TRAINING LATENT LANGUAGE DIFFUSION MODELS REQUIRES LESS THAN YOU THINK

Anonymous authors
Paper under double-blind review

ABSTRACT

Contextual latent diffusion offers a way to generate language in a representation space that captures linguistic dependencies beyond token identity, but existing approaches typically rely on pretrained language features or a separate representation-learning stage. We introduce **LDLM-s: Scratch Latent Language Diffusion Model**, to our knowledge, the first latent language diffusion framework to achieve strong generation while jointly learning its contextual representation space and generator from random initialization, without pretrained language components or a separate representation-pretraining stage. LDLM-s uses a language-modeling autoencoder to provide linguistic supervision to the latent space, while the diffusion objective simultaneously shapes the encoder, allowing the representation and denoiser to co-adapt throughout training. On OpenWebText, LDLM-s outperforms most of the recent continuous and discrete diffusion baselines on the generative-perplexity–token-entropy tradeoff. It also retains substantially more repetition-clean generations in the low-perplexity regime than the competing continuous baselines. Relative to LDLM, LDLM-s extends the tradeoff toward lower generative perplexity, while LDLM reaches higher-diversity operating points, despite LDLM-s seeing roughly $5\times$ fewer training tokens. More broadly, our results show that strong contextual latent diffusion can be learned without external language representations, providing a step toward self-contained latent language generation directly from text.

1 INTRODUCTION

Language diffusion models depend critically on the representation space in which denoising takes place. With static token representations, the clean target primarily identifies the observed token, leaving the denoiser to recover its relationships with the rest of the sequence. Contextual latents can encode these relationships directly in the clean representation, relieving the diffusion model from having to recover complex linguistic dependencies entirely through denoising. This makes contextual latent spaces an appealing substrate for language diffusion, but raises a fundamental question: where should their linguistic structure come from?

Most contextual latent diffusion systems obtain linguistic structure before generative training. TEncDM and ELF use pretrained contextual representations (Shabalin et al., 2025; Hu et al., 2026). TextLDM trains its latent representation with supervision from a frozen language-model teacher, while COSMOS learns an autoencoder on top of representations produced by a pretrained encoder (Jiang et al., 2026; Meshchaninov et al., 2025). CoLa primarily uses a separately pretrained autoencoder, while its all-scratch joint-training variant substantially underperforms the pretrained configuration (Guo et al., 2026). LDLM learns its latent space jointly with diffusion, but its trainable encoder still builds on frozen GPT-2 features (Meshchaninov et al., 2026). *This raises a more fundamental question: can a strong contextual latent space and its diffusion model be learned together directly from text, without pretrained representations, teacher models, or a separate representation-learning stage?*

A central difficulty is that token reconstruction alone provides little incentive to organize the latent space around linguistic structure. Recovering the observed sequence requires preserving token information, but does not explicitly reward representations for capturing the syntactic, semantic, and discourse-level dependencies that make language coherent and predictable. Prior work has similarly

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

found that reconstruction alone can yield latent spaces poorly suited for diffusion (Meshchaninov et al., 2025; 2026). We instead use language modeling itself as representation supervision. A causal next-token objective encourages the autoencoder to preserve information about the prefix that is useful for modeling language, providing contextual and semantic supervision directly from the same text used to train the generator.

We introduce LDLM-s, to our knowledge the first latent language diffusion framework to achieve strong unconditional generation while jointly learning its contextual representation space and latent diffusion generator from random initialization, without external representation supervision. The key is a language-modeling autoencoder trained together with continuous latent diffusion. Language modeling shapes the representation toward linguistically informative structure, while the diffusion objective simultaneously adapts the encoder toward a latent geometry that can be effectively corrupted and denoised. The denoiser, in turn, continuously tracks this evolving latent distribution. Rather than learning a representation first and fitting a generator afterward, LDLM-s allows the latent space and its generative model to co-adapt throughout training. This also yields a simpler joint-training recipe: unlike LDLM, we train all components together from the beginning without a diffusion-to-encoder warmup.

This joint setup makes the interaction between the objectives and the latent geometry particularly important. We find a simple division of labor. Language modeling provides the linguistic supervision that organizes the shared representation, while reconstruction is best treated as a readout from latents back to tokens. LatentNorm fixes the latent scale and stabilizes joint training from the start. Diffusion should still shape the encoder, but only through the noisy-input path: detaching the clean target substantially improves generation over both full target backpropagation and removing the diffusion gradient from the encoder. With normalized latents, decoder noise remains important, but the model becomes much less sensitive to its precise magnitude. Our gradient analysis exposes the asymmetry between the two diffusion routes, while the latent-geometry analysis shows that target stop-gradient is accompanied by a much more dispersed representation space. Together, these results yield a simple recipe for learning contextual latents and their diffusion model end to end from scratch.

On OpenWebText, LDLM-s outperforms recent continuous and discrete diffusion methods on the generative-perplexity–token-entropy tradeoff. Relative to LDLM, LDLM-s extends the attainable tradeoff toward lower generative perplexity, while LDLM reaches higher-diversity operating points; LDLM-s sees roughly $5\times$ fewer training tokens. In the low-perplexity regime, LDLM-s also suffers substantially less from four-gram repetition than the competing continuous and discrete baselines. More broadly, our results show that strong contextual latent diffusion can be learned without external language representations, providing a step toward latent language generation learned entirely from text.

Our contributions are:

- **Contextual latent diffusion learned entirely from text.** To our knowledge, LDLM-s is the first contextual latent language diffusion model to achieve strong generation while jointly learning its representation space and generative model from random initialization, without pretrained language components, teacher models, or a separate representation-learning stage. Language modeling provides the internal supervision that makes this joint representation learning possible.
- **A simple recipe for stable end-to-end latent learning.** Through controlled ablations and gradient and geometry analysis, we identify the roles of the main training signals: language modeling organizes the contextual representation, reconstruction serves as a detached token readout, LatentNorm fixes latent scale, nonzero decoder noise regularizes decoding, and diffusion shapes the encoder through the noisy-input path while its clean target is detached. This combination removes the diffusion-to-encoder warmup and allows all components to co-adapt from the first training step.
- **Strong generation without external representations.** On OpenWebText, LDLM-s outperforms most of the recent continuous and discrete diffusion baselines on the generative-perplexity–token-entropy tradeoff and exhibits markedly less low-perplexity repetition than the continuous baselines. Relative to LDLM, it extends the attainable tradeoff toward lower generative perplexity, while LDLM reaches higher-diversity operating points; LDLM-s sees roughly $5\times$ fewer training tokens.

2 Related Work

2.1 Continuous language diffusion

Continuous diffusion models generate text by denoising a continuous representation and mapping the result back to tokens. Diffusion-LM and PLAID learn token embeddings jointly with the generative model (Li et al., 2022; Gulrajani & Hashimoto, 2023). LangFlow connects embedding-space flow matching with token cross-entropy through Bregman divergence and studies noise scheduling and self-conditioning (Chen et al., 2026). CoBit applies continuous diffusion to analog bit representations of tokens (Batzolis et al., 2026). These methods operate in spaces whose clean targets are determined by token identity; a contextual denoiser can process such inputs, but the clean value itself does not depend on the prefix.

2.2 Contextual latent diffusion

Latent diffusion can instead model contextual encoder outputs. Latent Diffusion for Language Generation and TEncDM demonstrate the value of pretrained language representations (Lovelace et al., 2023; Shabalin et al., 2025). ELF performs flow matching over pretrained T5 contextual representations with a shared denoising-and-decoding network (Hu et al., 2026). TextLDM aligns a text VAE with a frozen language-model teacher before learning its flow model (Jiang et al., 2026). CoLa primarily pretrains a causal text VAE before joint VAE–flow training; its all-scratch variant substantially underperforms the pretrained configuration (Guo et al., 2026).

LDLM jointly learns the latent space and its diffusion model, coupling a trainable encoder above frozen GPT-2 features with a diffusion Transformer trained end to end (Meshchaninov et al., 2026). LDLM uses the tangent noise schedule introduced in TEncDM (Shabalin et al., 2025), adaptive timestep sampling following *Continuous Diffusion for Categorical Data* (Dieleman et al., 2022), and self-conditioning introduced by Chen et al. (2023). Our work learns contextual features directly from tokens using a randomly initialized language-modeling autoencoder. NTP trains the shared representation, reconstruction learns a token readout, and diffusion adapts the latents through its noisy-input path. The distinction is the *source and timing* of representation supervision: our encoder receives its training signal entirely from within the generative training process, without pretrained weights, a teacher, or a separate autoencoder-training stage.

2.3 Discrete diffusion

Discrete diffusion models operate directly on categorical tokens. MDLM uses absorbing-mask corruption (Sahoo et al., 2024); ReMDM enables already generated tokens to be remasked and refined (Wang et al., 2025). Duo studies uniform-state corruption through its connection to Gaussian diffusion. We include these methods as generation baselines.

3 Preliminaries

Continuous text diffusion. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathcal{V}^n$ denote a token sequence. Continuous text diffusion first maps it to a sequence of continuous representations $\mathbf{z}_0 = \mathcal{E}(\mathbf{w}) \in \mathbb{R}^{n \times d}$. A variance-preserving forward process gradually corrupts these representations with Gaussian noise,

$$\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \alpha_t^2 + \sigma_t^2 = 1, \tag{1}$$

where $t = 0$ corresponds to clean data and $t = 1$ to noise. A denoiser $\hat{\mathbf{z}}_\theta$ learns to recover the clean representation from $\mathbf{z}_t$, for example through

$$\mathcal{L}_{\text{Diff}} = \mathbb{E}_{\mathbf{w},t,\boldsymbol{\epsilon}} \left[\| \hat{\mathbf{z}}_\theta(\mathbf{z}_t, t) - \mathbf{z}_0 \|_2^2 \right]. \tag{2}$$

Generation starts from Gaussian noise, iteratively denoises the complete latent sequence, and maps the final representation back to tokens with a decoder $\mathcal{D}$. Unlike discrete diffusion, token identities need not be selected during intermediate denoising steps; consequently, the geometry and linguistic content of the continuous representation space play a central role in generation quality.

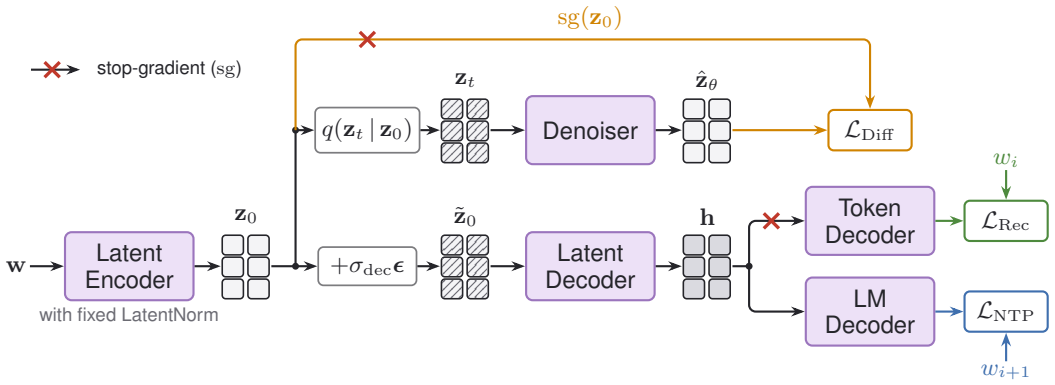

Figure 1: Overview of LDLM-s training. The Latent Encoder output $\mathbf{z}_0$ feeds three paths: the Latent Decoder reads $\tilde{\mathbf{z}}_0 = \mathbf{z}_0 + \sigma_{\text{dec}}\boldsymbol{\epsilon}$, the Denoiser reads $\mathbf{z}_t \sim q(\mathbf{z}_t \mid \mathbf{z}_0)$, and $\mathrm{sg}(\mathbf{z}_0)$ is the diffusion target. The reconstruction ($\mathcal{L}_{\text{Rec}}$), next-token ($\mathcal{L}_{\text{NTP}}$) and diffusion ($\mathcal{L}_{\text{Diff}}$) losses are summed, and all modules are trained jointly from scratch. $\mathcal{L}_{\text{NTP}}$ and $\mathcal{L}_{\text{Diff}}$ reach the encoder, the latter only through $\mathbf{z}_t$; $\mathcal{L}_{\text{Rec}}$ trains only the Token Decoder.

Joint latent-space learning in LDLM. LDLM (Meshchaninov et al., 2026) makes the latent representation itself trainable together with the diffusion model. It first obtains contextual hidden states $\mathbf{h} = \mathcal{E}_{\mathbf{h}}(\mathbf{w})$ from a frozen pretrained GPT-2 encoder and maps them through a trainable Latent Encoder, $\mathbf{z}_0 = \mathcal{E}_{\mathbf{z}}(\mathbf{h})$. A trainable Latent Decoder reconstructs the pretrained hidden states from noisy latents using an MSE objective, after which a separate token decoder maps the reconstructed states back to text. The token-level cross-entropy gradient is stopped before the Latent Decoder, so the learned latent space is shaped primarily by hidden-state reconstruction and diffusion. Crucially, the diffusion loss also backpropagates into the Latent Encoder, allowing the representation and denoiser to co-adapt rather than fixing the latent space in advance.

Naive joint optimization in LDLM is unstable: early in training, reconstruction and diffusion exert competing pressure on the learned representation. LDLM therefore uses a diffusion-to-encoder warmup, together with decoder-input noise and adaptive timestep sampling, to stabilize joint training (Meshchaninov et al., 2026). The resulting framework demonstrates the benefit of adapting contextual latents to the diffusion objective, but their linguistic structure still originates from a frozen pretrained language model. Our work takes this joint-learning principle as a starting point and asks whether the contextual representation itself can instead be learned within the same training process directly from text.

4 Method

We propose LDLM-s, a latent language diffusion model that learns its contextual representation space and generative model jointly from text. Language modeling provides linguistic supervision to the latent space, while diffusion simultaneously adapts the encoder and learns its distribution. A separate reconstruction pathway maps generated latents back to tokens. All components are trained jointly from the beginning, without pretrained language representations, autoencoder pretraining, or diffusion warmup. Figure 1 summarizes the framework.

4.1 Language-modeling supervision for causal autoencoding

Let $\mathbf{w} = (w_1, \ldots, w_n)$ be a token sequence. A causal *Latent Encoder* $\mathcal{E}^\theta_{\mathbf{z}}$ produces contextual latents

$$\mathbf{z}_0 = \mathcal{E}^\theta_{\mathbf{z}}(\mathbf{w}), \qquad \mathbf{z}_{0,i} = \mathcal{E}^\theta_{\mathbf{z}}(\mathbf{w}_{\le i}). \tag{3}$$

The encoder terminates with a fixed, non-trainable LatentNorm, which fixes the per-position latent scale so that

$$\frac{1}{d}\|\mathbf{z}_{0,i}\|_2^2 = 1. \tag{4}$$

Thus, the latent scale cannot drift during joint training, helping stabilize optimization without a diffusion warmup.

Before decoding, we perturb the latents with Gaussian noise,

$$\tilde{\mathbf{z}}_0 = \mathbf{z}_0 + \sigma_{\text{dec}}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{5}$$

and process them with a causal *Latent Decoder*,

$$\mathbf{h} = \mathcal{D}^{\theta}_{\mathbf{h}}(\tilde{\mathbf{z}}_0). \tag{6}$$

Following LDLM, we choose $\sigma_{\text{dec}} = 3$ as default. However, our ablations (Table 2) show similar generation quality for nonzero training noise.

Two separate token decoders operate on $\mathbf{h}$. The bidirectional *Token Decoder* reconstructs the observed sequence, while the causal *LM Decoder* predicts the next token:

$$\mathcal{L}_{\text{Rec}} = -\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log p^{\text{Rec}}_{\theta}\left(w_i \mid \text{sg}(\mathbf{h}_{1:n})\right)\right], \tag{7}$$

$$\mathcal{L}_{\text{NTP}} = -\mathbb{E}\left[\frac{1}{n-1}\sum_{i=1}^{n-1}\log p^{\text{LM}}_{\theta}\left(w_{i+1} \mid \mathbf{h}_{\leq i}\right)\right]. \tag{8}$$

The Token Decoder uses bidirectional attention because it maps a complete latent sequence back to text. The LM Decoder is causal.

In our final configuration, reconstruction receives $\text{sg}(\mathbf{h})$ and therefore trains only the Token Decoder. $\mathcal{L}_{\text{NTP}}$ remains differentiable through the Latent Decoder and Latent Encoder and provides language-modeling supervision to the shared representation. We study this gradient routing in Section 5.

4.2 Latent diffusion

We model the contextual latent distribution with a Diffusion Transformer (Peebles & Xie, 2023), using the variance-preserving process introduced in Section 3. Following LDLM (Meshchaninov et al., 2026), we use the tangent noise schedule and $\mathbf{z}_0$-prediction. The denoiser is trained with

$$\mathcal{L}_{\text{Diff}} = \mathbb{E}_{\mathbf{w},t,\boldsymbol{\epsilon}}\left[\left\|\hat{\mathbf{z}}_{\theta}(\mathbf{z}_t, t, \mathbf{x}_{\text{sc}}) - \text{sg}(\mathbf{z}_0)\right\|_2^2\right], \tag{9}$$

where $\mathbf{x}_{\text{sc}}$ is a detached self-conditioning estimate. Although the clean target is detached, gradients still reach the Latent Encoder through $\mathbf{z}_t$, allowing the latent representation and denoiser to co-adapt. We analyze this gradient routing in Section 5.3.

We also adopt LDLM's adaptive timestep sampler, using its evolving distribution both to sample diffusion times during training and to construct the reverse-time sampling grid. Details are given in Appendix A.2.

Self-conditioning and guidance. We use self-conditioning during both training and sampling. During training, with probability 0.5, a detached preliminary prediction of $\mathbf{z}_0$ is fed back to the denoiser as $\mathbf{x}_{\text{sc}}$; otherwise $\mathbf{x}_{\text{sc}} = \mathbf{0}$. During sampling, $\mathbf{x}_{\text{sc}}$ is the prediction from the preceding denoising step.

Following ELF (Hu et al., 2026), we use this signal for classifier-free-style self-conditioning guidance:

$$\hat{\mathbf{z}}^{(w)}_{\theta}(\mathbf{z}_t, t, \mathbf{x}_{\text{sc}}) = (1-w)\hat{\mathbf{z}}_{\theta}(\mathbf{z}_t, t, \mathbf{0}) + w\hat{\mathbf{z}}_{\theta}(\mathbf{z}_t, t, \mathbf{x}_{\text{sc}}). \tag{10}$$

At $w = 1$, this reduces to ordinary self-conditioning; increasing w provides an inference-time control over the quality–diversity tradeoff. We explicitly evaluate both branches, requiring two denoiser evaluations per guided sampling step.

Generation follows the reverse-time VP SDE with Euler–Maruyama and ends with a single pass through the Latent Decoder and Token Decoder. The Latent Encoder and LM Decoder are not needed at inference time.

The complete training objective is

$$\mathcal{L} = \mathcal{L}_{\text{Rec}} + \mathcal{L}_{\text{NTP}} + \mathcal{L}_{\text{Diff}}, \tag{11}$$

with unit weights. All components are optimized jointly from the first training step; unlike LDLM, we do not use a diffusion-to-encoder warmup.

5 Ablations and analysis

Experimental setup. All ablations use OWT-128, 100,000 training updates. We report generative perplexity (PPL), MAUVE, token entropy (H), and n-gram diversity (DIV); lower PPL and higher MAUVE, entropy, and DIV are better. Each configuration is evaluated with five generation seeds from a single trained checkpoint.

5.1 Language modeling and gradient routing

Representation supervision with next-token prediction. Removing NTP causes the largest degradation among the objective ablations. When reconstruction CE is left to shape the autoencoder without the NTP objective, generation deteriorates sharply, showing that token reconstruction alone is not sufficient to organize a useful contextual latent space.

Reconstruction as a token readout. Propagating reconstruction cross-entropy through the shared autoencoder does not improve the overall quality–diversity tradeoff. We therefore detach reconstruction CE at the shared hidden states and use it only as the final token readout, leaving language modeling and diffusion to shape the latent representation. These observations align with LDLM's findings.

Selective coupling to diffusion. The best behavior is obtained when diffusion shapes the encoder through the noisy input while the clean target is detached. Backpropagating through the clean target substantially degrades generation, whereas removing the diffusion gradient from the encoder also hurts. This points to an asymmetric role of the two gradient routes, which we examine directly in Section 5.3.

Table 1: OWT-128 ablations. Attached/detached CE: whether reconstruction CE updates the autoencoder.

Configuration	PPL↓	MAUVE↑	H ↑	DIV↑
No NTP, attached CE	$152.1_{\pm 1.0}$	$0.55_{\pm 0.04}$	$4.18_{\pm 0.01}$	$0.45_{\pm 0.01}$
NTP, attached CE	$77.3_{\pm 0.4}$	$0.84_{\pm 0.03}$	$4.17_{\pm 0.01}$	$0.46_{\pm 0.01}$
NTP, detached CE	$73.8_{\pm 0.6}$	$0.80_{\pm 0.02}$	$4.19_{\pm 0.01}$	$0.48_{\pm 0.01}$
Full target gradient	$142.9_{\pm 1.3}$	$0.70_{\pm 0.01}$	$4.20_{\pm 0.01}$	$0.60_{\pm 0.01}$
No diffusion input gradient	$100.8_{\pm 1.2}$	$0.84_{\pm 0.03}$	$4.20_{\pm 0.01}$	$0.50_{\pm 0.01}$

5.2 Fixed latent geometry and decoder noise

Fixed latent geometry. LatentNorm removes latent scale as a learnable degree of freedom: every token latent is centered and has unit RMS. This prevents the encoder from minimizing the diffusion loss by simply shrinking or expanding the representation and allows the complete model to train jointly from the first step without the diffusion-to-encoder warmup used by LDLM.

$$\mathbf{1}^\top \mathbf{z}_{0,i} = 0, \qquad \frac{1}{d}\|\mathbf{z}_{0,i}\|_2^2 = 1. \tag{12}$$

Decoder noise. Once latent RMS is fixed, σ_{dec} has a direct meaning: it sets the corruption magnitude relative to a unit-RMS latent and can no longer be absorbed by rescaling the representation. This makes generation much less sensitive to its precise value. Nonzero decoder noise remains essential: without it, generation degrades sharply, whereas $\sigma_{\text{dec}} \in \{1, 2, 3\}$ yields similar generation behavior. In this normalized space, $\sigma_{\text{dec}} = 1$ is a natural reference scale: the injected Gaussian noise has the same per-coordinate variance as the clean latent, giving an equal-RMS perturbation without introducing another latent-scale degree of freedom. This observation relieves the burden of tuning another hyperparameter and further simplifies the method.

Table 2: Decoder-noise ablation on OWT-128.

σ_{dec}	PPL↓	MAUVE↑	H ↑	DIV↑
0	$430.7_{\pm 3.6}$	$0.16_{\pm 0.02}$	$4.251_{\pm 0.002}$	$0.562_{\pm 0.002}$
1	$79.3_{\pm 0.6}$	$0.83_{\pm 0.01}$	$4.204_{\pm 0.001}$	$0.479_{\pm 0.002}$
2	$78.4_{\pm 0.7}$	$0.82_{\pm 0.03}$	$4.199_{\pm 0.002}$	$0.480_{\pm 0.002}$
3	$73.8_{\pm 0.6}$	$0.80_{\pm 0.02}$	$4.185_{\pm 0.001}$	$0.475_{\pm 0.004}$

5.3 Gradient-route analysis

The diffusion objective reaches the Latent Encoder through two gradient routes. Let

$$\mathbf{e}_t = \hat{\mathbf{z}}_\theta(\mathbf{z}_t, t) - \mathbf{z}_0, \qquad \mathbf{J}_t = \frac{\partial \hat{\mathbf{z}}_\theta(\mathbf{z}_t, t)}{\partial \mathbf{z}_t}, \tag{13}$$

where $\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \boldsymbol{\epsilon}$. For the squared-error diffusion objective with a differentiable clean target,

$$\nabla_{\mathbf{z}_0} \mathcal{L}_{\text{full}} = 2 \left(\alpha_t \mathbf{J}_t^\top - \mathbf{I} \right) \mathbf{e}_t. \tag{14}$$

Detaching the clean target removes only its direct derivative while preserving the path through the noisy input:

$$\nabla_{\mathbf{z}_0} \mathcal{L}_{\text{SG}} = 2 \alpha_t \mathbf{J}_t^\top \mathbf{e}_t. \tag{15}$$

The difference between the two gradients is therefore exactly

$$\nabla_{\mathbf{z}_0} \mathcal{L}_{\text{full}} - \nabla_{\mathbf{z}_0} \mathcal{L}_{\text{SG}} = 2 \left(\mathbf{z}_0 - \hat{\mathbf{z}}_\theta \right). \tag{16}$$

At low and intermediate noise, the factor $\alpha_t \mathbf{J}_t^\top$ allows diffusion to shape the representation through the corrupted input. Intermediate noise is particularly useful: the denoising problem is non-trivial, while the corrupted input still contains substantial information about the clean example. At high noise, however, α_t becomes small and the noisy-input route weakens, whereas the direct target-fitting term in Eq. (14) can remain large whenever the prediction error is large. Target stop-gradient therefore removes a gradient component that can dominate precisely when the noisy observation carries little information about the individual clean example, while retaining the route that couples representation learning to denoising.

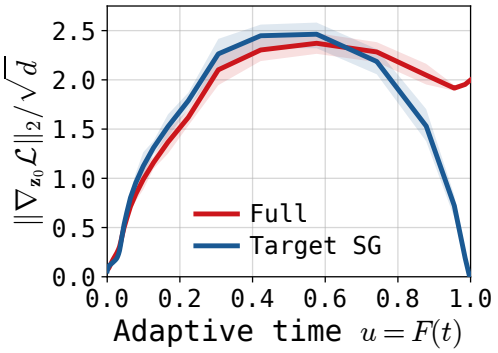

Figure 2: **Effect of diffusion gradient routing.** Clean-latent gradient magnitude with and without target stop-gradient.

Figure 2 shows the corresponding empirical gradient magnitudes. Under full-target backpropagation, the clean-latent gradient remains strong into the high-noise regime, whereas target stop-gradient tracks the weakening noisy-input route. The diagnostic protocol is given in Appendix B.

Clean-latent geometry. LatentNorm fixes the mean and scale of each clean latent but leaves its direction unconstrained. We therefore also measure the pairwise cosine similarity between clean latent representations. Full-target training produces strongly aligned latents, with mean pairwise cosine similarity of approximately 0.49, whereas target stop-gradient reduces it to approximately 0.00. Thus, the difference between the two gradient routes is accompanied by a substantial change in the directional organization of the latent space, rather than merely its scale.

6 Experiments

6.1 Main Results on OpenWebText

Main experiments use OWT-1024 and otherwise follow the ablation setup, except for the longer sequence length and training duration. The evaluated LDLM-s checkpoint is trained from random initialization for 195,000 updates, corresponding to approximately 102B GPT-2 tokens. We compare against the continuous diffusion and flow baselines ELF-B (Hu et al., 2026), LangFlow (Chen et al., 2026), and CoBit (Batzolis et al., 2026); the closest jointly trained latent-diffusion baseline LDLM (Meshchaninov et al., 2026); and the discrete diffusion baselines Duo (Sahoo et al., 2025) and MDLM (Sahoo et al., 2024). Detailed evaluation settings are provided in Appendix D.

Quality–diversity frontier. Figure 3 compares generative perplexity and token entropy across sampling configurations. LDLM-s establishes a stronger quality–diversity frontier than ELF-B, LangFlow, and CoBit. Relative to the closest latent-diffusion baseline, LDLM, LDLM-s extends the frontier substantially further toward low generative perplexity, while LDLM reaches higher-entropy operating points. This higher token entropy, however, does not translate into less sequence-level repetition at comparable generative perplexity, which we examine separately below.

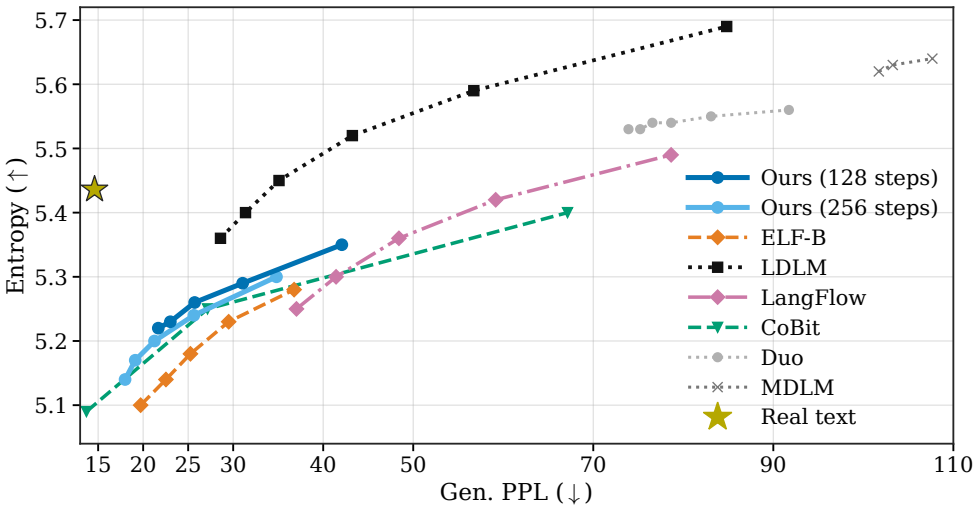

Figure 3: **Quality–diversity frontier on OWT.** LDLM-s is shown at two sampling budgets, with each curve traced by varying self-conditioning guidance. Higher token entropy and lower generative perplexity are better.

Training and inference efficiency. Strong generation emerges well before the end of training. After roughly 60B training tokens (∼100k updates), LDLM-s already establishes a stronger quality–diversity frontier than ELF-B. Continued training to roughly 100B tokens further improves the frontier, with a clear shift toward higher-diversity operating points while preserving the low-perplexity regime (Figure 4a). The final model is therefore obtained with roughly $5\times$ fewer training tokens than LDLM. Appendix C provides the complete training configuration.

Inference efficiency. Because self-conditioning guidance evaluates two denoiser branches, we compare methods by the number of denoiser evaluations (NFE), rather than sampling steps alone. Despite this additional guidance cost, LDLM-s reaches lower generative perplexity than the lowest-perplexity reported LDLM operating point using at least $4\times$ fewer denoiser evaluations, and extends the low-perplexity frontier at both evaluated sampling budgets.

Low perplexity is not repetition. Low generative perplexity can be misleading when a model achieves it by falling into highly predictable repetitive patterns rather than by producing better text. Recent work (Zhang et al., 2026) highlights this failure mode in continuous diffusion language models, so we evaluate sequence-level repetition alongside Gen-PPL. Figure 4b reports the fraction of generated samples whose SEQ-REP-4 score remains below the human-reference threshold. SEQ-REP-4 measures repeated word four-grams, so a higher fraction corresponds to fewer generations with strong local repetition.

LDLM-s reaches low generative perplexity without the sharp increase in sequence-level repetition observed for several competing continuous models. As ELF-B and CoBit move toward lower Gen-PPL, repetition becomes increasingly prominent, while LangFlow avoids this failure mode to a greater extent but remains in a substantially higher-perplexity regime. In contrast, LDLM-s retains a high fraction of repetition-clean generations throughout the low-perplexity regime and shows the same advantage over LDLM at comparable Gen-PPL. Together, these results indicate that the low Gen-PPL of LDLM-s is not primarily driven by repetitive degeneration.

7 CONCLUSION

We introduced LDLM-s, a latent language diffusion model that learns its contextual representation space and generative model jointly from random initialization, without pretrained language representations or a separate representation-learning stage. The central ingredient is a language-modeling autoencoder: next-token prediction provides the linguistic supervision needed to organize the latent space, while reconstruction serves as a token readout. Stable joint training further depends on

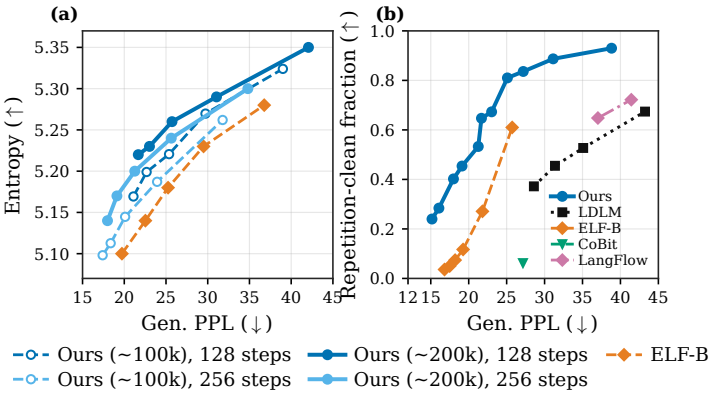

Figure 4: **Training progress and repetition.** (a) Quality–diversity frontiers for the early and final checkpoints. (b) Fraction of repetition-clean generations as a function of generative perplexity.

controlling the geometry and gradient routes of the shared representation: LatentNorm fixes latent scale, and diffusion shapes the encoder through the noisy-input path while the clean target is detached. Our gradient and latent-geometry analyses provide a direct view of how this routing changes the learned representation.

On OpenWebText, LDLM-s establishes a strong generative-perplexity–token-entropy frontier while maintaining substantially less sequence-level repetition in the low-perplexity regime than competing continuous diffusion models. Strong generation already emerges at a substantially smaller training budget, and continued training shifts the frontier toward higher-diversity operating points. Taken together, these results show that useful contextual latent spaces need not be inherited from pretrained language models or learned in a separate stage: representation learning and diffusion can co-adapt successfully within a single end-to-end training process. We hope this provides a simpler foundation for scaling latent language diffusion and for studying how linguistic structure should be organized when the representation itself is learned together with the generator.

Reproducibility Statement

We provide an anonymized code repository as supplementary material containing the training and evaluation code, experiment configurations, metric implementations, and scripts used to produce the main figures. Together with the repository, the appendices document the complete experimental setup. Appendix A specifies the model components and gradient routing, Appendix A.2 describes training-time sampling and generation, and Appendix C reports preprocessing, architecture, and optimization settings. The protocols for the gradient analysis and repetition evaluation are given in Appendices B and F, respectively.

AI Use Statement

Generative AI was used to assist with drafting and restructuring the manuscript, checking related work, deriving and verifying parts of the gradient analysis, preparing figures and LaTeX, and writing and modifying experimental code. AI assistance was also used during experiment execution, for example to help inspect logs, debug runs, and prepare evaluation scripts and commands.

The research questions, hypotheses, methodological ideas, experimental design, and overall scientific direction were developed by the authors. All AI-generated code, analysis, and manuscript content were reviewed by the authors before use. The authors manually verified the experimental setup, inspected the resulting outputs, and checked the reported measurements and figures. All final decisions about the methodology, implementation, analysis, interpretation, and manuscript content were made by the authors.

REFERENCES

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. CoBit: Language modeling with bitstream diffusion, 2026. URL https://arxiv.org/abs/2605.07013.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling, 2026. URL https://arxiv.org/abs/2604.11748.

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

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus, 2019. URL https://skylion007.github.io/OpenWebTextCorpus/.

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models, 2023. URL https://arxiv.org/abs/2305.18619.

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

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows, 2026. URL https://arxiv.org/abs/2605.10938.

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

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14217.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2492. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b2a2bd5d5051ff6af52e1ef60aefd255-Abstract-Conference.html.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.21170.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space, 2026. URL https://arxiv.org/abs/2605.07933.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

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

https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619. PMLR, 2025. URL https://proceedings.mlr.press/v267/sahoo25a.html.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. TEncDM: Understanding the properties of the diffusion model in the space of language model encodings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):25110–25118, 2025. doi: 10.1609/aaai.v39i23.34696. URL https://ojs.aaai.org/index.php/AAAI/article/view/34696.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling, 2025. URL https://arxiv.org/abs/2503.00307.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Shuai Zhang, Zijie Chen, Hongliang He, Lun Du, and Zhenzhong Lan. Low perplexity is repetition: A one-dimensional self-conditioning attractor in continuous diffusion LMs, 2026. URL https://arxiv.org/abs/2607.00588.

A TECHNICAL SPECIFICATION

A.1 HEADS, LOSSES, AND GRADIENT PATHS

Throughout, θ denotes the collection of trainable model parameters; repeated use of this symbol does not imply weight sharing. The Token Decoder and LM Decoder define p_θ^{Rec} and p_θ^{LM}, respectively, while the denoiser is $\hat{\mathbf{z}}_\theta$.

The Latent Encoder $\mathcal{E}_{\mathbf{z}}^\theta$ and Latent Decoder $\mathcal{D}_{\mathbf{h}}^\theta$ are causal Transformers. The Latent Decoder produces

$$\mathbf{h} = \mathcal{D}_{\mathbf{h}}^\theta(\tilde{\mathbf{z}}_0).$$

Reconstruction uses a bidirectional two-layer Token Decoder, while NTP uses a separate causal two-layer LM Decoder; each is followed by a vocabulary projection. In the final configuration, reconstruction reads $\mathrm{sg}(\mathbf{h})$, whereas NTP reads $\mathbf{h}$ directly.

Consequently, reconstruction CE updates only the Token Decoder and has no gradient path to the Latent Decoder or Latent Encoder. NTP updates the LM Decoder and provides the language-modeling gradient to both latent components.

The LM Decoder vocabulary projection is tied to the Latent Encoder input embedding matrix, while the Token Decoder vocabulary projection is independent. All remaining decoder parameters are distinct. Thus, weight sharing is restricted to the vocabulary matrix rather than the complete prediction head. Table 3 summarizes the resulting gradient routes.

Table 3: Gradient routes for the three training objectives. The tied input/output embedding matrix is counted with the Latent Encoder; the LM Decoder column contains its remaining untied parameters.

Loss	Latent Encoder	Latent Decoder	Token Decoder	LM Decoder	Denoiser
$\mathcal{L}_{\mathrm{Rec}}$	–	–	Yes	–	–
$\mathcal{L}_{\mathrm{NTP}}$	Yes	Yes	–	Yes	–
$\mathcal{L}_{\mathrm{Diff}}$	Noisy input	–	–	–	Yes

The diffusion input path satisfies $\partial \mathbf{z}_t / \partial \theta = \alpha_t \, \partial \mathbf{z}_0 / \partial \theta$. Self-conditioning is detached and provides no additional backward path. One update jointly evaluates reconstruction, NTP, and diffusion and sums the active gradients. Cross-entropy terms are averaged over their valid target positions, and diffusion training uses the mean squared error over valid latent elements. The norm in Eq. (9) suppresses this averaging for readability; the gradient diagnostic uses an explicitly summed loss instead.

A.2 ADAPTIVE TIME SAMPLING AND GENERATION

The tangent schedule and adaptive time sampler follow Meshchaninov et al. (2026). We use the variance-preserving tangent schedule

$$\alpha_t^2 = \left[1 + d_{\mathrm{sched}}^2 \tan^2\left(\frac{\pi t}{2}\right)\right]^{-1}, \qquad \sigma_t^2 = 1 - \alpha_t^2,$$

with $d_{\mathrm{sched}} = 3$. The adaptive sampler tracks the denoising-loss profile over time and allocates sampling probability according to its increase across noise levels. Let $q(t)$ denote the resulting density and $F(t) = \int_0^t q(s)\,\mathrm{d}s$ its CDF. Training samples $t \sim q$, while the reverse-time sampling grid is obtained by applying F^{-1} to uniformly spaced probability levels and traversing them from noise to data. We therefore use $u = F(t)$ as the adaptive time coordinate in our analysis.

In practice, we divide $[0, 1]$ into 100 equal bins, estimate the denoising loss at their boundaries, and sample each bin in proportion to the corresponding loss increase, followed by uniform sampling within the selected bin. We use an EMA coefficient of 0.9, refresh the bin weights every 5000 updates, and set the minimum bin weight to 10^{-4}.

Self-conditioning during training uses a detached preliminary prediction with probability 0.5. During generation, the conditioning input is initialized at zero and subsequently replaced by the previous clean prediction. We integrate the reverse VP SDE with Euler–Maruyama. For the evaluated LDLM-s model, the final Latent Decoder receives

$$\hat{\mathbf{z}}_0 + \sigma_{\mathrm{dec}}\boldsymbol{\epsilon}, \qquad \sigma_{\mathrm{dec}} = 3, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

matching the decoder corruption used during training. In contrast, LDLM decodes the generated latent without this additional noise.

For N trajectory steps, ordinary self-conditioning ($w = 1$) requires N denoiser evaluations, while two-branch self-conditioning guidance requires $2N$. The final Latent Decoder and Token Decoder pass is performed once after the denoising trajectory.

B Gradient diagnostic protocol

We evaluate clean-latent gradient magnitudes on 32 fixed OWT-1024 sequences, using four noise realizations per sequence and 40 measurements over the adaptive time coordinate $u = F(t)$. The full-target and target-stop-gradient objectives use identical sequences, noise, self-conditioning inputs, and model predictions. We report pooled root-mean-square gradient magnitudes with 95% batch-bootstrap intervals.

The diagnostic uses the trained encoder, LatentNorm, self-conditioning, and noise schedule in evaluation mode with autograd enabled. The two objectives reuse the same forward pass: only the clean target is detached for the target-stop-gradient variant, while the prediction retains its gradient through the noisy input. Reconstruction and NTP are excluded from the diagnostic.

For sequence i, noise realization k, and variant v, let

$$\mathbf{g}_{i,k}^{(v)}(t) = \nabla_{\mathbf{z}_0} L_v,$$

and let d_i denote the number of latent elements in sequence i. The pooled root-mean-square gradient magnitude is

$$G_v(t) = \sqrt{\frac{\sum_{i,k} \left\| \mathbf{g}_{i,k}^{(v)}(t) \right\|_2^2}{\sum_{i,k} d_i}}. \tag{17}$$

We aggregate squared gradient norms across sequences and noise realizations before normalization, avoiding cancellation that would arise from averaging gradient vectors directly. The transformation from t to $u = F(t)$ changes only the horizontal coordinate and does not reweight the loss.

As an implementation check, the paired objectives have identical forward loss values and identical denoiser-parameter gradients, while their clean-latent gradients satisfy

$$\mathbf{g}_{\mathrm{full}} - \mathbf{g}_{\mathrm{SG}} = 2\left(\mathbf{z}_0 - \hat{\mathbf{z}}_0\right), \tag{18}$$

for the summed clean-latent squared-error objective.

C Architecture and training

The final 100,000 OpenWebText documents (Gokaslan & Cohen, 2019) are reserved for validation and the remainder are used for training. Following LDLM preprocessing, the corpus is tokenized with GPT-2 and concatenated into a token stream with end-of-text separators preserving document boundaries. The stream is split into fixed-length sequences of 128 or 1024 tokens without padding. No pretrained language-model weights are used to initialize the model.

The OWT-1024 run uses eight distributed workers with per-worker batch size 16 and four gradient-accumulation steps, yielding a global batch size of 512. The model contains 324,231,168 trainable parameters (324,232,704 parameters in total).

Table 4: Architecture and training settings.

Setting	Value
Latent Encoder / Latent Decoder depth	6 / 6 Transformer layers
Token Decoder / LM Decoder	Bidirectional / causal, 2 layers each
Hidden / feed-forward width	768 / 3072
Attention heads	12
Denoiser	12-layer DiT
Latent dimension / LatentNorm target RMS	768 / 1
Initialization	Random
Dataset / tokenizer	OpenWebText / GPT-2
Ablation / main sequence length	128 / 1024
Ablation / main training updates	100,000 / 195,000
Global batch size	512 sequences
Optimizer	AdamW
Learning-rate schedule	5000-step linear warmup from 10^{-8} to 3×10^{-4}; constant thereafter
Adam (β_1, β_2) / ϵ	$(0.9, 0.98)$ / 10^{-8}
Weight decay / gradient clipping	0.01 / 1.0
Decoder noise σ_{dec}	3
Self-conditioning probability	0.5
Tangent schedule	$d_{\mathrm{sched}} = 3$
Adaptive timestep sampler	100 bins; EMA 0.9; refresh every 5000 steps; minimum weight 10^{-4}
Generation solver	Euler–Maruyama
Reconstruction CE routing	Detached at $\mathbf{h}$
Diffusion encoder route	Noisy input only
Loss weights	Unit weights

D Evaluation settings

OWT-128 ablations. All ablation models are trained for 100,000 updates and evaluated from a single checkpoint using five generation seeds. Each evaluation generates 128-token sequences with 128 sampling steps and uses the same decoder-noise scale at training and inference. Reported uncertainties are sample standard deviations across generation seeds, rather than across independently trained models. DIV is the product of the unique-to-total 2-, 3-, and 4-gram ratios. In the no-NTP ablation, reconstruction CE remains attached to the Latent Decoder and Latent Encoder.

OWT-1024 evaluation. All methods in the main comparison generate 1024-token sequences. ELF-B uses its 64-step SDE configuration at guidance 3, LangFlow uses 1024 denoiser evaluations, and CoBit uses its reported 128- and 256-step configurations. LDLM, Duo, and MDLM use their reported OWT-1024 operating points. The real-text reference shown in the quality–diversity figure has generative perplexity 14.6 and token entropy 5.436.

E Limitations

Our experiments focus on unconditional English web-text generation; conditional generation is outside the scope of this study. Perplexity and token entropy capture complementary aspects of generation, while semantic diversity, long-range coherence, and factual correctness require additional evaluation. The NTP ablation establishes its role as a useful representation-learning signal, but does not fully characterize the linguistic information encoded in the latent space. The gradient diagnostic measures clean-latent gradients rather than encoder-parameter updates. Finally, models trained on web text may reproduce biases, memorized content, and misleading statements present in the training distribution.

F REPETITION EVALUATION

F.1 SEQUENCE-LEVEL FOUR-GRAM REPETITION

We measure sequence-level repetition using Seq-Rep-4, following Welleck et al. (2020) and Zhang et al. (2026). Let $y = (y_1, \ldots, y_L)$ denote a generated text after whitespace tokenization, and let $\mathcal{U}_4(y)$ be the set of its distinct contiguous four-grams. Seq-Rep-4 is defined as

$$R_4(y) = 1 - \frac{|\mathcal{U}_4(y)|}{L-3} = \frac{\sum_{g \in \mathcal{U}_4(y)} (c_y(g) - 1)}{L-3}, \tag{19}$$

where $c_y(g)$ is the number of occurrences of four-gram g. Thus, the first occurrence of a four-gram does not count as repetition, while each subsequent occurrence does; $R_4(y) = 0$ when all four-grams are unique. Because the metric uses whitespace tokenization, its length L counts word-level tokens rather than GPT-2 tokens.

F.2 REPETITION-CLEAN FRACTION

Following Zhang et al. (2026), we use their human-reference threshold for Seq-Rep-4. It is defined as the 95th percentile over 1000 BBC/XSum reference articles, corresponding to $\tau = 0.0192$ (1.92%). This threshold is therefore an external human-text reference rather than an OpenWebText-specific calibration.

For M generated samples, we report the repetition-clean fraction

$$A(\tau) = \frac{1}{M} \sum_{j=1}^{M} \mathbf{1}\left[R_4\left(y^{(j)}\right) < \tau\right]. \tag{20}$$

A higher value indicates that a larger fraction of generations remains below the reference repetition threshold. Here, "repetition-clean" refers only to this Seq-Rep-4 criterion and does not imply the absence of other forms of repetition or establish text quality more broadly.