - \frac{V}{\bar{x}_{0,m}} + \sum_{j=1}^{V}\frac{\bar{x}_{0,j}}{\bar{x}_{0,m}}\log\frac{\bar{x}_{\theta,m}\bar{x}_{0,j}}{\bar{x}_{\theta,j}\bar{x}_{0,m}}\right]. \quad (3)$$

Summing over all positions ℓ gives the sentence-level training objective. We provide the SEDD loss in Appendix A.1.3 and show that it is equivalent to the Duo loss up to parameterization.

SDDLM (Zhu et al., 2025c) follows the same parameterization as Duo, but introduces empirically simplified objectives for more stable and efficient training. The SDDLM and SDDLM-v1 losses are:

$$\mathcal{L}_{\mathrm{SD}}^{\ell} = -\mathbb{E}_{t,\mathbf{x}_t}\mathbb{I}_{\mathbf{x}_t\neq\mathbf{x}_0}\log x_{\theta,y}, \quad (4)$$

$$\mathcal{L}_{\mathrm{SD1}}^{\ell} = -\mathbb{E}_{t,\mathbf{x}_t}\mathbb{I}_{\mathbf{x}_t\neq\mathbf{x}_0}\left[\log x_{\theta,y} - \frac{1}{V}\sum_{j=1}^{V}\log x_{\theta,j}\right].$$

Unlike $\mathcal{L}_{\mathrm{Duo}}$, which corresponds to a NELBO objective, these simplified losses are no longer NELBOs in theory, but have been observed to yield stronger empirical performance.

3 Methodology

In this section, we address the over-uniform and condition-target confusion that prevent UDLMs from scaling to complex tasks. The lack of a clean supervisory signal forces the model to rely on many denoising steps during sampling, yet complex tasks demand that simpler tokens be decoded quickly and serve as conditioning context. To resolve this tension, we first analyze the source of over-uniform and mitigate it with the LU loss (Sec. 3.1), then introduce per-token time embeddings that provide token-level corruption hints (Sec. 3.2). Together, the LU loss endows LUDI with stronger few-step generation capability, enabling it to rapidly decode high-confidence tokens and update their per-token time embeddings as conditions, thereby handling complex generation tasks effectively.

3.1 Less Uniform Loss

Over-uniform phenomenon. In MDLMs, training based on ELBO has achieved widespread success, being used for large-scale pretraining or AR-to-diffusion training. The ELBO objective in UDLMs, while theoretically well grounded, is substantially less effective in practice. We attribute this to the *mode-covering* property of the KL loss, which drives the model toward over-uniform and drowns out meaningful supervision.

To elucidate this issue, we simplify the complex original loss in Eq.(3) into a CE-like form, as stated in the following proposition.

Proposition 1 (Proved in the Appendix A.2.1) Under mild approximations, optimizing the ELBO loss $\mathcal{L}^{\ell}_{\text{ELBO}}$ in Eq. (3) at positions where $\mathbf{x}_t \neq \mathbf{x}_0$ can be approximated as optimizing

$$-\alpha_t' \mathbb{E}_{t,\mathbf{x}_t}\left[\frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} - \frac{1}{V\alpha_t}\sum_{i=1}^{V} \log \bar{x}_{\theta,i}\right]$$
$$= -\alpha_t' \mathbb{E}_{t,\mathbf{x}_t}\left[\frac{\text{CE}_{\text{LS}}^{1-\alpha_t}(\bar{\mathbf{x}}_\theta, y)}{\alpha_t(1-\alpha_t)} + \frac{\log \bar{x}_{\theta,m}}{1-\alpha_t}\right]. \quad (5)$$

For tokens where $\mathbf{x}_t^i = \mathbf{x}_0^i$, the contribution is $O(1/V)$ and negligible for large vocabularies.

When $\mathbf{x}_t^i \neq \mathbf{x}_0^i$, the loss decomposes into three terms: (1) suppression of the current erroneous token $\log \bar{x}_{\theta,m}$; (2) learning of the correct token $-\log \bar{x}_{\theta,y}$; (3) a smoothing term over the uniform distribution $-\sum_i \log \bar{x}_{\theta,i}$. Notably, (2) and (3) can form a label-smoothed cross-entropy objective CE_{LS} with effective target $\alpha_t \mathbf{x}_0 + (1-\alpha_t)\mathbf{1}/V$. The smoothing strength $1-\alpha_t$ is unusually large. Under the common schedule $\alpha_t = 1-t$ with $t \sim \mathcal{U}(0,1)$, on average only half of the probability mass is placed on clean token $\mathbf{x}_0$.

This stems from the mode-covering nature of the KL divergence. Whenever $p(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{x}_0)$ assigns probability to random tokens, the learned reverse transition kernel must cover those probabilities, otherwise it incurs a large penalty. Consequently, when t is large, the model receives little effective supervision toward the clean token and is instead driven toward overly uniform predictions.

Remark 1. This problem is circumvented in mask diffusion. Although mask diffusion also employs a KL-based ELBO, the loss decouples into t-dependent and t-independent terms. In MDLMs, the probability of a token staying masked is computed analytically rather than learned.

Remark 2. While SD and SDv1 in Eq. (4) heuristically remove the smoothing term $-\sum_i \log \bar{x}_{\theta,i}$ and even introduce anti-smoothing, they also discard the suppression loss $\log \bar{x}_{\theta,m}$ on the erroneous token, thereby weakening the model's correction capability and lacking theoretical support.

LU loss. To mitigate over-uniform, a simple fix is to directly remove term $-\sum_i \log \bar{x}_{\theta,i}$. We obtain a *less uniform* loss, termed the LU loss:

$$\mathcal{L}^{\ell}_{\text{LU}} = \frac{-\alpha_t'}{1-\alpha_t}\mathbb{E}_{t,\mathbf{x}_t}\left(\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}\right). \quad (6)$$

Despite its simplicity, we can prove that minimizing the LU loss is equivalent to minimizing the KL divergence between $p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$ and a Dirac delta at the clean token.

Theorem 1 (Proved in the Appendix A.2.2). At corrupted positions $m \neq y$, minimizing $\mathcal{L}^{\ell}_{\text{LU}}$ is equivalent to minimizing the finite θ-dependent part of the Dirac-target KL:

$$\lim_{\Delta t \to 0} \text{KL}\big(\delta(\mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)\big). \quad (7)$$

The LU loss simultaneously reduces the probability of the current incorrect token and increases the probability of the clean token $\mathbf{x}_0$. Moreover, the use of $\bar{\mathbf{x}}_\theta = V\alpha_t \mathbf{x}_\theta + (1-\alpha_t)\mathbf{1}$ prevents the logarithmic terms from becoming singular. As a result, the learned transition is consistently encouraged to move toward $\mathbf{x}_0$ for any time step. Intuitively, an ELBO-trained model behaves more like a *diffusion model*, whereas the LU loss aligns more closely with a *consistency model*, as illustrated in Figure 2 (a). Our LUDI loss offers two key advantages:

(*a*) LU loss avoids the excessive smoothing induced by the KL objective and instead aligns the parameterized transition rate with the clean token. This provides a stronger learning signal and improves few-step generation through consistency.

(*b*) Compared with the ELBO-based loss, our formulation involves only the two indices y and m, making it computationally efficient. With optimized operators, LUDI can achieve a $4.39\times$ speedup and a $3.45\times$ memory reduction. While the dense nature of the ELBO loss is computationally expensive for large vocabularies. (A detailed efficiency analysis is deferred to the Appendix C.3.)

3.2 Scaling up UDLM

Condition-target confusion. When scaling UDLMs to the billion-parameter regime for challenging downstream tasks such as mathematics and code generation, a central obstacle is the ambiguity between conditioning context and denoising targets. Difficult reasoning tasks require the model to first establish a reliable condition, such as a mathematical equation, and then infer the target, such as the solution. In masked diffusion, this separation is explicit: clean tokens serve as conditions, while mask tokens identify the denoising targets. In UDLMs, however, the model is given only the global time t, which indicates the overall corruption level but not the status of each

individual token. The model therefore struggles to distinguish condition tokens from target tokens and tends to denoise all positions simultaneously in a condition-independent manner. This behavior is particularly fatal for complex reasoning.

Per-token time embedding. To address this ambiguity, we introduce per-token time embeddings. The denoiser is written as $\mathbf{x}_\theta(\mathbf{x}_t, \boldsymbol{\tau})$, where $\boldsymbol{\tau} = (\tau^1, \ldots, \tau^L)$ assigns a distinct time value to each position, as shown in Figure 2(b). Each τ^i indicates the probability that token i has been randomly replaced, thereby providing the model with a token-level hint as to whether the token should be treated as condition or target.

To preserve the marginal distribution of $\mathbf{x}_t$, we use a hierarchical sampling procedure in the forward process. We first sample a global time $t \sim \mathcal{U}(0, 1)$. Then, for each position i, we sample a per-token time $\tau^i \sim q_t$ and set the token to its clean value with probability $1 - \tau^i$, or to a random vocabulary token with probability τ^i. As long as q_t has support on $[0, 1]$ and satisfies $\mathbb{E}_{q_t}[\tau^i] = t$, the marginal distribution $p(\mathbf{x}_t \mid t)$ remains unchanged.

In practice, we choose q_t to be a Beta distribution $\mathrm{Beta}(ct, c(1-t))$ with $c = 2$, and inject the time signal via AdaLN (Peebles and Xie, 2023). Note that the per-token t acts only as a hint; the LUDI loss itself still uses the global t.

Conditional sampling. Benefiting from per-token time embeddings, LUDI naturally supports position-wise non-uniform conditional sampling. At initialization, non-prompt tokens are sampled uniformly from the vocabulary, while prompt tokens are kept fixed. The corresponding per-token times can be obtained by Bayes' rule: τ^i for random tokens follows $\mathrm{Beta}(2, 1)$, whereas τ^i for prompt tokens follows $\mathrm{Beta}(1, 2)$. During denoising, non-prompt tokens are progressively updated until their token-level times τ^i approach zero.

Given the current token-level time τ_t^i, the next value $\tau_{t-\Delta t}^i$ need not follow a uniform schedule and can instead be chosen adaptively for each position. For example, a high-confidence token can be decoded in a single step by setting $\tau_{t-\Delta t}^i = 0$, after which it is treated as conditional tokens for subsequent generation. Consequently, LUDI can directly inherit samplers developed for MDLMs: each position either remains unchanged with $\tau_{t-\Delta t}^i = \tau_t^i$ or is committed by setting $\tau_{t-\Delta t}^i = 0$. Empirically, this easy-to-hard denoising schedule is critical for complex reasoning tasks. We provide pseudocode for training and sampling in the Appendix B.

AR to block-UDLM. Training a large-scale UDLM from scratch is costly. We therefore initialize from an autoregressive (AR) model to maximally preserve acquired knowledge. To continue-training the AR model as a block-UDLM, we adopt two adaptation strategies.

Label shifting and complementary noise (following Fast dLLMv2 (Wu et al., 2025a)). Since an AR model predicts token x_{i+1} from position x_i, we shift the labels left by one position. We also concatenate samples noised at t and $1 - t$ during training to reduce gradient variance.

Context-causal attention and AR loss (following NB-Diff (Tian et al., 2025)). We employ context-causal attention, where only the block currently being denoised receives bidirectional attention, while all preceding context retains causal attention. This design provides cleaner contextual signals for block generation and naturally enables inter-block KV-caching. We further regularize training with an auxiliary next-token prediction loss $\mathcal{L}_{\mathrm{AR}}$, which prevents the model from drifting excessively from its AR initialization. The overall objective is therefore $\mathcal{L}_{\mathrm{LUDI}} = \mathcal{L}_{\mathrm{LU}} + \lambda\, \mathcal{L}_{\mathrm{AR}}$, with λ typically set to 0.5 to supply supervision to the diffusion and AR components with comparable strength.

By combining these components, we present LUDI-7B, the first 7B-scale UDLM capable of tackling complex reasoning tasks such as math and code generation, marking the transition of the UDLM paradigm from a conceptual idea to practical usability. Further design details are provided in the Appendix B.2.

4 Experiments

We conduct experiments at two scales: small-scale pretraining (170M and 1B; Sec. 4.1) and large-scale AR-to-UDLM continue-training (LUDI-7B; Sec. 4.2). In the small-scale setting, where per-token time embeddings are not essential, we focus on the generation quality and training efficiency of the LU loss. In the large-scale setting, we evaluate LUDI-7B on complex reasoning tasks.

4.1 Small-Scale LUDI

Experimental setup. For the 170M experiments, we train four UDLMs under the same configuration, using LU loss, Duo loss, SDDLM loss, and SDDLM-v1 loss, respectively. All models are trained on OpenWebText (Gokaslan et al., 2019)

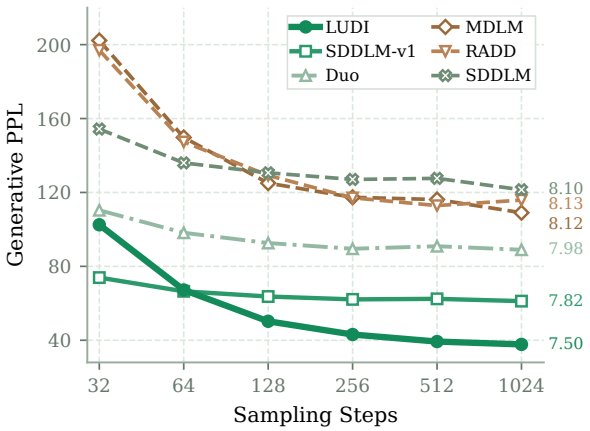


Figure 3: Generative perplexity from 32 to 1024 sampling steps. Lower PPL is better; endpoint values report entropy at 1024 steps.

with a sequence length of 1024, using the GPT-2 tokenizer (Radford et al., 2019). We use a batch size of 512 and a learning rate of 3×10^{-4}. For masked-diffusion baselines, we use pretrained RADD and MDLM models, trained on the same dataset for 400K and 1M steps, respectively. Generative perplexity is computed by GPT-2 Large on 512 samples. For the 1B experiments, we evaluate four UDLM models on downstream likelihood-based tasks. The models are trained on FineWeb (Penedo et al., 2024) for 500K steps using the Llama tokenizer (Touvron et al., 2023). The default sequence length is 2048, with 1% variable-length data. We use a batch size of 256 and a learning rate of 2×10^{-4}. Additional experimental details are provided in Appendix C.1.

Generative performance. Figure 3 compares generation quality under different numbers of sampling steps. With 1024 sampling steps, LUDI achieves a generative perplexity of 38.71 and an entropy of 7.50, outperforming both UDLM and MDLM baselines. Across most sampling budgets (≥ 64 steps), LU loss consistently yields stronger generation quality. Although LUDI yields lower entropy, our analysis of Gen.PPL and diversity (see the Appendix C.5) indicates that this reduction does not stem from mode collapse. UDLMs are often expected to enable few-step generation, but their quality can suffer from early saturation (Deschenaux et al., 2026): increasing the number of sampling steps does not necessarily improve generation. Our experiments confirm this behavior. UDLMs outperform MDLMs in low-step regimes, yet SDDLM-v1 and Duo rapidly reach a plateau. In contrast, LUDI continues to improve as the sampling budget increases. We attribute this behavior to the stronger correction ability induced by LU loss. Compared with Duo loss, which contains a smoothing term, and SDDLM-style losses, which lack a correction term, LU loss more directly encourages the model to revise potentially erroneous tokens throughout sampling.

Table 1: Generation results (Gen.PPL and Entropy) at 200K and 500K checkpoints. All models sampled with 1024 steps.

Model	200K Steps		500K Steps	
	Gen. PPL	Entropy	Gen. PPL	Entropy
Duo	99.11	8.00	88.99	7.98
SDDLM	126.97	8.10	121.47	8.10
SDDLM-v1	64.22	7.82	61.18	7.82
LUDI	**37.09**	7.53	**37.76**	7.50

Table 2: Zero-shot downstream evaluation of 1B models at 200k and 500k pretraining checkpoints. Full evaluation details are provided in Appendix C.1.

Task	LUDI	SDDLM	SDDLM-v1	Duo
200K Steps				
PIQA	**60.12**	57.40	55.33	58.11
SIQA	**36.13**	35.11	34.70	35.67
ARC-E	**42.46**	34.74	31.93	34.04
HellaSwag	**34.42**	30.85	30.05	31.82
OBQA	**26.60**	25.80	24.80	26.40
RACE	**29.32**	28.69	28.46	28.98
Avg.	**38.17**	35.43	34.21	35.83
500K Steps				
PIQA	**62.19**	61.15	60.28	61.37
SIQA	36.54	**37.26**	35.47	36.03
ARC-E	**50.35**	50.18	48.60	48.07
HellaSwag	**35.31**	34.22	33.31	35.28
OBQA	28.60	**29.80**	27.20	28.00
RACE	**29.82**	28.83	29.09	29.35
Avg.	**40.47**	40.24	38.99	39.68

Downstream task performance. Table 2 reports downstream results for 1B models on likelihood-based tasks, which probe the language modeling capability. LUDI attains the highest average score and leads on most tasks, indicating both superior performance and stability. This advantage is consistent with the behavior of LU loss. By reducing confidence on erroneous samples and increasing confidence on correct ones, LUDI enlarges the relative margin of the correct answer.

Training efficiency. Table 1 and Table 2 also compare models at 200K training steps. LUDI rapidly establishes clear advantages: the 170M model already exhibits stable generation, and the

Table 3: LUDI-7B results and ablations. Metrics cover code generation (HumanEval, MBPP, HumanEval+, MBPP+), mathematical reasoning (GSM8K), instruction following (IFEval), knowledge-intensive question answering (MMLU, GPQA), and an overall average score (Avg.). Code benchmarks are evaluated using the EvalPlus. The highest score in each column is marked in **bold**, and the second highest is underlined. AR → M/U denotes MDLM/UDLMs continue-trained from AR. In ablation, LUDI is based on Fast dLLM v2 (FD) and incorporates the LU loss, per-token time embedding, and NBDiff (ND) techniques. Full evaluation details are provided in Appendix C.2.

Model	Size	Type	HumanEval		MBPP		GSM8K	IFEval	MMLU	GPQA	Avg.
			Base	Plus	Base	Plus					
					Baselines						
LLaDA	8B	MDLM	35.4	31.7	31.5	28.6	78.6	59.9	65.5	31.8	45.4
LLaDA-1.5	8B	MDLM	52.4	–	42.8	–	83.3	58.2	66.0	**36.9**	–
LLaDA-MoE	7B	MDLM	61.6	–	70.0	–	82.4	59.3	67.2	–	–
Dream	7B	AR→M	57.9	53.7	68.3	56.1	81.0	62.5	67.0	33.0	59.9
Qwen2.5-7B	7B	AR	51.2	47.6	57.7	49.5	71.4	70.8	68.7	33.5	56.3
Fast-dLLM v2	7B	AR→D	63.4	58.5	63.0	52.3	**83.7**	61.4	66.6	31.9	60.1
LUDI-7B	7B	AR→U	**68.9**	**65.9**	**72.0**	**61.6**	75.1	**77.7**	**71.1**	33.3	**65.7**
					Ablation						
SD + FDNB	7B	AR→U	62.8	59.1	68.5	57.1	74.7	77.3	69.9	**35.3**	63.1
SDv1 + FDNB	7B	AR→U	62.2	55.5	67.7	57.9	73.5	75.7	**71.5**	33.2	62.2
LU + FD	7B	AR→U	59.1	54.9	67.7	56.3	64.7	75.1	70.8	32.4	60.1
FD	7B	AR→M	62.2	58.5	72.0	61.1	71.3	**77.7**	71.2	33.3	63.4
FDNB	7B	AR→M	68.3	62.8	**72.2**	59.8	**76.0**	77.2	70.4	31.0	64.7
LUDI-7B	7B	AR→U	**68.9**	**65.9**	72.0	**61.6**	75.1	**77.7**	71.1	33.3	**65.7**

1B LUDI outperforms all baselines across downstream tasks. We ascribe this to the cleaner learning objective provided by the LU loss. By avoiding the wasted supervision of label smoothing and directly matching $\mathbf{x}_0$, the LU loss offers a consistent signal: the model always learns to move away from erroneous tokens and toward the correct token. This improves training efficiency and data utilization, which is critical for large-scale training.

4.2 LUDI-7B

Experimental setup. We initialize our block UDLM from Qwen2.5-7B-Instruct (Team, 2024) and continue-train it on Dolci-Instruct-SFT (Olmo et al., 2025), a high-quality instruction dataset containing 2M samples. LUDI-7B is trained for 2500 steps with a batch size of 256 and a learning rate of 1×10^{-5}, following the protocol of Fast dLLMv2 (Wu et al., 2025a). The block size is fixed to 32. In each transformer layer, we insert a learnable, zero-initialized AdaLN module before the multi-head self-attention and the MLP. The AdaLN output is further multiplied by t, ensuring that for clean tokens the module remains inactive. These AdaLN modules together with the time embeddings introduce only 78.7M additional parameters ($\sim$1% of the total). During sampling, we adopt confidence-based parallel decoding.

Performance and speed. We compare LUDI with representative 7 to 8B autoregressive and diffusion models; the results are summarized in Table 3. LUDI-7B achieves an average score of 65.7, outperforming both the baselines and its AR initialization. It performs strongly on likelihood-based benchmarks (MMLU and GPQA) while also delivering competitive results on generation tasks. Notably, on code generation it attains 65.9 on HumanEval+ and 61.6 on MBPP+. With parallel decoding, LUDI-7B offers an attractive performance and efficiency trade-off. By adjusting the confidence threshold, LUDI-7B maintains stable accuracy on GSM8K (Figure 4) for thresholds above 0.7, while yielding a 2.6 to 3.0× speedup over AR decoding (1 token per step). These results confirm that UDLMs can handle complex generation tasks and possess substantial acceleration potential.

Ablation study. To systematically examine the contribution of each design choice, we conduct controlled experiments on Dolci-Instruct under identical conditions (Table 3). Our LUDI-7B builds upon Fast dLLM (FD) and integrates the LU loss, per-token time embeddings, and the tricks from NB-Diff (NB). (1) Without per-token time embeddings, the model suffers from severe condition-target confusion caused by position-uniform sampling, quickly collapsing to a trivial solution that repeats a single token. We therefore exclude this

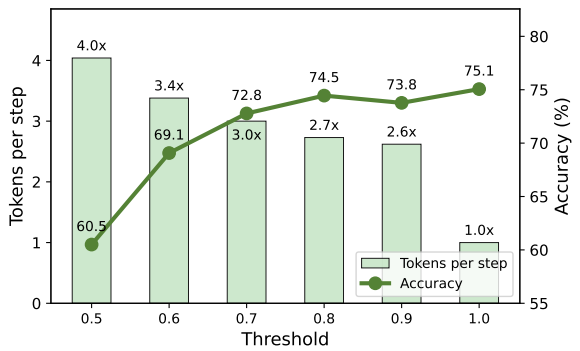


Figure 4: GSM8K performance and decoding speedup of LUDI-7B under different decoding thresholds.

variant from comparison. (2) Replacing the LU loss with the SDv1 loss, which also mitigates over-uniform, still yields reasonable generations but with a clear performance drop. This suggests that complex generation requires explicit error correction: the model must be steered away from the current erroneous token. (3) The AR loss and causal context attention from NB-Diff prove important, indicating that block decoding in UDLMs benefits from higher-quality context. (4) Compared to MDLM baselines, LUDI-7B surpasses FDNB trained under the same budget, demonstrating that UDLMs can profit from a more challenging training objective and may offer greater scaling potential under data-constrained regimes (von Rütte et al., 2025b).

5 Related Works

Masked diffusion language models. MDLMs have become a competitive generative paradigm against AR models. Through large-scale pretraining (Nie et al., 2025b; Bie et al., 2025), AR-to-diffusion adaptation (Wu et al., 2025b; Tian et al., 2025; Liu et al., 2025), and reinforcement-learning post-training (Zhao et al., 2025; Zhong et al., 2026; Ni et al., 2026), MDLMs have achieved performance comparable to frontier LLMs on complex reasoning tasks while offering substantial inference-speed advantages. To better handle challenging problems, MDLM sampling has also moved beyond position-uniform unmasking toward strategic policies based on confidence (Wu et al., 2025b) and entropy (Ben-Hamu et al., 2025). However, the binary mask/non-mask state space makes it difficult for MDLMs to correct already generated tokens. Moreover, the uninformative mask state naturally constrains parallel decoding (Chen et al., 2026). Consequently, changing the masked generation paradigm has become an important direction for raising the ceiling of diffusion language models (Ding et al., 2026; von Rütte et al., 2025a).

Uniform diffusion language models. UDLMs provide a complementary formulation where corrupted tokens are driven toward a uniform vocabulary distribution rather than an absorbing mask. This gives UDLMs potential advantages in few-step generation (Sahoo et al., 2025; Deschenaux et al., 2026), controllability (Schiff et al., 2025), self-correction (Schiff et al., 2026), and scaling behavior under data- or compute-limited regimes (von Rütte et al., 2025b). Nevertheless, existing analyses remain largely confined to small-scale settings or scaling-law studies, and UDLMs have not yet been scaled for complex reasoning tasks. Some works use uniform diffusion only as an auxiliary mechanism to improve the self-correction ability of MDLMs (Schiff et al., 2026; von Rütte et al., 2025a), or as a bridge toward continuous diffusion language modeling (Sahoo et al., 2025; Lee et al., 2026). Efforts dedicated to improving UDLMs mainly focus on mitigating the sampling plateau with predictor-corrector samplers (Deschenaux et al., 2026) or simplifying the training objective (Zhu et al., 2025c). However, these works either do not address, or only empirically touch upon, the over-uniform effect induced by UDLM training objectives; meanwhile, their samplers are not specifically designed for complex reasoning and largely remain position-uniform. These limitations create an inherent obstacle to scaling UDLMs, motivating our study.

6 Conclusion

This work presents LUDI as a step toward making UDLMs practical at scale. We show that the main barrier to scaling UDLMs lies not in the uniform noise prior itself, but in the excessive uniformity introduced by existing training objectives and sampling procedures. By introducing a less uniform loss and token-level time awareness, LUDI achieves cleaner supervision and a more efficient denoising process. Through LUDI-7B, we further demonstrate that UDLMs can be extended to complex reasoning tasks, narrowing the gap with AR and MDLM paradigms. These results suggest that UDLMs remain a promising yet underexplored direction. Further leveraging their distinctive properties, such as self-correction and few-step distillation, may be key to unlocking their full potential.

Limitations

This work presents LUDI as an initial effort to scale uniform diffusion language models to 7B parameters and to apply them to complex reasoning tasks. Nevertheless, several limitations remain. (1) We have not yet fully activated the self-correction potential of UDLMs. The errors that arise during self-correction stem from the model's own imperfect generations, which are qualitatively different from the random noise introduced during training. Consequently, instilling this ability may require dedicated post-training stages that explicitly teach the model to detect and amend its own mistakes. (2) Due to computational resource constraints, we have not been able to pre-train a large-scale UDLM from scratch. Our approach of continue-training from an AR checkpoint efficiently preserves the original model's knowledge, yet it may also retain certain inductive biases of the autoregressive generation paradigm. We view this work as providing a foundation and confidence for the UDLM community, and we hope it spurs future research into UDLMs pre-trained at scale from scratch, which may more fully exploit the unique advantages of the uniform diffusion framework.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. 2025. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Preprint*, arXiv:2505.24857.

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*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Peter Clark, 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 preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Justin Deschenaux, Caglar Gulcehre, and Subham Sekhar Sahoo. 2026. The diffusion duality, chapter II: $\backslash psi$-samplers and efficient curriculum. In *The Fourteenth International Conference on Learning Representations*.

Fangyu Ding, Ding Ding, Sijin Chen, Kaibo Wang, Peng Xu, Zijin Feng, Haoli Bai, Kai Han, Youliang Yan, Binhang Yuan, and Jiacheng Sun. 2026. Beyond masks: Efficient, flexible diffusion language models via deletion-insertion processes. *Preprint*, arXiv:2603.23507.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

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

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

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. 2026. The flexibility trap: Why arbitrary order limits reasoning potential in diffusion language models. *Preprint*, arXiv:2601.15165.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, volume 2025, pages 82974–82997.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large language diffusion models. *Preprint*, arXiv:2502.09992.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. Olmo 3. *Preprint*, arXiv:2512.13961.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, volume 2025, pages 64972–65009.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. 2025. The diffusion duality. In *International Conference on Machine Learning*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Yair Schiff, Omer Belhasin, Roy Uziel, Guanghan Wang, Marianne Arriola, Gilad Turok, Michael Elad, and Volodymyr Kuleshov. 2026. Learn from your mistakes: Self-correcting masked diffusion models. *Preprint*, arXiv:2602.11590.

Yair Schiff, Subham Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. 2025. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations*, volume 2025, pages 43776–43821.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, and 1 others. 2025. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. 2025a. Generalized interpolating discrete diffusion. *Preprint*, arXiv:2503.04482.

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. 2025b. Scaling behavior of discrete diffusion language models. *Preprint*, arXiv:2512.10858.

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. *Preprint*, arXiv:2505.22618.

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*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

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

Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. 2026. Stabilizing reinforcement learning for diffusion language models. *Preprint*, arXiv:2603.06743.

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

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025a. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, and 1 others. 2025b. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*.

Huaisheng Zhu, Zhengyu Chen, Shijie Zhou, Zhihui Xie, Yige Yuan, Shiqi Chen, Zhimeng Guo, Siyuan Xu, Hangfan Zhang, Vasant Honavar, and 1 others. 2025c. Simple denoising diffusion language models. *arXiv preprint arXiv:2510.22926*.

A Proofs

We collect the derivations omitted from the main text. Unless otherwise specified, all statements are for a single token position; sentence-level losses are obtained by summing the token-level losses over positions. Let $\mathcal{V} = \{1, \dots, V\}$ and let $\boldsymbol{e}_i$ denote the one-hot vector of token i. We write the clean token as $\boldsymbol{x}_0 = \boldsymbol{e}_y$ and the current noisy token as $\boldsymbol{x}_t = \boldsymbol{e}_m$. The schedule α_t is continuously differentiable, non-increasing, and satisfies $\alpha_0 = 1$ and $\alpha_1 = 0$. For any predicted clean-token distribution $\boldsymbol{p} \in \Delta^{V-1}$, define the scaled noisy marginal

$$\bar{p}_i(t) = V\alpha_t p_i + 1 - \alpha_t, \qquad \bar{\boldsymbol{p}}(t) = V\alpha_t \boldsymbol{p} + (1-\alpha_t)\mathbf{1}.$$

Thus $\bar{p}_i(t)/V = \alpha_t p_i + (1-\alpha_t)/V$. In particular, $\bar{x}_{0,i}$ denotes the same quantity with $\boldsymbol{p} = \boldsymbol{x}_0$, and $\bar{x}_{\theta,i}$ denotes it with $\boldsymbol{p} = \boldsymbol{x}_\theta(\boldsymbol{x}_t, t)$.

A.1 Uniform-State Diffusion and Mask Diffusion

A.1.1 Forward processes

We use the row-vector convention for Markov chains. For a continuous-time chain with generator Q_t, the infinitesimal transition is

$$p(\boldsymbol{x}_{t+\Delta t} = \boldsymbol{e}_j \mid \boldsymbol{x}_t = \boldsymbol{e}_i) = \mathbb{I}_{i=j} + Q_t(i,j)\Delta t + o(\Delta t), \tag{8}$$

and the conditional distribution $p_t(\cdot \mid y)$ satisfies the Kolmogorov forward equation

$$\frac{d}{dt} p_t(\cdot \mid y) = p_t(\cdot \mid y) Q_t. \tag{9}$$

Uniform-state diffusion. The uniform-state generator is

$$Q_t^{\mathrm{U}}(i,j) = -\frac{\alpha_t'}{\alpha_t}\left(\frac{1}{V} - \mathbb{I}_{i=j}\right). \tag{10}$$

Since $\alpha_t' \le 0$, the off-diagonal entries are non-negative: $Q_t^{\mathrm{U}}(i,j) = -\alpha_t'/(V\alpha_t)$ for $i \ne j$. The diagonal entry is $Q_t^{\mathrm{U}}(i,i) = \alpha_t'(1-1/V)/\alpha_t \le 0$, and each row sums to zero because

$$\sum_{j=1}^{V}\left(\frac{1}{V} - \mathbb{I}_{i=j}\right) = 0.$$

Starting from $\mathbf{x}_0 = \mathbf{e}_y$, the marginal distribution is

$$p_t^{\mathrm{U}}(j \mid y) = \alpha_t \mathbb{I}_{j=y} + (1-\alpha_t)\frac{1}{V}. \tag{11}$$

Indeed,

$$\frac{d}{dt} p_t^{\mathrm{U}}(j \mid y) = \alpha_t'\left(\mathbb{I}_{j=y} - \frac{1}{V}\right),$$

while the Kolmogorov forward equation gives

$$\left[p_t^{\mathrm{U}}(\cdot \mid y) Q_t^{\mathrm{U}}\right]_j = -\frac{\alpha_t'}{\alpha_t}\left(\frac{1}{V} - p_t^{\mathrm{U}}(j \mid y)\right) = \alpha_t'\left(\mathbb{I}_{j=y} - \frac{1}{V}\right).$$

The initial condition also holds, so

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathrm{Cat}\left(\cdot;\, \alpha_t \mathbf{x}_0 + (1-\alpha_t)\frac{\mathbf{1}}{V}\right). \tag{12}$$

Mask diffusion. For mask diffusion, the state space is $\mathcal{V} \cup \{\mathtt{m}\}$, where $\mathtt{m}$ is an absorbing mask token. Its generator is

$$\begin{aligned} &Q_t^{\mathrm{M}}(i,\mathtt{m}) = -\frac{\alpha_t'}{\alpha_t}, \quad Q_t^{\mathrm{M}}(i,i) = \frac{\alpha_t'}{\alpha_t}, \qquad i \in \mathcal{V}, \\ &Q_t^{\mathrm{M}}(i,j) = 0, \qquad Q_t^{\mathrm{M}}(\mathtt{m},j) = 0, \end{aligned} \tag{13}$$

where the first zero applies to $i, j \in \mathcal{V}$ with $j \ne i$. Again, off-diagonal entries are non-negative and rows sum to zero.

Starting from $x_0 = y$, the candidate marginal is

$$p_t^{\mathrm{M}}(y \mid y) = \alpha_t, \qquad p_t^{\mathrm{M}}(\mathtt{m} \mid y) = 1 - \alpha_t, \tag{14}$$

with all other clean tokens having probability zero. Its derivative is α_t' at y and $-\alpha_t'$ at $\mathtt{m}$. The forward equation gives

$$\left[p_t^{\mathrm{M}}(\cdot \mid y) Q_t^{\mathrm{M}}\right]_y = \alpha_t \frac{\alpha_t'}{\alpha_t} = \alpha_t', \qquad \left[p_t^{\mathrm{M}}(\cdot \mid y) Q_t^{\mathrm{M}}\right]_{\mathtt{m}} = \alpha_t \left(-\frac{\alpha_t'}{\alpha_t}\right) = -\alpha_t'.$$

Thus Eq. (14) satisfies the Kolmogorov forward equation and the correct initial condition.

A.1.2 Exact backward posteriors

Let $0 \le s < t \le 1$ and define $\alpha_{t|s} = \alpha_t / \alpha_s$. Because the same construction applies between times s and t, the uniform-state transition from s to t is

$$p(\boldsymbol{x}_t = \boldsymbol{e}_m \mid \boldsymbol{x}_s = \boldsymbol{e}_k) = \alpha_{t|s} \mathbb{I}_{m=k} + \frac{1 - \alpha_{t|s}}{V}. \tag{15}$$

Bayes' rule then gives, for $\boldsymbol{x}_0 = \boldsymbol{e}_y$,

$$p(\boldsymbol{x}_s = \boldsymbol{e}_k \mid \boldsymbol{x}_t = \boldsymbol{e}_m, \boldsymbol{x}_0 = \boldsymbol{e}_y) = \frac{\left(\alpha_{t|s}\mathbb{I}_{m=k} + \frac{1-\alpha_{t|s}}{V}\right)\left(\alpha_s \mathbb{I}_{k=y} + \frac{1-\alpha_s}{V}\right)}{\alpha_t \mathbb{I}_{m=y} + \frac{1-\alpha_t}{V}}. \tag{16}$$

To obtain the vector form, multiply the numerator and denominator by V and use $\alpha_{t|s}\alpha_s = \alpha_t$. The coordinate k of the numerator becomes

$$V\alpha_t \mathbb{I}_{m=k}\mathbb{I}_{k=y} + (\alpha_{t|s} - \alpha_t)\mathbb{I}_{m=k} + (\alpha_s - \alpha_t)\mathbb{I}_{k=y} + \frac{(1-\alpha_{t|s})(1-\alpha_s)}{V}.$$

Therefore

$$\boldsymbol{\pi}_{t\to s} = \frac{1}{\langle \bar{\boldsymbol{x}}_t(t), \boldsymbol{x}_0 \rangle}\left[V\alpha_t\, \boldsymbol{x}_t \odot \boldsymbol{x}_0 + (\alpha_{t|s} - \alpha_t)\boldsymbol{x}_t + (\alpha_s - \alpha_t)\boldsymbol{x}_0 + (1-\alpha_{t|s})(1-\alpha_s)\frac{\mathbf{1}}{V}\right], \tag{17}$$

where $\bar{\boldsymbol{x}}_t(t) = V\alpha_t \boldsymbol{x}_t + (1-\alpha_t)\mathbf{1}$ and $\langle \bar{\boldsymbol{x}}_t(t), \boldsymbol{x}_0 \rangle = V\alpha_t \mathbb{I}_{m=y} + 1 - \alpha_t$.

For mask diffusion, if $x_t = y$, the observation is possible only if the token has never been masked; hence $p(x_s = y \mid x_t = y, x_0 = y) = 1$. If $x_t = \mathtt{m}$, Bayes' rule and Eq. (14) give

$$\begin{aligned} p(x_s = y \mid x_t = \mathtt{m}, x_0 = y) &= \frac{p(x_t = \mathtt{m} \mid x_s = y)p(x_s = y \mid x_0 = y)}{p(x_t = \mathtt{m} \mid x_0 = y)} \\ &= \frac{(1-\alpha_{t|s})\alpha_s}{1-\alpha_t} = \frac{\alpha_s - \alpha_t}{1-\alpha_t}, \\ p(x_s = \mathtt{m} \mid x_t = \mathtt{m}, x_0 = y) &= \frac{1-\alpha_s}{1-\alpha_t}. \end{aligned} \tag{18}$$

These two probabilities sum to one, so the posterior is complete.

A.1.3 Reverse KL, SEDD, and Duo

The local term in the diffusion NELBO is the KL divergence between the exact one-step reverse posterior and the learned reverse transition,

$$\mathrm{KL}\big(p(\boldsymbol{x}_s \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_s \mid \boldsymbol{x}_t)\big), \qquad s < t. \tag{19}$$

Since

$$\mathrm{KL}(\pi \| q) = -\mathbb{E}_{z\sim\pi} \log q(z) - H(\pi), \tag{20}$$

only the posterior cross-entropy depends on the model. In continuous time, the NELBO is obtained by integrating the infinitesimal version of these local KL terms over t; SEDD derives the same infinitesimal objective as a discrete score entropy loss, and Duo uses an x_0-parameterization of the same reverse kernel (Lou et al., 2023; Sahoo et al., 2025).

For the special case of a Dirac target, Eq. (20) reduces to

$$\mathrm{KL}(\delta_{\boldsymbol{e}_y} \| q) = -\log q(\boldsymbol{e}_y). \tag{21}$$

For a non-degenerate posterior π, however, the pointwise quantity $-\log q(\boldsymbol{e}_y)$ is not by itself a general upper bound on $\mathrm{KL}(\pi\|q)$; the correct model-dependent term is the full posterior cross-entropy $-\mathbb{E}_{z\sim\pi}\log q(z)$.

SEDD parameterizes the reverse model through the ratio between a candidate token j and the current token m:

$$s_\theta(j,m,t) \approx \frac{p_t(\boldsymbol{e}_j \mid \boldsymbol{x}_0)}{p_t(\boldsymbol{e}_m \mid \boldsymbol{x}_0)}.$$

Conditioned on $\boldsymbol{x}_0$, the true ratio is

$$s^\star(j,m,t) = \frac{p_t(\boldsymbol{e}_j \mid \boldsymbol{x}_0)}{p_t(\boldsymbol{e}_m \mid \boldsymbol{x}_0)} = \frac{\bar{x}_{0,j}}{\bar{x}_{0,m}}. \tag{22}$$

The diffusion-weighted denoising score entropy at one token position is, up to terms independent of θ,

$$\mathcal{L}^\ell_{\mathrm{DSE}} \doteq \mathbb{E}_{t,\boldsymbol{x}_t} \sum_{j\neq m} Q_t^{\mathrm{U}}(m,j)\left[s_\theta(j,m,t) - s^\star(j,m,t)\log s_\theta(j,m,t)\right]. \tag{23}$$

Here $\doteq$ denotes equality up to additive constants independent of θ. For the uniform-state generator, $Q_t^{\mathrm{U}}(m,j) = -\alpha_t'/(V\alpha_t)$ for all $j \neq m$. Under the Duo x_0-parameterization,

$$s_\theta(j,m,t) = \frac{\bar{x}_{\theta,j}}{\bar{x}_{\theta,m}}, \qquad \bar{x}_{\theta,j} = V\alpha_t x_{\theta,j} + 1 - \alpha_t. \tag{24}$$

Adding the $j = m$ term to Eq. (23) changes the objective only by the constant $Q_t^{\mathrm{U}}(m,m)$-free normalization of the score-entropy summand, because $s_\theta(m,m,t) = s^\star(m,m,t) = 1$. Using $\sum_j \bar{x}_{\theta,j} = V$, we obtain

$$\begin{aligned}
\mathcal{L}^\ell_{\mathrm{DSE}} &\doteq \mathbb{E}_{t,\boldsymbol{x}_t}\frac{-\alpha_t'}{V\alpha_t}\left[\frac{V}{\bar{x}_{\theta,m}} - \sum_{j=1}^{V}\frac{\bar{x}_{0,j}}{\bar{x}_{0,m}}\log\frac{\bar{x}_{\theta,j}}{\bar{x}_{\theta,m}}\right] \\
&\doteq \mathbb{E}_{t,\boldsymbol{x}_t}\frac{-\alpha_t'}{V\alpha_t}\left[\frac{V}{\bar{x}_{\theta,m}} - \frac{V}{\bar{x}_{0,m}} + \sum_{j=1}^{V}\frac{\bar{x}_{0,j}}{\bar{x}_{0,m}}\log\frac{\bar{x}_{\theta,m}\bar{x}_{0,j}}{\bar{x}_{\theta,j}\bar{x}_{0,m}}\right].
\end{aligned} \tag{25}$$

The second line includes the θ-independent term $-V/\bar{x}_{0,m}$ and the θ-independent clean-ratio logarithms, so that the bracket is zero when $\boldsymbol{x}_\theta = \boldsymbol{x}_0$. This is the same reverse-KL/score-entropy objective used by Duo after the rate ratio is expressed through a predicted clean-token distribution rather than output directly as a ratio.

The mask-diffusion counterpart explains why the masked ELBO becomes a simple cross-entropy loss (Sahoo et al., 2024). When $x_t = \mathtt{m}$, define

$$a_{t,s} = \frac{\alpha_s - \alpha_t}{1-\alpha_t}, \qquad b_{t,s} = \frac{1-\alpha_s}{1-\alpha_t}.$$

The exact posterior assigns probability $a_{t,s}$ to the clean token y and $b_{t,s}$ to $\mathtt{m}$. The usual x_0-parameterized model assigns probability $a_{t,s}x_{\theta,j}$ to each clean token j and $b_{t,s}$ to $\mathtt{m}$. Hence

$$\mathrm{KL}\big(p^\star_{s|t}\|q^\theta_{s|t}\big) = a_{t,s}\log\frac{a_{t,s}}{a_{t,s}x_{\theta,y}} + b_{t,s}\log\frac{b_{t,s}}{b_{t,s}} = -a_{t,s}\log x_{\theta,y}. \tag{26}$$

All dependence on s and t is contained in the analytic scalar $a_{t,s}$; the learned part is the ordinary clean-token cross-entropy. This decoupling is the reason that the absorbing-state/masked objective is much simpler than the uniform-state objective.

A.2 ELBO Loss and LU Loss

A.2.1 Proof of Proposition 1

We first handle corrupted positions, i.e., $m \neq y$. From Eq. (25), and ignoring additive constants independent of θ, the token-level ELBO/Duo loss is

$$\mathcal{L}^\ell_{\mathrm{ELBO}} \doteq -\alpha_t'\mathbb{E}_{t,\boldsymbol{x}_t}\left[\frac{1}{\alpha_t\bar{x}_{\theta,m}} + \frac{V\alpha_t + 1 - \alpha_t}{V\alpha_t(1-\alpha_t)}\log\frac{\bar{x}_{\theta,m}}{\bar{x}_{\theta,y}} + \frac{1}{V\alpha_t}\sum_{j\neq y}\log\frac{\bar{x}_{\theta,m}}{\bar{x}_{\theta,j}}\right]. \tag{27}$$

Here we used $\bar{x}_{0,m} = 1 - \alpha_t$, $\bar{x}_{0,y} = V\alpha_t + 1 - \alpha_t$, and $\bar{x}_{0,j} = 1 - \alpha_t$ for $j \neq y$. Rearranging the logarithmic terms gives

$$\begin{aligned}
&\frac{V\alpha_t + 1 - \alpha_t}{V\alpha_t(1-\alpha_t)} \log \frac{\bar{x}_{\theta,m}}{\bar{x}_{\theta,y}} + \frac{1}{V\alpha_t} \sum_{j \neq y} \log \frac{\bar{x}_{\theta,m}}{\bar{x}_{\theta,j}} \\
&= \frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} + \frac{\log \bar{x}_{\theta,m}}{\alpha_t} - \frac{1}{V\alpha_t} \sum_{i=1}^{V} \log \bar{x}_{\theta,i}. \qquad (28)
\end{aligned}$$

Therefore the exact θ-dependent form is

$$\mathcal{L}^{\ell}_{\text{ELBO}} \doteq -\alpha_t' \mathbb{E}_{t,\boldsymbol{x}_t} \left[\frac{1}{\alpha_t \bar{x}_{\theta,m}} + \frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} + \frac{\log \bar{x}_{\theta,m}}{\alpha_t} - \frac{1}{V\alpha_t} \sum_{i=1}^{V} \log \bar{x}_{\theta,i} \right]. \qquad (29)$$

To obtain the CE-like expression in the main text, apply the expansion

$$\frac{1}{z} = 1 - \log z + R(z), \qquad R(z) = \frac{1}{z} - 1 + \log z. \qquad (30)$$

The remainder satisfies $R(1) = R'(1) = 0$ and is second order around $z = 1$. Substituting Eq. (30) with $z = \bar{x}_{\theta,m}$ into Eq. (29), the term $-\log \bar{x}_{\theta,m}/\alpha_t$ cancels the $+\log \bar{x}_{\theta,m}/\alpha_t$ term. After dropping the additive constant $1/\alpha_t$, we get

$$\mathcal{L}^{\ell}_{\text{ELBO}} \doteq -\alpha_t' \mathbb{E}_{t,\boldsymbol{x}_t} \left[\frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} - \frac{1}{V\alpha_t} \sum_{i=1}^{V} \log \bar{x}_{\theta,i} + \frac{R(\bar{x}_{\theta,m})}{\alpha_t} \right]. \qquad (31)$$

The proposition in the main text is obtained by neglecting the second-order remainder $R(\bar{x}_{\theta,m})/\alpha_t$ in the non-degenerate time region where α_t and $1 - \alpha_t$ are not too close to zero. Equivalently,

$$\mathcal{L}^{\ell}_{\text{ELBO}} \simeq -\alpha_t' \mathbb{E}_{t,\boldsymbol{x}_t} \left[\frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} - \frac{1}{V\alpha_t} \sum_{i=1}^{V} \log \bar{x}_{\theta,i} \right]. \qquad (32)$$

Define the scaled label-smoothed cross-entropy

$$\text{CE}^{1-\alpha_t}_{\text{LS}}(\bar{\boldsymbol{x}}_\theta, y) = -\left[\alpha_t \log \bar{x}_{\theta,y} + \frac{1-\alpha_t}{V} \sum_{i=1}^{V} \log \bar{x}_{\theta,i} \right]. \qquad (33)$$

This is the cross-entropy with target $\alpha_t \boldsymbol{x}_0 + (1-\alpha_t)\mathbf{1}/V$, evaluated on the scaled positive vector $\bar{\boldsymbol{x}}_\theta$; using the normalized vector $\bar{\boldsymbol{x}}_\theta/V$ would only add a constant independent of θ. A direct rearrangement yields

$$\frac{\text{CE}^{1-\alpha_t}_{\text{LS}}(\bar{\boldsymbol{x}}_\theta, y)}{\alpha_t(1-\alpha_t)} + \frac{\log \bar{x}_{\theta,m}}{1-\alpha_t} = \frac{\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}}{1-\alpha_t} - \frac{1}{V\alpha_t} \sum_{i=1}^{V} \log \bar{x}_{\theta,i}. \qquad (34)$$

This proves the two forms of Proposition 1, with the approximation and its remainder made explicit.

Now consider positions where $m = y$. From Eq. (25), the θ-dependent terms are

$$\mathcal{L}^{\ell}_{\text{ELBO},m=y} \doteq -\alpha_t' \mathbb{E}_{t,\boldsymbol{x}_t} \left[\frac{1}{\alpha_t \bar{x}_{\theta,y}} + \frac{1-\alpha_t}{V\alpha_t(V\alpha_t + 1 - \alpha_t)} \sum_{j \neq y} \log \frac{\bar{x}_{\theta,y}}{\bar{x}_{\theta,j}} \right]. \qquad (35)$$

The coefficient $(1-\alpha_t)/(V\alpha_t + 1 - \alpha_t)$ is the true ratio $\bar{x}_{0,j}/\bar{x}_{0,y}$ for every $j \neq y$. Thus, away from the terminal region where $\alpha_t \approx 0$, and for a reasonably learned denoiser that puts large mass on the already-clean token, these terms are suppressed by the large vocabulary size. More precisely, if α_t is bounded away from zero and $\bar{x}_{\theta,y} = \Theta(V\alpha_t)$ while the average logarithmic ratio is not pathological, Eq. (35) is negligible, typically of order $\tilde{O}(1/V)$. This is the qualified sense in which already-clean positions contribute negligibly.

A.2.2 Proof of Theorem 1

Let $s = t - \Delta t$ and consider a corrupted position $m \neq y$. Under the Duo parameterization, the learned reverse transition is obtained by replacing the unknown clean token distribution in the exact posterior with

$\mathbf{x}_\theta$. Therefore, the probability of jumping from the current token m to the clean token y is

$$p_\theta(\mathbf{x}_s = \mathbf{e}_y \mid \mathbf{x}_t = \mathbf{e}_m) = \frac{(\alpha_s - \alpha_t)x_{\theta,y} + (1 - \alpha_t/\alpha_s)(1 - \alpha_s)/V}{\bar{x}_{\theta,m}}. \tag{36}$$

As $\Delta t \to 0$, we have

$$1 - \frac{\alpha_t}{\alpha_s} = \frac{\alpha_s - \alpha_t}{\alpha_t} + o(\Delta t), \qquad 1 - \alpha_s = 1 - \alpha_t + o(1).$$

Thus

$$p_\theta(\mathbf{x}_s = \mathbf{e}_y \mid \mathbf{x}_t = \mathbf{e}_m) = \frac{\alpha_s - \alpha_t}{V\alpha_t}\frac{\bar{x}_{\theta,y}}{\bar{x}_{\theta,m}} + o(\Delta t). \tag{37}$$

The Dirac-target KL is therefore

$$\begin{aligned}\mathrm{KL}(\delta(\mathbf{x}_0)\|p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)) &= -\log p_\theta(\mathbf{x}_s = \mathbf{e}_y \mid \mathbf{x}_t = \mathbf{e}_m) \\ &= \log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y} - \log \frac{\alpha_s - \alpha_t}{V\alpha_t} + o(1).\end{aligned} \tag{38}$$

The logarithmic normalizer may diverge as $\Delta t \to 0$, but it is independent of θ. Hence the raw infinitesimal KL need not have a finite limit; the finite θ-dependent part is exactly

$$\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}.$$

Multiplying this finite part by the positive time weight $-\alpha_t'/(1 - \alpha_t)$ and averaging over $(t, \mathbf{x}_t)$ gives $\mathcal{L}_{\mathrm{LU}}^{\ell}$. This proves that LU minimizes the finite θ-dependent part of the infinitesimal Dirac-target KL, or equivalently minimizes the Dirac-target KL up to a θ-independent normalizer and an $o(1)$ term.

A.3 Per-Token Time Embeddings

This section spells out the probabilistic meaning of the per-token time variables used by LUDI. For simplicity, the main text uses the linear schedule $\alpha_t = 1 - t$. Given a global time t, we sample a token-level time $\tau^i \sim q_t$ for each position i. Conditional on τ^i, we draw a replacement indicator

$$R^i \mid \tau^i \sim \mathrm{Bernoulli}(\tau^i).$$

If $R^i = 0$, the token is kept clean; if $R^i = 1$, the token is replaced by a uniform vocabulary sample. Therefore, for $x_0^i = y$ and any $v \in \mathcal{V}$,

$$p(x_t^i = v \mid x_0^i = y, t) = \mathbb{E}_{\tau^i \sim q_t}\left[(1 - \tau^i)\mathbb{I}_{v=y} + \tau^i \frac{1}{V}\right] = \left(1 - \mathbb{E}_{q_t}\tau^i\right)\mathbb{I}_{v=y} + \frac{\mathbb{E}_{q_t}\tau^i}{V}. \tag{39}$$

Thus, if $\mathbb{E}_{q_t}[\tau^i] = t$, Eq. (39) becomes

$$p(x_t^i = v \mid x_0^i = y, t) = (1 - t)\mathbb{I}_{v=y} + \frac{t}{V} = \alpha_t \mathbb{I}_{v=y} + \frac{1 - \alpha_t}{V}, \tag{40}$$

which is exactly the uniform-state marginal. For a non-linear schedule, the same argument applies after replacing the condition $\mathbb{E}_{q_t}[\tau^i] = t$ by $\mathbb{E}_{q_t}[\tau^i] = 1 - \alpha_t$.

In our implementation, we use

$$q_t(\tau) = \mathrm{Beta}(ct, c(1 - t)), \qquad c = 2, \tag{41}$$

whose mean is t and whose concentration is controlled by c. Very close to $t = 0$ or $t = 1$, the Beta parameters are clipped to a small positive value for numerical stability. The token-level time is therefore a hint about the replacement probability: by construction, $p(R^i = 1 \mid \tau^i) = \tau^i$. Note that observing the token value $x_t^i = x_0^i$ is not identical to observing $R^i = 0$, because a uniform replacement can coincidentally redraw the clean token with probability $1/V$.

For conditional generation, prompt tokens are actively fixed and non-prompt tokens are actively initialized from the uniform vocabulary. A simple Bayesian initialization for their token-level times is obtained by placing the prior $\tau \sim \mathrm{Beta}(1, 1)$ at the start of sampling and using $R \mid \tau \sim \mathrm{Bernoulli}(\tau)$.

Then

$$p(\tau \mid R=1) = \frac{p(R=1 \mid \tau)p(\tau)}{p(R=1)} = \frac{\tau}{\int_0^1 u\,du} = 2\tau, \quad \tau \in [0,1],$$

$$p(\tau \mid R=0) = \frac{p(R=0 \mid \tau)p(\tau)}{p(R=0)} = \frac{1-\tau}{\int_0^1 (1-u)\,du} = 2(1-\tau). \tag{42}$$

Thus a known random token receives $\tau \sim \mathrm{Beta}(2,1)$, while a known fixed prompt token receives $\tau \sim \mathrm{Beta}(1,2)$. This initialization is not required to equal the marginal distribution of τ induced by the training mixture $t \sim \mathrm{Unif}(0,1)$ and $\tau \mid t \sim \mathrm{Beta}(ct, c(1-t))$; it is a boundary-aware sampling prior that encodes the facts "random tokens should start noisy" and "fixed prompt tokens should start clean". If one wanted the exact Bayes posterior under the training mixture, the prior density would be $\int_0^1 q_t(\tau)\,dt$ instead of the uniform density, giving $p(\tau \mid R=1) \propto \tau \int_0^1 q_t(\tau)dt$ and $p(\tau \mid R=0) \propto (1-\tau) \int_0^1 q_t(\tau)dt$.

During denoising, the model receives $(\boldsymbol{x}_t, \boldsymbol{\tau}_t)$ and predicts $\boldsymbol{x}_\theta$. Positions that are committed by the sampler are assigned $\tau^i = 0$ and subsequently treated as clean conditioning tokens; uncommitted positions keep positive token-level times and continue to be denoised. This is why per-token times make confidence-based easy-to-hard sampling possible in a uniform-state model.

Algorithm 1: Training of LUDI

Input : Clean data $\mathbf{x}_0$, noise schedule α_t, model $\mathbf{x}_\theta$
Output : Trained model parameters θ

for *each training step* **do**
  Sample global time $t \sim \mathcal{U}(0, 1)$;
  for *each token position* i **do**
    // sample token-wise corruption time
    $\tau^i \sim \mathrm{Beta}(2t,\ 2(1-t))$;
    Corrupt $\mathbf{x}_0^i$: with probability τ^i, replace by a uniform token $\rightarrow \mathbf{x}_t^i$;
  end
  // model predicts clean tokens from noisy inputs and per-token times
  $\hat{\mathbf{x}}_0 = \mathtt{Model}(\mathbf{x}_t,\ \boldsymbol{\tau})$;
  // compute LU loss on corrupted positions (Eq. 6)
  $\mathcal{L} = \frac{-\alpha_t'}{1-\alpha_t} \sum_{i:\mathbf{x}_t^i \neq \mathbf{x}_0^i} \left(\log \bar{x}_{\theta,m}^i - \log \bar{x}_{\theta,y}^i\right)$;
  Update θ using $\nabla_\theta \mathcal{L}$;
end

Algorithm 2: Sampling of LUDI

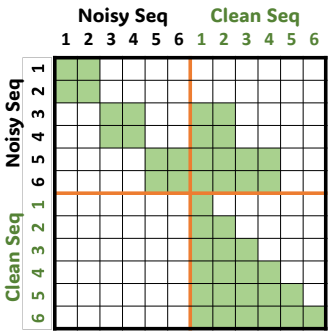
Input : Prompt tokens, trained model $\mathbf{x}_\theta$, scheduler function h, confidence threshold γ
Output : Generated clean sequence $\mathbf{x}$

Initialize $\mathbf{x}$: prompt tokens kept unchanged, all other tokens sampled uniformly from $\mathcal{V}$;
Initialize $\boldsymbol{\tau}$: for non-prompt tokens $\tau^i \sim \mathrm{Beta}(2, 1)$, for prompt tokens $\tau^i \sim \mathrm{Beta}(1, 2)$;
while $\max(\boldsymbol{\tau}) > 0$ **do**
  $\hat{\mathbf{x}}_0 = \mathtt{Model}(\mathbf{x},\ \boldsymbol{\tau})$;
  // compute next per-token times via scheduler h
  $\boldsymbol{\tau}_{\text{next}} = h(\boldsymbol{\tau}, \hat{\mathbf{x}}_0)$;
  // For confidence-based scheduler: high-confidence tokens are committed
  // $\tau_{\text{next}}^\ell = 0$ if $\max(\hat{x}_0^\ell) > \gamma$ else τ^ℓ
  update $\mathbf{x}$ from $\mathbf{x}_\tau$ to $\mathbf{x}_{\tau_{\text{next}}}$ using the reverse transition (Eq. 1);
  $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau}_{\text{next}}$;
end
return $\mathbf{x}$;

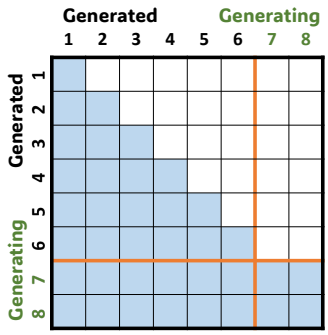
B Algorithm Details

B.1 Training and Sampling

The training procedure of LUDI is summarized in Algorithm 1. It introduces per-token time embeddings $\boldsymbol{\tau}$ to provide token-level corruption hints, and optimizes the less uniform (LU) loss (Eq. 6) that directly encourages each reverse transition toward the clean token. The conditional sampling, detailed in Algorithm 2, leverages a confidence-based scheduler h to progressively commit high-confidence tokens as conditions, enabling few-step generation for complex reasoning tasks.



(a) Training mask.



(b) Inference mask.

Figure 5: Context-Causal attention masks for block-diffusion adaptation. (a) Training mask: the prefix uses a lower-triangular causal mask; the active block attends bidirectionally within itself and causally to the prefix. (b) Inference mask: generated blocks form a causal prefix; only the current block is denoised bidirectionally.

B.2 Details of AR to Block Diffusion

Label shift. AR pretraining predicts x^ℓ from a clean prefix $x^{<\ell}$, while block diffusion forces the model to predict x^ℓ at position ℓ. We resolve this mismatch by shifting the labels left by one position, while still allowing token $\ell - 1$ to attend to token ℓ. This changes only the prediction position, preserving both the diffusion paradigm and the autoregressive prediction preference.

Complementary views. Following Fast dLLM v2 (Wu et al., 2025a), we train with two complementary mask patterns per sequence: positions kept clean in the first view are masked in the second, and vice versa. In the UDLM setting, unmasked positions receive $\mathbf{x}_0$ and masked positions receive a random token. Both views share a single forward pass, forcing every position to serve as both a clean target and a noisy input. This doubling of supervision is particularly valuable in our low-data regime ($\sim$1B tokens).

Context-causal attention mask. NBDiff (Tian et al., 2025) proposed the context-causal attention

mask. The prefix observes a strictly causal mask, while the active denoising block enjoys full intra-block bidirectionality and causal access to the prefix. This yields the attention mask pattern shown in Figure 5, where both noisy and clean sequences are used during training. The clean sequence provides keys and values from causal attention for the subsequent noisy block. This preserves the left-to-right inductive bias in already committed context, preventing instability from premature future exposure. At inference the same pattern holds (Figure 5(b)): the prompt and past blocks form a frozen causal prefix, and only the current block is denoised bidirectionally, enabling KV-cache reuse and parallel token decoding.

AR loss. An auxiliary autoregressive loss is computed on the clean sequence. This incurs no extra forward cost and serves dual purposes: it prevents catastrophic forgetting of the pretrained language model, and it supplies a stable gradient signal that counters the noisier diffusion loss early in training. The combined objective is $\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda \mathcal{L}_{\text{AR}}$ with a weight λ.

C Experiments

C.1 Experimental Setup for Small-Scale LUDI

170M models. We train four 170M UDLMs on OpenWebText (Gokaslan et al., 2019): LUDI with the LU loss, and baselines trained with Duo (Sahoo et al., 2025), SDDLM (Zhu et al., 2025c), and SDDLM-v1 (Zhu et al., 2025c) losses. The denoiser is a 12-block Transformer with hidden size 768, 12 attention heads, a 128-dimensional conditioning embedding, sequence length 1024, dropout 0.1, the GPT-2 vocabulary of 50,257 tokens (Radford et al., 2019), and bfloat16 precision. Each run is trained for 500K iterations with batch size 512. We use AdamW with learning rate 3×10^{-4}, betas $(0.9, 0.999)$. Generative perplexity is computed with GPT-2 Large (Radford et al., 2019), and sample diversity is measured by mean token-level entropy in bits. As in RADD (Ou et al., 2025), Gumbel-based categorical sampling is performed in float64 precision. For released masked-diffusion baselines, we evaluate RADD (Ou et al., 2025) and MDLM (Sahoo et al., 2024) checkpoints trained on the same dataset for 400K and 1M steps, respectively, using the same generation metrics.

1B models. We train four 1B-parameter UDLMs from scratch on FineWeb (Penedo et al., 2024) with the LLaMA tokenizer (Touvron et al., 2023). The denoiser is a 20-block Transformer with hidden size 2048, 16 attention heads, MLP ratio 3.5, a 128-dimensional conditioning embedding, sequence length 2048, dropout 0.0, a 32,000-token vocabulary and bfloat16 precision. Following the SMDM implementation (Nie et al., 2025a), we mix 1% variable-length FineWeb examples into training. Each run is trained for 500K iterations with batch size 256 . We use AdamW with learning rate 2×10^{-4}, betas $(0.9, 0.95)$, weight decay 0.1. We evaluate likelihood-based multiple-choice accuracy on PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), and RACE (Lai et al., 2017). Following SDDLM (Zhu et al., 2025c), each candidate answer is scored with the ELBO-based likelihood estimate induced by Eq. (3). We report accuracy (`acc`) for PIQA, SIQA, ARC-Easy, and RACE, and length-normalized accuracy (`acc_norm`) for HellaSwag and OpenBookQA.

C.2 Experimental Setup for LUDI-7B

Baselines. We compare LUDI-7B against autoregressive (AR) baselines and state-of-the-art diffusion language models at comparable scales. AR baselines include Qwen2.5-7B-Instruct (Team, 2024) (the initialization checkpoint). Masked diffusion baselines include Dream 7B (Ye et al., 2025) (adapted from Qwen2.5-7B), LLaDA 8B (Nie et al., 2025b), LLaDA-1.5 8B (Zhu et al., 2025a), LLaDA-MoE 7B (Zhu et al., 2025b) and Fast-dLLM v2 (Wu et al., 2025a).

Evaluation benchmarks. We evaluate on a diverse set of tasks spanning code generation, mathematical reasoning, instruction following, and knowledge-intensive question answering. For code generation, we use HumanEval (Chen et al., 2021) and HumanEval+ (Liu et al., 2023), as well as MBPP (Austin et al., 2021) and MBPP+ (Liu et al., 2023). Mathematical reasoning is assessed on GSM8K (Cobbe et al., 2021). Instruction following is measured with IFEval (Zhou et al., 2023). For knowledge-intensive tasks, we adopt MMLU (Hendrycks et al., 2020) and GPQA (Rein et al., 2023). All code benchmarks are evaluated using the EvalPlus framework (Liu et al., 2023). For all tasks we report standard accuracy metrics

following the evaluation protocol of Fast-dLLM v2 (Wu et al., 2025a).

C.3 Efficiency of the LU Loss

We analyze the computational advantage of the LU loss in Eq. (6) over the ELBO loss in Eq. (3), tracing it to the structural sparsity that enables a fused Triton kernel for the LU loss, while the dense operations in the Duo loss inherently resist such optimization.

Fused kernel for the LU loss Let $\mathbf{x}_\theta$ be the model's predicted probability vector (after softmax) and recall the parametrization $\bar{x}_{\theta,i} = \alpha_t V x_{\theta,i} + (1-\alpha_t)$. The LU loss is

$$\mathcal{L}_{\text{LU}}^{\ell} = \frac{-\alpha_t'}{1-\alpha_t}\big(\log \bar{x}_{\theta,m} - \log \bar{x}_{\theta,y}\big). \tag{43}$$

A naive implementation materializes the full tensor $\mathbf{x}_\theta \in \mathbb{R}^{B\times L\times V}$, incurring $O(BLV)$ memory traffic. Our fused Triton kernel avoids this through two observations.

Online log-sum-exp. The log-sum-exp over the vocabulary is computed in a single numerically stable pass, using only $O(1)$ auxiliary memory. From the log-sum-exp we directly obtain $\log \bar{x}_{\theta,m}$ and $\log \bar{x}_{\theta,y}$ by indexing the logits, without storing $\mathbf{x}_\theta$ or $\bar{\mathbf{x}}_\theta$. **Sparse gradients.** The gradient w.r.t. the logits z_k is

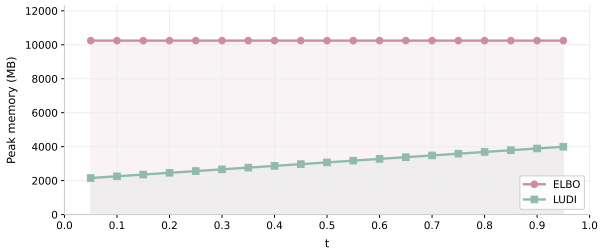
$$\frac{\partial \ell}{\partial z_k} = \begin{cases} \dfrac{\alpha_t V\, x_{\theta,m}}{\bar{x}_{\theta,m}} & k = m, \\ -\dfrac{\alpha_t V\, x_{\theta,y}}{\bar{x}_{\theta,y}} & k = y, \\ 0 & \text{otherwise.} \end{cases} \tag{44}$$

Only the two indices m and y receive non-zero gradients. The backward kernel exploits this sparsity: a 2D grid iterates over the vocabulary dimension but conditionally accumulates gradients only at the targeted positions, effectively performing $O(1)$ memory writes per token.

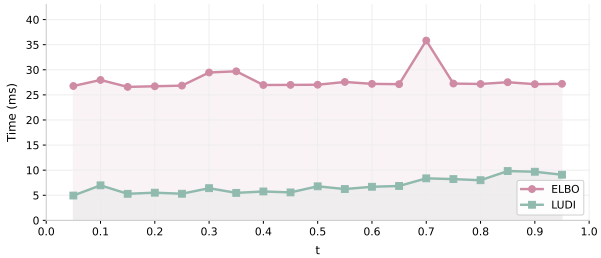
Together, the forward and backward kernels process each token row in $O(V)$ time with $O(1)$ working memory, fully coalescing memory accesses and avoiding materialization of x_θ or $\bar{x}_\theta$.

Memory complexity of the ELBO loss The ELBO loss Eq.(3) involves a sum over the entire vocabulary, $\sum_{j=1}^{V} \log \bar{x}_{\theta,j}$, in addition to point-wise terms at indices m and y. While a fused kernel can avoid storing the full softmax matrix by recomputing $\bar{x}_{\theta,j}$ on the fly and accumulating the sum in registers, this still requires a full pass over the vocabulary for every token. More critically, the backward pass propagates gradients to all V positions through the summation term, producing a dense gradient tensor. In contrast, the LU loss generates gradients at only two positions per token (Eq. 44). Thus, even with aggressive kernel fusion, the ELBO loss incurs $O(BLV)$ gradient writes, while the LU loss writes only $O(BL)$ gradients.

This fundamental sparsity gap makes the LU loss substantially more efficient at scale. As shown in Figure 6, optimized LU loss offers a $4.39\times$ speedup and $3.45\times$ memory saving over the PyTorch baseline.



(a) GPU Memory Usage.



(b) Forward and Backward Time.

Figure 6: Computational efficiency simulation in a long-sequence, large-vocabulary setting with batch size 10, sequence length 1024, and vocabulary size 50,000.

C.4 Convergence

Figure 7 compares the training trajectories of the four UDLM objectives under the same evaluation protocol. LUDI improves rapidly in the early stage and remains in a low-perplexity regime as training proceeds. At 500K steps, LUDI obtains the lowest generative perplexity under both 512-step and 1024-step sampling, whereas Duo and SDDLM plateau at substantially higher PPL. These trajectories support the main-text observation that the LU loss provides a cleaner and more sample-efficient training signal.

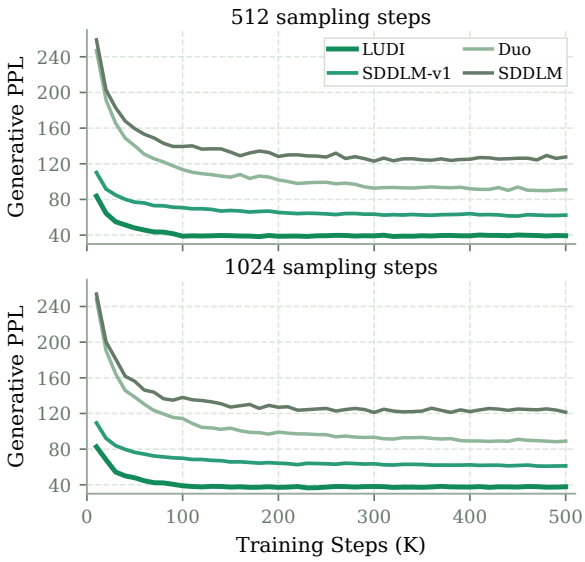


Figure 7: Generative perplexity over training checkpoints. Lower PPL is better. Lines connect all 10K-spaced checkpoints from 10K to 500K steps.

C.5 Quality and Diversity

Figure 8 compares the per-sample distributions of the four losses at the 500K checkpoint, using 1024 denoising steps and 1024 samples for each loss. LUDI places a large fraction of its samples in the low-Gen.PPL region, with a mean Gen.PPL of 37.9 and a mean entropy of 7.50. Its samples are not concentrated in an extremely low-entropy region, suggesting that the lower Gen.PPL is not accompanied by an obvious entropy collapse.

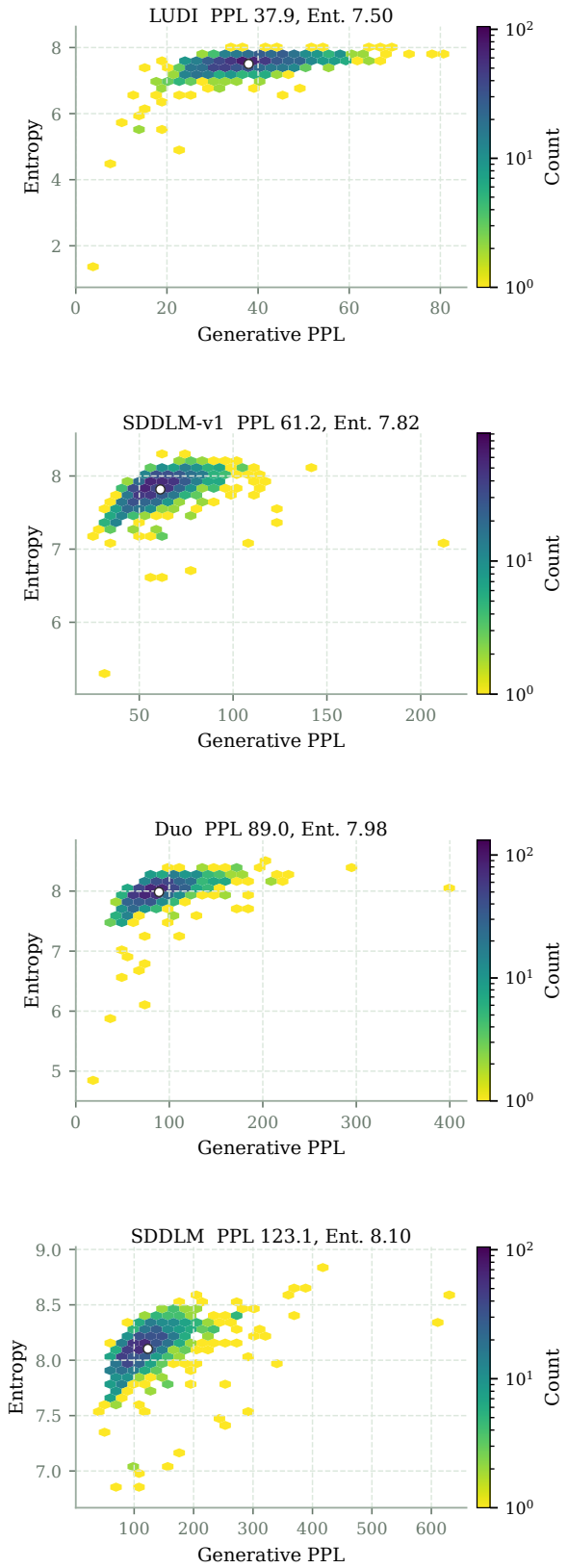


Figure 8: Density of per-sample generative PPL and sample entropy for the four losses at the 500K checkpoint, evaluated with 1024-step sampling and 1024 generated samples per loss. Each panel uses its own axis range and density scale for readability. Darker cells indicate more samples within the corresponding panel, and the white dot marks the method mean. Panel titles report mean Gen.PPL and entropy.