HIERARCHICAL CONTINUOUS DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion language models offer a compelling alternative to autoregressive generation for tasks demanding bidirectional reasoning and global constraint satisfaction. Yet they share a structural bottleneck: when decoding in parallel, each token is sampled independently from its marginal, severing the statistical dependencies among the tokens decoded together. Continuous diffusion language models avoid this by denoising a shared continuous state, but their denoiser sees only that state, so nothing ties it to a valid token configuration until it is finally decoded. To address this, we propose **Hierarchical Continuous Diffusion Language Models (HC-DLM)**, which couple discrete token generation with a continuous latent trajectory in a single, principled denoising process, whose training objective is derived from a variational bound on the token likelihood. In contrast to recent methods that attach continuous context to a self-contained discrete chain, HC-DLM makes the latent the only persistent generative state: tokens are read out from it at every step and feed back as a scaffold for the next latent update. On structured reasoning (Sudoku), mathematical planning (Countdown) and language modeling (LM1B), HC-DLM improves over discrete and continuous diffusion baselines at matched model size, in puzzle accuracy on Sudoku and Countdown and in generative perplexity on LM1B.

1 INTRODUCTION

Autoregressive (AR) language models generate text left-to-right. This sequential factorization is remarkably successful, yet it is a mismatch for tasks that need global constraint satisfaction or bidirectional computation, such as solving logical puzzles or planning arithmetic operations. On these tasks, AR models commit early to choices that are challenging to revise (Ye et al., 2025).

Discrete diffusion language models (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Nie et al., 2025) provide a principled alternative. By modeling the joint distribution over all tokens via an iterative denoising process, they enable the model to condition on arbitrary subsets of tokens and refine its predictions over many bidirectional passes, decoding multiple tokens in parallel at each step. This yields empirical gains on reasoning and planning (Ye et al., 2025; Kim et al., 2025) and, at scale, perplexity competitive with strong AR baselines (Nie et al., 2025; Lou et al., 2024).

However, a bottleneck remains at the heart of parallel decoding: *token dependence*. When the model unmasks multiple positions in a single denoising step, each position is independently sampled from its marginal conditioned on a partial sequence. Hence, the joint over the simultaneously decoded tokens is modeled as a product of marginals (Gu et al., 2018; Li et al., 2026; Hersche et al., 2026; Kim et al., 2026; Ringel et al., 2026), which fails precisely where tokens are tightly coupled by syntax, logic, or physical constraints.

Continuous diffusion language models address this by denoising a continuous state shared by all tokens: tokens decoded in parallel all depend on the same latent instead of being sampled on their own. They denoise either the token embeddings (Li et al., 2022; Gulrajani & Hashimoto, 2023; Chen et al., 2026) or a compressed latent produced by an encoder (Zhang et al., 2023; Lovelace et al., 2023; Guo et al., 2026), and decode tokens from the clean result. Their weakness mirrors the discrete one. Their denoiser takes only the continuous state as input, and no term of the objective involves tokens along the trajectory, removing ties to a valid token configuration until the final decoding step. Rounding (Li et al., 2022) and self-conditioning (Chen et al., 2023; 2026) feed the model's own token

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.

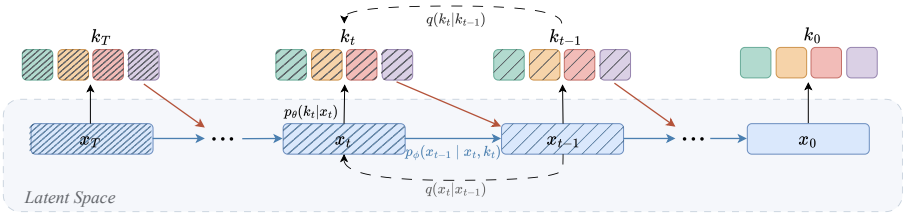


Figure 1: **HC-DLM coupled diffusion process.** The forward direction independently corrupts the continuous latent trajectory $x_0 \to x_T$ and the discrete token trajectory $k_0 \to k_T$ via $q(x_t \mid x_{t-1})$ and $q(k_t \mid k_{t-1})$. The reverse direction couples the two channels: $p_\phi(x_{t-1} \mid x_t, k_t)$ advances the continuous state conditioned on the current token state, while $p_\theta(k_t \mid x_t)$ reads token distributions from the latent state, iterating until the clean states (x_0, k_0) are reached.

estimate back as an input, but they share the same problem, since that estimate is neither a variable of the model nor a term of its objective.

The two weaknesses are complementary, and so are their remedies. Tokens decoded in parallel need a shared state to depend on, which a continuous latent provides, and a latent being denoised needs a discrete configuration to be tied to, which the tokens provide. We therefore propose **Hierarchical Continuous Diffusion Language Models (HC-DLM)**, a generative framework whose reverse process is a two-level chain (Fig. 1). The continuous latent trajectory $\{x_t\}$ is the only state that persists across steps. At every reverse step, tokens are read out from the current latent through a readout distribution $p_\theta(k_t \mid x_t)$, re-noised through the discrete forward kernel, and fed back as the condition of the next latent transition $p_\phi(x_{t-1} \mid x_t, k_t)$. The token state is thus a variable of the generative model with a forward kernel of its own, which is what separates the scaffold from a fed-back estimate. In the generative model, k_t depends on the past only through x_t, so the tokens carry no transition chain of their own and every position is read out anew at each step. We call this organization the hierarchical coupling. It determines both how the model is trained and how it samples: a single variational bound over the two-level trajectory gives the training objective (Section 3.2), and generation alternates latent denoising with token readout and re-noising (Section 3.4).

Hybrid discrete–continuous diffusion models also pair the two spaces, through per-token continuous hints (CADD) (Zheng et al., 2026), or an embedding chain denoised in parallel with the tokens (CCDD) (Zhou et al., 2026). However, separate transition chains are used as discussed in Section 5 and Appendix C.

Contributions.

1. **Framework.** We introduce HC-DLM (Section 3.1), a hierarchical generative structure for discrete sequences, in which a continuous latent is the only persistent state and tokens are per-step readouts that feed back as a conditioning scaffold.
2. **A likelihood bound for the hierarchical coupling chain.** We derive a variational lower bound for HC-DLM (Section 3.2) over the hierarchical trajectory, which establishes that reading tokens from the latent and feeding them back is a proper generative model of the token sequence.
3. **Both the continuous latent and the token feedback are needed.** We show on Sudoku, Countdown and LM1B that HC-DLM improves over discrete and continuous diffusion baselines at matched model size (Section 4), and that removing either one leaves a model that falls well short of the full one (Section 4.5).

2 Preliminaries

Discrete Diffusion Models. A discrete diffusion language model (Austin et al., 2021; Sahoo et al., 2024) defines a forward Markov chain that progressively corrupts a discrete length L token sequence $k_0 = (\dots, k_0^i, \dots) \in \mathcal{V}^L$ over vocabulary $\mathcal{V}$ ($|\mathcal{V}| = K$) with $i \in \{1, \dots, L\}$. Using a noise schedule $\alpha_0 = 1 \geq \alpha_1 \geq \cdots \geq \alpha_T \approx 0$, the marginal at time $t \in \{0, \dots, T\}$ interpolates between sequence k_0 and stationary distribution π:

$$q(k_t^i \mid k_0^i) = \alpha_t\, \delta(k_t^i - k_0^i) + (1-\alpha_t)\, \pi(k_t^i), \quad q(k_t^i \mid k_{t-1}^i) = \frac{\alpha_t}{\alpha_{t-1}}\, \delta(k_t^i - k_{t-1}^i) + \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right) \pi(k_t^i). \tag{1}$$

Following Austin et al. (2021), two common choices of π recover the standard discrete diffusion formulations: *(i) absorbing (*MASK*) state*, $\pi(k) = \delta(k - [\text{MASK}])$, which corrupts each token toward a special mask symbol and underlies most masked diffusion language models (Sahoo et al., 2024; Nie et al., 2025); *(ii) uniform state*, $\pi(k) = 1/K$, which replaces tokens by a uniform draw over the vocabulary. A parametric model $p_\theta(k_0 \mid k_t)$, typically a bidirectional transformer trained with token-level cross-entropy, reverses this corruption.

Token independence. During inference, the factored form $p_\theta(k_0 \mid k_t) = \prod_i p_\theta(k_0^i \mid k_t)$ samples each token independently. While the shared context k_t provides some global information, the joint distribution over simultaneously decoded tokens is modeled as a product of marginals which does not capture their statistical dependencies.

Continuous Diffusion Models. For continuous data $x_0 \in \mathbb{R}^d$, a DDPM (Ho et al., 2020) defines a Gaussian forward process with tractable posterior $q(x_{t-1} \mid x_t, x_0)$ and trains a denoiser via a weighted MSE objective. Flow Matching (FM) (Lipman et al., 2023; Liu et al., 2023) takes a more direct approach, training a velocity network v_ϕ to regress a target vector field transporting noise $\epsilon \sim \mathcal{N}(0, I)$ to data x_0. Using the affine interpolation $x_t = (1-t)\,x_0 + t\,\epsilon$, the Conditional Flow Matching objective is: $\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\, x_0,\, \epsilon}\big[\|v_\phi(x_t,\, t) - (\epsilon - x_0)\|^2\big]$. Samples are generated by integrating $\dot{x} = v_\phi(x, t)$ backward from $t = 1$ to $t = 0$, often with fewer function evaluations than DDPM sampling.

3 Hierarchical Continuous Diffusion Language Models

In this section, we first define a coupled continuous–discrete trajectory whose forward corruption remains tractable while its reverse dynamics are organized into two levels, with tokens read out from the latent state and fed back as a scaffold for the next latent update. We then derive a variational lower bound for this hierarchical process, convert it into a practical training objective (Fig. 2), and describe the resulting alternating sampler.

3.1 Generative Model and Forward Process

The token-independence bottleneck calls for a reverse process whose state carries cross-token information before any token is final, and the weakness of latent diffusion calls for that state to stay anchored to a token sequence while it is denoised. We therefore make the continuous representation itself a denoising trajectory and let every reverse step update both the continuous state and the token scaffold that conditions it.

For this, we model a sequence of discrete tokens $k_0 \in \mathcal{V}^L$ by introducing a coupled continuous latent state $x_0 \in \mathbb{R}^{M \times d}$. M is the latent sequence length and d is the per-position embedding dimension. The model maintains two trajectories, a continuous one $\{x_t\}_{t=0}^T$ and a discrete one $\{k_t\}_{t=0}^T$, that are corrupted independently in the forward direction but tightly coupled in the reverse direction.

Variational distribution (forward process). For independent noising, we use

$$q_\psi(x_{0:T},\, k_{1:T} \mid k_0) = q_\psi(x_0 \mid k_0) \prod_{t=1}^{T} q(x_t \mid x_{t-1})\; q(k_t \mid k_{t-1}). \tag{2}$$

The only learned component is the *encoder* $q_\psi(x_0 \mid k_0)$, which produces a continuous latent representation of the clean token sequence. Jointly training it with the generative model permits the latent geometry to adapt to the needs of denoising. The continuous noising $q(x_t \mid x_{t-1})$ is a standard Gaussian diffusion process with tractable marginals $q(x_t \mid x_0)$ and closed-form posteriors $q(x_{t-1} \mid x_t, x_0)$. It converges to a standard Gaussian prior $q(x_T \mid x_0) \approx \mathcal{N}(0, I)$ at the terminal time. For the discrete noising kernel $q(k_t \mid k_{t-1})$ we use the categorical forward process defined in Eq. (1), with either an absorbing or uniform state distribution. Note that the two forward chains in Eq. (2) are *independent*: x_t is corrupted by Gaussian noise while k_t is corrupted separately by token-level noise. This independence keeps both forward marginals tractable, exactly as in standard continuous and discrete diffusion.

Generative model (reverse process). While the forward chains are independent, the reverse chain deliberately breaks this symmetry: a continuous denoiser $p_\phi(x_{t-1} \mid x_t, k_t)$ conditions on the current

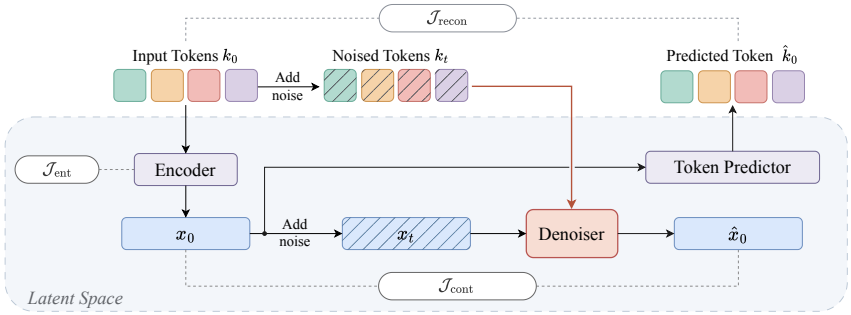


Figure 2: **HC-DLM training pipeline.** The encoder $q_\psi(x_0 \mid k_0)$ maps clean tokens k_0 to a continuous latent x_0, while independent forward kernels produce the noisy pair (x_t, k_t). The token-conditioned denoiser $d_\phi(x_t, k_t, t/T)$ produces the clean-latent estimate $\hat{x}_0$ (loss $\mathcal{J}_{\text{cont}}$), the token predictor $p_\theta(k_0 \mid x_0)$ decodes x_0 to $\hat{k}_0$ (loss $\mathcal{J}_{\text{recon}}$), and the encoder is regularized by $\mathcal{J}_{\text{ent}}$.

token state. Hence, information flows from discrete variables into the continuous trajectory at every step. Specifically, the joint generative model factorizes as follows:

$$p_{\theta,\phi}(k_{0:T},\, x_{0:T}) = p(x_T)\; p_\theta(k_0 \mid x_0) \prod_{t=1}^{T} p_\theta(k_t \mid x_t)\; p_\phi(x_{t-1} \mid x_t, k_t). \tag{3}$$

Two learned distributions drive the reverse direction of Fig. 1. The *token predictor* $p_\theta(k_t \mid x_t)$, with parameters θ, maps the noisy continuous state to a distribution over tokens at time t: at $t{=}0$ it produces the final discrete output, while at intermediate t it acts as a soft readout that maps the continuous state back to a distribution over the discrete vocabulary. The *continuous denoiser* $p_\phi(x_{t-1} \mid x_t, k_t)$, with parameters ϕ, advances the latent state conditioned on the current token state. Importantly, dependence of the denoising on k_t makes HC-DLM more than two parallel diffusion processes: the token state acts as a discrete scaffold that resolves continuous-space ambiguity. Confident token estimates constrain the directions in which x_t can move, while uncertain positions leave the continuous state free to explore alternative completions.

This intuition can be made precise. This reverse chain is a hierarchy rather than a pair of sibling processes, that is, two self-contained chains coupled at the same level. The token state has no transition kernel of its own: in Eq. (3), k_t depends on the past only through x_t, so all cross-step information travels through the continuous trajectory, and tokens are read out anew at every step, which keeps them revisable.

3.2 Variational Lower Bound

The two-level reverse chain also needs a principled training objective, and the question is whether coupling the reverse dynamics keeps the forward process simple enough for variational learning. It does: because the continuous and discrete corruptions in Eq. (2) are independent, the two processes above admit the following bound, which separates into interpretable reconstruction, prior-matching, boundary, and denoising terms.

Proposition 1 (HC-DLM ELBO)**.** *Given the generative model Eq.* (3) *and variational distribution Eq.* (2)*, we obtain* $\log p(k_0) \geq \mathcal{L}(k_0)$ *with*

$$\begin{aligned}\mathcal{L}(k_0) = {} & \underbrace{\mathbb{E}_{q_\psi(x_0|k_0)}\big[\log p_\theta(k_0 \mid x_0)\big]}_{\textit{(i) reconstruction}} - \underbrace{\mathbb{E}_{q_\psi(x_0|k_0)}\big[\mathrm{KL}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big)\big]}_{\textit{(ii) prior matching}} \\ & + \underbrace{\mathbb{E}_{q_\psi(x_0,x_1|k_0),\, q(k_1|k_0)}\big[\log p_\phi(x_0 \mid x_1, k_1)\big]}_{\textit{(iii.a) boundary continuous denoising}} \\ & + \underbrace{H\big(q_\psi(x_0 \mid k_0)\big)}_{\textit{(iii.b) encoder entropy}} - \underbrace{\mathbb{E}_{q_\psi(x_1|k_0)}\big[\mathrm{KL}\big(q(k_1 \mid k_0) \,\|\, p_\theta(k_1 \mid x_1)\big)\big]}_{\textit{(iii.c) boundary discrete token prediction}}\end{aligned}$$

$$- \underbrace{\sum_{t=2}^{T} \mathbb{E}_{q_\psi(x_0, x_t, k_{t-1} \mid k_0)} \big[\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) q(k_t \mid k_{t-1}) \| p_\phi(x_{t-1} \mid x_t, k_t) p_\theta(k_t \mid x_t) \big) \big]}_{\textit{(iv) denoising step}} . \tag{4}$$

The derivation is provided in Appendix A. We provide some intuition for each term next: **(i)** a data reconstruction log-likelihood. **(ii)** a terminal prior-matching, which doesn't contain trainable parameters. **(iii.a)** the boundary continuous denoising log-likelihood. **(iii.b)** an encoder entropy, which encourages uncertainty and hence a non-degenerate variational encoder. **(iii.c)** the boundary discrete token prediction cross-entropy. Finally, **(iv)** contributes two signals for every $t \geq 2$, since the reverse model predicts a token state and a continuous state at every step. Concretely, applying the chain rule of the KL divergence, and using the fact that $q(x_{t-1} \mid x_t, x_0)$ does not depend on k_t because the continuous and discrete forward kernels are independent in Eq. (2), yields

$$\begin{aligned} &\mathrm{KL}\Big(q(x_{t-1} \mid x_t, x_0)\, q(k_t \mid k_{t-1}) \,\Big\|\, p_\phi(x_{t-1} \mid x_t, k_t)\, p_\theta(k_t \mid x_t) \Big) \\ &= \underbrace{\mathrm{KL}\big(q(k_t \mid k_{t-1}) \,\|\, p_\theta(k_t \mid x_t) \big)}_{\text{discrete token prediction}} + \underbrace{\mathbb{E}_{q(k_t \mid k_{t-1})} \big[\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\phi(x_{t-1} \mid x_t, k_t) \big) \big]}_{\text{continuous denoising}} . \end{aligned} \tag{5}$$

The first sub-term is the discrete token prediction loss. The second sub-term is the continuous denoising loss, which we implement with flow matching rather than by fitting an explicit Gaussian reverse kernel. A detailed derivation of this identity is given in Appendix A.

Remark 1 (Connection to discrete diffusion)**.** *The discrete token prediction sub-term in Eq.* (5) *has the same functional form as the token-prediction KL used in categorical discrete diffusion, with x_t playing the role of the noisy conditioning state. HC-DLM therefore retains discrete-diffusion supervision while augmenting it with a continuous denoising signal.*

3.3 Practical Training Objective

The ELBO identifies three trainable signals: reconstruct the clean tokens from the clean latent, keep the encoder distribution non-degenerate, and denoise the continuous latent trajectory conditioned on the current token. We write expectations under q_ψ for samples from the forward/variational process.

Using the standard Gaussian parameterization of continuous diffusion, the continuous likelihood/KL terms reduce to a weighted MSE denoising objective (Ho et al., 2020; Luo, 2022), with schedule-dependent target $d_t(x_t, x_0)$ and weight ω_t. The explicit derivation is provided in Appendix A.

To obtain a loss, two points are notable. First, the boundary continuous denoising term **(iii.a)** has the same functional form as a $t{=}1$ instance of the continuous denoising sub-term in Eq. (5), so we include it in the sum over timesteps. Second, we use an x_0-prediction parameterization for the token predictor: from the noisy state x_t we form a clean-latent estimate $\hat{x}_0(x_t)$ via the denoiser d_ϕ (the explicit form depends on the parameterization of d_ϕ, Appendix B.1), and the token predictor maps $\hat{x}_0(x_t)$ to a clean-token distribution. The corresponding noisy-token distribution is obtained by applying the known discrete forward kernel. Under this parameterization, the intermediate noisy-token KL (the discrete token prediction sub-term in Eq. (5)) is controlled by a clean-token cross-entropy term; for the absorbing kernel it reduces to the same clean-token loss up to a schedule-dependent weight and constants, while for general categorical kernels it follows from an upper bound by the data processing inequality (Appendix A). Hence, term **(i)** and the boundary/intermediate discrete token prediction terms (from **(iii.c)** and Eq. (5)) are represented by $\mathcal{J}_{\mathrm{recon}}$; the boundary continuous denoising term **(iii.a)** and the continuous denoising sub-term of Eq. (5) becomes $\mathcal{J}_{\mathrm{cont}}$; **(iii.b)** becomes $\mathcal{J}_{\mathrm{ent}}$. Combined we have

$$\begin{aligned} \mathcal{J} = \lambda_{\mathrm{recon}} \underbrace{\mathbb{E}_{q_\psi(x_0 \mid k_0)} \big[-\log p_\theta(k_0 \mid x_0) \big]}_{\mathcal{J}_{\mathrm{recon}}\text{: reconstruction}} + \lambda_{\mathrm{ent}} \underbrace{\mathbb{E}_{q_\psi(x_0 \mid k_0)} \big[\log q_\psi(x_0 \mid k_0) \big]}_{\mathcal{J}_{\mathrm{ent}}\text{: encoder entropy}} \\ + \lambda_{\mathrm{cont}} \underbrace{\sum_{t=1}^{T} \mathbb{E}_{q_\psi(x_0, x_t, k_t \mid k_0)} \Big[\omega_t \big\| d_\phi(x_t, k_t, t/T) - d_t(x_t, x_0) \big\|^2 \Big]}_{\mathcal{J}_{\mathrm{cont}}\text{: continuous denoising}} . \end{aligned} \tag{6}$$

Algorithm 1 HC-DLM Training (single-sample Monte Carlo)

Require: Dataset $\mathcal{D}$; discrete noise schedule $\{\alpha_t\}$; loss weights $\lambda_{\text{recon}}, \lambda_{\text{cont}}, \lambda_{\text{ent}}$

repeat
  Sample $k_0 \sim \mathcal{D}$, $t \sim \text{Uniform}(1, T)$
  $x_0 \sim q_\psi(x_0 \mid k_0)$ ▷ Encode: tokens → continuous latent
  Sample $x_t \sim q(x_t \mid x_0)$ ▷ Continuous noising
  $k_t \sim q(k_t \mid k_0)$ ▷ Forward discrete noising
  Compute $\mathcal{J}$ following Eq. (6)
  Update θ, ϕ, ψ to minimize $\mathcal{J}$
until stopping criterion

Here, $\mathcal{J}_{\text{recon}}$ encourages that x_0 can be decoded to k_0, $\mathcal{J}_{\text{cont}}$ trains the token-conditioned continuous denoising trajectory, and $\mathcal{J}_{\text{ent}}$ prevents the Gaussian encoder from collapsing to a deterministic map (Kingma & Welling, 2014; Higgins et al., 2017). In experiments, we instantiate $\mathcal{J}_{\text{cont}}$ with Conditional Flow Matching (CFM). This implementation choice is detailed in Appendix A. $\mathcal{J}_{\text{recon}}$ evaluates the predictor at the encoder sample x_0 rather than at $\hat{x}_0(x_t)$, so token-level gradients do not reach the denoiser. In practice we sample a single timestep per training example and form a Monte Carlo estimate of Eq. (6). Algorithm 1 summarizes the inner loop.

3.4 Inference

Inference instantiates the coupled reverse chain in Eq. (3), made explicit in Algorithm 2: starting from $x_T \sim \mathcal{N}(0, I)$ and $k_T \sim \pi$, each step (i) forms the scaffold-conditioned clean-latent estimate $\hat{x}_0$ with the denoiser and advances the continuous state to x_{t-1}; (ii) reads clean tokens $\hat{k}_0$ out of $\hat{x}_0$ with the token predictor; and (iii) re-noises $\hat{k}_0$ through the known forward kernel to obtain k_{t-1}. After the final step, we return $\hat{k}_0$ from $p_\theta(k_0 \mid x_0)$. Under the absorbing kernel, step (iii) can replace random re-masking with adaptive, confidence-based re-masking (Section 4.5). Architecture and conditional-generation details are in Appendices B.1 and B.2.

4 Experiments

We evaluate HC-DLM on three tasks targeting complementary capabilities: structured reasoning (Sudoku), mathematical planning (Countdown), and general language generation (LM1B), comparing against autoregressive, discrete diffusion, and continuous or hybrid diffusion baselines.

4.1 Setup and Baselines

For Sudoku, we compare to the autoregressive model (ARM) of Shah et al. (2024), with and without its learned variable-ordering heuristic, and to masked diffusion (MDM) with the MDLM objective (Sahoo et al., 2024), reporting vanilla parallel sampling and the adaptive top-probability and top-probability-margin rules of Kim et al. (2025). For Countdown, we include the baseline published by Ye et al. (2025): GPT-2 Scratch, Stream-of-Search, LLaMA, VDM, D3PM, and RDM, together with the same 6M-parameter MDM decoding variants for a same-order-of-magnitude diffusion comparison. On both tasks we further include CCDD (Zhou et al., 2026), reproduced at a matched 6M-parameter scale under the same protocol with its original backbone replaced by our own for fair comparison, as a hybrid discrete–continuous baseline. Parameter counts in our tables refer to sampling-time generative parameters, excluding token embeddings (Appendix B.4). For LM1B, we compare

Table 1: Sudoku puzzle-solving accuracy (%↑) on easy and hard splits. Best result per column in **bold**.

Method	#Params	Easy	Hard
Autoregressive			
ARM (w/o ordering)	42M	9.73	–
ARM (with ordering)		87.18	32.57
Discrete diffusion			
MDM (vanilla)		6.88	3.62
MDM (top-prob.)	6M	18.51	9.44
MDM (top-prob. margin)		89.49	49.88
Hybrid discrete–continuous			
CCDD	6M	**94.65**	70.73
HC-DLM (ours)	6M	94.21	**72.41**

against an autoregressive Transformer (Vaswani et al., 2017), the baseline discrete diffusion models MDM (Sahoo et al., 2024), SEDD (Lou et al., 2024) and Duo (Sahoo et al., 2025), and the continuous diffusion models Plaid (Gulrajani & Hashimoto, 2023) and LangFlow (Chen et al., 2026). Implementation details are given in Appendix B.

4.2 Token-Dependent Reasoning: Sudoku

Setup. Sudoku puzzles are 9×9 grids ($L = 81$, $K = 10$), given as partially filled grids. The model must fill all blank cells, and a prediction counts as correct only when all 81 cells match the unique solution. We follow the data and evaluation protocol of Kim et al. (2025): the standard split of Shah et al. (2024) contains puzzles solvable by a fixed set of seven logical strategies, and the hard split consists of the remaining puzzles, which require a strategy outside that set (Appendix B.3).

Results. Table 1 compares HC-DLM to autoregressive and diffusion baselines, with the autoregressive and masked-diffusion results taken from Kim et al. (2025). HC-DLM is competitive on both splits. On the in-distribution Easy set, it surpasses the best masked-diffusion variant and the much larger 42M-parameter autoregressive model, while the matched CCDD implementation attains comparable and marginally higher accuracy (94.65 vs. 94.21). On the out-of-distribution Hard split, HC-DLM instead leads CCDD (72.41 vs. 70.73), indicating that its advantage is most pronounced beyond the solution strategies represented in training.

4.3 Mathematical Reasoning: Countdown

Setup. Countdown (Ye et al., 2025) is a mathematical reasoning task that generalizes the Game of 24: given n numbers and a target integer, the model must produce a chain of arithmetic steps that reaches the target exactly. We follow Ye et al. (2025), with 500k problems and 10% of the target values held out for out-of-distribution evaluation; CD4 and CD5 use four and five input numbers. Since a Countdown puzzle admits many valid solutions, we report answer-correct accuracy (Appendix B.3).

Table 2: Countdown accuracy (%↑). Best overall result per subtask in **bold**; best result at the 6M parameter scale underlined.

Method	#Params	CD4	CD5
Autoregressive			
GPT-2 Scratch	6M	31.9	4.3
	85M	45.8	5.1
	303M	41.3	4.5
Stream-of-Search	250M	54.2	–
LLaMA	7B	41.1	6.7
	13B	51.1	7.4
Discrete diffusion			
VDM	85M	73.4	16.3
D3PM	85M	83.1	27.6
RDM	85M	**87.0**	**45.8**
MDM (vanilla)	6M	6.7	0.3
MDM (top-prob.)	6M	47.4	21.4
MDM (top-prob. margin)	6M	50.8	21.3
Hybrid discrete–continuous			
CCDD	6M	81.18	25.35
HC-DLM (ours)	6M	<u>84.41</u>	<u>37.52</u>

Results. Table 2 compares HC-DLM to autoregressive and diffusion baselines. Results for the non-MDM baselines are taken from Ye et al. (2025), while the MDM and CCDD rows are our reproduced 6M-parameter baseline runs under the same Countdown setting. Most diffusion variants outperform substantially larger autoregressive models; following Ye et al. (2025), we read this gap as any-order decoding suiting subgoal-imbalanced planning, not as a property of any single method. The informative comparison for HC-DLM is therefore with diffusion models of the same order of magnitude, and there it holds a clear advantage: it surpasses the matched hybrid CCDD on both subtasks, with the margin widening on the longer-horizon CD5 (37.52 vs. 25.35). Much larger discrete diffusion models still achieve the best overall numbers, but HC-DLM remains highly competitive at a fraction of the parameter cost.

4.4 Language Modeling: LM1B

Setup. To evaluate our method on natural language modeling, we test unconditional generation on the One Billion Word benchmark (LM1B; Chelba et al., 2013) at sequence length 128 with 118M parameters in the denoiser and token decoder, excluding embeddings (Appendix B.4), and report generative perplexity (Gen. PPL) under GPT-2-Large (Radford et al., 2019) over 1,024 samples at 128 sampling steps, following the protocol of LangFlow (Chen et al., 2026); sampling uses classifier-free guidance with $w = 2.75$ (Appendix B.7).

Results. Table 3 compares HC-DLM with autoregressive, discrete-diffusion and continuous-diffusion baselines on LM1B, with baseline results taken from LangFlow (Chen et al., 2026). HC-DLM achieves the best generative perplexity among the evaluated diffusion models, outperforming all discrete diffusion baselines and both continuous models that decode only once at the end. Entropy remains stable under moderate reductions in NFE and across the CFG sweep (Table 7 and Fig. 5). At large batch sizes, HC-DLM also maintains lower wall-clock sampling time, suggesting favorable efficiency for batched parallel generation (Fig. 4). Together with the structured-task ablations below, these comparisons are consistent with a benefit from iterative token feedback over decoding the continuous representation only once at the end. Implementation and training details are provided in Appendix B.4.

Table 3: Unconditional generation results on LM1B. Baseline results are from models retrained by the LangFlow authors (Chen et al., 2026).

Method	#Params	Gen. PPL ↓
Autoregressive		
Transformer	108M	66.7
Diffusion		
MDM	116M	103.9
SEDD	116M	115.9
Duo	116M	97.6
Plaid	109M	77.3
LangFlow	117M	92.2
HC-DLM (ours)	118M	75.5
Ground truth	–	40.4

4.5 Ablation Studies

We test HC-DLM's key design choices and analyze its generation dynamics; unless otherwise specified, ablations use Sudoku.

Framework Choices. Our central hypothesis is that the gains come from the hierarchical coupling itself, i.e., a latent plan with per-step token readout and scaffold feedback. To test this, we remove each level in turn. Latent DM removes the token level by dropping the token state k_t from the denoiser d_ϕ, which leaves a standard continuous latent diffusion model that decodes tokens only at the final step. The MDM baseline from Table 1 removes the latent level altogether, leaving a purely discrete chain.

As shown in Table 4, removing either level degrades performance, particularly on the Hard split where global constraint satisfaction is critical. Latent DM falls far below HC-DLM, demonstrating that a continuous latent without a discrete scaffold can over-smooth precise token-level constraints. Conversely, MDM's bottleneck highlights the limitations of independent discrete sampling.

Table 4: Component ablation on Sudoku: accuracy (%↑). **Cont. latent**: latent trajectory x_t present. **Disc. cond.**: k_t passed to denoiser d_ϕ.

Method	Cont. latent	Disc. cond.	Easy	Hard
MDM (top-prob. margin)	×	×	89.49	49.88
Latent DM (w/o disc. cond.)	✓	×	50.46	24.74
HC-DLM (ours)	✓	✓	**94.21**	**72.41**

Discrete Forward Process Design. We next evaluate the discrete forward process choice: Table 5 compares the *absorbing* and *uniform* kernels. For the absorbing kernel, the presence of distinct masked tokens allows for explicit decoding schedules. We compare *random* selection to adaptive strategies: *top-prob.* (unmasking the token with the highest predicted probability) and *top-prob. margin* (top-two probability gap (Kim et al., 2025)). Because the uniform kernel corrupts all tokens uniformly without a separate masked set, we apply the standard parallel update, re-noising all positions through the forward kernel at each step.

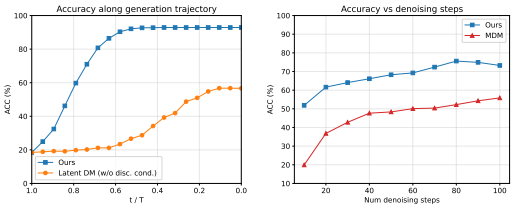
The results demonstrate that while adaptive ordering modestly improves the absorbing kernel, the uniform kernel is substantially stronger for our hierarchical sampler. We therefore use the uniform kernel as the default in all main experiments. Two properties of the hierarchy explain this gap: the discrete state serves the latent denoiser as an informative conditioning scaffold, which absorbing corruption empties of content, and per-step readout keeps every position revisable, which adaptive unmasking forecloses; Appendix B.11 expands this analysis.

Table 5: Ablation on Sudoku: discrete noise type and token-ordering heuristic. Accuracy (%↑). Best result in **bold**.

Noise type	Token ordering	Easy	Hard
Absorbing-state	Random	69.60	43.74
	Top-prob.	74.72	47.50
	Top-prob. margin	75.59	48.25
Uniform-state	Parallel update	**94.21**	**72.41**

Information Emergence. To understand why HC-DLM outperforms the baselines, we analyze how information crystallizes along the reverse trajectory: at each step t we decode the intermediate clean-latent prediction $\hat{x}_0$ into tokens and measure the resulting solution accuracy. As shown in Fig. 3a, accuracy is near chance at $t/T = 1.0$, where the latent is near Gaussian noise, but rises significantly earlier and reaches a much higher plateau than Latent DM. Conditioning the continuous denoiser on the discrete scaffold k_t "locks in" structural constraints sooner, refining the solution progressively through the feedback loop rather than resolving it at the final boundary step.

Inference Efficiency. We further evaluate accuracy across varying numbers of denoising steps on the Hard Sudoku split. Fig. 3b illustrates that HC-DLM is robust to reduced step counts, maintaining strong predictive accuracy. In contrast, the purely discrete MDM baseline degrades more sharply as the number of steps decreases. This pattern is consistent with parallel MDM predictions relying on repeated refinement to compensate for their conditional-independence approximation. In HC-DLM, all parallel readouts are tied to one shared latent state, and the scaffold conditions the latent update inside every step. Appendix B.10 further shows that our implementation is faster in wall-clock time than the reproduced MDM implementation at a comparable parameter scale.



(a) Trajectory decoding. (b) Step-count robustness.

Figure 3: **(a)** Accuracy of the decoded intermediate prediction $\hat{x}_0$ at each normalized step t/T; HC-DLM improves faster. **(b)** Final accuracy on Hard Sudoku vs. step count; HC-DLM stays robust with fewer steps.

5 Related Work

Discrete diffusion language models. D3PM (Austin et al., 2021), SEDD (Lou et al., 2024), MDLM (Sahoo et al., 2024), RDM (Zheng et al., 2023) and LLaDA (Nie et al., 2025) introduced absorbing- and uniform-state diffusion over tokens from small to 8B scale, with block diffusion (Arriola et al., 2025) and discrete flow matching (Gat et al., 2024) as alternative decompositions. MGDM (Ye et al., 2025) and Kim et al. (2025) established their advantage on planning tasks and the role of decoding order. HC-DLM preserves parallel token updates and adds a continuous variable.

Continuous diffusion for text. Continuous models denoise a continuous state of the sequence, either per-token embeddings (Li et al., 2022; Gong et al., 2023; Gulrajani & Hashimoto, 2023; Chen et al., 2026; Hu et al., 2026; Shen et al., 2026) or an encoder-compressed latent (Zhang et al., 2023; Lovelace et al., 2023; Guo et al., 2026; Meshchaninov et al., 2026), with a denoiser that takes only the continuous state as input and tokens decoded from the clean result. HC-DLM reads tokens out of the latent at every step and feeds them back into its denoiser, so the two levels refine each other throughout generation (Section 3.1).

Hybrid discrete–continuous diffusion. Hybrid models pair a discrete chain with a continuous channel, whether a global latent drawn once (VMD) (Zhang et al., 2025), per-token hints (CADD) (Zheng et al., 2026), or a parallel embedding chain (CCDD) (Zhou et al., 2026), and in these designs the tokens keep a transition chain of their own. Further couplings are explored by Shariatian et al. (2025), Pynadath et al. (2025) and Lemercier et al. (2026). HC-DLM instead reads tokens out of it at every step, so the tokens carry no transition kernel of their own. See Appendix C for more.

6 Conclusion

We proposed HC-DLM, a diffusion framework for discrete sequences whose reverse process is a two-level chain, with tokens read out of a continuous latent at every step and fed back into its denoiser. A single variational lower bound yields the coupled objective, and the hierarchical coupling addresses the token-independence bottleneck of parallel decoding by routing cross-token dependence through a compact learned latent. Experiments on Sudoku, Countdown, and LM1B show that the framework applies across globally constrained generation and unconditional language modeling. Structured-task comparisons show that scaffold feedback is important and that the coupled model outperforms continuous-only and discrete alternatives.

AI USE STATEMENT

Large language models were used to assist with editing and polishing the writing of this paper. All scientific content, experimental results, and claims were reviewed and verified by the authors, who take responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies training methods for diffusion language models. All experiments use publicly available datasets under their released licenses; we do not collect new data or involve human subjects. The trained models are at a scale comparable to prior academic DLM work and do not, to our knowledge, introduce misuse risks beyond those already documented for autoregressive language models of similar size.

REPRODUCIBILITY STATEMENT

We will open-source our code, data and pre-trained checkpoints upon publication.

REFERENCES

Xinyue Ai, Yutong Kelly He, Albert Gu, Russ Salakhutdinov, Zico Kolter, Nicholas Boffi, and Max Simchowitz. Joint distillation for fast likelihood evaluation and sampling in flow-based models. In *ICLR*, 2026.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *ICLR*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2013. arXiv preprint arXiv:1312.3005.

Chen Chen, Rui Qian, Wenze Hu, Tsu-Jui Fu, Jialing Tong, Xinze Wang, Lezhi Li, Bowen Zhang, Alex Schwing, Wei Liu, and Yinfei Yang. Dit-air: Revisiting the efficiency of diffusion model architecture design in text to image generation, 2025. arXiv preprint arXiv:2503.10618.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *ICLR*, 2023.

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

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *NeurIPS*, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *ICLR*, 2023.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR*, 2018.

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. In *NeurIPS*, 2023.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, et al. Continuous latent diffusion language model, 2026. arXiv preprint arXiv:2605.06548.

Michael Hersche, Nicolas Menet, Ronan Tanios, and Abbas Rahimi. Locally coherent parallel decoding in diffusion language models, 2026. arXiv preprint arXiv:2603.20216.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows, 2026. arXiv preprint arXiv:2605.10938.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. Dependency-aware parallel decoding via attention for diffusion llms, 2026. arXiv preprint arXiv:2603.12996.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *ICML*, 2025.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Jean-Marie Lemercier, Tomas Geffner, Karsten Kreis, Morteza Mardani, Arash Vahdat, and Ante Jukić. Diladiff: Distilled latent-augmented diffusion for language modeling, 2026. arXiv preprint arXiv:2605.23605.

Ian Li, Zilei Shao, Benjie Wang, Rose Yu, Guy Van den Broeck, and Anji Liu. Breaking the factorization barrier in diffusion language models, 2026. arXiv preprint arXiv:2603.00045.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In *NeurIPS*, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. In *NeurIPS*, 2023.

Calvin Luo. Understanding diffusion models: A unified perspective, 2022. arXiv preprint arXiv:2208.11970.

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. arXiv preprint arXiv:2605.07933.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models, 2025. arXiv preprint arXiv:2510.22510.

David Radcliffe. 3 million sudoku puzzles with ratings, 2020. Kaggle, `https://www.kaggle.com/datasets/radcliffe/3-million-sudoku-puzzles-with-ratings`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-guided parallel decoding in discrete diffusion language models, 2026. arXiv preprint arXiv:2604.02560.

Subham S 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 *NeurIPS*, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *ICML*, 2025.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *NeurIPS*, 2024.

Dario Shariatian, Alain Durmus, Umut Simsekli, and Stefano Peluchetti. Latent-augmented discrete diffusion models, 2025. arXiv preprint arXiv:2510.18114.

Junzhe Shen, Jieru Zhao, Ziwei He, and Zhouhan Lin. Codar: Continuous diffusion language models are more powerful than you think, 2026. arXiv preprint arXiv:2603.02547.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *ICLR*, 2025.

Yichi Zhang, Alex Schwing, and Zhizhen Zhao. Variational masked diffusion models, 2025. arXiv preprint arXiv:2510.23606.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua Susskind, and Navdeep Jaitly. Planner: Generating diversified paragraph via latent language diffusion model. In *NeurIPS*, 2023.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling. In *ICLR*, 2026.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation, 2023. arXiv preprint arXiv:2302.05737.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. In *ICML*, 2026.

A Proof of Proposition 1

We derive the ELBO step by step, starting from the definitions in Section 3.1.

Step 1: Jensen's inequality. Starting from the marginal likelihood of k_0, we make use of the variational distribution given in Eq. (2) and apply Jensen's inequality:

$$\begin{aligned}\log p(k_0) &= \log \int \sum_{k_{1:T}} p_{\theta,\phi}(k_{0:T}, x_{0:T})\, dx_{0:T} \\ &= \log \mathbb{E}_{q_\psi(x_{0:T}, k_{1:T}|k_0)}\left[\frac{p_{\theta,\phi}(k_{0:T}, x_{0:T})}{q_\psi(x_{0:T}, k_{1:T} \mid k_0)}\right] \\ &\geq \mathbb{E}_{q_\psi(x_{0:T}, k_{1:T}|k_0)}\left[\log \frac{p_{\theta,\phi}(k_{0:T}, x_{0:T})}{q_\psi(x_{0:T}, k_{1:T} \mid k_0)}\right] =: \mathcal{L}(k_0). \end{aligned} \tag{7}$$

For readability, we use $\mathbb{E}_{q_\psi}$ to denote the expectation under $q_\psi(x_{0:T}, k_{1:T} \mid k_0)$ in what follows.

Step 2: Substituting the factorizations. Substituting Eq. (3) and Eq. (2) and taking logs:

$$\begin{aligned}\mathcal{L}(k_0) = \mathbb{E}_{q_\psi}\Big[&\log p(x_T) + \log p_\theta(k_0 \mid x_0) + \textstyle\sum_{t=1}^T \log p_\theta(k_t \mid x_t) + \sum_{t=1}^T \log p_\phi(x_{t-1} \mid x_t, k_t) \\ &- \log q_\psi(x_0 \mid k_0) - \textstyle\sum_{t=1}^T \log q(x_t \mid x_{t-1}) - \sum_{t=1}^T \log q(k_t \mid k_{t-1})\Big]. \end{aligned} \tag{8}$$

We separate the $t = 1$ boundary term from the $t \geq 2$ terms:

$$\begin{aligned}\mathcal{L}(k_0) = &\mathbb{E}_{q_\psi}\Big[\log p_\theta(k_0 \mid x_0) - \log q_\psi(x_0 \mid k_0)\Big] \\ &+ \mathbb{E}_{q_\psi}\Big[\log p_\theta(k_1 \mid x_1) + \log p_\phi(x_0 \mid x_1, k_1) - \log q(x_1 \mid x_0) - \log q(k_1 \mid k_0)\Big] \\ &+ \sum_{t=2}^T \mathbb{E}_{q_\psi}\Big[\log p_\theta(k_t \mid x_t) + \log p_\phi(x_{t-1} \mid x_t, k_t) - \log q(x_t \mid x_{t-1}) - \log q(k_t \mid k_{t-1})\Big] \\ &+ \mathbb{E}_{q_\psi}\big[\log p(x_T)\big]. \end{aligned} \tag{9}$$

Step 3: Telescoping on continuous terms. For $t \geq 2$, we apply the diffusion posterior identity:

$$q(x_t \mid x_{t-1}) = q(x_{t-1} \mid x_t, x_0)\, \frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)}, \tag{10}$$

which gives

$$\log q(x_t \mid x_{t-1}) = \log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0). \tag{11}$$

Substituting this into the sum over $t = 2, \ldots, T$ in Eq. (9), the terms $-\log q(x_t \mid x_0) + \log q(x_{t-1} \mid x_0)$ telescope:

$$\sum_{t=2}^T \big(-\log q(x_t \mid x_0) + \log q(x_{t-1} \mid x_0)\big) = -\log q(x_T \mid x_0) + \log q(x_1 \mid x_0). \tag{12}$$

The resulting $+\log q(x_1 \mid x_0)$ cancels the $-\log q(x_1 \mid x_0)$ in the t=1 boundary term. After telescoping:

$$\begin{aligned}\mathcal{L}(k_0) = &\mathbb{E}_{q_\psi}\Big[\log p_\theta(k_0 \mid x_0) - \log q_\psi(x_0 \mid k_0)\Big] \\ &+ \mathbb{E}_{q_\psi}\Big[\log p_\theta(k_1 \mid x_1) + \log p_\phi(x_0 \mid x_1, k_1) - \log q(k_1 \mid k_0)\Big] \\ &+ \sum_{t=2}^T \mathbb{E}_{q_\psi}\big[\log p_\theta(k_t \mid x_t) + \log p_\phi(x_{t-1} \mid x_t, k_t) \\ &\qquad - \log q(x_{t-1} \mid x_t, x_0) - \log q(k_t \mid k_{t-1})\big] \\ &+ \mathbb{E}_{q_\psi}\big[\log p(x_T) - \log q(x_T \mid x_0)\big]. \end{aligned} \tag{13}$$

The last line is the negative prior KL:

$$\mathbb{E}_{q_\psi}\big[\log p(x_T) - \log q(x_T \mid x_0)\big] = -\mathbb{E}_{q_\psi(x_0|k_0)}\Big[\mathrm{KL}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big)\Big]. \tag{14}$$

Step 4: Rewriting as KL divergences. *Boundary term ($t = 1$).* Rearranging the t=1 terms in Eq. (13):

$$\mathbb{E}_{q_\psi}\Big[\log p_\theta(k_1 \mid x_1) + \log p_\phi(x_0 \mid x_1, k_1) - \log q(k_1 \mid k_0)\Big]$$
$$= \mathbb{E}_{q_\psi(x_0,x_1|k_0),\, q(k_1|k_0)}\big[\log p_\phi(x_0 \mid x_1, k_1)\big] - \mathbb{E}_{q_\psi(x_1|k_0)}\big[\mathrm{KL}\big(q(k_1 \mid k_0) \,\|\, p_\theta(k_1 \mid x_1)\big)\big], \quad (15)$$

where we used the definition of the KL divergence for the discrete part. The remaining term from Eq. (13), $-\mathbb{E}_{q_\psi(x_0|k_0)}[\log q_\psi(x_0 \mid k_0)]$, is the encoder entropy $H(q_\psi(x_0 \mid k_0))$. Together with the reconstruction term and the prior-matching term, this gives terms (i)–(iii) in Eq. (4).

Intermediate terms ($t \geq 2$). For each $t \geq 2$, define

$$\mathcal{A}_t := \mathbb{E}_{q_\psi}\big[\log p_\theta(k_t \mid x_t) + \log p_\phi(x_{t-1} \mid x_t, k_t) - \log q(x_{t-1} \mid x_t, x_0) - \log q(k_t \mid k_{t-1})\big]. \quad (16)$$

This can be rewritten as an expectation of a log-ratio over the joint variational distribution $q(x_{t-1}, k_t \mid x_t, x_0, k_{t-1}) = q(x_{t-1} \mid x_t, x_0)\, q(k_t \mid k_{t-1})$:

$$\mathcal{A}_t = \mathbb{E}_{q_\psi(x_0,x_t,k_{t-1}|k_0)}\mathbb{E}_{q(x_{t-1}|x_t,x_0)\, q(k_t|k_{t-1})}\left[\log \frac{p_\phi(x_{t-1} \mid x_t, k_t)\, p_\theta(k_t \mid x_t)}{q(x_{t-1} \mid x_t, x_0)\, q(k_t \mid k_{t-1})}\right]$$
$$= -\mathbb{E}_{q_\psi(x_0,x_t,k_{t-1}|k_0)}\Big[\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0)\, q(k_t \mid k_{t-1}) \| p_\phi(x_{t-1} \mid x_t, k_t)\, p_\theta(k_t \mid x_t)\big)\Big]. \quad (17)$$

Combining terms (i)–(iv) gives Proposition 1. □

Chain rule decomposition (Eq. (5)). We now derive the decomposition of the joint KL in term (iv) into the two complementary learning signals shown in Eq. (5).

Abbreviate $Q_{t-1} := q(x_{t-1} \mid x_t, x_0)$, $Q_t^k := q(k_t \mid k_{t-1})$, $P_{t-1} := p_\phi(x_{t-1} \mid x_t, k_t)$, $P_t^k := p_\theta(k_t \mid x_t)$. The joint KL expands by definition:

$$\begin{aligned}
&\mathrm{KL}\big(Q_t^k\, Q_{t-1} \,\big\|\, P_t^k\, P_{t-1}\big) \\
&= \mathbb{E}_{Q_t^k\, Q_{t-1}}\big[\log Q_t^k + \log Q_{t-1} - \log P_t^k - \log P_{t-1}\big] \\
&= \mathbb{E}_{Q_t^k}\big[\log Q_t^k - \log P_t^k\big] + \mathbb{E}_{Q_t^k}\big[\mathbb{E}_{Q_{t-1}}\big[\log Q_{t-1} - \log P_{t-1}\big]\big], \qquad (18)
\end{aligned}$$

where the split uses the fact that $Q_{t-1} = q(x_{t-1} \mid x_t, x_0)$ does not depend on k_t, while $P_{t-1} = p_\phi(x_{t-1} \mid x_t, k_t)$ does. Therefore:

$$\begin{aligned}
\mathrm{KL}\big(Q_t^k\, Q_{t-1} \,\big\|\, P_t^k\, P_{t-1}\big) &= \underbrace{\mathrm{KL}\big(q(k_t \mid k_{t-1}) \,\|\, p_\theta(k_t \mid x_t)\big)}_{\text{discrete token prediction}} \\
&\quad + \underbrace{\mathbb{E}_{q(k_t|k_{t-1})}\big[\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\phi(x_{t-1} \mid x_t, k_t)\big)\big]}_{\text{continuous denoising}}, \qquad (19)
\end{aligned}$$

which is exactly Eq. (5). The key observation is that the chain rule of the KL divergence,

$$\mathrm{KL}\big(p(A)\, p(B \mid A) \| q(A)\, q(B \mid A)\big) = \mathrm{KL}\big(p(A) \| q(A)\big) + \mathbb{E}_{p(A)}\big[\mathrm{KL}\big(p(B \mid A) \| q(B \mid A)\big)\big], \quad (20)$$

applies here with $A = k_t$ and $B = x_{t-1}$: the "forward" distribution of x_{t-1} given k_t is $q(x_{t-1} \mid x_t, x_0)$ (constant in k_t), while the generative distribution is $p_\phi(x_{t-1} \mid x_t, k_t)$ (dependent on k_t). □

Step 5: Practical parameterization. The ELBO above is derived for a Markov continuous noising chain with tractable Gaussian posteriors $q(x_{t-1} \mid x_t, x_0)$. Concretely, write the marginal as

$$q(x_t \mid x_0) = \mathcal{N}\Big(\sqrt{\bar{\alpha}_t^x}\, x_0,\, (1 - \bar{\alpha}_t^x) I\Big), \quad x_t = \sqrt{\bar{\alpha}_t^x}\, x_0 + \sqrt{1 - \bar{\alpha}_t^x}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (21)$$

so that $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\tilde{\mu}_t(x_t, x_0), \tilde{\Sigma}_t)$. If $p_\phi(x_{t-1} \mid x_t, k_t) = \mathcal{N}(\mu_\phi(x_t, k_t, t), \sigma_t^2 I)$, then

$$\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\phi(x_{t-1} \mid x_t, k_t)\big) = \frac{1}{2\sigma_t^2}\big\|\mu_\phi(x_t, k_t, t) - \tilde{\mu}_t(x_t, x_0)\big\|^2 + C_t, \quad (22)$$

where C_t is independent of learned parameters. With the usual noise-prediction parameterization, this is equivalent to a weighted MSE

$$\mathcal{J}_t^{\mathrm{cont}} = \mathbb{E}_{q_\psi(x_0,x_t,k_t|k_0)}\Big[\omega_t \big\|\epsilon_\phi(x_t, k_t, t/T) - \epsilon\big\|^2\Big], \quad (23)$$

for a known schedule-dependent weight ω_t (Ho et al., 2020; Luo, 2022). For the discrete token prediction term, the categorical kernel gives $\mathrm{KL}(q(k_t \mid k_{t-1}) \| p_\theta(k_t \mid x_t)) = \mathbb{E}_{q(k_t|k_{t-1})}[-\log p_\theta(k_t \mid x_t)] + C'$ with constant C' independent of θ. After averaging over $q(k_{t-1} \mid k_0)$, the trainable cross-entropy part is equivalently

$$\mathbb{E}_{q(k_t|k_0)}\big[-\log p_\theta(k_t \mid x_t)\big], \tag{24}$$

again up to constants independent of θ.

Clean-token surrogate for the discrete KL. To optimize the discrete part of the ELBO we must evaluate the noisy-token cross-entropy in Eq. (24), which requires the noisy-token distribution $p_\theta(k_t \mid x_t)$ at every noise level t. The challenge is that this is one head per noise level, while the rest of the objective only ever supervises the model at the clean endpoint. We highlight two obstacles, then resolve both with a single x_0-prediction parameterization that ties everything back to the clean-token predictor $p_\theta(k_0 \mid \cdot)$ already used in term (i).

Obstacle 1: a per-t predictor is wasteful and inconsistent. The underlying signal at every t is the same clean sequence k_0 passed through the known categorical kernel $q(k_t \mid k_0)$ from Eq. (1), so a separate head per t would relearn information that is already analytically available. It is also inconsistent with the clean-endpoint head $p_\theta(k_0 \mid x_0)$ supervised by term (i): we would maintain two discrete heads for what is essentially the same task.

We therefore reuse a single, time-invariant clean-token predictor $p_\theta(k_0 \mid \cdot)$, the same head appearing in $\mathcal{J}_{\mathrm{recon}}$, and compose it with the known forward kernel from Eq. (1):

$$p_\theta(k_t \mid x_t) = \sum_{\tilde{k}_0} q(k_t \mid \tilde{k}_0)\, p_\theta(\tilde{k}_0 \mid \hat{x}_0(x_t)), \tag{25}$$

where $\hat{x}_0(x_t)$ is the clean-latent estimate produced by the denoiser. Its explicit form depends on the parameterization of d_ϕ. In our implementation the denoiser predicts the clean latent directly, $\hat{x}_0(x_t) = d_\phi(x_t, k_t, t/T)$, an x-prediction parameterization (see Appendix B.1); an equivalent velocity parameterization would instead set $\hat{x}_0(x_t) = x_t - (t/T)\, v_\phi(x_t, k_t, t/T)$. The data-processing argument that follows does not rely on either form. With this parameterization the noisy-token distribution at every t is determined in closed form once $p_\theta(k_0 \mid \cdot)$ is learned, and a single head is shared across $t = 0, 1, \ldots, T$.

Obstacle 2: directly optimizing the noisy-token KL gives a smeared gradient. Even after Eq. (25), optimizing $-\log p_\theta(k_t \mid x_t)$ requires differentiating through the mixture induced by $q(k_t \mid k_0)$. Under the absorbing kernel, in particular, $q(k_t \mid k_0)$ maps many distinct candidate clean tokens to the same noisy symbol [MASK], so the gradient is averaged over a large equivalence class and the per-position learning signal on $p_\theta(k_0 \mid \cdot)$ is diffuse. To recover a sharp per-position cross-entropy, and to match the functional form of term (i), we replace the noisy-token KL by a clean-token upper bound obtained from the data-processing inequality.

For a fixed clean sequence k_0, the noisy-token KL induced by the model family in Eq. (25) is

$$\mathrm{KL}\Big(q(k_t \mid k_0) \,\Big\|\, \textstyle\sum_{\tilde{k}_0} q(k_t \mid \tilde{k}_0)\, p_\theta(\tilde{k}_0 \mid \hat{x}_0(x_t))\Big). \tag{26}$$

Both arguments are obtained by pushing a clean-token distribution (δ_{k_0} on the left, $p_\theta(\cdot \mid \hat{x}_0(x_t))$ on the right) through the same channel $q(k_t \mid k_0)$, so the data-processing inequality gives

$$(26) \le \mathrm{KL}(\delta_{k_0} \,\|\, p_\theta(\cdot \mid \hat{x}_0(x_t))) = -\log p_\theta(k_0 \mid \hat{x}_0(x_t)). \tag{27}$$

Thus the clean-token cross-entropy upper-bounds the trainable noisy-token KL, up to entropy terms determined only by the forward kernel. For the absorbing kernel, if the token predictor places probability mass only on non-mask data tokens, this relation tightens to the usual masked-diffusion form: the trainable part of Eq. (24) is $\alpha_t\big[-\log p_\theta(k_0 \mid \hat{x}_0(x_t))\big]$, with all remaining terms independent of θ.

In the practical objective we further supervise the same token predictor at the clean endpoint, replacing $\hat{x}_0(x_t)$ by the variational clean latent x_0 to obtain

$$\mathcal{J}_{\mathrm{recon}} = \mathbb{E}_{q_\psi(x_0|k_0)}\big[-\log p_\theta(k_0 \mid x_0)\big], \tag{28}$$

while the continuous denoising loss separately drives $\hat{x}_0(x_t)$ toward x_0 along the reverse trajectory. This replacement is a practical surrogate for the discrete KL terms rather than an additional exact algebraic identity. These substitutions yield the compact practical objective in Eq. (6). In our implementation, we then replace the Gaussian noise-prediction term with a token-conditioned flow-matching interpolant in x-prediction form. Using $s = t/T$, the linear interpolant is

$$x_t = (1 - s)\, x_0 + s\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{29}$$

and the implemented regression loss targets the clean latent,

$$\mathcal{J}_{\text{cont}} = \sum_{t=1}^{T} \mathbb{E}_{q_\psi(x_0, x_t, k_t \mid k_0)} \left[\| d_\phi(x_t, k_t, t/T) - x_0 \|^2 \right]. \tag{30}$$

Under Eq. (29), x_0 and $\epsilon - x_0$ are related by an affine reparameterization of (x_t, t), so regressing the clean endpoint is equivalent to regressing the CFM velocity target $\epsilon - x_0$ up to a schedule-dependent reweighting. This is a practical surrogate for the continuous denoising term, not an additional exact algebraic step in the ELBO (Lipman et al., 2023). This final replacement gives the implemented $\mathcal{J}_{\text{cont}}$ used in Eq. (6). □

B Experimental Details

B.1 Architecture

Overview. The practical objective in Eq. (6) is implemented with three jointly trained transformer modules: an encoder, a token predictor, and a continuous denoiser. The reconstruction and entropy terms require a variational encoder whose latent remains decodable, while the continuous denoising term requires a denoiser network that conditions on both the noisy latent and the current token state. The interaction between these modules during training is shown in Fig. 2.

Encoder. The encoder $q_\psi(x_0 \mid k_0)$ is a bidirectional transformer that maps the token sequence k_0 to a continuous sequence $x_0 \in \mathbb{R}^{M \times d}$. We use a fully factorized Gaussian parameterization $q_\psi(x_0 \mid k_0) = \mathcal{N}\big(\mu_\psi(k_0),\, \mathrm{diag}(\sigma^2_\psi(k_0))\big)$. The final hidden states are projected to per-position mean $\mu_\psi(k_0)$ and log-variance $\log \sigma^2_\psi(k_0)$ via two independent linear heads; continuous latents are sampled by the reparameterization trick applied position-wise. This Gaussian form gives a closed-form $\log q_\psi$ for the entropy regularizer $\mathcal{J}_{\text{ent}}$.

Denoiser. The continuous denoiser $d_\phi(x_t, k_t, t/T)$ uses an x-prediction parameterization in our flow-matching implementation: it outputs the clean-latent estimate $\hat{x}_0(x_t)$ directly and is regressed against x_0 as in Eq. (30), which under the linear interpolant is equivalent to velocity regression up to a schedule-dependent reweighting. Its inputs are formed by concatenating the continuous state $x_t \in \mathbb{R}^{M \times d}$ and the learned token embeddings of the discrete state $k_t \in \mathbb{R}^{L \times d}$ along the sequence dimension, yielding a joint sequence of length $M + L$ that is processed by the transformer's self-attention. The normalized timestep t/T is injected via adaptive layer norm (adaLN) following Peebles & Xie (2023), with scale and shift parameters produced by a two-layer MLP applied to a sinusoidal embedding of t/T.

Token predictor. Rather than reading the noisy state directly, the token predictor is implemented as a lightweight transformer that consumes a clean latent and emits per-position logits over $\mathcal{V}$, which defines $p_\theta(k_0 \mid \cdot)$. At inference it reads the denoiser estimate

$$\hat{x}_0(x_t) := d_\phi(x_t, k_t, t/T),$$

and during training it reads the encoder sample x_0, following $\mathcal{J}_{\text{recon}}$ in Eq. (6). The cross-entropy therefore trains only the encoder and the predictor, and carries no gradient into the denoiser. The noisy-token distribution is obtained by marginalizing this clean-token distribution through the discrete forward kernel from Eq. (1), i.e.,

$$p_\theta(k_t \mid x_t) \;=\; \sum_{k_0} q(k_t \mid k_0)\, p_\theta\big(k_0 \mid \hat{x}_0(x_t)\big). \tag{31}$$

Algorithm 2 HC-DLM Sampling

Require: Denoiser d_ϕ; token predictor p_θ; discrete forward kernel $q(k_t \mid k_0)$ (Eq. (1)); steps T

$x_T \sim \mathcal{N}(0, I)$; $k_T \sim \pi$ ▷ Stationary token distribution

for $t = T, \ldots, 1$ **do**

  $\hat{x}_0 \leftarrow d_\phi(x_t, k_t, t/T)$ ▷ Scaffold-conditioned clean-latent estimate

  $x_{t-1} \leftarrow x_t - \frac{1}{t}(x_t - \hat{x}_0)$ ▷ Euler step (linear schedule)

  $\hat{k}_0 \leftarrow \arg\max_{k_0} p_\theta(k_0 \mid \hat{x}_0)$ ▷ Read tokens out of the latent

  $k_{t-1} \sim q(k_{t-1} \mid k_0{=}\hat{k}_0)$ ▷ Re-noise readout to level $t{-}1$

end for

return $\arg\max_{k_0} p_\theta(k_0 \mid x_0)$

This is an exact marginal for the model family that first decodes a clean sequence and then applies the known categorical noising kernel. It mirrors the conditional independence structure of Eq. (2): given k_0, the discrete corruption does not require additional information from x_t. Because the token predictor always sees an estimate of the clean latent, it is not conditioned on t and shares no parameters with the encoder or denoiser.

Sampling algorithm. Algorithm 2 lists the full sampling procedure described in Section 3.4.

B.2 Conditional Generation

For a condition c (e.g., puzzle clues, a class label, or a target sequence property), the conditional model follows by replacing each *learned* component with its conditioned counterpart while leaving the unconditional forward kernels untouched:

$$p_{\theta,\phi}(k_{0:T},\, x_{0:T} \mid c) = p(x_T)\; p_\theta(k_0 \mid x_0, c) \prod_{t=1}^{T} p_\theta(k_t \mid x_t, c)\; p_\phi(x_{t-1} \mid x_t, k_t, c), \tag{32}$$

$$q_\psi(x_{0:T},\, k_{1:T} \mid k_0, c) = q_\psi(x_0 \mid k_0, c) \prod_{t=1}^{T} q(x_t \mid x_{t-1})\; q(k_t \mid k_{t-1}). \tag{33}$$

Substituting Eqs. (32) and (33) into the derivation of Proposition 1 yields the conditional bound $\log p(k_0 \mid c) \geq \mathcal{L}(k_0, c)$ with the same four-term structure as Eq. (4), with every learned distribution conditioned on c.

The practical objective is obtained by the same substitutions in Eq. (6): the reconstruction term uses $p_\theta(k_0 \mid x_0, c)$, the continuous denoising term uses $d_\phi(x_t, k_t, t/T, c)$ in our flow-matching implementation, and the entropy term uses $q_\psi(x_0 \mid k_0, c)$. At inference time, we use the alternating sampler from Section 3.4 with these conditioned networks. In implementation, c is injected into all three learned components (the encoder q_ψ, the token predictor p_θ, and the continuous denoiser d_ϕ) as an additional input embedding fed alongside their existing inputs. We keep the three-argument signature $d_\phi(x_t, k_t, t/T)$ in the main text for notational consistency, with c understood to be appended to the discrete-state stream when conditioning is present. This is exactly the conditioning prescribed by the conditional ELBO and requires no architectural changes beyond the extra embedding.

B.3 Data and Serialization Details

Sudoku. Puzzles are drawn from the dataset of Radcliffe (2020) and split as in Kim et al. (2025). The standard split follows Shah et al. (2024), who select puzzles solvable by a fixed set of seven logical strategies without backtracking (1.8M training, 0.1M test puzzles); the hard split consists of the complementary 1.1M puzzles, which require a strategy outside this set, and we evaluate on 0.1M of them. Per-puzzle clue counts are nearly identical across the two splits (mean 24.2 vs. 24.5, range 20–31), so difficulty arises from solution structure rather than from the number of givens, and each puzzle has a unique solution. We report exact-solution accuracy (%). Quizzes and solutions are serialized as 81-token sequences over $\{1, \ldots, 9\}$, with 0 marking blank cells.

Countdown. Problems are generated following Ye et al. (2025) with targets from 10 to 100. CD4 and CD5 use four and five input numbers; CD5 requires a longer calculation chain and has a much larger search space. A prediction counts as correct only if the entire decoded chain is valid: it uses only the four permitted arithmetic operations, does not use any provided number more than once, uses only available intermediate results, permits division only when it is exact, and ends with the target value. Agreement with the dataset solution is not required. Each example is serialized as a prompt "$n_1, \ldots, n_k$ = target" (e.g., "$58, 84, 48, 62 = 96$") and a response that is a comma-separated chain of sub-equations (e.g., "$62 - 58 = 4, 48/4 = 12, 84 + 12 = 96$"). Tokens are drawn from a 17-symbol vocabulary (digits 0–9, operators $+, -, \times, \div, =$, comma, and PAD).

B.4 Training Details

Sudoku, Countdown. Our latent flow-matching model has three modules, all operating at hidden size 256. We use a single latent token ($M = 1$) in the Sudoku and Countdown experiments; an ablation over M is provided in Appendix B.9. The encoder ($\approx$ 8M parameters) consists of 8 DiT blocks following Arriola et al. (2025), in which learnable latent queries cross-attend to the embedded input sequence and produce latent tokens of dimension 256. The continuous denoiser ($\approx$ 4.9M) uses the DiT backbone from Sahoo et al. (2025) with 8 blocks, applying concatenated self-attention over $[c, x_t, k_t]$. We train with continuous-time flow matching under a linear schedule and use 50 Euler steps for the main structured-task results. The token predictor ($\approx$ 1.7M) consists of 2 DiT blocks following Arriola et al. (2025), with concatenated self-attention over $[c, \hat{x}_0]$. Both the token predictor and the denoiser share AdaLN and attention parameters across blocks following DiT-Air (Chen et al., 2025). The sampling-time generative parameters reported in the main tables comprise the denoiser and the token predictor; the encoder is used only during training. For our implementation of CCDD, we adopt its pipeline while replacing its original backbone with our continuous denoiser for fair comparison. Following CCDD, the token sequence is noised at the same level t as the latent, embedded by a separate table, and concatenated into the denoiser's self-attention stream together with the condition and the noisy latent; a second adaLN-zero head reads the token positions of the same residual stream and predicts the clean tokens, so one forward pass emits the continuous and discrete predictions in parallel. Its cross-entropy is added to our objective with a loss weight of 0.2, with all other losses and hyperparameters unchanged.

We use Adam with learning rate 1×10^{-4}, batch size 256 per GPU over 7 NVIDIA L40s GPUs. We set $\lambda_{\text{recon}} = 1.0, \lambda_{\text{ent}} = 8\times10^{-3}, \lambda_{\text{cont}} = 0.5$, dropout rate 0.1. Additionally, an Exponential Moving Average (EMA) was applied to the model weights with a decay factor of 0.999. For all tasks, we additionally use a BERT-style reconstruction loss with weight 0.2. Each input token is independently masked with probability 0.15; the encoder receives the resulting partially masked sequence, and the token decoder is trained to recover the original clean tokens. We repeat each of our reported configurations over multiple independent runs to verify stability.

LM1B. The LM1B model uses the same encoder–denoiser–predictor design and sampling procedure as the structured-task models, but scales the flow backbone to a 107.8M-parameter DiT-small with hidden size 768, 14 blocks, and 12 attention heads. The 118M parameters reported in Table 3 comprise the sampling-time denoiser and token decoder, excluding token embeddings and the training-only encoder; the complete training model has approximately 200.5M parameters. We use the Qwen3 tokenizer (Yang et al., 2025) ($|\mathcal{V}| = 151{,}669$) at sequence length 128. The LangFlow baselines use the bert-base-uncased tokenizer at the same length; GPT-2-Large scores the detokenized text, placing all models in a common evaluation space. At inference, we generate 1,024 unconditional samples with 128 sampling steps and token-channel classifier-free guidance with guidance weight $w = 2.75$ (Appendix B.7), and report their mean GPT-2-Large perplexity[1]; the ground-truth row of Table 3 is measured in the same way on LM1B text. The baseline results were obtained by the LangFlow authors after retraining the corresponding models on LM1B for 1M steps. Baseline parameter counts follow the same sampling-time, non-embedding rule. Plaid has only 0.5M embedding parameters, which we omit from its reported 109M count. LM1B training uses two stages and 660K steps in total: In the first stage, the encoder and token predictor are trained for 80K steps with a KL loss weighted by 1×10^{-4}. In the second stage, the denoiser is trained while the encoder and token predictor are

[1]Following ELF (Hu et al., 2026), we do not report validation perplexity, since likelihood evaluation for flow-based models can require additional likelihood-specific training (Ai et al., 2026).

kept fixed. We use AdamW with a global batch size of 224 and learning rate 3×10^{-4} under a linear continuous-time noise schedule. Training is performed on a single NVIDIA RTX PRO 6000 GPU with bfloat16 precision and takes approximately 48 hours.

LM1B training efficiency. Table 6 provides the reported wall-clock training times as compute context rather than as a hardware-controlled efficiency comparison. Our model is trained in approximately 48 hours on a single NVIDIA RTX PRO 6000 GPU using bfloat16 precision. Table 4 of the LangFlow paper reports approximately 292 hours for LangFlow and the LM1B baselines retrained by its authors (Transformer, MDM, SEDD Absorb, and Duo), without a separate duration for each retrained model, and approximately 375 hours for Plaid. Because the hardware and training schedules differ, these figures should be interpreted as reported end-to-end training costs rather than direct throughput measurements.

Table 6: Reported wall-clock training time for LM1B models.

Method	Training time
LangFlow and retrained baselines[†]	∼292 h
Plaid	∼375 h
HC-DLM (ours)	∼48 h

[†] LangFlow reports a shared duration for LangFlow and its retrained baselines rather than per-model measurements; hardware and training schedules also differ across the reported runs.

B.5 Sample Quality

Table 7 reports the generative perplexity and entropy across different numbers of neural function evaluations (NFEs). Both metrics are computed over 1,024 unconditional samples at each NFE.

Table 7: Sample quality across sampling budgets. Lower Gen. PPL is better.

NFE	Gen. PPL	Entropy
128	75.5	4.21
64	80.1	4.21
32	93.0	4.21
16	117.2	4.18
8	168.8	4.02
Ground truth	40.4	4.32

B.6 Inference Speed

Figure 4 compares wall-clock sampling time per batch for HC-DLM and LangFlow under the same sampling budget and sequence length on a single NVIDIA RTX PRO 6000 GPU. At larger batch sizes, HC-DLM maintains a lower time per batch than LangFlow, suggesting favorable efficiency for batched parallel generation.

B.7 Classifier-Free Guidance

Classifier-free guidance on the token channel. Although LM1B generation is unconditional at the sequence level, we use classifier-free guidance (CFG) to control how strongly the continuous denoiser follows its internal discrete token scaffold. At reverse step t, the denoiser takes the noisy continuous latent x_t, normalized time t/T, and token scaffold k_t, and predicts the clean latent, $d_\phi(x_t, k_t, t/T)$. During training, we first construct k_t normally. With probability 0.2, we replace the entire row by the maximally noised channel at normalized time $t = 1$, in which every position is an independently and uniformly sampled random token ID; we denote this uninformative channel by $\varnothing$. Because $\varnothing$ contains no information about the clean sequence, the same denoiser learns both

$$d_\phi(x_t, k_t, t/T) \approx \mathbb{E}[x_0 \mid x_t, k_t], \qquad d_\phi(x_t, \varnothing, t/T) \approx \mathbb{E}[x_0 \mid x_t]. \tag{34}$$

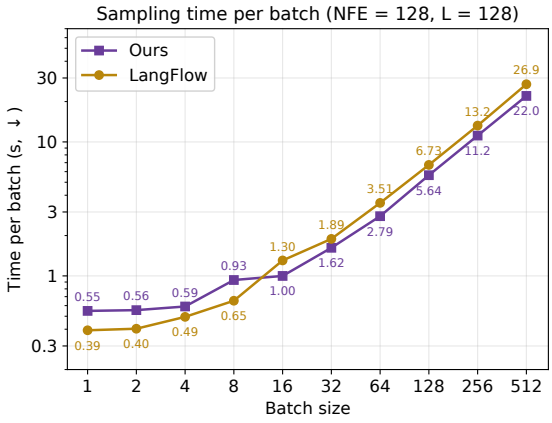


Figure 4: Wall-clock LM1B sampling time per batch at 128 NFEs and sequence length 128. HC-DLM is faster than LangFlow across all evaluated batch sizes. Lower is better.

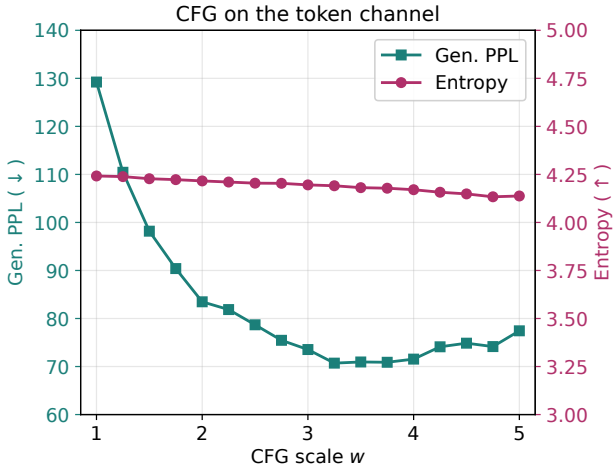


Figure 5: Effect of token-channel classifier-free guidance. We report generative perplexity (lower is better) and entropy (higher is better) as the guidance scale w varies. The main LM1B result uses $w = 2.75$; $w = 1$ corresponds to sampling without guidance extrapolation.

At each sampling step, we evaluate the denoiser twice at the same continuous state:

$$\hat{x}_{0,c} = d_\phi(x_t, k_t, t/T), \qquad \hat{x}_{0,u} = d_\phi(x_t, \varnothing_t, t/T), \tag{35}$$

where a fresh random-token sequence $\varnothing_t$ is sampled at every reverse step. We then extrapolate along the direction contributed by the token scaffold:

$$\hat{x}_0^{\mathrm{CFG}} = \hat{x}_{0,u} + w\big(\hat{x}_{0,c} - \hat{x}_{0,u}\big). \tag{36}$$

Thus, $w = 1$ recovers the ordinary scaffold-conditioned prediction, while $w > 1$ increases reliance on the token channel. We use $\hat{x}_0^{\mathrm{CFG}}$ in the Euler update of the continuous state. The token predictor then decodes $\hat{x}_0^{\mathrm{CFG}}$ into a clean token sequence, which is corrupted according to the categorical forward kernel at the next noise level to produce the next scaffold $k_{t_{\mathrm{next}}}$. This guided continuous update and token refresh are repeated at every sampling step. Fig. 5 compares generative perplexity and entropy as w varies. Across the CFG sweep, proper guidance lowers generative perplexity while preserving entropy, indicating improved sample quality without reducing diversity.

B.8 LM1B Generation Samples

Table 8 presents three unconditional generation examples from the LM1B experiment using 128 NFEs.

Table 8: Unconditional samples generated by HC-DLM on LM1B.

Example	Generated text
Example 1	first 21 points of the quarter because they rallied by 21 points .<|endoftext|>It is , after all , Obama 's favourite to win for president .<|endoftext|>Around 700,000 to 800,000 are required to pay their retails , including winter holidays , year-course , morning holidays , trips , and rigs taxes .<|endoftext|>Even with his lawyer , Dorothy Kinaitz , who has planned a separate appeal , he could soon appeal to the jury .<|endoftext|>Although Geoff Karlfeld 's Irish blood was redbed by judges Alec Dominan and Dr. McGerson , the American men
Example 2	alcohol .<|endoftext|>This year , NDO has cut its profit forecast for the second quarter of 2009 .<|endoftext|>Ÿes , you are going to have a real team that is going to get the support and remain out of the race as to how very good they 're going through , M̈cCain said .<|endoftext|>An aggressive Revolutionary Union militia , Tim Lappba , were killed in the fighting .<|endoftext|>The impact of the April quake was striking , with anticipated power managers warning from the city 's summer thunderstorms .<|endoftext|>The Blue Millions Rural Care is likely to be offered if the county fails to comply .<|endoftext|>Publishing professionals
Example 3	.<|endoftext|>Ẅe have to work together to make sure that it is difficult to do all of the things that we don 't do in the future , ̈he said .<|endoftext|>Meanwhile , a couple of elections in the northern port city of Yivazoum sent a complaint saying his support for Mr Tsvangirai and Mr Mugabe to a disastrous bid for a second term .<|endoftext|>Bill Greenberg , a Republican on the Finance Committee , said that if Congress was for delay , the State program linked to Bush 's overhaul of the financial institutions was not upstate to slip out the majority .<|endoftext|>Undle Sarben

B.9 Latent Sequence Length

The latent sequence length M controls the capacity of the continuous channel: larger M provides more continuous tokens for the continuous denoiser to operate on at the cost of additional compute. Table 9 reports Sudoku accuracy on the Easy and Hard splits as M varies in $\{1, 2, 4, 8, 16\}$, with all other hyperparameters held fixed at the structured-task values in Appendix B.4. The table suggests that increasing M improves accuracy up to a specific capacity ($M = 4$), hinting at an optimal level of latent expressivity before performance degradation.

Table 9: Ablation on latent sequence length M (Sudoku, accuracy %↑). All other hyperparameters follow the main configuration in Appendix B.4; $M = 1$ is the setting used throughout the Sudoku and Countdown experiments, while LM1B defaults to $M = 128$, with one latent per token.. For a fair comparison of early-to-mid stage optimization efficiency, we report accuracy at the same checkpoint (560 epochs).

M	Easy	Hard
1	81.73	55.81
2	91.59	67.34
4	**93.75**	**71.11**
8	71.82	45.52
16	78.80	51.89

B.10 Wall-Clock Inference Time

The step-efficiency results in Fig. 3b compare final accuracy as the number of denoising steps is reduced. Since each reverse step may have a different computational cost, we additionally report

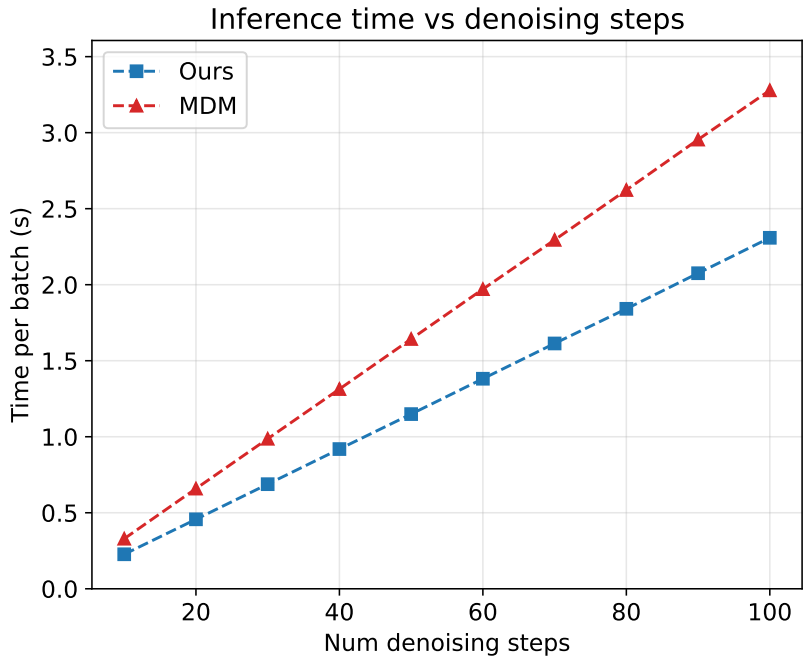


Figure 6: **Wall-clock inference time versus denoising steps.** Runtime per generated batch scales approximately linearly for both methods. Across tested step counts, HC-DLM is consistently faster, reflecting its efficient DiT implementation.

wall-clock inference time in Fig. 6. The measurement uses the same evaluation implementation and batch setting as the denoising-step ablation.

Fig. 6 shows that HC-DLM is consistently faster than MDM at the same number of steps despite using an additional continuous state. At 100 denoising steps, HC-DLM takes roughly 2.3 seconds per batch, compared with about 3.3 seconds for MDM. Together with Fig. 3b, this indicates that HC-DLM improves solution quality without increasing measured sampling latency in our implementation.

B.11 Why the Uniform Kernel Suits the Hierarchy

This section expands the analysis of the uniform–absorbing gap observed in Table 5. Two properties of the hierarchy explain it. The first is that the discrete state acts as a conditioning scaffold for the latent denoiser rather than as a ledger of commitments. A uniformly corrupted sequence is a noisy but complete hypothesis in which every position carries a concrete token, whereas absorbing corruption replaces positions with [MASK] symbols that carry no information. In our hierarchy, the "information void" that motivates continuous augmentation in CADD (Zheng et al., 2026) instead impoverishes the scaffold itself. The second is that tokens are read out from the latent anew at every step, so the uniform kernel keeps every position revisable throughout generation. Adaptive unmasking under the absorbing kernel instead freezes commitments one by one, which is costly in constraint-dense tasks where a single early error invalidates the whole solution.

C Extended Comparison with Hybrid Discrete-Continuous Diffusion

This section expands the comparison of Section 5 with three representative hybrid models that attach the continuous state to a discrete chain in different ways, namely a global latent drawn once (VMD), per-token hints (CADD), and a parallel embedding chain (CCDD). Table 10 summarizes the structural differences, and the paragraphs below give the details.

Method	Continuous variable	Update of the continuous variable	Input of the token predictor	Decoded token can change later
MDM	none	—	k_t	no
VMD	global latent	not updated (sampled once)	$k_t,\ z$	no
CADD	noised token embeddings	no learned update	$k_t,\ x_t$	no
CCDD	frozen token embeddings	learned denoiser	$k_t,\ x_t$	no
HC-DLM (ours)	global latent	learned denoiser	$\hat{x}_0$ only	yes

Table 10: Structural comparison with hybrid discrete–continuous diffusion models. MDM (Sahoo et al., 2024), VMD (Zhang et al., 2025), CADD (Zheng et al., 2026) and CCDD (Zhou et al., 2026) use the absorbing kernel, under which a token stays fixed once it is unmasked. For HC-DLM, $\hat{x}_0$ is the denoiser's clean-latent estimate.

C.1 VMD

Variational Masked Diffusion (VMD) (Zhang et al., 2025) extends masked diffusion with a single global latent z drawn once from a Gaussian prior and held fixed for all generation steps, and derives a variational bound that jointly trains an encoder of z and the masked diffusion model conditioned on it. The latent supplies static global context that cannot adapt as the partially decoded sequence changes, and the discrete chain remains the generative process. HC-DLM keeps VMD's learned variational latent but turns it into a denoising trajectory of its own, refined at every step under feedback from the token scaffold.

C.2 CADD

CADD (Zheng et al., 2026) keeps absorbing masked diffusion as the primary process and pairs each token with a continuous hint. The clean hint is the token's row in the model's own learnable embedding table w_θ, and it is corrupted by Gaussian noise only from the moment its token is masked. The hint has no learned denoiser. A position revealed at a step copies the embedding of its sampled token, and a position that stays masked takes a Gaussian posterior step towards an estimate computed from the predicted token distribution $\pi_\theta(\cdot \mid k_t, x_t)$, by default the embedding of its most likely token. The hint therefore stores the model's own earlier predictions and feeds them back to the token predictor as an extra input, so it carries no information that the token predictor did not already produce. Two consequences follow. The discrete part of their posterior does not depend on the hint (their Proposition 2), so the discrete chain alone decides which positions unmask and when. And because the hint is corrupted only once its token is masked, the construction requires the absorbing kernel and cannot use the uniform one. In HC-DLM the dependence runs the other way. The latent has a learned denoiser of its own, and the tokens are read out of it.

C.3 CCDD

CCDD (Zhou et al., 2026) defines a joint process over tokens and *frozen* embeddings of the clean sequence from a pretrained text encoder (one contextualized vector per position, e.g., from Qwen3-Embedding). As in our Eq. (2), the forward process corrupts the tokens and the embeddings independently. The reverse process is symmetric. A single backbone with two parallel heads predicts both clean states from the joint input, and each modality is then updated by its own standard rule, so the two channels interact only through the shared network inputs. Its objective is the weighted sum of the two single-channel diffusion losses. The token head reads the joint hidden state, so the token update sees k_t directly, and under the absorbing kernel an unmasked token stays fixed. HC-DLM has no separate token head, since its tokens come from the latent alone. Our experiments use this design on our backbone and learned latent rather than frozen per-token embeddings (Appendix B.4).

C.4 Summary

HC-DLM differs from all three hybrids in the last two columns of Table 10. Its token predictor reads only the latent, so every token is re-read from the latent at each step instead of being carried over

once decoded. This suits the uniform kernel, which in our experiments works clearly better than the absorbing one (Table 5).

D LIMITATIONS

HC-DLM jointly optimizes an encoder, a continuous denoiser, and a token predictor, which makes each training step more expensive than in a purely discrete masked diffusion baseline. This overhead is confined to training: the encoder is discarded at sampling time, and inference remains faster than the baseline at matched step counts (Appendix B.10). Our concatenation-based conditioning on k_t is simple and effective, though more structured fusion mechanisms may further improve sample efficiency. Finally, our experiments target structured reasoning and planning benchmarks at moderate scale, where correctness can be verified exactly and controlled comparisons fit an academic compute budget. Since all modules are standard DiT blocks and the latent channel adds only $M{=}1$ token, the design carries no scale-specific components; scaling HC-DLM to larger pretrained backbones is therefore mainly a question of compute, which exceeds the resources typically available in academia, and we consider it the most promising direction for future work.