HIERARCHICAL MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models have emerged as an alternative to autoregressive models, enabling parallel token generation. However, standard masked diffusion language models use a shared noise schedule across positions without explicitly encoding which positions should be resolved first. Many tasks exhibit a natural priority among positions: document titles and section headings can guide the generation of body text, while Sudoku cells determined directly from the initial clues can help resolve more difficult cells. To incorporate this notion of priority (or hierarchy) among token positions, we introduce the Hierarchical Masked Diffusion Language Model (HMDLM), which augments each token position with a hierarchy level available as part of the training data, and requires zero or minimal additional annotations. It factors generation into two passes: the first predicts a hierarchy distribution over masked positions, the second predicts token distributions conditioned on it. Giving each hierarchy level its own noise schedule induces a soft, controllable preference for revealing higher-priority positions earlier while keeping decoding fully parallel. Our formulation is mathematically grounded, and we re-derive the forward and reverse processes and the NELBO for this factorization. Our evaluation shows that for the task of language generation on LM1B, HMDLM reduces perplexity by more than 1.5 points over Anchored Diffusion Language Model (ADLM), a recently proposed strong diffusion baseline; on WikiHow, we see a drop in perplexity of upto 7 pts on ADLM and upto 4.4 pts against MDLM (Masked Diffusion Language Model). A LLM judge prefers HMDLM generations more than $2\times$ as often as the baseline. On the task of solving Sudoku puzzles requiring multi-step reasoning, it improves solution accuracy by up to 8.6% absolute points over existing diffusion baselines.

1 INTRODUCTION

Large language models (LLMs) have revolutionized the AI space, from general-purpose text generation to solving complex problems such as mathematical reasoning and theorem proving. Despite their success, limitations remain. One of the criticisms of modern-day LLMs, which are largely auto-regressive in nature, is that they generate text only in a strictly left-to-right manner. It has been argued that this style of decoding has specific disadvantages, such as not being able to correct past mistakes, and involving large computational costs. As a remedy, recent literature has proposed another class of models, referred to as *Diffusion Models* (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Nie et al., 2025c), which do not subscribe to the left-to-right generation style, but rather generate multiple tokens in parallel. The formulation is based on a forward diffusion process, which translates the text into noise, and on learning the reverse process, can generate the desired text by iteratively denoising from complete noise. This line of work has shown significant promise for the future of language generation, both for increasing computational efficiency and solving complex tasks (e.g., solving Sudoku or generating structured text).

A key open question in masked diffusion language models is how to choose the order in which positions should be denoised. Standard masked diffusion treats token positions symmetrically: with a shared masking schedule and no task-dependent notion of priority. Recent work shows that token ordering and decoding strategies can substantially affect generation quality (Kim et al., 2025; Garg et al., 2025; Hong et al., 2026; Fox et al., 2026). Recent approaches address this by identifying important anchor tokens (Rout et al., 2025; Xue et al., 2026), or adding planning-aware training

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.

and decoding mechanisms (Peng et al., 2025; Israel et al., 2025). These methods suggest that not all positions should be treated equally, but none provides a general hierarchy-aware forward and backward diffusion process. Such priority is often available: many problems carry a natural hierarchy over positions. In document generation, titles and section headings or summary often provide high-level semantic structure before the surrounding body text. In Sudoku, more constrained cells are easier to fill first, and their priority belongs to the position, not the digit it will contain.

We introduce HMDLM, a hierarchical extension of masked diffusion language models that associates each token position with a hierarchy variable. During the forward process, both the token and its hierarchy state are diffused toward a masked state. During the reverse process, the model first predicts the hierarchy for masked positions and then predicts token identities conditioned on the predicted hierarchy. This two-stage factorization allows the model to learn which hierarchy level each position belongs to before decoding its token value. Unlike anchor-based methods (Rout et al., 2025), which tie importance to token identities, HMDLM assigns hierarchy to positions, with a separate noise schedule per level to control how fast each level is unmasked (see Section 2). We also derive the negative evidence lower bound (NELBO) for this process which serves as our training objective 4.

We experimentally validate HMDLM across several benchmark domains. On language modeling benchmarks like LM1B, we train the model from scratch and compare with diffusion baselines (like MDLM (Sahoo et al., 2024), SEDD (Lou et al., 2024), the Anchored Diffusion Language Model (ADLM) (Rout et al., 2025), and DiffusionBERT (He et al., 2023)) on both perplexity and generation quality. On Wikihow, we fine-tune on pretrained checkpoints and compare with ADLM and MDLM on perplexity and qualitative generation. On Sudoku, we train from scratch, and more reliably produce accurate solutions than the diffusion baselines (like MGDM (Ye et al., 2025) and MDLM). These results demonstrate that explicitly modeling hierarchy over token positions can improve masked diffusion generation across both natural language and structured domains.

Our work makes the following contributions: (a) We present a novel formulation for incorporating the notion of hierarchy in the discrete diffusion models, referred to as HMDLM (b) We re-formulate the standard diffusion formulation to have a two stage training and inference process, where the first stage predicts distribution over hierarchy levels, and second one predicts token distribution given the hierarchy prediction. (c) We re-derive the ELBO to incorporate the notion of hierarchy in the forward and backward diffusion process. (d) We present experiments showing gains over masked diffusion baselines in perplexity and generation quality on LM1B and WikiHow, and report accuracy on Sudoku. We also plan to release the code for HMDLM on paper acceptance.

2 Related Work

Masked diffusion language models: Diffusion models have recently been adapted to language generation as an alternative to left-to-right autoregressive modeling. Early approaches explored diffusion in continuous word-vector spaces, as in Diffusion-LM (Li et al., 2022), or directly over discrete tokens, as in D3PMs (Austin et al., 2021) with structured transition matrices and absorbing `[MASK]` states. More recent masked diffusion language models formulate generation as iterative denoising from partially masked sequences: the forward process gradually replaces tokens with `[MASK]`, and the reverse model reconstructs the sequence from bidirectional context. SEDD (Lou et al., 2024), reparameterized discrete diffusion (Zheng et al., 2023), DiffusionBERT (He et al., 2023), MD4 (Shi et al., 2024) and MDLM (Sahoo et al., 2024) improve the objectives and sampling procedures for this setting, while large-scale systems such as MDM (Nie et al., 2025a), LLaDA (Nie et al., 2025b) and LLaDA2.0 (Bie et al., 2025) show that masked diffusion can support instruction-following LLMs. However, these models typically employ a uniform masking schedule, ignoring the inherent semantic or structural hierarchy of the task.

Generation order and decoding strategies: Masked diffusion is attractive because generation need not proceed left-to-right; tokens can be revealed in arbitrary or partially parallel orders. This flexibility also makes the choice of denoising order important. Prior work shows that adaptive or learned ordering can significantly improve masked diffusion, through inference-time token ordering (Kim et al., 2025), planner-aware training (Peng et al., 2025), planned generation where the model first constructs a high-level plan and then denoises the planned spans in parallel (Israel et al., 2025), variational order learning (Fox et al., 2026), or unified learnable-order formulations (Hong et al.,

2026). Other methods (Wang et al., 2025; Huang et al., 2025) modify sampling trajectories to address irreversibility or decoding bias. Our work is complementary: rather than learning a free-form position policy or adding an external planner, HMDLM encodes a task-dependent hierarchy in the diffusion state itself and uses per-level schedules to induce a controllable soft generation order.

Anchoring, planning, and structured priors: Several recent works also argue that diffusion language models benefit from identifying important tokens or high-level structure before completing the full sequence. Anchored Diffusion Language Models (Rout et al., 2025) observe that masking important words early can remove useful context, and use a two-stage model that predicts anchor-token distributions before predicting missing tokens. Planned Diffusion (Israel et al., 2025) first constructs an explicit plan and then generates spans in parallel. In code generation, AnCoder (Xue et al., 2026) uses syntax-tree priors to prioritize keywords and identifiers, and code-focused diffusion models such as DiffuCoder and Dream-Coder further show the value of domain-specific structure (Gong et al., 2025; Xie et al., 2025). These works share our motivation that not all sequence positions should be treated equally. However, HMDLM differs by attaching hierarchy to positions rather than only token identities, supporting multiple task-defined hierarchy levels, and using these levels to control both the forward corruption process and the reverse denoising dynamics through level-specific schedules.

3 BACKGROUND

3.1 NOTATIONS

Let x denote a sequence of discrete tokens each token x^i is represented as one-hot column vector, and $\mathcal{V} = \{x^i \in \{0,1\}^D : \sum_{j=1}^{D} x^i[j] = 1\}$ at position i as the set of all such vectors. Define $Cat(.;\pi)$ as the categorical distribution over $\mathcal{V}$ classes, where $\pi \in \mathbb{R}^D$ and $\sum_d \pi[d] = 1$. Also we assume a special $[MASK]$ token, $m \in \mathcal{V}$ and treat it as the D^{th} category, i.e., $m[D] = 1$. Let x_t denote the state of the diffusion process at time step t. The following description follows the treatment provided in Sahoo et al. (2024).

3.2 DISCRETE DIFFUSION MODELS

Forward process. Discrete diffusion process is defined using a forward process that gradually corrupts data over T time steps, and replaces each token with a special MASK token. Let $t \in \{0, 1, \ldots, T\}$, with $t = 0$ corresponding to clean data and $t = T$ to fully masked data, and $x_t^i \in \mathcal{V}$ denote progressively noisier latent variables where the timestep t ranges from 0 (zero noise) to T (maximum noise). The forward noising process q at time t and position i is defined as:

$$q(x_t^i \mid x_0^i) = \mathrm{Cat}\big(x_t^i; \alpha_t x_0^i + (1-\alpha_t)m\big)\,, \tag{1}$$

where x_0^i is the clean state, α_t is the masking schedule where $\alpha_t \in [0,1]$ is a strictly decreasing function in t, with $\alpha_0 = 1$ and $\alpha_T = 0$ controlling the proportion of unmasked tokens at time t. The corresponding posterior transition with $\lambda_t = \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}$ is:

$$q(x_{t-1}^i \mid x_t^i, x_0^i) = \begin{cases} \mathrm{Cat}(x_{t-1}^i; x_t^i), & x_t^i \neq m, \\ \mathrm{Cat}\big(x_{t-1}^i; (1-\lambda_t)m + \lambda_t x_0^i\big)\,, & x_t^i = m. \end{cases} \tag{2}$$

Reverse process. The reverse process iteratively denoises the data from timestep t to $t-1$ until the sequence becomes fully unmasked. Diffusion model aims to approximate the reverse process by learning a backward model $p_\theta(x_{t-1}^i | x_t^i)$. Since the true posterior depends on the unknown clean token x_0^i, it is approximated with a neural network $x_\theta^i(x_t, t) \in \Delta^{\mathcal{V}}$, which predicts the clean token distribution from the noisy input x_t^i at time t. During the reverse diffusion process, unmasked tokens stay fixed, and clean inputs are never masked:

$$p_\theta(x_{t-1}^i \mid x_t) = \begin{cases} \mathrm{Cat}(x_{t-1}^i; x_t^i), & x_t^i \neq m, \\ \mathrm{Cat}\big(x_{t-1}^i; (1-\lambda_t)m + \lambda_t\, x_\theta^i(x_t, t)\big)\,, & x_t^i = m. \end{cases} \tag{3}$$

NELBO. The denoising model $x_\theta(x_t, t)$ is trained using the following Negative ELBO:

$$\mathcal{L}_{\mathrm{NELBO}}(x;\theta) = \mathbb{E}_{q,t}\left[-\log p_\theta(x_0 \mid x_1) + T\left(\frac{\alpha_t - \alpha_{t-1}}{1-\alpha_t}\log\langle x_\theta(x_t,t), x_0\rangle\right)\right] \tag{4}$$

4 HIERARCHICAL MASKED DIFFUSION LANGUAGE MODEL

In this section we introduce the Hierarchical Masked Diffusion Language Model (HMDLM), which extends discrete diffusion models by incorporating a notion of hierarchy into both the forward and backward processes. Our framework augments each token position with a hierarchy level. Let h_0^i denote the gold hierarchy at position i, represented as a one-hot vector over K levels, i.e., $h_0^i \in \{0,1\}^K$. We introduce a distinct hierarchy mask state h_{m}, giving $K{+}1$ possible hierarchy states. Positions to be unmasked at time step t are governed by the level dependent masking schedule $\alpha_k(t)$ defined for time step $t \in \{0,\dots,T\}$ and level $k \in \{0,\dots,K-1\}$ which is a decreasing function. If a position i at time step t is sampled or is already unmasked, we then set $b_t^i = 0$ and $b_t^i = 1$ otherwise.

$$s_t^i = (x_t^i, h_t^i) = (x_0^i, h_0^i) \text{ if } b_t^i = 0 \text{ (revealed)}, \qquad (m, h_{\mathrm{m}}) \text{ if } b_t^i = 1 \text{ (masked)}. \tag{5}$$

Since b_t^i is shared, a token and its hierarchy level are always masked or revealed together, ruling out decoupled states such as a masked token with a revealed level.

4.1 FORWARD AND REVERSE PROCESS

Forward Process The forward process is a fixed Markov chain, that progressively masks tokens from the initial state s_0 to a noisy state s_t at each time step t over T time steps, according to the hierarchy conditioned noise schedule $\alpha_k(t)$. Tokens at lower hierarchy levels are masked more aggressively at early time-steps, than tokens at higher hierarchy levels. The forward marginal at a single token position i with hierarchy k and joint state $s_{\mathrm{t}}^i = (x_{\mathrm{t}}^i, h_{\mathrm{t}}^i)$ is given by:

$$q(s_t^i \mid s_0^i) = \mathrm{Cat}\left(s_t^i;\, \alpha_{h_0^i}(t)\, s_0^i + (1 - \alpha_{h_0^i}(t))\, s_m\right) \tag{6}$$

At $t = T$, all tokens are fully masked regardless of the hierarchy level, i.e. $\alpha_k(T) = 0$. Let $\lambda_k(t) = \frac{\alpha_k(t-1) - \alpha_k(t)}{1 - \alpha_k(t)}$ denote the probability that a masked level-k position is revealed at step t. The true joint posterior is:

$$q(s_{t-1}^i \mid s_t^i, s_0^i) = \begin{cases} \mathrm{Cat}(s_{t-1}^i; s_t^i), & s_t^i \neq \mathbf{s}_m, \\ \mathrm{Cat}\Big(s_{t-1}^i; (1 - \lambda_{h_0^i}(t))\mathbf{s}_m + \lambda_{h_0^i}(t) s_0^i\Big), & s_t^i = \mathbf{s}_m. \end{cases} \tag{7}$$

Reverse Process The reverse process aims to recover the clean sequence s_0 from a fully masked sequence s_T over T denoising steps. We approximate the reverse process $p_\theta(s_{t-1} \mid s_t, t)$ using the neural network $s_\theta(s_t, t)$, which is composed of two joined networks - the hierarchy head, $h_\theta(x_t, h_t, t)$ and the token head, $x_\theta(x_t, h_{t-1}, t)$. It takes as input the noisy state $s_t{=}(x_t{,}h_t)$ at timestep t and predicts the clean joint state s_0. Revealed positions are carried over unchanged, $p_\theta(s_{t-1}^i \mid s_t^i) = \mathrm{Cat}(s_{t-1}^i; s_t^i)$ if $s_t^i \neq \mathbf{s}_m$. For a masked position ($s_t^i = \mathbf{s}_m$), the model first draws the hierarchy state and then the token conditioned on it:

$$\begin{aligned} p_\theta\big(h_{t-1}^i = k \mid x_t, h_t\big) &= \lambda_k(t)\, g_k^i(t), && k \in \{0,\dots,K-1\}, \\ p_\theta\big(h_{t-1}^i = h_{\mathrm{m}} \mid x_t, h_t\big) &= 1 - \pi_i(t), && \\ p_\theta\big(x_{t-1}^i \mid x_t, h_{t-1}\big) &= \mathrm{Cat}\big(x_{t-1}^i;\, f^i(t)\big), && h_{t-1}^i \neq h_{\mathrm{m}}, \\ p_\theta\big(x_{t-1}^i = m \mid x_t, h_{t-1}\big) &= 1, && h_{t-1}^i = h_{\mathrm{m}}, \end{aligned} \tag{8}$$

where $g^i(t) = \mathrm{softmax}\big(h_\theta^i(x_t, h_t, t)\big)$ is the hierarchy head's distribution over the K levels, $f^i(t) = \mathrm{softmax}\big(x_\theta^i(x_t, h_{t-1}, t)\big)$ is the token head's distribution over the vocabulary, and $\pi_i(t) = \sum_{k=0}^{K-1} \lambda_k(t)\, g_k^i(t) = \mathbb{E}_{k \sim g^i(t)}\big[\lambda_k(t)\big]$, is the total probability that position i is revealed at step t. Since the gold hierarchy h_0^i is unknown in the reverse process, $\pi_i(t)$ replaces the true reveal probability $\lambda_{h_0^i}(t)$ of equation 7 by its expectation under the predicted hierarchy distribution. Because $\lambda_{k'}(t) > \lambda_k(t)$ for $k' > k$, a masked position is more likely to be revealed at a higher level than at a lower one even when $g^i(t)$ is uniform, which is what induces the soft generation order (Appendix D).

4.2 NELBO

We derive the discrete time NELBO 9 and its continuous-time limit 77 for the generative process $p_\theta(s_0)$. The detailed derivation is given in the Appendix A. The discrete ELBO is given as:

$$\log p_\theta(\mathbf{s}_0) \;\geq\; \underbrace{\mathbb{E}_q\big[\log p_\theta(\mathbf{s}_0 \mid \mathbf{s}_1)\big]}_{L_0} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_q\Big[D_{\mathrm{KL}}\big(q(\mathbf{s}_{t-1} \mid \mathbf{s}_t, \mathbf{s}_0) \parallel p_\theta(\mathbf{s}_{t-1} \mid \mathbf{s}_t)\big)\Big]}_{L_t}. \tag{9}$$

Each L_t is a KL over the state $(\mathbf{s}_t) = (x_t, h_t)$. By the chain rule of KL divergence, L_t can be expanded as:

$$= \underbrace{D_{\mathrm{KL}}\big(q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \parallel p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)}_{\text{hierarchy KL (Pass 1)}}$$

$$+ \underbrace{\mathbb{E}_{q(\mathbf{h}_{t-1}|\mathbf{x}_t,\mathbf{h}_t,\mathbf{x}_0,\mathbf{h}_0)}\Big[D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1})\big)\Big]}_{\text{token KL (Pass 2)}}. \tag{10}$$

The KL divergence splits into two terms 10, which we model using two stages in our architecture introduced in 4.3.

4.3 Model Architecture

4.3.1 Two Stage Model

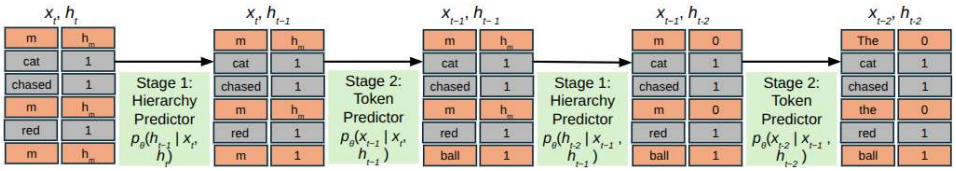


Figure 1: **HMDLM inference example.** Positions $i = 0, 3, 5$ are masked in (x_t, h_t). Step t: Stage 1 reveals $h^5_{t-1} = 1$, then Stage 2 predicts $x^5_{t-1} =$ ball. Step $t-1$: Stage 1 reveals $h^0_{t-2} = 0$ and $h^3_{t-2} = 0$, then Stage 2 predicts $x^0_{t-2} =$ The and $x^3_{t-2} =$ the.

To model the factorized posterior, we introduce the following two stage architecture 1.

First Stage: Given the partially denoised sequence (x_t, h_t) at time step t, the *first stage* predicts the hierarchy distribution $g^i(t)$ at every masked position $i \in \mathcal{M}_t$, where $\mathcal{M}_t$ is the set of masked positions at time t; together with the noise schedule, this defines $p_\theta\big(h^i_{t-1} \mid x_t, h_t\big)$ in equation 8. Masked positions are fed the masked state $\mathbf{s}_m$ as input to Stage I during both training and inference. During training, unmasked positions are provided with their gold pairs (x^i_0, h^i_0) via teacher forcing. During inference, they are provided with the token and hierarchy sampled at the step when the position was first revealed, which remain fixed thereafter. At each inference step t, every masked position i draws its next hierarchy state from equation 8: it is revealed at level $k \in \{0, \ldots, K-1\}$ with probability $\lambda_k(t)\, g^i_k(t)$ and stays masked with probability $1 - \pi_i(t)$. This single draw decides both whether position i is revealed at this step and at which level, which forms the set of revealed positions $\mathcal{U}_t \subseteq \mathcal{M}_t$.

Second Stage: The *second stage* predicts the token distribution at each masked position i at time step t: $p_\theta(x^i_{t-1} \mid x_t, h_{t-1})$. For masked positions, it receives as input (x_t, h_{t-1}) with the hierarchy at the reveal set filled in: positions $i \in \mathcal{U}_t$ are assigned gold hierarchy h^i_0, (m, h^i_0) during training and the Stage-I sample (m, h^i_{t-1}) during inference. For remaining masked positions, we pass the mask state s_m = (m, h_m) as input. For unmasked positions, the token and hierarchy are copied directly from x_t and h_t without modification: $p_\theta(s^i_{t-1} \mid s^i_t) = \mathrm{Cat}\,\big(s^i_{t-1};\, s^i_t\big) \quad$ if $s^i_t \neq \mathbf{s_m}$ during inference and passed s^i_0=(x^i_0,h^i_0) during training. Both stages use weight tying, with separate output heads (hierarchy predictor head and token predictor head) attached to the final hidden layer and a softmax applied at the end of each head. This design reduces the parameter count while ensuring that both stages benefit from the same rich contextual representation.

Note: Across iterations, the token and hierarchy at each position, $s_t^i = (x_t^i, h_t^i)$, are masked and unmasked jointly through a shared Bernoulli variable 5 . Within a single iteration, however, Stage 1 predicts the hierarchy before Stage 2 predicts the token. The partially unmasked state $(m,\ h_{t-1}^i)$ with $h_{t-1}^i \neq h_{\mathrm{m}}$ therefore arises only between the two stages of the same iteration.

4.3.2 Per-level Noise Schedule and Hierarchy Embeddings

We use a different masking schedule for each hierarchy level k. Tokens that are more informative are given a higher hierarchy level and a slower masking schedule. Consequently, during the forward noising process, these tokens are masked more slowly than lower level tokens, preserving them for longer. They are denoised faster than other tokens during the reverse de-noising process. These tokens are unmasked earlier than lower-hierarchy tokens, allowing the model to condition on higher level tokens when predicting the remaining masked positions.

The hierarchy of a position is represented by a row of the learnable hierarchy embedding matrix $W \in \mathbb{R}^{(K+1)\times H}$, where K is the number of hierarchy levels and H is the hidden dimension. Row W_k is the embedding of level $k \in \{0, \ldots, K-1\}$, and the last row W_K is the embedding of the hierarchy mask state h_{m}. The hierarchy embedding for each position i is added to the token embedding at position i and fed as input to the backbone.

4.4 Training and Inference

Training The HMDLM model is trained end-to-end across both the passes. We sample a time step t, and mask each token independently according to the token's hierarchy level masking schedule. The input to the Stage 1 is (x_t, h_t) and it predicts the hierarchy probabilities - $p_\theta(h_{t-1}^i \mid x_t, h_t)$, we pass updated (x_t, h_{t-1}) to Stage 2 to get token distribution - $p_\theta(x_{t-1}^i \mid x_t, h_{t-1})$ for each masked position. The final training loss is governed by:

$$\mathcal{L} = \mathbb{E}_{t,q}\left[\sum_{i=1}^{L} \mathbf{1}[x_t^i = m]\left(\lambda'_{h_0^i}(t)\left(-\log f_{x_0^i}^i(t) - \beta_{\mathrm{hier}} \log g_{h_0^i}^i(t)\right) + \beta_{\mathrm{cons}}\left(\textstyle\sum_k g_k^i(t)\,\lambda'_k(t) - \lambda'_{h_0^i}(t)\right)\right)\right] \tag{11}$$

where β_h and β_e are tunable hyperparameters, $\lambda'_k(t) = \frac{-\alpha'_k(t)}{1-\alpha_k(t)}$, and h_0^i is the gold hierarchy and x_0^i is the gold token, $g_k^i(t)$ and $f_v^i(t)$ are the hierarchy-head and token-head probabilities of equation 8 at position i. The full training procedure is given in Algorithm 1.

Inference During the reverse process, we start from a fully masked sequence and iteratively unmask tokens at each time step t. At time step $t = T$ all the tokens are $[MASK]$ tokens and hence are assigned masked state, s_m. In Stage I, we sample hierarchy value for each masked token position i at time step t from the predicted hierarchy probabilities as:

$$h_{t-1}^i \sim \mathrm{Cat}\big(\lambda_0(t)\, g_0^i(t),\ \ldots,\ \lambda_{K-1}(t)\, g_{K-1}^i(t),\ \underbrace{1 - \pi_i(t)}_{\text{stay masked}}\big) \tag{12}$$

with $g^i(t)$ and $\pi_i(t)$ as in equation 8, and $\lambda_k(t)$ the per-step reveal probability of level k. Higher levels are masked more slowly, and for the schedules we use $\lambda_k(t) \leq \lambda_{k'}(t)$ for $k < k'$ at every step at which level k' is still masked (Appendix D). A position predicted to be at level k' therefore receives a larger reveal mass $\lambda_{k'}(t)\, g_{k'}^i(t)$ and is revealed earlier. The full inference algorithm is given in the Appendix (Algorithm 2).

5 Experiments

In this section, we evaluate the proposed Hierarchical Masked Diffusion Language Model (HMDLM) along two axes: (1) its ability to model language distributions, measured via perplexity, and (2) its effectiveness as a puzzle solver, measured by accuracy. We consider three representative benchmarks: the One Billion Word Benchmark (LM1B) (Chelba et al., 2013) and WikiHow (Koupaee & Wang, 2018) for generative modeling, and Sudoku (Palm et al., 2018)

5.1 Language Generation Tasks

Setup. For both LM1B and WikiHow, HMDLM adopts the MDLM backbone (Sahoo et al., 2024): 12 transformer blocks, hidden size 768, 12 attention heads, rotary positional embeddings (Su et al., 2024), and the dataset-dependent tokenizer for each corpus: BERT-base-uncased for LM1B and GPT-2 for WikiHow. The hierarchy-specific components ($K = 2$ levels) are (i) a shared embedding matrix $W \in \mathbb{R}^{(K+1)\times H} = \mathbb{R}^{3\times 768}$, with K learnable level embeddings plus a learnable [MASK]-hierarchy row for unrevealed positions, and (ii) a hierarchy-prediction head that mirrors the token head's adaLN-modulated (Peebles & Xie, 2023) final layer, mapping $\mathbb{R}^{768} \to \mathbb{R}^2$. Together these add ≈ 0.2M parameters, about 0.15% of the 139M-parameter LM1B backbone and 0.12% of the 170M-parameter WikiHow backbone. The Sudoku architecture differs and is described in Section 5.2.

Metrics. We evaluate generation quality with four complementary measures. Model PPL assesses density modeling. Generative PPL assesses fluency of sampled text, but can be artificially lowered by leaning on a few high-likelihood phrases, producing repetitive, low-entropy output. We therefore read it alongside two diversity measures: entropy, where a value close to the reference or a high value indicates the model has not collapsed onto safe phrases, and MAUVE, which measures the similarity between the model-generated and reference text distributions. To corroborate the above metrics, we also run a pairwise LLM-as-judge study (GPT-5 judge) that reports how often each model's generations are preferred.

5.1.1 LM1B Dataset

Task Description: We first evaluate the generative modeling capability of HMDLM by training it from scratch on LM1B, a standard dataset for language modeling. We compare test perplexity against several diffusion-based baselines: MDLM (Sahoo et al., 2024), SEDD (Lou et al., 2024), the Anchored Diffusion Language Model (ADLM) (Rout et al., 2025), and DiffusionBERT (He et al., 2023).

Setup: Using the shared language backbone (Section 5.1), we set a context length of 128 and train HMDLM from scratch for 1.25M and 2.5M steps ($\approx$ 33B and $\approx$ 65B tokens) on LM1B (see Section B).

Hierarchy Assignment: We define two hierarchy levels on LM1B by in-sentence word frequency (τ) following Rout et al. (2025). Tokens appearing at most τ times in a sample are assigned to level 1 (important words) and the rest to level 0 (common words). Although ADLM reports numbers with $\tau = 5$, this setting provides limited hierarchy separation: majority of the tokens get hierarchy 1 (99.16%). Under $\tau = 1$, around 79% tokens get level 1, and the rest get assigned level 0 - yielding a more meaningful hierarchy.

Table 1: Test perplexities (PPL ↓) on LM1B. Best value is bolded.

Model	PPL (↓)
33B tokens	
Diffusion BERT (He et al., 2023)	≤ 63.78
SEDD (Lou et al., 2024)	≤ 32.79
MDLM (Sahoo et al., 2024)	≤ 27.04
ADLM (Rout et al., 2025)	≤ 26.40
HMDLM	$\leq \mathbf{24.30}$
65B tokens	
MDLM (Sahoo et al., 2024)	≤ 25.49
ADLM (Rout et al., 2025)	≤ 24.46
HMDLM	$\leq \mathbf{22.88}$

Noise Schedule: We use a linear denoising schedule (identical across all models for a fair PPL comparison) with a *level offset* $\delta = 0.20$, defined for HMDLM as $\alpha_0(t) = 1-(1-\varepsilon)t$ and $\alpha_1(t) = 1$ for $t \in [0, \delta]$, while $\alpha_1(t) = 1 - (1-\varepsilon)\frac{t-\delta}{1-\delta}$ for $t \in (\delta, 1]$.

Results: We generate 1000 samples across three seeds for each model (128 token length and 256 steps). As can be seen from the Table 1 and Table 2, HMDLM attains the best model PPL and generative perplexity (measured using GPT-2 Large) among the diffusion baselines at the 65B-token budget, cutting generative PPL by

Table 2: LM1B generative evaluation on 65B checkpoints. Mean ± std over 3 seeds.

Model	Gen. PPL (↓)	MAUVE* (↑)	Entropy
MDLM	$177.94_{\pm 0.82}$	$0.66_{\pm 0.03}$	$6.022_{\pm 0.003}$
HMDLM	$\mathbf{83.35}_{\pm 0.83}$	$\mathbf{0.76}_{\pm 0.06}$	$5.876_{\pm 0.014}$

*MAUVE is reported here in %.

nearly 95 points relative to MDLM [1]. The
reduction in generative PPL is accompanied by only a small decrease in entropy (5.876 vs. 6.022), providing no clear evidence of severe diversity collapse. MAUVE scores, though low, are higher for HMDLM as compared to MDLM. An independent LLM-as-judge evaluation (prompt: G) is consistent with the results: in blinded pairwise comparisons with randomized A/B assignment (100 samples per seed, 3 seeds, 2 runs each; 600 comparisons total), the judge preferred HMDLM to MDLM 56% to 44%, with no ties. Under the $\tau = 5$ setting, HMDLM performs similar to MDLM, as nearly all real tokens are assigned to the same hierarchy level, leaving little scope for the hierarchical mechanism to provide an advantage.

5.1.2 WIKIHOW DATASET

Task Description: We next evaluate generative modeling on WikiHow. We initialize HMDLM from the publicly released MDLM checkpoint pretrained on OpenWebText (OWT) (Gokaslan & Cohen, 2019), which we refer to as MDLM-OWT, and fully fine-tune it on WikiHow. As baselines, we fine-tune the MDLM-OWT and ADLM-OWT checkpoints on WikiHow under an identical setting.

Table 3: Model PPL (↓) across sequence lengths on WikiHow. Best value is in bold.

L	MDLM	HMDLM-P[†]	ADLM	HMDLM-F[‡]
256	24.60 ± 0.22	23.89 ± 0.11	22.05 ± 0.21	$\mathbf{17.62 \pm 0.12}$
512	21.04 ± 0.06	21.76 ± 0.15	20.15 ± 0.10	$\mathbf{16.64 \pm 0.06}$
1024	22.65 ± 0.04	22.58 ± 0.04	20.92 ± 0.05	$\mathbf{16.90 \pm 0.09}$

[†]HMDLM-P: position-based hierarchy. [‡]HMDLM-F: frequency-based hierarchy.

Setup: We reuse the shared language backbone (Section 5.1), fully fine-tuning from the MDLM-OWT checkpoint. We experiment with maximum sequence lengths of $L \in \{256, 512, 1024\}$, filtering WikiHow to corresponding token length. The train, val, and test set corresponding to each token length L is kept consistent across all models for fair comparison. During inference, we use 256 sampling steps for all three sequence lengths. All the baseline models are trained for 30 epochs. Both the HMDLM variants are trained slightly longer because of lower expected masking fraction. Detailed discussion in Appendix C.

Hierarchy Assignment: We consider two hierarchy-assignment strategies. In the *position-based* scheme, we assign level 1 to summary tokens and level 0 to the rest; here we compare directly against the fine-tuned MDLM model, since ADLM defines its anchor tokens by frequency rather than position and is therefore not comparable under this assignment. In the *Frequency-based* scheme, we reuse the frequency rule from LM1B, assigning level 1 to tokens occurring at most once in a sentence and level 0 otherwise; this is directly comparable with ADLM, as we apply the same rule to define its anchor positions.

Table 4: Generative evaluation on WikiHow across sequence lengths $L \in \{256, 512, 1024\}$. HMDLM-P/F: position-/frequency-based hierarchy. Gold is the reference data.

	Gen. PPL			MAUVE (↑)			Entropy		
Model	256	512	1024	256	512	1024	256	512	1024
Gold	25.2	21.0	20.8	1.00	1.00	1.00	5.86	6.42	6.70
MDLM	76.0	67.4	62.7	0.33	0.11	0.12	6.47	7.10	7.60
ADLM	84.0	82.1	93.81	0.40	0.21	0.08	6.49	7.17	7.77
HMDLM-P	23.1	17.6	14.6	0.38	0.13	0.07	5.91	6.26	6.45
HMDLM-F	29.2	25.4	19.92	**0.52**	**0.41**	**0.20**	5.98	6.60	6.83

Noise Schedule: We use the power-function schedule $\alpha_h(t) = 1 - (1 - \epsilon)\, t^{\omega_h}$ with $\omega_h[0] = 1.0$ and $\omega_h[1] = 2.0$. Setting $\omega_h[0] = 1.0$ recovers the log-linear schedule of MDLM.

Results: HMDLM-F (Frequency based hierarchy) achieves the best model PPL and its generative PPL is closest to reference distribution. HMDLM-P (Position based hierarchy) achieves the best gen PPL, on all three generation lengths (Tables 3 and 4), cutting gen PPL by nearly 50 points relative to the baselines. The improvement is not repetition-driven: MAUVE is higher for HMDLM-F at every length, and entropy stays close to that of the reference dataset, indicating a closer match in terms of token distribution. LLM-as-judge evaluation (prompt: G) is consistent with the results: in blinded comparisons (400 samples, 2 runs each; 800 comparisons total), the judge preferred HMDLM-P 40.75% times, HMDLM-F 31.88% times, ADLM 15.25%, and MDLM 12.12%. Therefore, HMDLM

[1]All the baseline results in Table 1 are taken from Rout et al. (2025). We tried training the ADLM model for comparison in Table 2, but the gradients were exploding.

combined was preferred more than twice the times baselines were selected. We also find that, for this dataset, HMDLM-F has lower model PPL on test set, better MAUVE scores and entropy closer to that of reference data than the HMDLM-P for all lengths.

Position- vs. frequency-based hierarchy: On this dataset, HMDLM-F outperforms HMDLM-P in model PPL, MAUVE, and entropy (closer to reference distribution) at all lengths. We hypothesize that generating key tokens first matters more than their positions, since these tokens provide enough context to decode the rest of the text. This is domain-dependent: for tasks like Sudoku, position-based hierarchy is likely essential. Both variants improve over MDLM, and while ADLM, which is implicitly frequency-driven, is competitive with HMDLM-P, HMDLM-F consistently outperforms ADLM. This highlights the benefit of our principled treatment of hierarchy in diffusion models.

5.2 Sudoku Dataset

Task Description: Sudoku is a useful test case for structured reasoning. Some cells can be filled directly from the given numbers, while others can be resolved only after several other cells have been filled; revealing the easy cells first therefore makes the harder cells easier to predict. In contrast, the current diffusion models (MDLM and MGDM (Ye et al., 2025)) reveal cells without exploiting such an order. Each puzzle has 81 cells and some given cells. We use the Sudoku dataset of Palm et al. (2018) with 180k training, 18k validation, and 18k test puzzles. The given cells are never masked and are excluded from the loss.

Hierarchy labels. Starting from the given cells, we repeatedly fill the masked cells that can be determined directly. A cell is assigned a level according to the round in which it is first determined. We use six levels ($K = 6$): level 5 contains cells determined directly from the given cells, while lower levels contain cells that require more rounds of solving. Level 0 contains cells that are determined only after six or more rounds, or that are never determined by these rules and therefore require search. At test time, the levels of blank cells are predicted by the hierarchy head. Architecture details for both variants (25M and 6M) are provided in Appendix F.

Table 5: Sudoku-hard results: 25M parameter models under random & top-k=1 decoding.

Metric	Model	stochastic	Greedy
Board Acc.	MDLM	0.81	0.83
	HMDLM	**0.86**	**0.88**
Cell Acc.	MDLM	0.94	0.94
	HMDLM	**0.96**	**0.97**

Results. HMDLM consistently outperforms the baseline in both solve rate and cell accuracy: for the 25M model it wins under both stochastic and greedy decoding (Table 5), and the same advantage holds for the compact 6M model (Table 6). We also analyze the cell accuracy for each hierarchy level (Table 8), HMDLM shows the largest gains, up to 8% over MDLM for level 0 cells (or the cells that are most difficult to fill). Additionally, we ran a qualitative analysis on Sudoku puzzles and found that, for the 25M model, among 287 puzzles solved by HMDLM but not MDLM across all three seeds, 45.6% showed MDLM making an early commitment to a search cell, while HMDLM delayed filling the same cell until fewer candidates remained. Details in Appendix F

Table 6: Sudoku-hard test set (18k puzzles). Results are from 6M-parameter models trained for 100k steps with $T = 81$ and confidence-based sampling. Mean $\pm$ std over 3 sampling seeds.

Model	Board Acc.	Cell Acc.
MGDM	63.07 ± 0.18	85.79 ± 0.10
HMDLM	$\mathbf{71.69 \pm 0.26}$	$\mathbf{91.13 \pm 0.07}$

6 Conclusion and Limitations

We introduced HMDLM, a hierarchical extension of masked diffusion language models that assigns priority to token positions and incorporates it into both the forward and reverse processes. By using level-specific noise schedules and a two-stage architecture that first predicts hierarchy and then tokens, HMDLM enables structured and controllable generation while remaining lightweight and modular. Our model consistently outperforms strong diffusion baselines by decoding semantically important tokens early and using them to better condition the rest of the sequence. While this improvement comes at the cost of additional computation due to the two-stage design, our results highlight hierarchy-aware generation as a promising direction for diffusion-based language models.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implement methods, clean and reformat datasets, writing proofs. We have not used generative AI tools to formulate mathematical claims, develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, interpret results, propose or refine hypotheses, or design or provide feedback on research methodology or experiments. Generating synthetic datasets, assisting with translation, and supporting qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or modify figures, draft parts of the paper, and search for information. We have reviewed all AI-assisted work. In particular, LLM-generated code and proofs were verified and tested for correctness by at least two authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

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. *arXiv preprint arXiv:1312.3005*, 2013.

David Fox, Sam Bowyer, Song Liu, Laurence Aitchison, Raul Santos-Rodriguez, and Mengyue Yang. Learning generation orders for masked discrete diffusion models via variational inference. In *2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy, ICLR 2026*, 2026.

Prateek Garg, Bhavya Kohli, and Sunita Sarawagi. Masked diffusion models are secretly learned-order autoregressive models. In *EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM)*, 2025.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. 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)*, pp. 4521–4534, 2023.

Chunsan Hong, Sanghyun Lee, and Jong Chul Ye. Unifying masked diffusion models with various generation orders and beyond. *arXiv preprint arXiv:2602.02112*, 2026.

Pengcheng Huang, Shuhao Liu, Zhenghao Liu, Yukun Yan, Shuo Wang, Zulong Chen, and Tong Xiao. Pc-sampler: Position-aware calibration of decoding bias in masked diffusion models. *arXiv preprint arXiv:2508.13021*, 2025.

Daniel Israel, Tian Jin, Ellie Cheng, Guy Van den Broeck, Aditya Grover, Suvinay Subramanian, and Michael Carbin. Planned diffusion. *arXiv preprint arXiv:2510.18087*, 2025.

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 *International Conference on Machine Learning*, pp. 30749–30768. PMLR, 2025.

Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset, 2018. URL `https://arxiv.org/abs/1810.09305`.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

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*, pp. 32819–32848, 2024.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 82974–82997, 2025a. URL `https://proceedings.iclr.cc/paper_files/paper/2025/file/ce1c1ff5d94079dea348a2317a889281-Paper-Conference.pdf`.

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. *arXiv preprint arXiv:2502.09992*, 2025b.

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, 2025c. URL https://arxiv.org/abs/2502.09992.

Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 3372–3382. Curran Associates, Inc., 2018. URL http://nips.cc.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Anru R Zhang, Michael Bronstein, Alexander Tong, and Avishek Joey Bose. Planner aware path learning in diffusion language models training. *arXiv preprint arXiv:2509.23405*, 2025.

Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. Anchored diffusion language model, 2025. URL https://arxiv.org/abs/2505.18456.

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. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Guanghan Wang, Yair Schiff, Subham Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=IJryQAOy0p.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

Anton Xue, Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. Ancoder: Anchored code generation via discrete diffusion models. *arXiv preprint arXiv:2602.17688*, 2026.

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 *International Conference on Learning Representations (ICLR)*, 2025.

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

A DISCRETE-TIME ELBO

A.1 JOINT GENERATIVE MODEL $p_\theta(\mathbf{x}_0, \mathbf{h}_0)$

Hierarchy is jointly modelled with tokens. We represent the state of the model at time step t as $\mathbf{s_t} = (\mathbf{x_t}, \mathbf{h_t})$. When token i is masked ($x_t^i = M$), *both* the token and its hierarchy level are masked: the token is replaced by `[MASK]` and h_t^i is set to masked hierarchy state (h_{MASK}). HMDLM therefore targets the joint distribution:

$$\log p_\theta(\mathbf{x}_0, \mathbf{h}_0). \tag{13}$$

For $s < t$ and a masked position ($x_t^i = m$), both x_0^i and h_0^i are revealed together upon unmasking, with the hierarchy fixed consistently to $h_0^i = f(x_0^i)$:

$$(x_s^i,\ h_s^i) = \begin{cases} (x_0^i,\ h_0^i) & \text{with prob.} \quad \dfrac{\alpha_{h_0^i}(s) - \alpha_{h_0^i}(t)}{1 - \alpha_{h_0^i}(t)} \quad \text{(jointly revealed)} \\ (m,\ h_{mask}) & \text{with prob.} \quad \dfrac{1 - \alpha_{h_0^i}(s)}{1 - \alpha_{h_0^i}(t)} \quad \text{(jointly remain masked)} \end{cases} \tag{14}$$

A.2 STEP 1 — INTRODUCE THE JOINT DIFFUSION PROCESS

We target the *joint* distribution $\log p_\theta(\mathbf{x}_0, \mathbf{h}_0)$. The joint $(\mathbf{x}_t, \mathbf{h}_t)$ is the diffusion state at time t: when $x_t^i = M$, both the token and its hierarchy label are masked and unobserved; when $x_t^i \neq M$, both are revealed together. Introduce the latent noisy sequence of joint states $(\mathbf{x}_{1:T}, \mathbf{h}_{1:T})$ at T discrete time steps $t_1 < \cdots < t_T = 1$:

$$\log p_\theta(\mathbf{x}_0, \mathbf{h}_0) = \log \int p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T})\ d\mathbf{x}_{1:T}\, d\mathbf{h}_{1:T}. \tag{15}$$

Because each denoising step depends only on the current state and not on the whole history, the joint model breaks into a product of one-step reverse predictions:

$$p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T}) = p(\mathbf{x}_T, \mathbf{h}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t) \tag{16}$$

We now split each reverse step into two parts: first predict the hierarchy, then predict the token given that hierarchy. We also drop $\mathbf{h}_t$ from the token prediction, since $\mathbf{h}_{t-1}$ already carries the hierarchy information needed. Finally, since the last state is fully masked and fixed, its prior splits into a product $p(\mathbf{x}_T)\, p(\mathbf{h}_T)$. This gives:

$$p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T}) = p(\mathbf{x}_T)\, p(\mathbf{h}_T) \prod_{t=1}^{T} \underbrace{p_\theta(\mathbf{h}_{t-1} \mid \mathbf{h}_t,\ \mathbf{x}_t)}_{\text{Pass 1: hierarchy}} \cdot \underbrace{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{h}_{t-1},\ \mathbf{x}_t)}_{\text{Pass 2: tokens}}. \tag{17}$$

Multiply and divide by the forward process $q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0)$ in 15:

$$\log p_\theta(\mathbf{x}_0, \mathbf{h}_0) = \log \mathbb{E}_{q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T})}{q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0)} \right]. \tag{18}$$

A.3 STEP 2 — APPLY JENSEN'S INEQUALITY

Since $\log$ is concave, Jensen's inequality gives:

$$\log p_\theta(\mathbf{x}_0, \mathbf{h}_0)\ \geq\ \mathbb{E}_{q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0)} \left[\log \frac{p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T})}{q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0)} \right]. \tag{19}$$

A.4 Step 3 — Expand the Log of the Joint Model

The forward process factorises as:

$$q(\mathbf{x}_{1:T}, \mathbf{h}_{1:T} \mid \mathbf{x}_0, \mathbf{h}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1}). \tag{20}$$

Taking the log of equation 17 gives:

$$\begin{aligned}\log p_\theta(\mathbf{x}_{0:T}, \mathbf{h}_{0:T}) = \log p(\mathbf{x}_T) + \log p(\mathbf{h}_T) + \underbrace{\sum_{t=1}^{T} \log p_\theta(\mathbf{h}_{t-1} \mid \mathbf{h}_t,\, \mathbf{x}_t)}_{\text{Pass 1: hierarchy}} \\ + \underbrace{\sum_{t=1}^{T} \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{h}_{t-1},\, \mathbf{x}_t)}_{\text{Pass 2: tokens}}.\end{aligned} \tag{21}$$

Substituting into equation 19:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0, \mathbf{h}_0) \;\geq\; \mathbb{E}_q\Big[&\log p(\mathbf{x}_T) + \log p(\mathbf{h}_T) \\ &+ \sum_{t=1}^{T} \log p_\theta(\mathbf{h}_{t-1} \mid \mathbf{h}_t,\, \mathbf{x}_t) \\ &+ \sum_{t=1}^{T} \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{h}_{t-1},\, \mathbf{x}_t) \\ &- \sum_{t=1}^{T} \log q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1})\Big].\end{aligned} \tag{22}$$

A.5 Step 4 — Telescope the Forward Terms

Key identity. By Bayes' rule on the joint forward process:

$$q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) = \frac{q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1})\; q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_0, \mathbf{h}_0)}{q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_0, \mathbf{h}_0)}. \tag{23}$$

Rearranging:

$$\begin{aligned}\log q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1}) = \log q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \\ + \log q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_0, \mathbf{h}_0) - \log q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_0, \mathbf{h}_0).\end{aligned} \tag{24}$$

Telescoping. Substituting equation 24 into $\sum_{t=1}^{T} \log q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1})$ and using $q(\mathbf{x}_0, \mathbf{h}_0 \mid \mathbf{x}_0, \mathbf{h}_0) = 1$:

$$\sum_{t=1}^{T} \log q(\mathbf{x}_t, \mathbf{h}_t \mid \mathbf{x}_{t-1}, \mathbf{h}_{t-1}) = \sum_{t=1}^{T} \log q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) + \log q(\mathbf{x}_T, \mathbf{h}_T \mid \mathbf{x}_0, \mathbf{h}_0). \tag{25}$$

Substitute back. Applying equation 25 to equation 22, grouping $t = 1$ separately, and identifying each $(\log p_\theta - \log q)$ pair as $-D_{\mathrm{KL}}(q \| p)$:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0, \mathbf{h}_0) \;\geq\; &\underbrace{\mathbb{E}_q\big[\log p_\theta(\mathbf{x}_0, \mathbf{h}_0 \mid \mathbf{x}_1, \mathbf{h}_1)\big]}_{L_0} \\ &- \sum_{t=2}^{T} \underbrace{\mathbb{E}_q\Big[D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \;\|\; p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)\Big]}_{L_t} \\ &- \underbrace{D_{\mathrm{KL}}\big(q(\mathbf{x}_T, \mathbf{h}_T \mid \mathbf{x}_0, \mathbf{h}_0) \;\|\; p(\mathbf{x}_T, \mathbf{h}_T)\big)}_{L_T}.\end{aligned} \tag{26}$$

Here $p(\mathbf{x}_T, \mathbf{h}_T) = p(\mathbf{x}_T)\, p(\mathbf{h}_T)$ since the terminal prior factorises. The reverse model factorises as

$$p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t) = p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t) \cdot p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}) \tag{27}$$

We approximate the reverse process $p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}|\mathbf{x}_t, \mathbf{h}_t)$ using the neural network $s_\theta(\mathbf{x}_t, \mathbf{h}_t, t)$, which is composed of two joined networks - $p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)$ is modeled as $h_\theta(\mathbf{x}_t, \mathbf{h}_t, t)$ and $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1})$ is modeled as $x_\theta(\mathbf{x}_t, \mathbf{h}_{t-1}, t)$.

A.6 Step 5 — Decompose the Joint KL Term L_t (Full Derivation)

Each L_t is a KL over the joint state $(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$, so that the term of interest is $L_t = D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)$.

Chain rule of KL divergence. For any two joint distributions $q(a, b \mid c)$ and $p(a, b \mid d)$ that are each factorised in the *same* variable order, the KL divergence obeys the exact identity

$$D_{\mathrm{KL}}\big(q(a, b \mid c) \,\|\, p(a, b \mid d)\big) = D_{\mathrm{KL}}\big(q(a \mid c) \,\|\, p(a \mid d)\big) + \mathbb{E}_{q(a|c)}\Big[D_{\mathrm{KL}}\big(q(b \mid a, c) \,\|\, p(b \mid a, d)\big)\Big]. \tag{28}$$

Hierarchy-first factorisation. We apply equation 28 with $a = \mathbf{h}_{t-1}$ (hierarchy) and $b = \mathbf{x}_{t-1}$ (token), matching the two-stage structure of the model. The true posterior factorises by the product rule ,

$$q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) = q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)\, q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0), \tag{29}$$

Note: (We drop h_t since h_{t-1} contains the latest state of the model The reverse model is *defined* to factorise in the same order - Pass 1 predicts $\mathbf{h}_{t-1}$ and Pass 2 predicts $\mathbf{x}_{t-1}$ conditioned on it

$$p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) = \underbrace{p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)}_{\text{Pass 1: hierarchy head}}\, \underbrace{p_\theta(\mathbf{x}_{t-1} \mid, \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0)}_{\text{Pass 2: token head}}. \tag{30}$$

Apply the chain rule. Substituting the factorisations equation 29 and equation 30 into the identity equation 28 yields the two-term decomposition

$$\begin{aligned} &D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big) \\ &\quad = \underbrace{D_{\mathrm{KL}}\big(q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)}_{\text{hierarchy KL (Pass 1)}} \\ &\qquad + \underbrace{\mathbb{E}_{q(\mathbf{h}_{t-1}|\mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)}\Big[D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1})\big)\Big]}_{\text{token KL (Pass 2)}}. \end{aligned} \tag{31}$$

Assemble. Expanding the abbreviations $\mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0 = (\mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)$ and $\mathbf{x}_t, \mathbf{h}_t = (\mathbf{x}_t, \mathbf{h}_t)$:

$$\begin{aligned} &D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}, \mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big) \\ &= \underbrace{D_{\mathrm{KL}}\big(q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)}_{\text{hierarchy KL (Pass 1)}} \\ &\quad + \underbrace{\mathbb{E}_{q(\mathbf{h}_{t-1}|\mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)}\Big[D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1})\big)\Big]}_{\text{token KL (Pass 2)}}. \end{aligned} \tag{32}$$

A.7 Step 6 — Simplify the Terminal Term L_T

At $t = T$, every token and hierarchy label is fully masked with probability $1 - \alpha_k(1) \approx 1$. The joint prior $p(\mathbf{x}_T, \mathbf{h}_T)$ is therefore also the fully-masked state, so

$$L_T = D_{\mathrm{KL}}\big(q(\mathbf{x}_T, \mathbf{h}_T \mid \mathbf{x}_0, \mathbf{h}_0) \,\|\, p(\mathbf{x}_T, \mathbf{h}_T)\big) \approx 0. \tag{33}$$

We drop this term.

A.8 Step 7 — Full Joint Discrete ELBO

Dropping $L_T \approx 0$:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0, \mathbf{h}_0) \;\geq\; & \underbrace{\mathbb{E}_q\big[\log p_\theta(\mathbf{x}_0, \mathbf{h}_0 \mid \mathbf{x}_1, \mathbf{h}_1)\big]}_{L_0} \\ & - \sum_{t=2}^{T} \underbrace{\mathbb{E}_q\Big[D_{\mathrm{KL}}\big(q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \parallel p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t)\big)\Big]}_{L_t^{\text{hier}} \text{ (hierarchy head)}} \\ & - \sum_{t=2}^{T} \underbrace{\mathbb{E}_q\Big[\mathbb{E}_{q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)}\Big[D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{x}_0, \mathbf{h}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_{t-1})\big)\Big]\Big]}_{L_t^{\text{tok}} \text{ (token head)}}.\end{aligned} \tag{34}$$

A.9 Step 8 — Expand the Hierarchy KL (Pass 1)

From equation 34, the hierarchy contribution at reverse step t is:

$$L_t^{\text{hier}} = \mathbb{E}_q[D_{\mathrm{KL}}(q(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0) \parallel p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t))]\,. \tag{35}$$

Since the forward process factorizes independently across sequence positions, and the Pass-1 hierarchy head predicts each position independently given the noisy state $(\mathbf{x}_t, \mathbf{h}_t)$,

$$q(\mathbf{h}_{t-1} \mid \cdots) = \prod_{i=1}^{L} q(h_{t-1}^i \mid \cdots), \tag{36}$$

$$p_\theta(\mathbf{h}_{t-1} \mid \mathbf{x}_t, \mathbf{h}_t) = \prod_{i=1}^{L} p_\theta(h_{t-1}^i \mid \mathbf{x}_t, \mathbf{h}_t). \tag{37}$$

The joint KL therefore decomposes into a sum of independent per-position KLs,

$$L_t^{\text{hier}} = \mathbb{E}_q\left[\sum_{i=1}^{L} D_{\mathrm{KL}}\big(q(h_{t-1}^i \mid \cdots) \parallel p_\theta(h_{t-1}^i \mid \mathbf{x}_t, \mathbf{h}_t)\big)\right]. \tag{38}$$

Notation. Let

$$g_h^i(t) := [h_\theta^i(\mathbf{x}_t, \mathbf{h}_t, t)]_h, \qquad h \in \{1, \ldots, K\}, \qquad \sum_{h=1}^{K} g_h^i(t) = 1, \tag{39}$$

denote the Pass-1 softmax distribution over the K hierarchy classes at position i. For convenience we write

$$\lambda \equiv \lambda_{h_0^i}(t), \qquad \pi \equiv \pi_i(t) = \sum_{k=1}^{K} g_k^i(t)\lambda_k(t), \tag{40}$$

where $\lambda_{h_0^i}(t)$ is the gold reveal probability associated with the true hierarchy class h_0^i, and $\lambda_k(t)$ is the (schedule-determined) reveal probability of class k. The quantity $\pi_i(t)$ is the model's total predicted reveal probability, obtained by averaging the class-dependent reveal rates $\lambda_k(t)$ under the Pass-1 distribution $g^i(t)$.

Forward posterior at masked positions. At an unmasked position ($s_t^i \neq s_m$) we have $s_{t-1}^i = s_0^i$ deterministically, contributing zero KL. At a masked position the absorbing-state posterior is

$$q(s_{t-1}^i \mid s_t^i, s_0^i) = \begin{cases} \mathrm{Cat}\big(s_{t-1}^i;\, s_t^i\big), & s_t^i \neq \mathbf{s_m}, \\ \mathrm{Cat}\left(s_{t-1}^i;\, \dfrac{(1 - \alpha_{h_0^i}(t-1))\,\mathbf{s_m} + (\alpha_{h_0^i}(t-1) - \alpha_{h_0^i}(t))\,s_0^i}{1 - \alpha_{h_0^i}(t)}\right), & s_t^i = \mathbf{s_m}. \end{cases} \tag{41}$$

Reverse hierarchy model (Pass 1). The head first assigns position i to class h with probability $g_h^i(t)$, and each class then reveals at its own *known* schedule rate $\lambda_h(t)$. The reverse model is thus

$$p_\theta(h_{t-1}^i = h \mid \mathbf{x}_t, \mathbf{h}_t) = \lambda_h(t)\, g_h^i(t), \qquad h \in \{1, \dots, K\}, \tag{42}$$

$$p_\theta(h_{t-1}^i = m \mid \mathbf{x}_t, \mathbf{h}_t) = 1 - \pi, \qquad \pi_i(t) = \sum_{k=1}^{K} g_k^i(t)\lambda_k(t). \tag{43}$$

Each class here reveals at its own rate $\lambda_h(t)$, and on the gold class this gives the *gold* rate directly,

$$p_\theta(h_{t-1}^i = h_0^i \mid \mathbf{x}_t, \mathbf{h}_t) = \lambda_{h_0^i}(t)\, g_{h_0^i}^i(t) = \lambda\, g_{h_0^i}^i(t). \tag{44}$$

Per-position KL.

h_{t-1}^i	q	p_θ
h_0^i	λ	$\lambda\, g_{h_0^i}^i(t)$
$h \neq h_0^i$	0	$\lambda_h(t)\, g_h^i(t)$
h_{mask}	$1-\lambda$	$1-\pi$

The KL divergence is

$$D_{\mathrm{KL}}(q \| p_\theta) = \sum_v q(v) \log \frac{q(v)}{p_\theta(v)}, \qquad v \in \{1, \dots, K, h_{\text{mask}}\}. \tag{45}$$

Since $q(v) = 0$ for every incorrect hierarchy class ($h \neq h_0^i$), all such terms vanish — the off-gold entries $\lambda_h(t)\, g_h^i(t)$ never enter the KL and serve only to normalize p_θ. Only the gold-class and mask rows remain,

$$D_{\mathrm{KL}} = \lambda \log \frac{\lambda}{\lambda\, g_{h_0^i}^i(t)} + (1-\lambda) \log \frac{1-\lambda}{1-\pi}. \tag{46}$$

In the gold-class row the reveal rate λ cancels between numerator and denominator, leaving a pure cross-entropy on g^i:

$$\begin{aligned} D_{\mathrm{KL}} &= \lambda \log \frac{1}{g_{h_0^i}^i(t)} + (1-\lambda) \log \frac{1-\lambda}{1-\pi} \\ &= -\lambda \log g_{h_0^i}^i(t) + (1-\lambda) \log \frac{1-\lambda}{1-\pi}. \end{aligned} \tag{47}$$

Hence

$$D_{\mathrm{KL}}^{\mathrm{hier,i}} = \underbrace{-\lambda \log g_{h_0^i}^i(t)}_{\text{reveal-weighted cross-entropy}} + \underbrace{(1-\lambda) \log \frac{1-\lambda}{1-\pi}}_{\text{extra-term}} \tag{48}$$

Summing over all the positions,

$$L_t^{\mathrm{hier}} = \sum_{i=1}^{L} \mathbb{E}_q \left[-\lambda_{h_0^i}(t)\, \log g_{h_0^i}^i(t) \;+\; \big(1 - \lambda_{h_0^i}(t)\big) \log \frac{1 - \lambda_{h_0^i}(t)}{1 - \pi_i(t)} \right]. \tag{49}$$

A.10 Step 9 — Expand the Token KL (Pass 2)

We expand the per-position token contribution

$$L_t^{\mathrm{tok},i} = \mathbb{E}_{q(h_{t-1}^i \mid \mathbf{x}_t, \mathbf{h}_t, \mathbf{x}_0, \mathbf{h}_0)} \Big[D_{\mathrm{KL}}\big(q(x_{t-1}^i \mid h_{t-1}^i, \cdots, \mathbf{x}_0, \mathbf{h}_0) \,\big\|\, p_\theta(x_{t-1}^i \mid h_{t-1}^i, \mathbf{x}_t, \mathbf{h}_t)\big) \Big], \tag{50}$$

By the shared reveal variable (equation 5), position i is governed by a single Bernoulli $b_t^i \sim \mathrm{Bernoulli}\big(1 - \alpha_{h_0^i}(t)\big)$, and the joint state $s_t^i = (x_t^i, h_t^i)$ is a deterministic push-forward of b_t^i: both streams reveal to (x_0^i, h_0^i) when the position is kept and mask to (m, h_{mask}) otherwise. Consequently the joint reverse posterior over (x_{t-1}^i, h_{t-1}^i) is equation 41.

Branch decomposition. The outer expectation in equation 50 runs over the two values $h^i_{t-1} \in \{h^i_0,\, h_{\text{mask}}\}$, with marginal weights λ and $1-\lambda$ from equation 41. We treat each branch in turn.

Revealed branch ($h^i_{t-1} = h^i_0$, weight λ).

$p_\theta(x^i_{t-1} = v \mid h^i_0, x_t, h_t) = f^i_v(t)$ with $f^i_v(t) := [x_\theta(x_t, h_t, t)]_v$ and $\sum_{v=1}^V f^i_v(t) = 1$:

x^i_{t-1}	q	p_θ
x^i_0	1	$f^i_{x^i_0}(t)$
$v \neq x^i_0$	0	$f^i_v(t)$
m	0	0

$$\implies \qquad D_{\text{KL}}\big(q \,\|\, p_\theta\big)\big|_{\text{rev}} = \sum_v q_v \log \frac{q_v}{p_{\theta,v}} = -\log f^i_{x^i_0}(t). \tag{51}$$

Masked branch ($h^i_{t-1} = h_{\text{mask}}$, weight $1-\lambda$).

x^i_{t-1}	q	p_θ
x^i_0	0	0
$v \neq x^i_0$	0	0
m	1	1

$$\implies \qquad D_{\text{KL}}\big(q \,\|\, p_\theta\big)\big|_{\text{mask}} = 0. \tag{52}$$

Taking the outer expectation.

$$\begin{aligned} L^{\text{tok},i}_t &= \mathbb{E}_q\Big[\underbrace{\lambda}_{P(h^i_{t-1}=h^i_0)} \cdot\big(-\log f^i_{x^i_0}(t)\big) \;+\; \underbrace{(1-\lambda)}_{P(h^i_{t-1}=h_{\text{mask}})} \cdot 0\Big] \\ &= -\,\mathbb{E}_q\big[\,\lambda_{h^i_0}(t)\,\log f^i_{x^i_0}(t)\big]. \end{aligned} \tag{53}$$

Summing over position,

$$L^{\text{tok}}_t = \sum_{i=1}^L L^{\text{tok},i}_t = -\sum_{i=1}^L \mathbb{E}_q\big[\,\lambda_{h^i_0}(t)\,\log f^i_{x^i_0}(t)\big]. \tag{54}$$

The token loss is thus a *reveal-weighted cross-entropy* on the Pass-2 softmax: each position contributes the negative log-likelihood of its clean token x^i_0, weighted by the class-dependent probability $\lambda_{h^i_0}(t)$ that the position is revealed at step $t-1$.

DISCRETE → CONTINUOUS

Setup. Let $s = t - \frac{1}{T}$, and let $\alpha_k(t)$ be the class-k survival (unmasked) probability, decreasing with $\alpha_k(0) = 1$, $\alpha_k(1) = 0$.

Step A — per-step masked-position KL

$$L^{\text{tok},i}_t = -\,\frac{\alpha_{h^i_0}(s) - \alpha_{h^i_0}(t)}{1 - \alpha_{h^i_0}(t)}\,\log f^i_{x^i_0}(t). \tag{55}$$

Step B — sum over timesteps.

$$\mathcal{L}^{\text{tok}} = \sum_t L^{\text{tok}}_t = -\sum_t \sum_i \mathbb{E}_q\left[\frac{\alpha_{h^i_0}(s) - \alpha_{h^i_0}(t)}{1 - \alpha_{h^i_0}(t)}\,\log f^i_{x^i_0}(t)\right]. \tag{56}$$

Step C — expose the mask indicator.

$$\mathcal{L}^{\text{tok}} = -\sum_t \mathbb{E}_q\left[\sum_i 1[x^i_t = M]\,\frac{\alpha_{h^i_0}(s) - \alpha_{h^i_0}(t)}{1 - \alpha_{h^i_0}(t)}\,\log f^i_{x^i_0}(t)\right]. \tag{57}$$

Step D — Riemann rescaling ($\sum_t \to \mathbb{E}_t$).

$$\sum_t [\,\cdot\,] = \underbrace{\tfrac{1}{T}\sum_t}_{\mathbb{E}_{t\in\{1/T,\dots,1\}}} \underbrace{T[\,\cdot\,]}_{O(1)}, \tag{58}$$

$$\mathcal{L}^{\text{tok}} = -\,\mathbb{E}_t\,\mathbb{E}_q\left[\sum_i 1[x_t^i = M]\left(T\cdot\tfrac{\alpha_{h_0^i}(s)-\alpha_{h_0^i}(t)}{1-\alpha_{h_0^i}(t)}\right)\log f^i_{x_0^i}(t)\right]. \tag{59}$$

Step E — limit $T\to\infty$: weight $\to$ reveal hazard.

$$T\big(\alpha_{h_0^i}(s)-\alpha_{h_0^i}(t)\big) = \frac{\alpha_{h_0^i}(t-\frac{1}{T})-\alpha_{h_0^i}(t)}{1/T} \;\xrightarrow[T\to\infty]{}\; -\alpha'_{h_0^i}(t), \tag{60}$$

$$T\cdot\frac{\alpha_{h_0^i}(s)-\alpha_{h_0^i}(t)}{1-\alpha_{h_0^i}(t)} \;\longrightarrow\; \frac{-\alpha'_{h_0^i}(t)}{1-\alpha_{h_0^i}(t)} =: \lambda'_{h_0^i}(t). \tag{61}$$

Step F — continuous-time loss.

$$\mathcal{L}_{\text{LM}} = \mathbb{E}_{t,\,q}\left[\sum_i 1[x_t^i = M]\,\lambda'_{h_0^i}(t)\,\big(-\log f^i_{x_0^i}(t)\big)\right]. \tag{61}$$

Step G — Hierarchy head (same derivation, $f\to g$).

$$\mathcal{L}_{\text{hier}} = \mathbb{E}_{t,\,q}\left[\sum_i 1[x_t^i = M]\,\lambda'_{h_0^i}(t)\,\big(-\log g^i_{h_0^i}(t)\big)\right]. \tag{62}$$

Let $s = t - \frac{1}{T}$, and

$$\mathcal{E}_t^i = \big(1-\lambda_{h_0^i}(t)\big)\,\log\frac{1-\lambda_{h_0^i}(t)}{1-\pi_i(t)}, \tag{62}$$

Both $\lambda_{h_0^i}(t)$ and every $\lambda_k(t)$ are $O(1/T)$, hence so is $\pi_i(t)$.

Step H — expose the mask indicator. Only masked positions contribute (unmasked positions give zero KL, cf. equation 50):

$$\mathcal{E}_t^{\text{hier}} = \mathbb{E}_q\left[\sum_i 1[x_t^i = M]\,\big(1-\lambda_{h_0^i}(t)\big)\log\frac{1-\lambda_{h_0^i}(t)}{1-\pi_i(t)}\right]. \tag{63}$$

Step I — small-step expansion. Write $\lambda \equiv \lambda_{h_0^i}(t)$ and $\pi \equiv \pi_i(t)$, both $O(1/T)$. Using $\log(1-z) = -z - \frac{z^2}{2} - \cdots$,

$$\log\frac{1-\lambda}{1-\pi} = (\pi-\lambda) - \tfrac{1}{2}\big(\lambda^2-\pi^2\big) + O(T^{-3}). \tag{64}$$

Multiplying by $(1-\lambda)$ and collecting terms by order,

$$\mathcal{E}_t^i = \underbrace{(\pi-\lambda)}_{O(1/T)} + \underbrace{\tfrac{1}{2}(\lambda-\pi)^2}_{O(1/T^2)} + O(T^{-3}). \tag{65}$$

Step J — Riemann rescaling ($\sum_t \to \mathbb{E}_t$).

$$\sum_t [\,\cdot\,] = \underbrace{\frac{1}{T}\sum_t}_{\mathbb{E}_{t\in\{1/T,\dots,1\}}} \underbrace{T\,[\,\cdot\,]}_{O(1)}, \qquad T\,\mathcal{E}_t^i = \underbrace{T(\pi-\lambda)}_{O(1)} + \underbrace{\tfrac{T}{2}(\lambda-\pi)^2}_{O(1/T)\to 0}. \tag{66}$$

Step K — limit $T \to \infty$: rate $\to$ reveal hazard. For each class k, as in Step E of the token derivation,

$$T\,\lambda_k(t) = \frac{\alpha_k\big(t - \frac{1}{T}\big) - \alpha_k(t)}{1/T} \cdot \frac{1}{1-\alpha_k(t)} \xrightarrow[T\to\infty]{} \frac{-\alpha_k'(t)}{1-\alpha_k(t)} =: \lambda_k'(t). \tag{67}$$

Applying this to λ and, term by term, to π,

$$T\lambda = T\lambda_{h_0^i}(t) \to \lambda'_{h_0^i}(t), \qquad T\pi = \sum_k g_k^i(t)\, T\lambda_k(t) \to \sum_k g_k^i(t)\, \lambda_k'(t), \tag{68}$$

so that

$$T(\pi - \lambda) \xrightarrow[T\to\infty]{} \sum_k g_k^i(t)\, \lambda_k'(t) - \lambda'_{h_0^i}(t), \tag{69}$$

while the quadratic remainder vanishes.

Now, the $1 - \lambda_k(t)$ term,

$$1 - \lambda_k(t) = \frac{1 - \alpha_k\big(t - \frac{1}{T}\big)}{1 - \alpha_k(t)}. \tag{70}$$

Since α_k is continuous, $\alpha_k\big(t - \frac{1}{T}\big) \to \alpha_k(t)$ as $T \to \infty$. Moreover, $t > 0$ implies $1 - \alpha_k(t) > 0$, so the denominator is nonzero. Therefore

$$1 - \lambda_k(t) \xrightarrow[T\to\infty]{} \frac{1-\alpha_k(t)}{1-\alpha_k(t)} = 1. \tag{71}$$

The only singular point is $t = 0$: since $1 - \alpha_k(0) = 0$,

$$\lambda_k'(t) \xrightarrow[t\to 0^+]{} \infty, \tag{72}$$

while $1 - \lambda_k(t) \in (0, 1]$ remains bounded and is undefined only at $t = 0$ itself.

In pratice, we sample $t \sim \mathcal{U}[\varepsilon, 1]$, and therefore never sample $t = 0$ and hence the expression always stays bounded.

Step L — continuous-time extra term.

$$\mathcal{L}_{\text{extra}}^{\text{hier}} = \mathbb{E}_{t,\,q}\left[\sum_i 1[x_t^i = M]\Big(\underbrace{\textstyle\sum_k g_k^i(t)\,\lambda_k'(t)}_{\text{predicted reveal hazard}} - \lambda'_{h_0^i}(t)\Big)\right], \qquad \lambda_k'(t) = \frac{-\alpha_k'(t)}{1-\alpha_k(t)}. \tag{73}$$

A.11 Step 10 — Final Training Loss

Training losses. The final training loss consists of the following three components:

$$\mathcal{L}_{\text{LM}} = \mathbb{E}_{t,q}\left[\sum_{i=1}^{L} \mathbf{1}[x_t^i = M]\,\lambda'_{h_0^i}(t)\,\big(-\log f^i_{x_0^i}(t)\big)\right], \tag{74}$$

$$\mathcal{L}_{\text{hier}} = \mathbb{E}_{t,q}\left[\sum_{i=1}^{L} \mathbf{1}[x_t^i = M]\,\lambda'_{h_0^i}(t)\,\big(-\log g^i_{h_0^i}(t)\big)\right], \tag{75}$$

$$\mathcal{L}_{\text{extra}}^{\text{hier}} = \mathbb{E}_{t,\,q}\left[\sum_{i=1}^{L} \mathbf{1}[x_t^i = M]\,\Big(\textstyle\sum_k g_k^i(t)\,\lambda_k'(t) \;-\; \lambda'_{h_0^i}(t)\Big)\right], \tag{76}$$

In practice we optimise a reweighted sum of the same three terms, replacing the tied weights above by tunable coefficients:

$$\mathcal{L}_{\text{HMDLM}} = \mathcal{L}_{\text{LM}} \;+\; \beta_{\text{hier}}\,\mathcal{L}_{\text{hier}} \;+\; \beta_{\text{cons}}\,\mathcal{L}_{\text{cons}}. \tag{77}$$

[2]We tried training the second stage with all gold hierarchy. However, the performance during inference was not as expected since the model never learned to predict the correct token distribution without gold hierarchy, which is the case during inference (inference sees masked hierarchy during second stage). Therefore, we added steps 10 and 11 where in we sample token positions and pass gold hierarchy during training only for this subset, and rest of the positions mask their hierarchy.

Algorithm 1 HMDLM training[2]

Require: data $\mathcal{D}$ of clean pairs $(\mathbf{x}_0, \mathbf{h}_0)$, with $x_{0,i} \in \{1, \dots, V\}$, $h_0^i \in \{1, \dots, K\}$; per-class keep schedules $\{\alpha_k(\cdot)\}$ with reveal hazards $\lambda_k(\cdot)$; hierarchy head h_θ (Pass 1), token head x_θ (Pass 2); weights $\beta_{\text{hier}}, \beta_{\text{cons}}$; learning rate η.

while not converged **do**
  sample $(\mathbf{x}_0, \mathbf{h}_0) \sim \mathcal{D}$, $t \sim \mathcal{U}(0, 1)$
  // Forward corruption to level t
  for $i = 1, \dots, L$ **do**
    $m_i \sim \text{Bernoulli}\big(1 - \alpha_{h_0^i}(t)\big)$ ▷ $m_i = 1$: masked at t
    $(x_t^i, h_t^i) \leftarrow (M,\ h_{\text{mask}})$ if $m_i = 1$ else $(x_0^i,\ h_0^i)$
  end for
  Pass 1 — hierarchy.
  $g_i(t) \leftarrow \text{softmax}\big(h_\theta^i(\mathbf{x}_t, \mathbf{h}_t, t)\big)$, $\lambda_i' \leftarrow \lambda'_{h_0^i}(t)$, $\pi_i' \leftarrow \sum_k g_k^i(t)\, \lambda_k'(t)$ for all i
  $\mathcal{L}_{\text{hier}} \leftarrow \sum_i m_i\, \lambda_i' \left(-\log g^i_{h_0^i}(t)\right)$
  $\mathcal{L}_{\text{cons}} \leftarrow \sum_i m_i \left(\sum_k g_k^i(t)\, \lambda_k'(t) \ - \ \lambda'_{h_0^i}(t)\right)$
  for i with $m_i = 1$ **do**
    $r_i \sim \text{Bernoulli}(\lambda_i)$ ▷ $r_i = 1$: hierarchy revealed at $t-1$
  end for
  $\mathcal{R} \leftarrow \{\, i : m_i = 1,\ r_i = 1 \,\}$
  $h_{t-1}^i \leftarrow h_0^i$ for $i \in \mathcal{R}$; $h_{t-1}^i \leftarrow h_t^i$ otherwise ▷ Pass-2 *context* only: gold hierarchy at $\mathcal{R}$, h_{mask} elsewhere
  Pass 2 — token.
  $f_i(t) \leftarrow \text{softmax}\big(x_\theta^i(\mathbf{x}_t,\ \mathbf{h}_{t-1},\ t)\big)$ for all i
  $\mathcal{L}_{\text{LM}} \leftarrow \sum_i m_i\, \lambda_i' \left(-\log f^i_{x_0^i}(t)\right)$ ▷ closed-form token CE over *all* masked positions
  // Combine and update
  $\mathcal{L} \leftarrow (\mathcal{L}_{\text{LM}} + \beta_{\text{hier}}\, \mathcal{L}_{\text{hier}} + \beta_{\text{cons}}\, \mathcal{L}_{\text{cons}})/L$
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}$
end while

B Expected Mask Fraction under Hierarchical Log-Linear Schedule

We derive the expected fraction of masked tokens per training step under our hierarchical masking schedule, which is needed to compute the effective training-token budget and compare fairly against baseline diffusion language models (Sahoo et al., 2024).

Setup. Let K denote the number of hierarchy levels and let $\text{offset}_k \in [0, 1)$ denote the per-level time offset. The effective time at level k is

$$t_{\text{eff},k} \ = \ \text{clip}\left(\frac{t - \text{offset}_k}{1 - \text{offset}_k},\ 0,\ 1\right), \tag{78}$$

and the log-linear schedule for level k is

$$\alpha_k(t) \ = \ 1 - (1 - \varepsilon)\, t_{\text{eff},k}, \qquad \varepsilon = 10^{-3}. \tag{79}$$

The probability that a token at level k is masked at time t is therefore

$$p_k(t) \ = \ 1 - \alpha_k(t) \ = \ (1 - \varepsilon)\, t_{\text{eff},k}. \tag{80}$$

Per-level expected mask probability. We compute $\mathbb{E}_{t \sim \mathcal{U}[0,1]}[p_k(t)] = (1 - \varepsilon)\, \mathbb{E}[t_{\text{eff},k}]$. Splitting the integral at the clamp boundary $t = \text{offset}_k$:

$$\mathbb{E}[t_{\text{eff},k}] = \int_0^{\text{offset}_k} 0\, dt + \int_{\text{offset}_k}^1 \frac{t - \text{offset}_k}{1 - \text{offset}_k}\, dt. \tag{81}$$

Algorithm 2 HMDLM ancestral sampling (reverse process)

Require: hierarchy head h_θ (Pass 1), token head x_θ (Pass 2); per-class keep schedules $\{\alpha_k(\cdot)\}_{k=1}^{K}$; number of steps T; optional token temperature τ.

1: $x^i \leftarrow M, \quad h^i \leftarrow h_{\text{mask}}$ for all $i = 1, \ldots, L$ ▷ $t = 1$: fully masked
2: **for** $n = T, T-1, \ldots, 1$ **do**
3: $t \leftarrow n/T, \quad s \leftarrow (n-1)/T$ ▷ reverse step $t \to s$; $\mathbf{x}, \mathbf{h}$ persist across iterations
Pass 1 — hierarchy.
4: $g^i \leftarrow \text{softmax}\big(h^i_\theta(\mathbf{x}, \mathbf{h}, t)\big)$ for all i with $h^i = h_{\text{mask}}$
5: $\mathcal{R} \leftarrow \varnothing$
6: **for** i with $h^i = h_{\text{mask}}$ **do**
7: $\lambda_k \leftarrow \dfrac{\alpha_k(s) - \alpha_k(t)}{1 - \alpha_k(t)}$ for $k = 1, \ldots, K$ ▷ per-step reveal hazard of class k
8: sample $\hat{h}^i \sim \text{Cat}\big(\lambda_1 g^i_1, \ \ldots, \ \lambda_K g^i_K, \ \underbrace{1 - \textstyle\sum_k \lambda_k g^i_k}_{\text{stay masked}}\big)$; $h^i \leftarrow \hat{h}^i$ ▷ commit the sampled class (or h_{mask}) into $\mathbf{h}$
9: **if** $h^i \neq h_{\text{mask}}$ **then**
10: $\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$ ▷ position revealed this step (shared across both streams)
11: **end if**
12: **end for**
Pass 2 — token. *Fill tokens at $\mathcal{R}$, conditioned on the **just-sampled** hierarchy $\hat{h}^i$.*
13: $f^i \leftarrow \text{softmax}\big(x^i_\theta(\mathbf{x}, \mathbf{h}, t)\big)$ for all $i \in \mathcal{R}$ ▷ $\mathbf{h}$ carries $\hat{h}^i$ at $i \in \mathcal{R}$ from line 9
14: **for** $i \in \mathcal{R}$ **do**
15: sample $x^i \sim \text{Cat}(f^i)$ ▷ reveal token; masked positions keep $x^i = M$
16: **end for**
17: **freeze** (x^i, h^i) for all $i \in \mathcal{R}$; carry all of $(\mathbf{x}, \mathbf{h})$ into step $n-1$ ▷ committed positions are never revisited
18: **end for**
19: **return** $(\mathbf{x}, \mathbf{h})$ ▷ clean sample $(\mathbf{x}_0, \mathbf{h}_0)$

With the substitution $u = t - \text{offset}_k$, the second integral becomes

$$\int_0^{1-\text{offset}_k} \frac{u}{1-\text{offset}_k}\, du \;=\; \frac{(1-\text{offset}_k)^2}{2(1-\text{offset}_k)} \;=\; \frac{1-\text{offset}_k}{2}. \tag{82}$$

Hence,

$$\mathbb{E}_{t\sim\mathcal{U}[0,1]}\big[\, p_k(t)\,\big] \;=\; (1-\varepsilon)\cdot\frac{1-\text{offset}_k}{2}. \tag{83}$$

Plugging in our configuration. With $K = 2$, level offsets $\text{offset}_1 = 0.25$ (rare tokens) and $\text{offset}_0 = 0.0$ (frequent tokens), and $\varepsilon = 10^{-3}$:

$$\mathbb{E}[p_1(t)] = 0.999 \cdot \tfrac{1-0.25}{2} = 0.999 \cdot 0.375 = 0.374625, \tag{84}$$

$$\mathbb{E}[p_0(t)] = 0.999 \cdot \tfrac{1-0.0}{2} = 0.999 \cdot 0.500 = 0.499500. \tag{85}$$

Overall expected mask fraction. Let f_k denote the fraction of *real* token positions assigned to level k, with $f_0 + f_1 = 1$. The overall expected mask fraction is

$$\mathbb{E}[\text{mask fraction}] \;=\; f_0 \cdot \mathbb{E}[p_0(t)] + f_1 \cdot \mathbb{E}[p_1(t)]. \tag{86}$$

For LM1B with the `bert-base-uncased` tokenizer and context length $L = 128$, we assign a token to level 1 if it occurs at most τ times within its own sequence (at $\tau = 1$, exactly once). Measured over the full training split (30,301,028 sequences, 922,347,589 real tokens), level 1 holds 737,634,266 real tokens at $\tau = 1$ and 914,638,961 at $\tau = 5$, so

$$\tau = 1: \quad f_1 = 0.799736, \quad f_0 = 0.200264, \tag{87}$$

$$\tau = 5: \quad f_1 = 0.991642, \quad f_0 = 0.008358. \tag{88}$$

Substituting into equation 86:

$$\begin{aligned}\tau = 1: \quad \mathbb{E}[\text{mask fraction}] &= 0.200264(0.499500) + 0.799736(0.374625) \\ &= 0.100032 + 0.299601 \approx 0.399633, \qquad (89)\\ \tau = 5: \quad \mathbb{E}[\text{mask fraction}] &= 0.008358(0.499500) + 0.991642(0.374625) \\ &= 0.004175 + 0.371494 \approx 0.375669. \qquad (90)\end{aligned}$$

Effective training-token budget. The budget is denominated in *supervised* positions, i.e. those that are both masked and real. Padding is excluded from the loss.[3] Given training steps S, batch size B, and mean real length $\bar{R}$, the expected number of supervised tokens is

$$\mathbb{E}[\text{supervised tokens}] \;=\; S \cdot B \cdot \bar{R} \cdot \mathbb{E}[\text{mask fraction}]. \qquad (91)$$

With $B = 512$ and $\bar{R} = 30.44$, each step presents $B \cdot \bar{R} = 15{,}585$ real positions and therefore supervises $15{,}585 \cdot 0.399633 \approx 6{,}228.3$ tokens at $\tau = 1$ and $15{,}585 \cdot 0.375669 \approx 5{,}854.8$ at $\tau = 5$, against MDLM's $15{,}585 \cdot 0.499500 \approx 7{,}784.7$. To reach a target budget N, the required number of steps is

$$S^\star \;=\; \frac{N}{B \cdot \bar{R} \cdot \mathbb{E}[\text{mask fraction}]}. \qquad (92)$$

We instantiate this at two budgets. MDLM's 33B tokens on LM1B counts masked positions over the padded context ($10^6 \cdot 65{,}536 \cdot 0.4995 = 32.74$B); of these, a fraction $\bar{R}/L = 0.2378$ are real, so the same run supervises $N_{33} = 7.7847$B tokens. Matching it requires

$$S^\star_{33} = \frac{7.7847 \times 10^9}{15{,}585 \cdot 0.399633} \approx 1.2499 \times 10^6 \text{ steps} \qquad (\tau = 1), \qquad (93)$$

$$S^\star_{33} = \frac{7.7847 \times 10^9}{15{,}585 \cdot 0.375669} \approx 1.3296 \times 10^6 \text{ steps} \qquad (\tau = 5). \qquad (94)$$

For the extended budget, twice as large at $N_{65} = 15.5694$B supervised tokens (65.47B masked positions),

$$S^\star_{65} = \frac{15.5694 \times 10^9}{15{,}585 \cdot 0.399633} \approx 2.4998 \times 10^6 \text{ steps} \qquad (\tau = 1), \qquad (95)$$

$$S^\star_{65} = \frac{15.5694 \times 10^9}{15{,}585 \cdot 0.375669} \approx 2.6593 \times 10^6 \text{ steps} \qquad (\tau = 5). \qquad (96)$$

C Expected MAsking Fraction for WikiHow

Under the hierarchical power schedule, a response token at level ℓ with exponent w_ℓ is masked at diffusion time $t \sim \mathcal{U}(0,1)$ with probability $(1-\epsilon)\, t^{w_\ell}$. Its expected masking rate is therefore

$$\bar{m}_\ell \;=\; (1-\epsilon) \int_0^1 t^{w_\ell}\, dt \;=\; \frac{1-\epsilon}{w_\ell + 1}. \qquad (97)$$

Let g_ℓ be the fraction of tokens assigned to level ℓ, pooled over the training set and counted over diffused positions only (the prompt and padding are excluded). The expected masking rate of HMDLM is then

$$\bar{m}_{\text{HMDLM}} \;=\; (1-\epsilon) \sum_\ell \frac{g_\ell}{w_\ell + 1}. \qquad (98)$$

MDLM corresponds to a single level with $w = 1$, giving $\bar{m}_{\text{MDLM}} = (1-\epsilon)/2$. With two levels and exponents $w_0 = 1$, $w_1 = 2$,

$$\bar{m}_{\text{HMDLM}} \;=\; (1-\epsilon) \left[\frac{1-g_1}{2} + \frac{g_1}{3} \right] \;=\; (1-\epsilon) \left(\frac{1}{2} - \frac{g_1}{6} \right), \qquad (99)$$

so the ratio to MDLM, in which ϵ cancels, is

$$\rho \;=\; \frac{\bar{m}_{\text{HMDLM}}}{\bar{m}_{\text{MDLM}}} \;=\; 1 - \frac{g_1}{3}. \qquad (100)$$

[3] LM1B sequences are not packed, so a 128-position row holds $\bar{R} = 922{,}347{,}589/30{,}301{,}028 = 30.44$ real tokens on average and the remainder is padding.

Table 7: Step-matched training budgets on WikiHow at context lengths 512 and 1024. g_1 is the fraction of tokens at level 1. Under the summary rule, level 1 is the article summary. Under the count rule (τ=1), level 1 is any token occurring at most once in its own sample.

Model	g_1	$\rho = 1 - g_1/3$	Steps	Extra steps (epochs)
$L = 512$				
MDLM	–	1.000	28,650	–
HMDLM (summary)	22.34%	0.926	30,955	+2,305 (+2.40)
HMDLM (τ=1)	30.82%	0.897	31,930	+3,280 (+3.41)
$L = 1024$				
MDLM	–	1.000	41,137	–
HMDLM (summary)	16.41%	0.945	43,517	+2,380 (+1.72)
HMDLM (τ=1)	27.12%	0.910	45,225	+4,088 (+2.96)

To match the expected number of masked (loss-bearing) tokens seen by MDLM in S_{MDLM} steps at the same batch size, HMDLM is trained for

$$S_{\text{HMDLM}} = \frac{S_{\text{MDLM}}}{\rho} = \frac{3\,S_{\text{MDLM}}}{3 - g_1}, \qquad \Delta S = S_{\text{HMDLM}} - S_{\text{MDLM}} = S_{\text{MDLM}}\,\frac{g_1}{3 - g_1} \tag{101}$$

steps.

On WikiHow, MDLM is trained for ≈ 29.8 epochs at each context length, with a global batch of 128: $S_{\text{MDLM}} = 28{,}650$ steps at $L = 512$ (123,046 training examples, 961.3 steps per epoch) and $S_{\text{MDLM}} = 41{,}137$ steps at $L = 1024$ (176,696 training examples, 1,380.4 steps per epoch). Table 7 gives the resulting budgets.

D Per-Level Schedules and the Induced Reveal Order

Levels are indexed so that a higher level is masked more slowly. Formally, for any two levels $k < k'$ we require

$$\alpha_k(t) \;\le\; \alpha_{k'}(t) \quad \text{for all } t, \tag{102}$$

with strict inequality at every step at which level k' is active, i.e. $\alpha_{k'}(t) < 1$. Since $\alpha_k(0) = 1$ and $\alpha_k(T) = 0$ for every k, the inequality is necessarily an equality at the two endpoints. Fewer level-k' positions are therefore masked at any interior point of the forward process.

The reverse dynamics are governed by the per-step reveal probability $\lambda_k(t)$, and equation 102 does *not* by itself imply an ordering of the λ_k. We therefore state the property we use directly: for $k < k'$,

$$\lambda_k(t) \;\le\; \lambda_{k'}(t) \qquad \text{whenever } \alpha_{k'}(t) < 1. \tag{103}$$

Outside that interval, $\alpha_{k'}(t) = 1$, no level-k' position is masked and $\lambda_{k'}(t) = 0$; the ordering is vacuous there, since there is nothing left at level k' to reveal. Both schedules used in our experiments satisfy equation 103, as the continuous-time hazards $\lambda'_k(t) = -\alpha'_k(t)/(1 - \alpha_k(t))$ show:

- **Log-linear with per-level offset** equation 79, used on LM1B: $\lambda'_k(t) = 1/(t - \delta_k)$ for $t > \delta_k$ and 0 otherwise. For $\delta_k < \delta_{k'}$ and $t > \delta_{k'}$ we have $t - \delta_k > t - \delta_{k'} > 0$, hence $\lambda'_k(t) < \lambda'_{k'}(t)$.
- **Power schedule** $\alpha_k(t) = 1 - (1 - \varepsilon)\,t^{\omega_k}$, used on WikiHow: $\lambda'_k(t) = \omega_k/t$, so the ordering holds exactly when $\omega_k < \omega_{k'}$. We use $\omega = (1.0, 2.0)$, and $\omega_k = 1$ recovers the log-linear schedule of MDLM.

The discrete $\lambda_k(t)$ inherit the ordering on the step grids used in our experiments ($T = 128$ on LM1B, $T \in \{256, 384\}$ on WikiHow): the only steps at which $\lambda_0(t) > \lambda_1(t)$ are the 25 LM1B steps with $t \le \delta_1$, where level 1 is inactive and $\lambda_1(t) = 0$.

Consequently, while level k' is still being revealed, a position that Stage I predicts to be at level k' receives a larger reveal mass $\lambda_{k'}(t)\,g^i_{k'}(t)$ in equation 8 and is revealed earlier, which is the soft generation order HMDLM is designed to induce.

E PERPLEXITY: CONTINUOUS-TIME ELBO WEIGHT

We evaluate test-set perplexity under the continuous-time limit of the HMDLM ELBO. For a single hierarchy level k, the masking schedule is linear with offset δ_k:

$$\alpha_k(t) \;=\; \begin{cases} 1 & t \le \delta_k, \\ 1 - (1-\varepsilon)\,\dfrac{t-\delta_k}{1-\delta_k} & \delta_k < t \le 1, \end{cases} \qquad \delta_0 = 0, \delta_1 = 0.20, \tag{104}$$

where $\alpha_k(t)$ is the survival (clean) probability of a level-k token at time t, ε is a small floor.

Per-level continuous-time weight. The standard absorbing-state ELBO weight is

$$w_k(t) \;=\; -\,\frac{\alpha_k'(t)}{1-\alpha_k(t)}, \tag{105}$$

For the schedule in equation 104, on the active interval $t > \delta_k$,

$$\alpha_k'(t) \;=\; -\,\frac{1-\varepsilon}{1-\delta_k}, \qquad\qquad 1-\alpha_k(t) \;=\; (1-\varepsilon)\,\frac{t-\delta_k}{1-\delta_k}. \tag{106}$$

Substituting into equation 105,

$$w_k(t) \;=\; \frac{(1-\varepsilon)/(1-\delta_k)}{(1-\varepsilon)\,(t-\delta_k)/(1-\delta_k)} \;=\; \frac{1}{t-\delta_k}. \tag{107}$$

For $t \le \delta_k$ the level is fully unmasked ($1-\alpha_k(t) = 0$) and contributes no loss; we therefore extend w_k by zero in that regime:

$$\lambda_k(t) \;:=\; \begin{cases} \dfrac{1}{t-\delta_k} & t > \delta_k, \\ 0 & t \le \delta_k. \end{cases} \tag{108}$$

$\delta_0 = 0$ recovers the MDLM weight $1/t$ used by Sahoo et al. (2024), so equation 108 is a strict generalisation.

Posterior-weighted per-token weight. Since at evaluation, the gold hierarchy is not available, we replace the gold-hierarchy weight $\lambda_{h_0^i}(t)$ with its expectation under the model's posterior over hierarchy levels at time t:

$$\pi_i(t) \;=\; \sum_{k=0}^{K-1} p_\theta\big(h_{t-1}^i = k \,\big|\, \mathbf{x}_t,\, \mathbf{h}_t\big)\, \lambda_k(t), \tag{109}$$

where $p_\theta(h_i = k \mid \cdot)$ is the Pass-1 hierarchy distribution.

Token-level perplexity. Let N denote the total number of tokens (non-padding) in the test set, and let $\mathbf{x}_t, \mathbf{h}_t$ be sampled by the joint forward process at time $t \sim \mathcal{U}(\varepsilon, 1)$. The per-token negative log-likelihood is

$$\mathrm{NLL}_{\mathrm{tok}} \;=\; \frac{1}{N}\, \mathbb{E}_{t,\,\mathbf{x}_t}\left[\sum_{i:\, x_t^i = M} \pi_i(t)\, \big(-\log p_\theta(x_0^i \mid \mathbf{x}_t)\big) \right], \tag{110}$$

and the reported test-set perplexity is

$$\mathrm{PPL} \;=\; \exp\big(\mathrm{NLL}_{\mathrm{tok}}\big). \tag{111}$$

Numerical handling of $t \to \delta_k^+$. At $t = \delta_k$ is bounded in code by a denominator floor:

$$\lambda_k(t) \;\approx\; \frac{1}{\max(t-\delta_k,\; \eta)}, \qquad \eta = 10^{-8},$$

which leaves the weight unchanged everywhere except in a vanishingly small neighbourhood of δ_k and prevents single-sample gradient spikes from the Monte-Carlo estimator.

F Architecture Details and Qualitative analysis on sudoku

F.1 Architecture Details

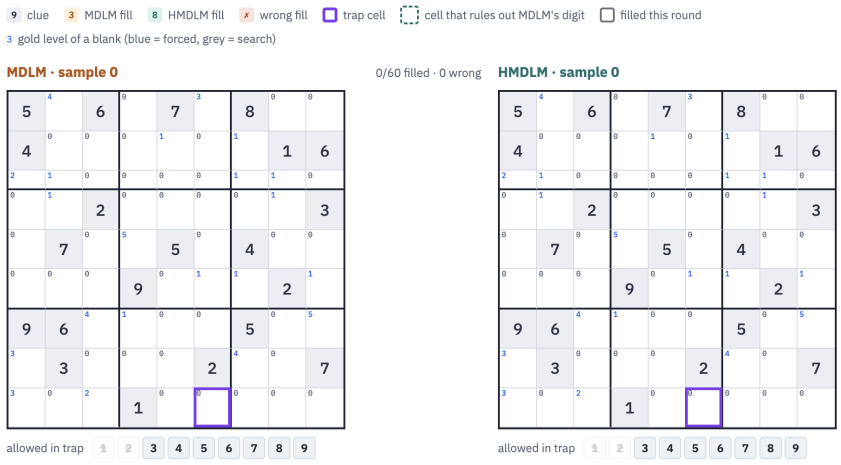
For the 6M MGDM and HMDLM models, we use a compact 3-layer GPT-2 Transformer (5.5M parameters) with hidden size 384, 12 attention heads of dimension 32, a 1536-dimensional GELU feed-forward layer, learned absolute position embeddings, and a 31-token character-level vocabulary whose output head shares weights with the input embedding and bidirectional attention. The hierarchical variant (HMDLM) adds only a level embedding and a level-prediction head, about 5K extra parameters.

For the 25M MDLM and HMDLM models, we use a 12-layer diffusion transformer (DiT, 25.0M parameters) with hidden size 384, 6 attention heads of dimension 64, a 1536-dimensional GELU feed-forward layer, rotary position embeddings, and bidirectional attention over a fixed 81-token sequence (one token per cell). Both the models are trained from scratch. The hierarchical variant (HMDLM, K=6 levels) adds a 7x384 level embedding (6 levels plus a [MASK] level), which is summed with the token embedding, and a level-prediction head that has the same form as the token output head. Together they add about 104K parameters (25.10M versus 24.99M).

F.2 Qualitative Analysis

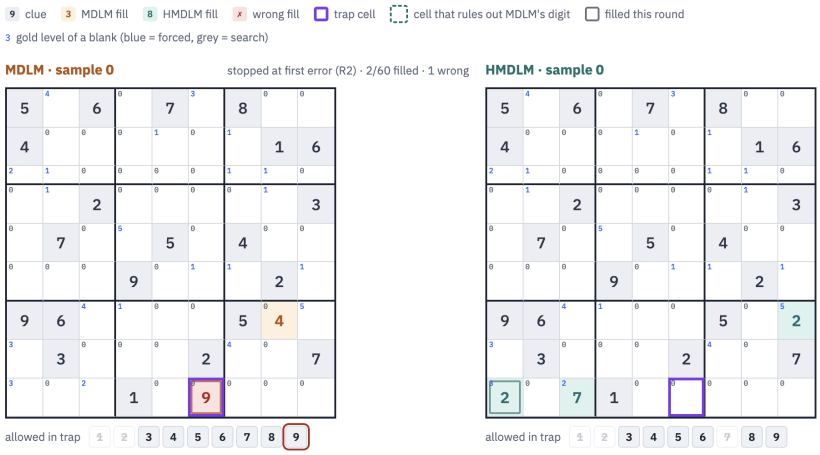
For the sudoku 25M hmdlm and mdlm models - we identified 287 puzzles where MDLM failed to solve across all three seeds but HMDLM solved correctly across all three. We then performed 8 additional inference runs per model on these puzzles. We pick some of these examples and trace their unmasking order to show how HMDLM helped in getting the right answer:

Example 1: `506070800400000001600000000000200000307005040000090002096000050003000200700010000 0`



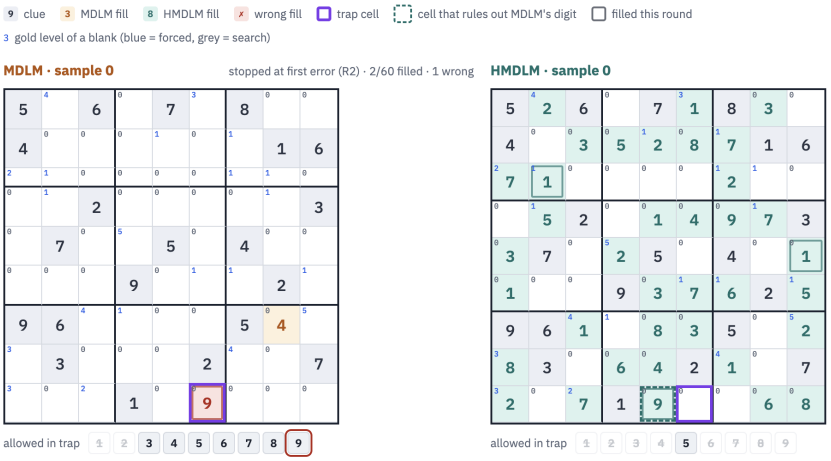
Start. Initial board.

Figure 2: Example 1, Start.



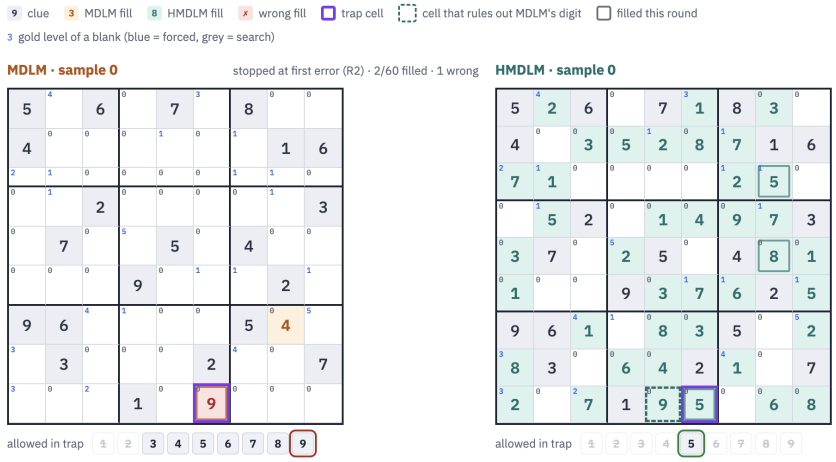
Round 2. MDLM predicts the R9C6 cell and gets it wrong. There are multiple possible values for the cell 3,4,5,6,7,8,9. HMDLM on the other hand, pushes the unmasking of this cell to later rounds.

Figure 3: Example 1, round 2.



Round 25. HMDLM defers R9C6 and fills other cells first.

Figure 4: Example 1, round 25.



Round 26. Finally, HMDLM gets the R9C6 cell right at round 25, when the only possible value left is 5 and gets it correct.

Figure 5: Example 1, round 26.

Example 2: `008100026000807000900000000010008900030000010000450000000309800200004005000203 00`



Start. Initial Board.

Figure 6: Example 2, Start.



Round 1. MDLM predicts the R7C1 cell and gets it wrong. There are multiple possible values for the cell 1,4,6,7. HMDLM on the other hand, pushes the unmasking of this cell to later rounds.

Figure 7: Example 2, round 1.



Round 29. HMDLM predicts the neighbouring cells first.

Figure 8: Example 2, round 29.



Round 36. HMDLM predicts cell R7C1 correctly.

Figure 9: Example 2, round 36.

Both the samples show the same pattern, MDLM tends to unmask ambiguous cells in the beginning and gets the prediction wrong. On the other hand, HMDLM tends to defer the unmasking of such cells to later rounds until neighbouring cells get filled first. Across 287 puzzles (where HMDLM passed and MDLM failed) with 8 samples per model, we find 131 cases where MDLM's first error is an early commit to a search cell and HMDLM fills that same cell only once a single candidate remains.

F.3 Cell Accuracy per Hierarchy Level

Table 8: Cell accuracy (%) by hierarchy level.

Level	Cells	MDLM	HMDLM
5	givens + round 0	99.76	**100.00**
4	round 1	99.59	**99.99**
3	round 2	99.29	**99.98**
2	round 3	98.73	**99.92**
1	rounds 4–5	96.77	**99.45**
0	round 6+ / never forced	78.78	**86.57**

Each cell is assigned a level by the constraint-propagation round in which it becomes forced: level 5 cells are givens or forced immediately (round 0), while level 0 cells are forced only after 6+ rounds or never, and so require search. We see that HMDLM improves the accuracy for the most difficult cells (label 0) by up to 8% than MDLM.

G Prompt for LLM-as-judge

G.1 LM1B Prompt

For evaluating the LM1B generated samples under LLM-as-judge, we pass the following prompt:

"""You are an expert evaluator judging two pieces of machine-generated English text. Your task is to determine which text is BETTER in overall quality. You must choose exactly ONE of:

A - Sample A is better

B - Sample B is better

tie - The two samples are approximately equal in quality

There is NO prompt or instruction the text is responding to; these are standalone generated passages. Evaluate each text holistically, considering:

1. Fluency: Is the text grammatical and natural English?

2. Coherence: Do the sentences follow each other in a logical, connected way?

3. Consistency: Does the text stay on a consistent topic without contradictions or abrupt, incoherent drift?

4. Overall readability: Does the passage read like plausible, well-formed human-written text?

SCORING RULES:

1. Judge ONLY the two texts.

2. Do NOT assume that either sample is better because it is longer.

3. Do NOT penalize a response merely for being brief or terse.

4. Judge each text independently on its own quality.

5. Do NOT use any information about the source/model that generated the text.

6. The labels A and B are randomly assigned and have no meaning.

7. Choose tie only when the two texts are genuinely comparable in overall quality.

Return ONLY valid JSON: "winner": "A" | "B" | "tie" """

G.2 WIKIHOW PROMPT

On similar lines, for running the LLM-as-judge on WikiHow dataset we use the following prompt:

"""You are an expert evaluator judging four responses to the same how-to prompt.

Your task is to determine which ONE response is the BEST for the given prompt.

You must choose exactly ONE of: A, B, C, D.

Evaluate the responses holistically, considering:

1. Adherence: Does the response address the specific topic in the prompt, or does it drift?

2. Coherence: Do the steps follow a logical, connected order?

3. Fluency: Is the response grammatical and natural English?

4. Overall usefulness: Would the response plausibly help someone accomplish what the prompt asks?

SCORING RULES:

1. Judge ONLY the prompt and the four responses. 2. Do NOT assume that a sample is better because it is longer. 3. Do NOT penalize a response merely for being brief or terse if it adequately answers the prompt. 4. Judge each response independently against the prompt. 5. Do NOT compare based on writing style alone; prioritize usefulness and correctness for the prompt. 6. Do NOT use any information about the source/model that generated the responses. 7. The labels A, B, C and D are randomly assigned and have no meaning. 8. If several responses are close, still pick the single best one.

Return ONLY valid JSON:

"winner": "A" | "B" | "C" | "D" """

H SOCIETAL IMPACT OF THE MODEL

The proposed HMDLM advances masked diffusion language modeling by introducing hierarchical decoding, which improves reasoning and text generation quality. The positive societal impacts include more efficient and accurate language models that can benefit downstream NLP applications such as question answering, text generation, and reasoning tasks. On the negative side, as with all pre-trained language models, HMDLM could potentially be misused to generate misleading or harmful text.

I COMPUTATIONAL RESOURCES

For language modelling, all experiments were conducted on NVIDIA A100 80GB GPUs. The HMDLM model was trained for approximately four days on 16 GPUs with a batch size of 512, completing 2.5M (65B tokens) training steps on LM1B dataset.

For WikiHow, All experiments were conducted on NVIDIA A100 80GB GPUs. The HMDLM model was trained for approximately 8 hours on 4 GPUs.

For Sudoku, again we used the same machine and trained on 4GPUs for 4 hours.