

CoDAR: Continuous Diffusion Language Models are More Powerful Than You Think

Junzhe Shen¹, Jieru Zhao³, Ziwei He², Zhouhan Lin^{1,2,3,4,*}

¹LUMIA Lab, School of Artificial Intelligence, Shanghai Jiao Tong University

²Shanghai Innovation Institute

³School of Computer Science, Shanghai Jiao Tong University

⁴Shanghai AI Laboratory

junzheshen@sjtu.edu.cn, lin.zhouhan@gmail.com

*Corresponding author.

Abstract

We study why continuous diffusion language models (DLMs) have lagged behind discrete diffusion approaches despite their appealing continuous generative dynamics. Under a controlled token-recovery study, we identify token rounding, the final projection from denoised embeddings to tokens, as a primary bottleneck. Building on these insights, we propose CoDAR (Continuous Diffusion with Contextual AutoRegressive Decoder), a two-stage framework that keeps diffusion entirely continuous in an embedding space while learning a strong, context-conditional discretizer: an autoregressive Transformer decoder that cross-attends to the denoised embedding sequence and performs contextualized rounding to tokens. Experiments on LM1B and OpenWebText demonstrate that CoDAR substantially improves generation quality over latent diffusion and narrows the gap with strong discrete DLMs, while exposing a simple decoder-temperature knob to navigate the fluency–diversity trade off.

1 Introduction

Continuous diffusion models have achieved remarkable success in domains such as image generation and structured latent spaces, where they demonstrate strong modeling capacity and sample quality. However, their adoption and scalability in natural language processing remain limited. In contrast to continuous media, language is inherently discrete, leading to fundamental challenges when applying continuous generative processes directly to text. While continuous formulations offer theoretical advantages (Zhou et al., 2025; Pynadath et al., 2025), including stronger theoretical expressivity and smoother latent reasoning, continuous space diffusion language models (DLMs) have fallen behind discrete DLMs (He et al., 2023; Li et al., 2025).

A core difficulty lies in the misalignment between continuous diffusion and the discrete nature

of language: diffusion processes typically operate over continuous spaces, whereas linguistic units such as tokens are inherently categorical. Prior works on continuous DLMs have attempted to bridge this gap using a rounding step (Li et al., 2022; Strudel et al., 2023; Lin et al., 2023; Gao et al., 2024) or formulating diffusion in the logit space (Mahabadi et al., 2024; Tae et al., 2025). In parallel, embedding-based latent diffusion methods such as *Latent Diffusion for Language Generation* (Lovelace et al., 2023) learn diffusion in the latent space of a language autoencoder built on a pre-trained encoder–decoder LM. While this provides a principled continuous space that is decodable by construction, it ties the method to encoder–decoder language models and thus narrows the choice of representations.

Beyond the continuous–discrete mismatch, a second difficulty lies in the high dimensionality of latent representations. In latent diffusion for images, recent studies document an optimization dilemma where increasing latent capacity (often via higher-dimensional latents) improves reconstruction but can hinder diffusion training and generation quality unless additional alignment or regularization is introduced (Yao et al., 2025; Lai et al., 2025; Zheng et al., 2026a). Although language differs from vision, analogous pressures can arise when diffusing over high-dimensional embedding sequences. Due to these difficulties, research largely pivoted toward discrete modeling strategies that match the token-level structure of language (Lou et al., 2024; Sahoo et al., 2024) as the field matured.

We argue that the performance gap between continuous DLMs and discrete DLMs is not solely due to the diffusion objective itself, but rather to a *rounding bottleneck*: the challenge of mapping noisy continuous embeddings back to discrete tokens in a large space. Most existing embedding-space DLMs typically rely on a rounding operator to recover tokens (Li et al., 2022; Lin et al., 2023).

Such operators treat each position largely independently and provide limited ability to leverage linguistic context when the denoised embedding is ambiguous or off manifold. Conversely, discrete or simplex-space DLMs avoid explicit rounding but must operate directly in a categorical state space, which changes the learning dynamics and often shifts complexity into the diffusion transition design and sampling procedure (Lou et al., 2024; Sahoo et al., 2024; Tae et al., 2025).

In this work, we analyze the rounding bottleneck of continuous DLMs both theoretically and empirically, and propose CODAR (**C**ontinuous **D**iffusion with **C**ontextual **A**uto**R**egressive **D**ecoder), a novel framework that effectively resolves the two aforementioned problems by *keeping diffusion entirely continuous while learning a powerful, context-conditional rounding module*. Concretely, CODAR factorizes generation into (i) a continuous diffusion process in an embedding space that is favorable for diffusion, and (ii) an autoregressive Transformer decoder that maps the generated continuous states to discrete tokens via cross-attention. This design keeps the diffusion component simple and fully continuous, and delegates the hardest part of decoding to a model class that is known to excel at sequence transduction.

We summarize our main contributions as follows:

- We identify token rounding, especially under low dimensional hidden states, as a principal bottleneck for continuous embedding DLMs theoretically and empirically, showing why pointwise classifiers can be suboptimal and validating this with controlled token-recovery experiments.
- We propose CODAR, a two-stage continuous diffusion language modeling framework: a continuous diffusion generator produces embedding sequences, and an autoregressive Transformer decoder performs contextual rounding back to tokens.
- We show that CODAR improves generation quality over latent diffusion and narrows the gap with strong discrete DLMs.

2 Related Work

2.1 Continuous Diffusion Language Models

Diffusion-LM (Li et al., 2022) models sequences as Gaussian vectors that are iteratively denoised

into word vectors, enabling plug-and-play controllable generation. Building on the embedding-space perspective, Self-conditioned Embedding Diffusion (Strudel et al., 2023) introduces self-conditioning for diffusion over token embeddings and demonstrates competitive generation quality with potential inference efficiency benefits. Diffformer (Gao et al., 2024) analyzes optimization pathologies in embedding diffusion (e.g., embedding collapse and schedule issues) and proposes anchor loss and noise rescaling to stabilize training and improve generation across tasks. GENIE (Lin et al., 2023) frames diffusion for pre-training, using an encoder plus diffusion-based decoder and a continuous paragraph denoise objective to reconstruct coherent paragraphs from corrupted inputs. LD4LG (Lovelace et al., 2023) trains diffusion in the latent space of an encoder-decoder language autoencoder, sampling compact continuous latents that are decoded by a pretrained decoder. More recent variants explore where to diffuse and how to better respect token discreteness: TESS (Mahabadi et al., 2024) performs diffusion on the logit simplex (rather than learned embeddings) with self-conditioning for fully non-autoregressive text-to-text generation. Smoothie (Shabalin et al., 2025) proposes progressively smoothing token embeddings by semantic similarity to combine semantic structure with a more natural decoding process.

2.2 Hybrid Architectures

AR-Diffusion Hybrid A growing line of work explores hybrid diffusion-autoregressive architectures that aim to combine diffusion’s global refinement/parallelism with AR decoding’s fluency and KV-cache-friendly generation. AR-Diffusion (Wu et al., 2023) injects causal, left-to-right structure into diffusion by using a position-dependent (dynamic) number of denoising steps so left tokens “settle” earlier and condition later ones. DGLM (Lovelace et al., 2024) uses a diffusion model to generate continuous semantic proposals (soft prompts) that steer a strong AR LM toward desired attributes. SDLM (Liu et al., 2025b) further “retrofit” pretrained AR LMs by performing diffusion within masked blocks but decoding consecutive subsequences adaptively (via Next Sequence Prediction) to maintain KV-cache compatibility and handle variable uncertainty across the sequence. Moving toward single-model synergy, TiDAR (Liu et al., 2025a) explicitly separates roles by drafting in diffusion while sampling final

outputs autoregressively using structured attention masks to achieve high throughput without sacrificing AR-quality. For reasoning-centric settings, LaDiR (Kang et al., 2026) augments an existing LLM with a VAE-defined latent “thought” space and a latent diffusion model that iteratively refines blockwise reasoning trajectories, enabling more holistic revision than pure AR chain-of-thought. Planner and Executor (Berrayana et al., 2025) studies explicit collaboration where a discrete diffusion model plans and an AR model executes, showing that shifting diffusion to AR communication from text to latent space via a learned projector can markedly improve reasoning accuracy while reducing token-cost.

Continuous-Discrete Hybrid Hybrid continuous-discrete diffusion LMs model tokens and continuous representations jointly, rather than diffusing in only one space. CCDD (Zhou et al., 2025) co-evolves discrete tokens and continuous states in a single joint diffusion, aiming to preserve continuous latent expressivity while improving trainability via explicit discrete structure. CANDI (Pynadath et al., 2025) decouples discrete identity corruption from continuous geometric degradation to avoid a mismatch in effective noise regimes, enabling useful continuous gradients while retaining conditional structure. CADD (Zheng et al., 2026b) augments masked diffusion with informative continuous latents. Compared with these interleaved hybrid processes, our approach keeps diffusion entirely continuous in embedding space and delegates discretization to a separate contextualized decoder.

3 Theoretical Analysis

Let $Y = (Y_1, \dots, Y_L)$ be a length- L token sequence and let $X \in \mathbb{R}^{L \times d}$ denote the denoised continuous sequence produced by the diffusion generator. Any rounding (discretization) procedure is implicitly performing posterior inference:

$$\hat{y} \in \arg \max_y p(y | X). \quad (1)$$

Many embedding-space DLMs implement rounding with a position-wise linear head (Li et al., 2022; Gao et al., 2024; Lin et al., 2023), which corresponds to the approximate factorization $p(y | X) \approx \prod_{i=1}^L p(y_i | X_i)$, treating token recovery as independent classification at each position i .

Entropy and conditional total correlation.

Throughout, $H(\cdot)$ denotes Shannon entropy (in

nats when log is natural). For discrete random variables,

$$H(Y | X) = \mathbb{E}_x \left[- \sum_y p(y | x) \log p(y | x) \right], \quad (2)$$

and similarly $H(Y_i | X_i)$ is the remaining uncertainty of Y_i after observing only the local vector X_i . The *conditional total correlation* (conditional TC) measures residual dependence among $(Y_1, \dots, Y_L)$ given X :

$$\begin{aligned} \text{TC}(Y | X) &= \mathbb{E}_x \left[D_{\text{KL}} \left(p(y | x) \parallel \prod_{i=1}^L p(y_i | x) \right) \right] \\ &= \sum_{i=1}^L H(Y_i | X) - H(Y | X) \geq 0. \end{aligned} \quad (3)$$

Locality gap vs. dependence gap. Because X contains (weakly) more information than X_i , conditioning reduces entropy: $H(Y_i | X) \leq H(Y_i | X_i)$. This yields a useful decomposition:

$$\begin{aligned} &\sum_{i=1}^L H(Y_i | X_i) - H(Y | X) \\ &= \underbrace{\left(\sum_{i=1}^L H(Y_i | X) - H(Y | X) \right)}_{\text{TC}(Y|X)} \\ &\quad + \underbrace{\sum_{i=1}^L \left(H(Y_i | X_i) - H(Y_i | X) \right)}_{\text{locality gap}} \geq \text{TC}(Y | X). \end{aligned} \quad (4)$$

Intuitively, $\text{TC}(Y | X)$ captures *intrinsic sequence coupling* (syntax/semantics, long-range constraints) that remains even if the decoder sees the entire X , whereas the locality gap captures extra uncertainty introduced when a decoder is restricted to per-position evidence X_i rather than the full context X .

Proposition 1 (Optimality gap of pointwise decoding). *Let $\mathcal{D}_{\text{pw}}$ be the set of all decoders that factorize as $\prod_{i=1}^L q_i(y_i | X_i)$, and let $\mathcal{D}_{\text{seq}}$ be the set of all conditional sequence decoders $q(y | X)$. Consider the expected negative log-likelihood (NLL) risk*

$$\mathcal{R}(q) = \mathbb{E}_{(X,Y)} [-\log q(Y | X)]. \quad (5)$$

Then,

$$\begin{aligned}
& \min_{q \in \mathcal{D}_{\text{pw}}} \mathcal{R}(q) - \min_{q \in \mathcal{D}_{\text{seq}}} \mathcal{R}(q) \\
&= \sum_{i=1}^L H(Y_i | X_i) - H(Y | X) \quad (6) \\
&\geq \text{TC}(Y | X) \geq 0.
\end{aligned}$$

Refer to Appendix H for proof.

Interpretation and implications for rounding.

Equation (6) formalizes two reasons why a linear LM head can be brittle for rounding diffusion outputs: (i) *sequence dependence*: the gap vanishes only if $Y_1, \dots, Y_L$ are conditionally independent given X (i.e., $\text{TC}(Y | X) = 0$), which is rarely true for natural language; (ii) *local evidence restriction*: even if a full-context decoder could in principle exploit X to resolve ambiguity, a per-position head that only sees X_i incurs the locality gap in (4). In realistic diffusion sampling, X is an imperfect denoised embedding sequence: it may be off-manifold, slightly inconsistent across positions, or contain structured errors. Such imperfections typically increase both conditional dependence (larger $\text{TC}(Y | X)$) and locality gap, making point-wise decoding strictly suboptimal.

Why increasing d helps but does not eliminate the issue. A larger embedding dimension d increases the capacity of each X_i to encode information about Y_i . Formally,

$$H(Y_i | X_i) = H(Y_i) - I(Y_i; X_i), \quad (7)$$

so improving the mutual information $I(Y_i; X_i)$ (often easier when d is larger) reduces *marginal* ambiguity. However, lowering these marginal ambiguities does not imply that sequence recovery becomes pointwise: the approximation $p(y | X) \approx \prod_i p(y_i | X_i)$ would require that (i) tokens become conditionally independent given the full X , and (ii) local evidence X_i is as informative as the global context X . In general, neither holds. Moreover, even if the clean embedding manifold were perfectly invertible, diffusion-generated X can be slightly off-manifold. In that case, X may be consistent with multiple nearby token sequences. Choosing the one that is globally coherent requires reasoning over the whole sequence, not each position independently. This explains why improving per-position separability (e.g., by increasing d) can improve linear head accuracy, yet still leaves a substantial gap to contextual decoding.

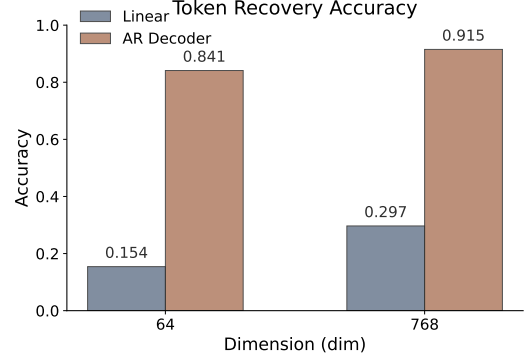


Figure 1: Token recovery rate of point-wise linear classifier and autoregressive Transformer decoder under different sizes of hidden states.

To test whether the theoretical gap in (6) is practically significant, we run a controlled study. We train two families of token decoders to map hidden states h_i back to tokens x_i : (i) a position-wise linear classifier $p(x_i | h_i)$; and (ii) an autoregressive Transformer decoder that predicts x_i conditioned on previously recovered tokens and cross-attends to the full continuous sequence $H = [h_1, \dots, h_L]$. For each position, we select the token with highest predicted probability as the predicted token, and token recovery is computed as the rate at which the predicted token matches the true token. The token recovery accuracy is shown in fig. 1.

Key observation. As shown in fig. 1, the Transformer decoder recovers tokens with high accuracy across both low- and high-dimensional representations (e.g., 0.841 at $d=64$ and 0.915 at $d=768$), while the Linear baseline performs poorly (0.154 and 0.297, respectively). This is consistent with (6): a linear head is constrained to $\mathcal{D}_{\text{pw}}$ and therefore cannot exploit sequence-level coupling (nonzero $\text{TC}(Y | X)$) nor full-context evidence (locality gap), both of which are crucial for reliable rounding when h_i are imperfect.

These results motivate our two-stage approach, where continuous diffusion generation and contextual token rounding are decoupled and separately learned:

- a continuous diffusion model generates a sequence of token embeddings in $\mathbb{R}^{L \times d}$;
- an autoregressive Transformer decoder with cross-attention maps the generated embeddings back to discrete tokens.

This design addresses the previously noted trainability bottleneck by explicitly learning a dedicated

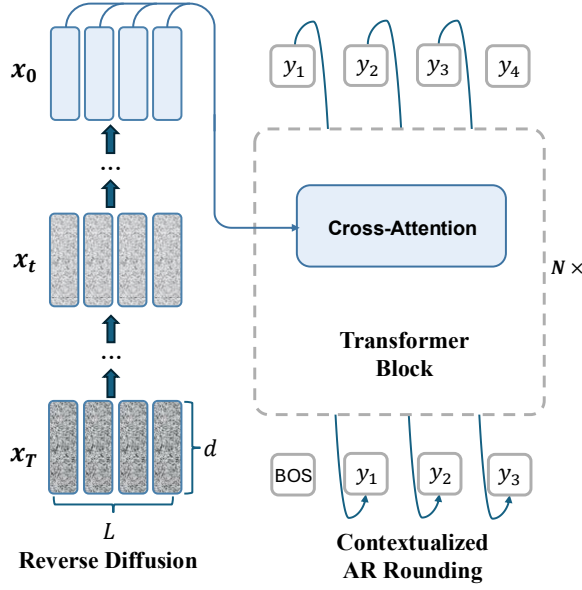


Figure 2: Framework of CODAR . Starting from a noisy latent sequence x_T , a reverse diffusion process progressively denoises the hidden states to x_0 . $x_T, \dots, x_0 \in \mathbb{R}^{L \times d}$, where L denotes the sequence length and d denotes the size of hidden states. After that, an autoregressive Transformer decoder conditions on the denoised x_0 with cross-attention to translate x_0 to discrete tokens $y_1, \dots, y_L$

embedding-to-token decoder, while keeping the diffusion process entirely continuous and allowing free choice of embeddings. In particular, diffusion no longer needs to land exactly on token embeddings at every position; it only needs to generate continuous states that are *decodable under context*, allowing the decoder to leverage linguistic regularities to resolve residual dependence and local ambiguity predicted by (4).

4 Continuous Diffusion with Contextual AutoRegressive Decoder

Let $\mathbf{y} = (y_1, \dots, y_L)$ be a token sequence with vocabulary size $|V|$. A frozen embedding model $E : V^L \rightarrow \mathbb{R}^{L \times d}$ maps tokens to continuous representations $\mathbf{x}_0 = E(\mathbf{y})$. We denote the diffusion state at time $t \in [0, 1]$ as $\mathbf{x}_t \in \mathbb{R}^{L \times d}$. Our denoiser $f_\theta(\mathbf{x}_t, t)$ operates purely in the continuous space, while a separate autoregressive decoder $p_\phi(\mathbf{y} | \hat{\mathbf{x}}_0)$ performs discrete realization. Unless stated otherwise, $E(\cdot)$ is fixed and only (θ, ϕ) are optimized. The framework of CODAR is shown in fig. 2.

4.1 Continuous Diffusion for Embedding Generation

Given a tokenized sequence $\mathbf{y} = (y_1, \dots, y_L)$, we obtain continuous embeddings

$$\mathbf{x}_0 = E(\mathbf{y}) \in \mathbb{R}^{L \times d},$$

where $E(\cdot)$ can be an arbitrary pretrained text embedding model.

We define a variance-preserving (VP) continuous diffusion process (Song et al., 2021) on embeddings by corrupting $\mathbf{x}_0$ with Gaussian noise:

$$\mathbf{x}_t = \alpha(t)\mathbf{x}_0 + \sigma(t)\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $t \sim \mathcal{U}(0, 1)$, and $\alpha(t), \sigma(t)$ are a noise schedule satisfying $\alpha(t)^2 + \sigma(t)^2 = 1$. We use cosine schedule proposed by Nichol and Dhariwal (2021).

We use the velocity parameterization (Salimans and Ho, 2022) because it interpolates between predicting noise and predicting data, and has been shown to improve stability—especially when sampling with few steps—relative to direct ϵ -prediction. Concretely, we train the denoiser to predict the *velocity*:

$$\mathbf{v}_t \triangleq \alpha(t)\epsilon - \sigma(t)\mathbf{x}_0,$$

We parameterize the denoiser as

$$\hat{\mathbf{v}}_\theta = f_\theta(\mathbf{x}_t, t),$$

and recover estimates of $\mathbf{x}_0$ and ϵ via

$$\hat{\mathbf{x}}_0 = \alpha(t)\mathbf{x}_t - \sigma(t)\hat{\mathbf{v}}_\theta, \quad \hat{\epsilon} = \sigma(t)\mathbf{x}_t + \alpha(t)\hat{\mathbf{v}}_\theta.$$

Under the VP formulation with v-prediction, the denoiser’s objective is to align its output $\hat{\mathbf{v}}_\theta$ with the ground-truth velocity $\mathbf{v}_t = \alpha(t)\epsilon - \sigma(t)\mathbf{x}_0$. We optimize the following velocity prediction loss:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(0,1), \mathbf{x}_0, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[w(t) \cdot \|f_\theta(\mathbf{x}_t, t) - \mathbf{v}_t\|_2^2 \right],$$

where $w(t)$ is a time-dependent weighting function that can improve optimization stability (for example, constant weighting or SNR-based schemes used in image diffusion literature (Ho et al., 2020; Nichol and Dhariwal, 2021; Karras et al., 2022)).

4.2 Contextualized Rounding with AR Decoder

To map the generated continuous embeddings back to text, we employ a Transformer decoder p_ϕ that uses cross-attention over the denoised embedding

sequence. Formally, the conditional likelihood of a token sequence $\mathbf{y} = (y_1, \dots, y_L)$ given the recovered embeddings is factorized as

$$p_\phi(\mathbf{y} \mid \hat{\mathbf{x}}_0) = \prod_{i=1}^L p_\phi(y_i \mid y_{<i}, \hat{\mathbf{x}}_0),$$

This two-stage strategy first runs diffusion in a continuous latent space, and then decodes the result into discrete text. This is similar to the latent diffusion for language setup in LD4LG (Lovelace et al., 2023), which combines a latent model with a encoder-decoder language model to produce text from latent vectors. Unlike LD4LG, we do not need to rely on an encoder-decoder language model and can use off-the-shelf, state-of-the-art text embedding models instead.

Under teacher forcing, the decoder is trained to reconstruct the ground-truth tokens from the (ideally denoised) continuous embeddings with the standard cross-entropy objective:

$$\mathcal{L}(\phi; \theta) = \mathbb{E} \left[- \sum_{i=1}^L \log p_\phi(y_i \mid y_{<i}, \mathbf{x}_0) \right].$$

4.3 Inference

At inference time, generation proceeds in two stages. First, we sample a continuous embedding sequence by running the learned reverse diffusion process: starting from Gaussian noise $\mathbf{x}_{t=1} \sim \mathcal{N}(0, I)$, we iteratively apply the denoiser $f_\theta(\mathbf{x}_t, t)$ (with a chosen numerical solver) to progressively remove noise and obtain a final denoised embedding sequence $\mathbf{x}_0 \in \mathbb{R}^{L \times d}$. Second, we translate these hidden states into tokens using the autoregressive Transformer decoder p_ϕ conditioning on $\hat{\mathbf{x}}_0$ via cross-attention. The decoder samples $y_i \sim p_\phi(y_i \mid y_{<i}, \hat{\mathbf{x}}_0)$ until an end-of-sequence token is produced (or a length limit is reached). In this way, diffusion handles global continuous-sequence generation, while the AR decoder performs discrete token realization from the generated hidden states.

5 Experiment

5.1 Setup

We evaluate the language modeling capability of our model via unconditional text generation, focusing on the quality of generated samples. We compare CODAR against a latent diffusion language model (LD4LG (Lovelace et al., 2023)) and two

strong discrete diffusion baselines, MDLM (Sahoo et al., 2024) and SEDD (Lou et al., 2024). For all comparisons, we report generative perplexity (Gen. PPL) as a proxy for fluency (lower is better), and an n-gram-based diversity metric as a proxy for lexical variety (higher is better) following previous work (Lovelace et al., 2023).

To compare against the latent diffusion baseline LD4LG, we train CODAR and LD4LG on the One Billion Word Benchmark (LM1B, English, Apache-2.0) (Chelba et al., 2014) using the standard data split. We train models using sequence length $L = 128$ with sentence packing. We train both models for 250k steps. For LD4LG, we use the BART variant and follow the baseline’s standard autoencoder setup and training procedure as reported in the original work.

For the comparison with discrete baselines, we train all models on the OpenWebText (Gokaslan and Cohen, 2019) (Primarily English, extracted from Reddit submissions, cc0-1.0) dataset. Since OpenWebText is substantially larger and more diverse than LM1B, it provides a stronger stress test for unconditional generation quality. We train CODAR for 250k steps using context length $L = 512$ with sequence packing. For comparison, we re-train MDLM and SEDD following prior practice (Sahoo et al., 2024; Lou et al., 2024; Sahoo et al., 2025) using standard training iterations (1M steps) but matching our context length of 512. We use the Qwen2 tokenizer (Zhang et al., 2025) for CODAR and for baselines when possible. LD4LG cannot use the Qwen2 tokenizer because its BART-based autoencoder is tied to the BART vocabulary. MDLM does not work well with Qwen2 tokenizer (resulting in very high generative perplexity), which is consistent with observations in Zhou et al. (2025). We therefore use the GPT-2 tokenizer for MDLM (marked as * in table 1 and 4).

Evaluation metrics. For each method, we generate 1,000 unconditional samples and compute mean generative perplexity (Gen. PPL) using GPT-2 Large (Radford et al., 2019) to measure fluency. To quantify diversity, we use the metric used in LD4LG, defined as the product of distinct n-gram ratios for $n \in \{2, 3, 4\}$:

$$\text{Div} = \prod_{n=2}^4 \frac{|\text{unique } n\text{-grams}(\{\mathbf{w}_i\})|}{|\text{total } n\text{-grams}(\{\mathbf{w}_i\})|},$$

where $\{\mathbf{w}_i\}$ is a set of generated samples.

To contextualize the generative perplexity and diversity scores, we report three reference points.

- Training set: Text drawn directly from the *training set*, serving as a data-distribution anchor under our evaluation protocol.
- Recovered training set: obtained by decoding ground-truth text embeddings, reflecting reconstruction error induced by the decoder.
- Decoded noise: Decoding pure Gaussian noise with our contextualized decoder, which exhibits low Gen. PPL but extremely low diversity, indicative of a degenerate mode of highly repetitive text.

These probes show that (i) the decoder is capable of producing fluent text, but (ii) a meaningful generative model must match *both* fluency and diversity, not fluency alone.

Training Details For the diffusion model, we use the same model architecture as MDLM(Sahoo et al., 2024). For the contextualized autoregressive decoder, we use the same model architecture as GPT2-small (Radford et al., 2019) but with additional cross attention. Model sizes and decoder training details are provided in appendix A. We choose Qwen3-Embedding (Zhang et al., 2025) as CoDAR’s pretrained embedding as it supports custom dimensions. We set the dimension of embeddings to 64 unless otherwise specified. All embeddings are normalized following Rombach et al. (2022).

5.2 Main Results

On OpenWebText unconditional generation (table 1), CoDAR spans a smooth fluency–diversity frontier by varying the decoder temperature while keeping the model and procedure fixed. At low temperatures, CoDAR’s Gen. PPL drops to 49.05 ($T=0.00$) and 53.50 ($T=0.25$), while diversity remains non-trivial and far from the collapse of decoded noise (diversity 0.0380). As temperature increases, diversity rises monotonically ($0.2697 \rightarrow 0.3741 \rightarrow 0.4870$ for $T=0.50, 0.75, 1.00$) with a corresponding increase in Gen. PPL ($69.04 \rightarrow 102.05 \rightarrow 160.39$), forming a clear Pareto trade-off. Crucially, at $T=1.00$ CoDAR reaches diversity 0.4870, slightly exceeding MDLM (0.4784) and SEDD (0.4742), showing that CoDAR provides a controllable fluency–diversity trade-off and can match the high-diversity regime of discrete

Table 1: Unconditional generation on OpenWebText. We compare CoDAR to discrete baselines (MDLM and SEDD) using 1,000 generated samples per method, reporting generative perplexity and an n-gram–based diversity score. We also include three diagnostic reference points: training set text, decoded ground-truth embeddings (“recovered training set”), and decoded Gaussian noise, to contextualize the metrics. We use 250 sampling steps for all runs while varying the decoder temperature for CoDAR.

Model	Gen. PPL(↓)	Diversity(↑)
Training set	16.75	0.2191
Recovered training set	25.07	0.2282
Decoded noise	14.24	0.0380
MDLM*	123.73	0.4784
SEDD	129.57	0.4742
CoDAR ($T = 0.00$)	49.05	0.1868
CoDAR ($T = 0.25$)	53.50	0.2139
CoDAR ($T = 0.50$)	69.04	0.2697
CoDAR ($T = 0.75$)	102.05	0.3741
CoDAR ($T = 1.00$)	160.39	0.4870

Table 2: Unconditional generation on LM1b.

Model	Gen. PPL(↓)	Diversity(↑)
LD4LG	167.47	0.5797
CoDAR	106.93	0.3282

diffusion baselines. On LM1B unconditional generation, we reuse the decoder trained with OpenWebText and the decoder’s temperature is set to 1. The results are shown in table 2, CoDAR significantly outperforms LD4LG in terms of fluency while maintaining nontrivial diversity.

5.3 Ablations

5.3.1 Sampler and Sampling Steps

As the diffusion process of CoDAR is fully continuous, we can leverage higher-order numerical solvers to improve few-step sampling. We compare standard ancestral sampling to DPM-Solver (Lu et al., 2022) while varying the number of sampling steps from 250 down to 25. We fix the decoder temperature to 1.0 for all runs. The results are shown in table 3.

Across most step budgets, DPM-Solver improves fluency (lower Gen. PPL) over ancestral sampling while maintaining high diversity. At 250

steps, DPM-Solver reduces Gen. PPL from 160.39 (ancestral) to 145.94 and increases diversity from 0.4870 to 0.5027. The improvement is also clear at 100 steps, where DPM-Solver achieves 156.23 Gen. PPL versus 179.16 for ancestral sampling, while keeping diversity high (0.4891). At 50 steps, DPM-Solver improves Gen. PPL from 195.79 to 179.88 and raises diversity from 0.4798 to 0.5020. At the most aggressive 25-step setting, ancestral sampling has slightly lower Gen. PPL (195.33 vs. 199.33), but DPM-Solver provides substantially higher diversity (0.4895 vs. 0.4100). Overall, these results show that advanced solvers make CoDAR effective for fast generation, typically improving sample quality while preserving or improving diversity.

Table 3: Solver ablation for CoDAR . We compare ancestral sampling to DPM-Solver for unconditional generation across sampling budgets with decoder temperature set to 1.0.

Solver	Steps	Gen. PPL(↓)	Div.(↑)
Ancestral	25	195.33	0.4100
	50	195.79	0.4798
	100	179.16	0.4955
	250	160.39	0.4870
DPM-Solver	25	199.33	0.4895
	50	179.88	0.5020
	100	156.23	0.4891
	250	145.94	0.5027

Now we compare CoDAR with DPM-Solver against discrete baselines (MDLM/SEDD) in few-step generation. Table 4 reports Gen. PPL and diversity at matched sampling budgets, with decoder temperature fixed to $T = 1$ for CoDAR .

At the most aggressive budget of 25 steps, CoDAR achieves the best fluency among all methods (199.33 Gen. PPL), while retaining strong diversity (0.4895). As the step budget increases, MDLM/SEDD become more fluent, but CoDAR remains competitive and stays in the same diversity regime: at 50–250 steps, CoDAR yields diversity around 0.49–0.50, comparable to or higher than the baselines. For instance, at 250 steps, CoDAR attains 0.5027 diversity versus 0.4784/0.4742 for MDLM/SEDD, though with higher Gen. PPL. At intermediate budgets, CoDAR reaches 179.88 Gen. PPL with 0.5020 diversity at 50 steps and

Table 4: Few-step unconditional generation on OpenWebText. We compare our model sampled with DPM-Solver (decoder temperature $T = 1$) against discrete diffusion baselines under matched step budgets.

Steps	Model	Gen. PPL(↓)	Div.(↑)
25	MDLM*	232.78	0.5287
	SEDD	221.63	0.5171
	CoDAR	199.33	0.4895
50	MDLM*	165.71	0.5046
	SEDD	164.24	0.492
	CoDAR	179.88	0.5020
100	MDLM*	137.62	0.4877
	SEDD	143.19	0.481
	CoDAR	156.23	0.4891
250	MDLM*	123.73	0.4784
	SEDD	131.96	0.4742
	CoDAR	145.94	0.5027

156.23 Gen. PPL with 0.4891 diversity at 100 steps, maintaining competitive diversity while trailing the discrete baselines in fluency.

Overall, these results highlight two takeaways: (i) thanks to the continuous formulation, CoDAR can exploit advanced solvers to enable high-quality fast sampling, and (ii) CoDAR is especially strong in the very low-step regime, achieving the best fluency at 25 steps while maintaining diversity comparable to discrete diffusion baselines; at larger budgets, it trades some fluency for higher diversity.

5.3.2 Choice of Decoder

We study the effect of decoder architecture by comparing a linear head, an autoregressive Transformer decoder, and a bidirectional Transformer decoder (table 5). As noted in section 3, linear heads perform poorly in token recovery; we examine whether this limitation extends to text generation. Although the linear decoder attains a slightly lower Gen. PPL

Table 5: Ablation on choice of decoder

Decoder	Gen. PPL(↓)	Div.(↑)
Linear	153.44	0.1238
Bidirectional Transformer	390.15	0.7636
CoDAR	160.39	0.4870

(153.44 vs. 160.39), it suffers from extremely low diversity (0.1238), indicating severe repetition and mode collapse, which substantially degrades text quality in practice. In contrast, the autoregressive Transformer decoder achieves much higher diversity (0.4870) while maintaining competitive perplexity, highlighting the importance of contextual modeling for high-quality decoding.

We further test whether autoregressive factorization is required for contextual rounding by evaluating a non-autoregressive alternative: a bidirectional Transformer decoder that predicts all tokens in parallel from the generated continuous embeddings. As shown in table 5, this parallel decoder obtains substantially worse Gen. PPL than CoDAR (390.15 vs. 160.39), indicating much less fluent text. Although the bidirectional decoder has substantial contextual capacity, it performs much worse than the autoregressive decoder. Thus, the gain is not explained by parameter count or contextual capacity alone; autoregressive factorization is important for contextual token realization. Together, these ablations show that stronger decoder capacity alone is insufficient: linear decoders lack sufficient expressive capacity, while bidirectional parallel decoding lacks the autoregressive factorization needed for fluent token realization from long sequences of continuous representations.

The noisy-embedding stress test in table 7 and fig. 3 further shows that the decoder relies on informative diffusion representations: as the conditioning signal-to-noise ratio decreases, both token recovery and generation quality degrade sharply, ruling out the decoder’s language prior alone as the source of the improvement.

5.4 Throughput Comparison with MDLM

We additionally report wall-clock inference throughput to complement the sampling-step comparisons in the main paper. We compare CoDAR and MDLM under the same generation budget: both use 250 diffusion steps and batch size 100, and we only measure throughput without changing the generation configuration that determines sample quality. As shown in table 6, CoDAR reaches 1129.24 tokens/second, compared with 570.47 tokens/second for MDLM, corresponding to a $1.98\times$ speedup. On a per-sample basis, CoDAR spends 0.29s in the diffusion phase and 0.15s in the autoregressive decoder, for a total of 0.44s, whereas MDLM takes 0.89s per sample.

The decoder therefore adds measurable over-

Table 6: Inference throughput comparison with MDLM. CoDAR’s total latency is the sum of its diffusion and decoder phases.

Model / Phase	Time / sample (s) ↓	Throughput (tok/s) ↑	Relative speed ↑
CoDAR (diffusion)	0.29	—	—
CoDAR (decoder)	0.15	—	—
CoDAR total	0.44	1129.24	$1.98\times$
MDLM	0.89	570.47	$1.00\times$

head, but it does not dominate inference time. The main speed advantage comes from where diffusion is performed: CoDAR diffuses in compact continuous hidden states (64 dimensions in our setup), while MDLM performs diffusion in token space. Through profiling, we found that MDLM is dominated by token-space indexing and transfer overhead, with `cudaMemcpyAsync` accounting for 56.9% (40.5s) and `cudaStreamSynchronize` accounting for 30.5% (21.7s) among the major profiled operators. Thus, even with an additional autoregressive decoding stage, CoDAR can be faster in wall-clock throughput because each diffusion step is substantially cheaper.

6 Conclusion

In this work, we argue that the performance gap between continuous and discrete diffusion language models is partly attributable to a token-rounding bottleneck. Through theoretical analysis and controlled token-recovery experiments, we show that rounding is inherently sequence dependent and that pointwise linear heads are provably suboptimal for mapping continuous representations back to tokens. Building on this insight, we propose CoDAR, a two-stage framework that performs diffusion entirely in an embedding space while using a context-aware autoregressive Transformer decoder to realize discrete tokens. Experiments on LM1B and OpenWebText demonstrate that CoDAR substantially improves over latent diffusion baselines and narrows the gap with strong discrete DLMS, while exposing a simple decoder-temperature mechanism to smoothly trade off fluency and diversity. Together, these results suggest that continuous diffusion and discrete language modeling are complementary rather than competing, and that treating rounding as a contextual problem unlocks much of the unrealized potential of continuous diffusion language models.

7 Limitations

Our current study has two main limitations. First, the model scale remains limited: our experiments are conducted with a relatively small diffusion backbone and decoder, so the scaling behavior of the proposed framework has not yet been fully explored. Second, training speed is constrained by the need to obtain continuous embeddings from pre-trained models before training the diffusion model. This embedding-extraction stage introduces additional preprocessing cost and limits the overall training throughput. In addition, our two-stage training does not jointly optimize the diffusion backbone and autoregressive decoder end to end; future work should investigate joint training to better align continuous representation generation with contextual token realization.

8 Ethical Considerations

CODAR is a generative language model and may produce harmful, biased, misleading, or private content without proper alignment and safety filtering. Since our contextual decoder improves the fluency of generated text, it could also make unsafe outputs more convincing if misused.

Our work focuses on modeling and analysis rather than deployment. Practical systems based on this framework should include data filtering, toxicity and bias evaluation, privacy checks, and output moderation. The model may also inherit biases or memorized content from pretrained components and training data, which should be carefully evaluated before real-world use.

We use only existing, publicly available benchmark datasets, OpenWebText and LM1B, and do not collect new user data or release any new dataset containing raw text. The datasets are used as intended. OpenWebText is derived from web pages linked from Reddit submissions, while LM1B is based on WMT 2011 News Crawl. Because these corpora are web/news-derived, they may contain personal names, identifying information, or offensive content. We do not attempt to identify individuals, do not report individual examples, and do not redistribute raw dataset contents. We did not perform additional anonymization beyond the preprocessing of the original dataset releases.

References

- Lina Berrayana, Ahmed Heakl, Muhammad Abdullah Sohail, Thomas Hofmann, Salman Khan, and Wei Chen. 2025. *Planner and executor: Collaboration between discrete diffusion and autoregressive models in reasoning*. Preprint, arXiv:2510.15244.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. *One billion word benchmark for measuring progress in statistical language modeling*. In *Inter-speech 2014*, pages 2635–2639.
- Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2024. Empowering diffusion models on the embedding space for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4664–4683.
- Aaron Gokaslan and Vanya Cohen. 2019. Open-webtext corpus. <http://Skylion007.github.io/OpenWebTextCorpus>.
- Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2023. *DiffusionBERT: Improving generative masked language models with diffusion models*. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4521–4534, Toronto, Canada. Association for Computational Linguistics.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.
- Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yian Ma, and Lianhui Qin. 2026. *Ladir: Latent diffusion enhances LLMs for text reasoning*. In *The Fourteenth International Conference on Learning Representations*.
- Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577.
- Bolin Lai, Xudong Wang, Saketh Rambhatla, James M Rehg, Zsolt Kira, Rohit Girdhar, and Ishan Misra. 2025. Toward diffusible high-dimensional latent spaces: A frequency perspective. *arXiv preprint arXiv:2511.22249*.
- Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. 2025. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.
- Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022. *Diffusion-LM improves controllable text generation*. In *Advances in Neural Information Processing Systems*.

- Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. 2023. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise. In *International Conference on Machine Learning*, pages 21051–21064. PMLR.
- Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. 2025a. [Tidar: Think in diffusion, talk in autoregression](#). *Preprint*, arXiv:2511.08923.
- Yangzhou Liu, Yue Cao, Hao Li, Gen Luo, Zhe Chen, Weiyun Wang, Xiaobo Liang, Bqing Qi, Lijun Wu, Changyao Tian, Yanting Zhang, Yuqiang Li, Tong Lu, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025b. [Sequential diffusion language models](#). *Preprint*, arXiv:2509.24007.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, pages 32819–32848. PMLR.
- Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Weinberger. 2024. Diffusion guided language modeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14936–14952.
- Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. 2023. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025.
- Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787.
- Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E Peters, and Arman Cohan. 2024. Tess: Text-to-text self-conditioned simplex diffusion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2347–2361.
- Alexander Quinn Nichol and Prafulla Dhariwal. 2021. [Improved denoising diffusion probabilistic models](#). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR.
- Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. 2025. Candi: Hybrid discrete-continuous diffusion models. *arXiv preprint arXiv:2510.22510*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. 2025. [The diffusion duality](#). In *Forty-second International Conference on Machine Learning*.
- Tim Salimans and Jonathan Ho. 2022. [Progressive distillation for fast sampling of diffusion models](#). In *International Conference on Learning Representations*.
- Alexander Shabalin, Viacheslav Meshchaninov, and Dmitry Vetrov. 2025. Smoothie: Smoothing diffusion on token embeddings for text generation. *arXiv preprint arXiv:2505.18853*.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. [Score-based generative modeling through stochastic differential equations](#). In *International Conference on Learning Representations*.
- Robin Strudel, Corentin Tallec, Florent Althé, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Sussman Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. 2023. [Self-conditioned embedding diffusion for text generation](#).
- Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. 2025. [TESS 2: A large-scale generalist diffusion language model](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21171–21188, Vienna, Austria. Association for Computational Linguistics.
- Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, yelong shen, Jian Jiao, Juntao Li, zhongyu wei, Jian Guo, Nan Duan, and Weizhu Chen. 2023. [AR-diffusion: Auto-regressive diffusion model for text generation](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Jingfeng Yao, Bin Yang, and Xinggang Wang. 2025. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15703–15712.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren

- Zhou. 2025. [Qwen3 embedding: Advancing text embedding and reranking through foundation models](#). *Preprint*, arXiv:2506.05176.
- Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. 2026a. [Diffusion transformers with representation autoencoders](#). In *The Fourteenth International Conference on Learning Representations*.
- Huangjie Zheng, Shansan Gong, Ruixiang ZHANG, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. 2026b. [Continuously augmented discrete diffusion model for categorical generative modeling](#). In *The Fourteenth International Conference on Learning Representations*.
- Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. 2025. [Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner](#). *Preprint*, arXiv:2510.03206.

A More Experimental Details

The diffusion backbone has approximately 88M non-embedding parameters, and the contextualized decoder has approximately 100M parameters. The decoder is trained for 1 epoch on OpenWebText with a batch size of 512. The Adam optimizer was used with a learning rate of 1.0×10^{-3} , weight decay of 1.0×10^{-1} , $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at a maximum norm of 1.0. The learning rate followed a cosine annealing schedule, with a warmup ratio of 5%. For the DiT training, we also use a batch size of 512 and Adam optimizer. The learning rate is set to 4.0×10^{-4} with a constant learning rate schedule. The weight decay is set to 0.02, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at a maximum norm of 1.0. The number of warmup steps is set to 10000.

B Noisy-Embedding Stress Test

Table 7: Noisy-embedding stress test. We corrupt ground-truth embeddings according to the diffusion noise schedule and decode them with the trained autoregressive decoder. As SNR decreases, token recovery and generation quality degrade rapidly, indicating that the decoder depends on informative continuous embeddings rather than acting as a standalone language model.

t	α_t	σ_t	SNR	Token Acc.	Gen. PPL
0	1.0	0	-	0.87	72.76
0.050	1.0	0.089	124.11	0.78	97.99
0.10	0.99	0.17	34.83	0.56	134.69
0.15	0.97	0.24	15.87	0.37	188.97
0.20	0.95	0.32	8.87	0.22	320.58
0.25	0.92	0.39	5.54	0.10	563.13
0.30	0.89	0.46	3.69	0.04	785.28
0.35	0.85	0.53	2.57	0.01	1192.19
0.40	0.80	0.59	1.84	0.01	1281.70
0.50	0.70	0.71	0.98	≈ 0.00	1864.09
0.60	0.58	0.81	0.52	≈ 0.00	2155.97
0.70	0.45	0.89	0.25	≈ 0.00	1984.34
0.80	0.31	0.95	0.10	≈ 0.00	2303.47
0.90	0.16	0.99	0.02	≈ 0.00	2432.56
0.99	0.01	1.0	0.00	≈ 0.00	2565.16

To test whether the decoder uses the continuous representations rather than simply behaving as an unconditional language model, we corrupt clean token embeddings using the same forward noise schedule as the diffusion model and then feed the noisy embeddings directly to the trained decoder. This produces inputs of the form $\alpha_t \mathbf{e} + \sigma_t \epsilon$, where larger t corresponds to lower signal-to-noise ratio (SNR). The results in table 7 and fig. 3 show a

clear monotonic degradation as the conditioning embeddings become less informative. At $t = 0.1$, the SNR is still 34.83, token accuracy drops to 0.560, and Gen. PPL is at 134.69. By $t = 0.2$, SNR falls to 8.87, token accuracy drops to 0.221, and Gen. PPL worsens to 320.58; at $t = 0.25$, token accuracy further drops to 0.100 and Gen. PPL reaches 563.13. Once SNR falls below 1, around $t \geq 0.5$, token recovery is essentially zero and Gen. PPL exceeds 1800. These results demonstrate that CODAR relies critically on informative continuous embeddings, and that the autoregressive decoder was not acting as a strong standalone language model.

C Conditional Generation

We also evaluate zero-shot conditional generation to test whether CODAR can transfer beyond unconditional generation. Following the SEDD conditional-generation setting, we report MAUVE in table 8. CODAR obtains 0.342 MAUVE, slightly above SSD-LM (0.312), but below SEDD (0.957). These results suggest that CODAR can be applied to conditional generation, while also indicating substantial room for improvement. A complete treatment of conditional generation in continuous diffusion language models is beyond the scope of this paper, since it involves a broad design space of conditioning mechanisms and downstream tasks.

The conditional generation procedure is:

1. Given a text prefix, map the prefix tokens into the continuous embedding space.
2. During reverse diffusion, keep the prefix embeddings fixed so that they serve as the conditioning context.
3. Sample the remaining suffix embeddings with the diffusion model while conditioning on the frozen prefix embeddings.
4. Decode the full embedding sequence with the autoregressive decoder, conditioning the decoder on both the generated continuous embeddings and the text prefix, to obtain the final continuation.

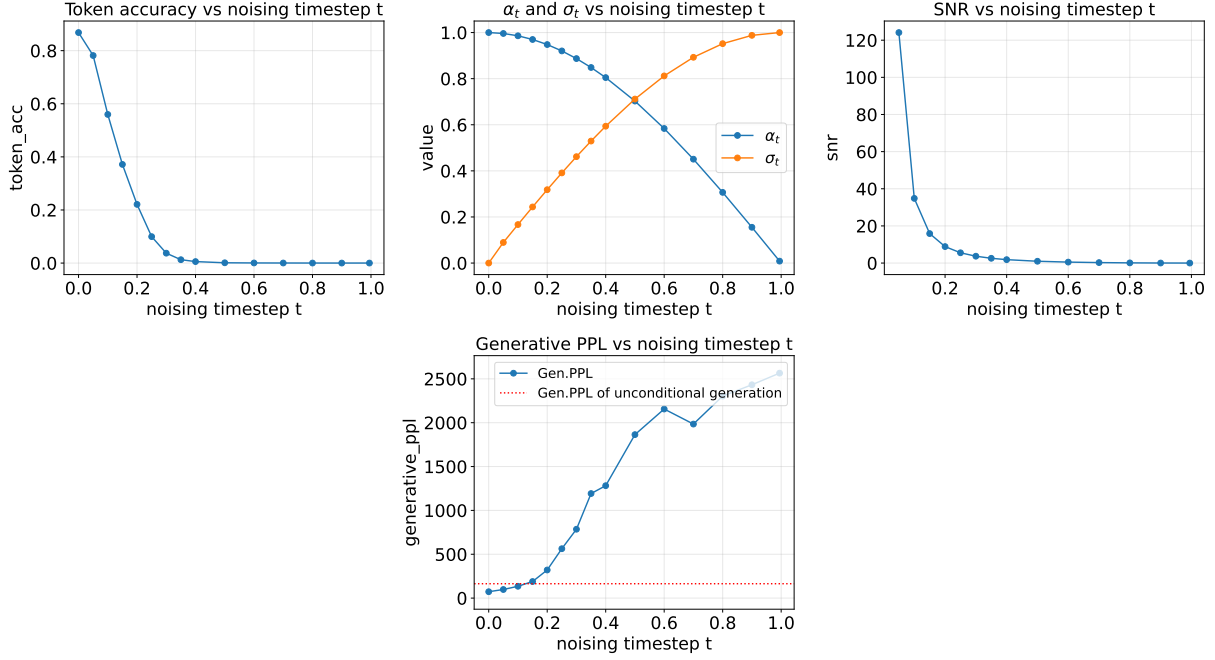


Figure 3: Visualization of table 7

Table 8: Zero-shot conditional generation results measured by MAUVE. Higher is better.

Model	MAUVE($\uparrow$)
SSD-LM	0.312
CoDAR	0.342
SEDD	0.957

D Hidden State Dimension Ablation

Table 9: Ablation on dimension of hidden states.

Hidden Dim.	Solver	Gen. PPL($\downarrow$)	Div.($\uparrow$)
768	Ancestral	523.07	0.5764
	DPM-Solver	546.10	0.6212
256	Ancestral	294.42	0.5056
	DPM-Solver	300.01	0.5475
64	Ancestral	160.39	0.4870
	DPM-Solver	145.94	0.5027

We study the effect of the hidden state dimension on the overall generation quality. Specifically, we vary $d \in \{64, 256, 768\}$ while keeping the decoder temperature to 1. As shown in table 9, increasing the hidden dimension does not translate into better text quality. While larger hidden states generally provide the decoder with higher representational

capacity, they hinder the diffusion process (Yao et al., 2025; Lai et al., 2025). This leads to a degradation in overall text quality: the generative perplexity increases substantially from 160.39 at $d = 64$ to 294.42 at $d = 256$, and further to 523.07 at $d = 768$. Interestingly, even the DPM-Solver, which is designed to handle diffusion more efficiently, struggles with the increased complexity brought by higher hidden dimensions. As the hidden state dimension grows, the DPM-Solver’s generative perplexity surpasses that of the ancestral sampler, indicating that the solver cannot mitigate the added difficulty of a larger state space.

E Decoder Depth Ablation

A 6-layer autoregressive decoder achieves 106.67 Gen. PPL and 0.3913 diversity. In contrast, the 1- and 2-layer variants are too weak and produce repetitive generations, indicating that sufficient decoder depth is necessary for effective contextual token realization.

F Training Cost and Model Scale

Training CODAR requires two components: the continuous diffusion model and the autoregressive decoder. On a single-node A800 setup, the diffusion model takes 155 hours to train and the decoder takes 31 hours.

G Future Work

An important direction is joint or end-to-end training. In the current two-stage implementation, the diffusion model and the decoder are trained separately: the diffusion model learns to generate continuous representations, while the decoder is trained afterward to map continuous embeddings to discrete text. This decomposition makes the problem modular and easier to analyze, but it may be suboptimal because the diffusion backbone is not directly optimized for the downstream autoregressive decoding objective. End-to-end training could allow the diffusion model to produce representations that are more aligned with the decoder’s token-realization behavior, potentially improving fluency, controllability, and robustness. At the same time, such training introduces nontrivial challenges, including memory cost, optimization stability, gradient flow through long denoising trajectories, and the need to balance diffusion losses with sequence-level decoding losses. We therefore leave joint training of the continuous diffusion model and contextual decoder as promising future work.

H Proof

Detailed proof of Proposition 1. Let $p(x, y)$ denote the true joint distribution of (X, Y) , and let Δ be the probability simplex over $\mathcal{V}^L$ (all length- L token sequences). We assume each candidate decoder specifies, for p -a.e. x , a valid conditional distribution $q_x \in \Delta$, with $q_x(y) = q(y | x)$.

Proof. We prove the two minimizers and then the lower bound by conditional total correlation.

Step 1: Bayes-optimal decoder over $\mathcal{D}_{\text{seq}}$. Fix any conditional distribution $q(y | x)$. The expected NLL risk can be rewritten by conditioning on X :

$$\begin{aligned} \mathcal{R}(q) &= \mathbb{E}_X \left[\mathbb{E}_{Y|X} [-\log q(Y | X)] \right] \\ &= \mathbb{E}_X \left[\sum_y p(y | X) (-\log q(y | X)) \right]. \end{aligned} \quad (8)$$

For each fixed value x , define $p_x(y) = p(y | x)$ and $q_x(y) = q(y | x)$. The *cross-entropy* between these two distributions is

$$H(p_x, q_x) := - \sum_y p_x(y) \log q_x(y). \quad (9)$$

A standard identity (cross-entropy decomposition) states that for any two distributions P, Q on the

same space,

$$H(P, Q) = H(P) + D_{\text{KL}}(P \| Q), \quad (10)$$

hence, applying it pointwise at each x yields

$$\begin{aligned} & - \sum_y p(y | x) \log q(y | x) \\ &= H(p_x) + D_{\text{KL}}(p_x \| q_x). \end{aligned} \quad (11)$$

Taking expectation over X gives

$$\mathcal{R}(q) = \underbrace{\mathbb{E}_X [H(p_{Y|X})]}_{H(Y|X)} + \mathbb{E}_X [D_{\text{KL}}(p_{Y|X} \| q_{Y|X})]. \quad (12)$$

Since KL divergence is nonnegative and equals 0 iff the two distributions agree almost surely, the second term in (12) is minimized (to 0) by choosing

$$q^*(y | x) = p(y | x) \quad \text{for } p\text{-a.e. } x. \quad (13)$$

Therefore,

$$\min_{q \in \mathcal{D}_{\text{seq}}} \mathcal{R}(q) = H(Y | X). \quad (14)$$

(These facts follow from the cross-entropy/KL identity and nonnegativity of KL.)

Step 2: Bayes-optimal decoder over $\mathcal{D}_{\text{pw}}$. Now restrict to pointwise-factorized decoders

$$q(y | x) = \prod_{i=1}^L q_i(y_i | x_i), \quad x = (x_1, \dots, x_L). \quad (15)$$

Then the log-likelihood separates:

$$- \log q(Y | X) = - \sum_{i=1}^L \log q_i(Y_i | X_i), \quad (16)$$

and by linearity of expectation,

$$\mathcal{R}(q) = \sum_{i=1}^L \mathbb{E} [-\log q_i(Y_i | X_i)]. \quad (17)$$

For each position i , the inner expectation depends only on the joint law of (X_i, Y_i) . Repeating the same cross-entropy decomposition as in Step 1 but for the conditional distribution $p(y_i | x_i)$ gives, for any conditional decoder $q_i(y_i | x_i)$,

$$\begin{aligned} & \mathbb{E} [-\log q_i(Y_i | X_i)] \\ &= H(Y_i | X_i) + \mathbb{E}_{X_i} [D_{\text{KL}}(p_{Y_i|X_i} \| q_{i,Y_i|X_i})], \end{aligned} \quad (18)$$

where $p_{Y_i|X_i}$ is the true conditional law of Y_i given X_i , and $q_{i,Y_i|X_i}$ is the corresponding conditional law induced by q_i . The KL term is minimized to 0 iff

$$q_i^*(y_i | x_i) = p(y_i | x_i) \quad \text{for } p\text{-a.e. } x_i. \quad (19)$$

Plugging into (17) yields

$$\min_{q \in \mathcal{D}_{\text{pw}}} \mathcal{R}(q) = \sum_{i=1}^L H(Y_i | X_i). \quad (20)$$

Step 3: The exact optimality gap. Subtracting the two minima obtained above gives the equality:

$$\begin{aligned} \min_{q \in \mathcal{D}_{\text{pw}}} \mathcal{R}(q) - \min_{q \in \mathcal{D}_{\text{seq}}} \mathcal{R}(q) \\ = \sum_{i=1}^L H(Y_i | X_i) - H(Y | X). \end{aligned} \quad (21)$$

Step 4: Lower bound by conditional total correlation and nonnegativity. By the definition of conditional total correlation,

$$\begin{aligned} \text{TC}(Y | X) &= \mathbb{E}_X \left[D_{\text{KL}} \left(p_{Y|X} \parallel \prod_{i=1}^L p_{Y_i|X} \right) \right] \\ &= \sum_{i=1}^L H(Y_i | X) - H(Y | X) \geq 0, \end{aligned} \quad (22)$$

where the conditional laws inside the KL are evaluated at the realized value of X , and nonnegativity follows from $\text{KL} \geq 0$. Moreover, since X_i is a component of X , conditioning on the full X cannot increase entropy:

$$H(Y_i | X) \leq H(Y_i | X_i) \quad \forall i. \quad (23)$$

Therefore, the optimality gap admits the exact decomposition

$$\begin{aligned} \sum_{i=1}^L H(Y_i | X_i) - H(Y | X) \\ = \underbrace{\sum_{i=1}^L H(Y_i | X) - H(Y | X)}_{\text{TC}(Y|X)} + \sum_{i=1}^L (H(Y_i | X_i) - H(Y_i | X)) \\ \geq \text{TC}(Y | X) \geq 0, \end{aligned} \quad (24)$$

which proves Proposition 1. $\square$