BINARY DIFFUSION LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion replaces corrupted tokens with the same mask symbol, removing all local identity information. Instead of representing a token with one-hot vector, one can also represent them with binary vectors. A binary code makes corruption gradual: partial evidence about token identity survives at every noise level. We introduce BinaryDLM, a diffusion language model that flips bits in token codes learned through language-modeling objective, so that code similarity reflects lexical similarity. During training, we sample noise independently across positions, mixing near-clean and highly corrupted states so that informative neighbors appear alongside difficult prediction targets. During generation, high-confidence predictions are temporarily anchored as clean context for their neighbors, and blocks are refined iteratively with early stopping. Across 0.3B–4B models trained on a shared math mixture for up to 100B target tokens, BinaryDLM outperforms both masked diffusion (+0.7–10.2 points) and autoregressive baselines (+2.4–8.5 points) on GSM8K accuracy. Adapting pretrained Qwen3-8B to binary diffusion yields competitive performance with state-of-the-art masked diffusion models of the same scale. These results suggest that the corruption representation is a key design axis for diffusion language models, complementary to the denoising architecture and noise schedule.

1 INTRODUCTION

Autoregressive language models generate text one token at a time. Diffusion language models instead generate text through iterative denoising, predicting multiple tokens in parallel at each step (Sahoo et al., 2024; Nie et al., 2025). A common approach is masked diffusion, in which each position contains either a vocabulary token or a mask. Each revealed token becomes clean context for the positions that remain masked, much as visible tokens do in masked language modeling (Devlin et al., 2019). However, the mask carries no partial information about token identity: until a position is unmasked, it contributes nothing to its neighbors. This makes the mask a binary information bottleneck, a position is either resolved or fully opaque and hinder denoising process, especially if we want to unmask multiple tokens in a single steps (Chen et al., 2026; Kim et al., 2026; Li et al., 2026).

Recent work replaces the mask with binary or continuous encodings so that corrupted states can retain partial token information (Chao et al., 2025; 2026; Batzolis et al., 2026). Two design limitations remain. First, these methods use fixed, index-derived codes, so code similarity is not learned to reflect linguistic relationships. For example, corrupting the code of `happy` with random bit flips still yields high affinity to `glad` and `joyful` but low affinity to `stone` (Figure 1a), whereas a random code offers no such guarantee. Second, continuous bit-based models denoise positions at a shared noise level within each block or sequence (Batzolis et al., 2026), so the model never encounters positions at different stages of resolution during training and cannot use nearly-resolved positions to guide uncertain ones during generation (Figure 1b).

A binary code makes corruption gradual rather than all-or-nothing: every position always carries some signal about its token identity, and that signal strengthens as noise decreases. Our central hypothesis is that *preserving emerging token information* allows tokens within a block to provide useful context for one another. This information takes two forms: Learned binary codes offer a way to retain partial evidence about related token alternatives, so even a corrupted code narrows the set of plausible predictions. For high-confidence predictions, we keep their predicted clean codes without adding noise for the next step, letting them serve as stable context while other positions resolve their uncertainty. Every position is still re-predicted at each update, so earlier predictions remain revisable.

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.

We introduce *BinaryDLM* to explore this view of denoising as progressively building and revising context. We learn binary token codes through the language-modeling objective so that code similarity reflects lexical similarity. We sample noise independently at each position, mixing near-clean and highly corrupted states within the same block, so that informative neighbors appear alongside difficult prediction targets. And we predict clean vocabulary tokens directly with cross-entropy, giving the model a task-relevant training signal at every position regardless of noise level. Generation iteratively refines blocks conditioned on a clean prefix, with confidence-based token anchoring and early stopping.

We perform controlled experiments at 0.3B–4B parameters, training from scratch on a shared math mixture for up to 100B target tokens. BinaryDLM outperforms masked diffusion by 0.7–10.2 percentage points and autoregressive baselines by 2.4–8.5 points on GSM8K accuracy. Unlike prior binary diffusion studies, which focus on small-scale experiments and generative perplexity, we evaluate on downstream task accuracy across multiple scales and training budgets. Ablations confirm that each component contributes: learned codes outperform random codes, near-clean training positions improve accuracy, and confidence-based anchoring outperforms shared-timestep denoising. Beyond training from scratch, we adapt pretrained Qwen3-8B to binary diffusion and obtain competitive results on math and code benchmarks.

2 Related Work

Language diffusion. Diffusion language models differ in how they represent and corrupt text. Diffusion-LM denoises continuous word embeddings (Li et al., 2022), whereas discrete models corrupt tokens directly: D3PM (Austin et al., 2021) defines structured transition matrices over the vocabulary, SEDD (Lou et al., 2023) learns ratios of the data distribution, and masked diffusion replaces tokens with an absorbing mask, DiffusionGemma (DiffusionGemma Team, 2026) leverage uniform diffusion where tokens are randomly changed to another token in vocabulary. Block diffusion combines these models with autoregression: blocks are generated left to right, so earlier blocks can be cached, while tokens within a block are denoised in parallel (Arriola et al., 2025).

Enriched denoiser inputs. Soft-Masked DLMs and x-Prediction Flow enrich denoiser inputs through continuous prediction feedback (Hersche et al., 2025; Wang et al., 2026). CADD and CCDD augment discrete denoising with evolving continuous latents, while CANDI combines discrete and continuous corruption (Zheng et al., 2026; Zhou et al., 2025; Pynadath et al., 2025). Our learned binary codes retain partial token information directly in discrete states, without an auxiliary continuous diffusion process.

Binary representations. Analog Bits and CoBit apply continuous diffusion to binary encodings (Chen et al., 2022; Batzolis et al., 2026), while BDPM models binary image latents with discrete corruption (Kinakh & Voloshynovskiy, 2025). CoBit uses fixed token-ID codes and bitwise outputs, whereas MDM-Prime and Prime-v2 partially mask fixed encodings and retain vocabulary decoding (Chao et al., 2025; 2026). SemDLM+ preserves local structure by corrupting tokens within semantic neighborhoods (Jiang et al., 2026). Prior work have shown that representation geometry affects denoising (Feng et al., 2026); RePlaid reports lower perplexity bounds with learned continuous embeddings than with fixed random embeddings (Yang et al., 2026). We learn binary codes through a BSQ bottleneck (Zhao et al., 2024) with next-token prediction and apply discrete bit flips, retaining vocabulary decoding.

Heterogeneous noise and adaptive decoding. Diffusion Forcing trains on independently corrupted sequence elements; Spatial Reasoning Models and Patch Forcing study heterogeneous noise in spatial settings (Chen et al., 2024; Wewer et al., 2025; Schusterbauer et al., 2026). Specifically, Wewer et al. (2025) found that diffusion models are good at making each local piece look plausible, but they can fail badly when many pieces must satisfy a global constraint together. The authors argue that this failure can be reduced by turning parallel denoising into a sequential reasoning process.

3 Method

BinaryDLM has three components (Figure 1). First, we learn binary token codes whose similarity reflects lexical similarity, so that even corrupted codes carry partial evidence about token identity

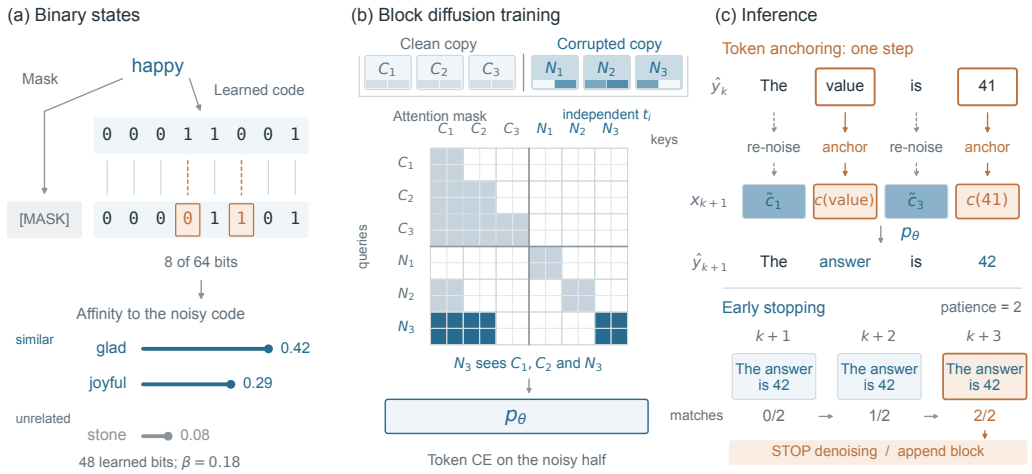


Figure 1: Binary states, clean-context block training, and adaptive decoding. (a) Selected token comparisons for one corruption of *happy*; affinity $1 - 2d_H/48$ uses the learned bits. (b) We use block diffusion for training and concatenated clean (C) and corrupted (N) copies. Filled cells permit attention; blue highlights N_3 attending to C_1, C_2, N_3, with its clean target C_3 blocked. Noise is position-wise; vocabulary CE applies only to the noisy half. (c) During inference, we perform token anchoring where we don't renoise high-confidence tokens. We allow the denoising process of a block terminate early if the output doens't change after some iterations.

(§3.1). Second, we train a block-diffusion model with independently sampled noise at each position, mixing near-clean and heavily corrupted states within the same block (§3.2). Third, we decode by iteratively refining blocks, anchoring high-confidence predictions as clean context and stopping early when predictions stabilize (§3.3).

3.1 Learned Binary States

How we encode tokens affects how easily the model can recover them from noise. To learn binary token embeddings, we insert a Binary Spherical Quantization bottleneck (Zhao et al., 2024) after the embedding layer of pretrained Qwen3-4B and continue next-token training. A learned projection and sign quantization produce 48 bits per token. We then train the model on 100B tokens of (English) web, stem and code corpus. As we only train the model with next-token prediction objective, there are unavoidable collision of tokens with similar meaning e.g., " `apple`" (with spacing) and "`apple`". We add deterministic bits to resolve code collisions and pad to 64 bits for easier sharding. As illustrated in Figure 2, tokens in similar semantic groups share common bit patterns. Note that for all our experiments, we share the same Qwen-3 tokenizer. Thus, we only need to learn the binary embedding one time.

Learned versus random codes. We verify the effectiveness of learned representation, we conduct ablation study that replacing trained binary embedding with a fixed random 64-bit table while retaining the same noise distribution, token objective, and decoder. Learned codes substantially improve final GSM8K accuracy by 8.7 percentage points at 0.6B and 13.6 at 1B (Figure 2).

3.2 Token-Level Diffusion Forcing

Given a learned codebook, we corrupt codes by flipping bits, sample noise levels independently across positions, and train a block-diffusion model to predict clean tokens from the corrupted inputs.

Corruption process. Following binary diffusion (Kinakh & Voloshynovskiy, 2025), we corrupt the binary code $c(y) \in \{0,1\}^{64}$ of token y as $\mathbf{x}_t = c(y) \oplus \mathbf{z}$, where $\oplus$ denotes bitwise XOR and $z_j \sim \text{Bernoulli}(\beta(t))$ independently.

$$\beta(t) = (1-t)^2\beta_{\min} + \left[1-(1-t)^2\right]\beta_{\max}, \qquad (\beta_{\min}, \beta_{\max}) = (10^{-3}, 0.5). \tag{1}$$

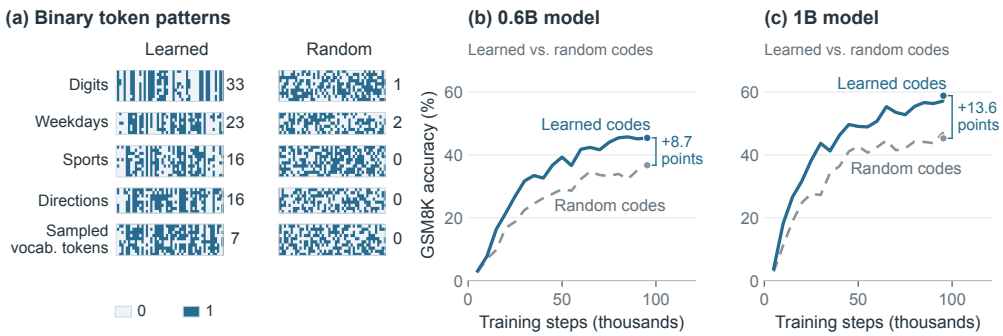


Figure 2: Learned versus random binary codebooks. (a) The same token groups encoded using the first 48 bits of each experimental codebook. Each row is a token and each column a bit in the vector; counts give bits shared by every token in that group. The last group contains ten randomly sampled vocabulary tokens. (b–c) GSM8K accuracy at 0.6B and 1B, holding the training configuration fixed.

Here $\beta(t)$ measures corruption relative to the clean code, rather than the probability of a transition between adjacent timesteps. Equivalently, each bit is replaced by a fair random bit with probability $2\beta(t)$. It is worth noticing that the resulting vector need not be a valid codeword.

Position-wise noise with near-clean states. Continuous diffusion language models commonly advance all positions using a shared noise schedule. Tokens within a block can differ substantially in predictability: some positions may already be confidently resolved, while others remain uncertain and require additional context. Our token-anchoring decoder (§3.3) explicitly exploits this asymmetry by keeping high-confidence predictions clean as context while continuing to denoise the remaining positions. As shown in Figure 3c, combining position-wise noise with token anchoring substantially outperforms conventional shared-timestep denoising, suggesting that allowing easier positions to become clean context is important for resolving more difficult ones.

As a result, inference naturally produces *mixed-noise* blocks in which clean or nearly resolved tokens coexist with heavily corrupted ones. Training with a single shared timestep for the entire block does not expose the model to such states, creating a mismatch between training and inference. To expose the model to these heterogeneous states during training, we follow Diffusion Forcing (Chen et al., 2024) and sample the noise level independently for each token. Specifically, for each position i, we sample

$$t_i = \begin{cases} 0, & \text{with probability } \pi, \\ \mathrm{clip}(u_i, \epsilon, 1-\epsilon), & \text{otherwise,} \end{cases} \qquad u_i \sim \mathcal{N}(\mu, \sigma^2). \tag{2}$$

We use $(\pi, \mu, \sigma) = (0.15, 0.8, 0.15)$ and clip the Gaussian samples with $\epsilon = 10^{-4}$. The point mass at $t_i = 0$ introduces a small fraction of clean positions into each noisy block, mirroring the anchored tokens that arise during inference. These positions can therefore serve as reliable context for predicting more heavily corrupted tokens, while the independently sampled noise levels expose the model to a broad range of intermediate states.

This design has a large empirical effect. Holding the nonzero-noise distribution fixed, introducing 15% clean positions improves GSM8K accuracy by 17.4 and 25.3 percentage points for the 0.6B and 1B models, respectively, after approximately 100B training tokens (Figure 3a). We further vary the clean-position probability in Figure 3b. Both the 10% and 20% settings outperform training without clean positions at both model sizes, indicating that the gain is not specific to a finely tuned choice of $\pi = 0.15$, but instead reflects the broader benefit of exposing the model to mixed-noise states during training.

Block training and objective. We adopt block diffusion (Arriola et al., 2025; Fu et al., 2025) for our training. We concatenates clean and corrupted copies as in Figure 1b: noisy blocks attend bidirectionally within themselves and to earlier clean blocks; clean blocks use block-causal attention (Arriola et al., 2025; Fu et al., 2025). Writing $\mathbf{x}$ for corrupted codes and $b(i)$ for the block containing position

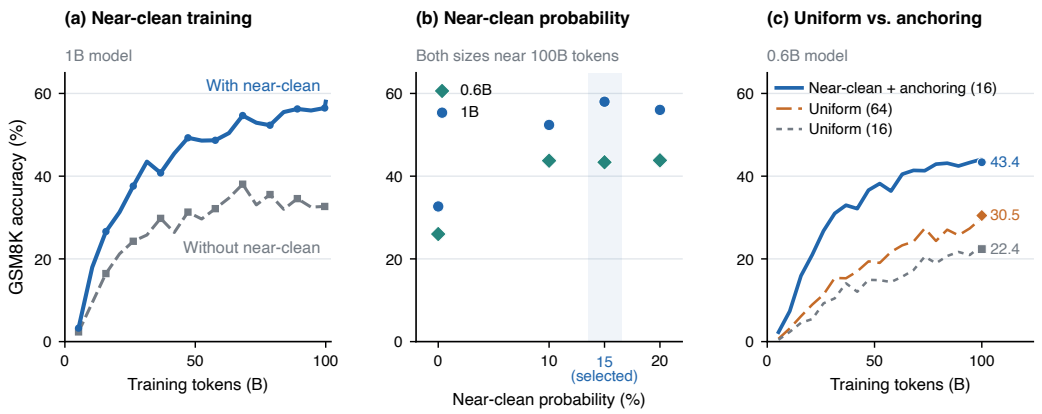


Figure 3: Training noise and token anchoring on GSM8K (strict extraction). (a) 1B trajectories with 0% and 15% near-clean positions. (b) we sweep the model trained with different near-clean probability while fixing other hyperparameters (c) We compare two variant of 0.6B model one trained with proposed noise scheduler and enable token anchoring versus shared-uniform training with conventional diffusion denoising; Parentheses indicate denoising step budgets.

i, the objective is

$$\mathcal{L} = \mathbb{E}_{\mathbf{y},\mathbf{t},\mathbf{x}}\left[-\frac{1}{L}\sum_{i=1}^{L}\log p_\theta(y_i \mid \mathbf{x}_{b(i)}, \mathbf{y}_{<b(i)})\right]. \tag{3}$$

We use x_0-prediction and enforce model output vocabulary logits rather than binary logits, this allow the model to perform confidence-based token anchoring during inference in a straightforward way as in conventional masked diffusion. Vocabulary cross-entropy supervises every target position, including near-clean ones. The model receives corrupted states without explicit timestep embeddings.

3.3 Confidence-Guided Generation

At inference, each block starts from random bits and is iteratively denoised conditioned on a clean prefix. The token-level diffusion forcing training exposure model to input blocks where some tokens carry more information than others. We exploit this through token anchoring and early stopping.

Token anchoring. Some tokens become predictable before the rest of a block. Re-corrupting their predictions can weaken the context available to uncertain positions, so we anchor selected predictions as clean inputs for the next update. At each step, we rank positions by confidence $s_i = \max_y p_\theta(y_i = y)$ and select anchors using a confidence threshold and a growing budget. The budget limits how many predictions become clean at once, while a minimum floor ensures progress (tokens are revealed) when few predictions pass the threshold. Anchored positions receive a deterministic bit update without re-corruption as in normal reverse step of diffusion; other tokens are sampled and re-corrupted at a decreasing noise level as in standard diffusion. Every position is predicted again and the anchors are reselected at the next step, allowing earlier mistakes to be corrected as context improves.

Early stopping. We found that a fixed denoising budget can spend additional updates on blocks whose predictions have already stabilized or are cycling among the same alternatives. We therefore stop based on repeated agreement with recent predictions (Figure 1c). After each update, we compare the entire predicted token block with the blocks in a short history. An exact match to any of them increments a counter; a miss resets it. Once the counter reaches the patience threshold, we append the current prediction to the prefix and begin the next block.

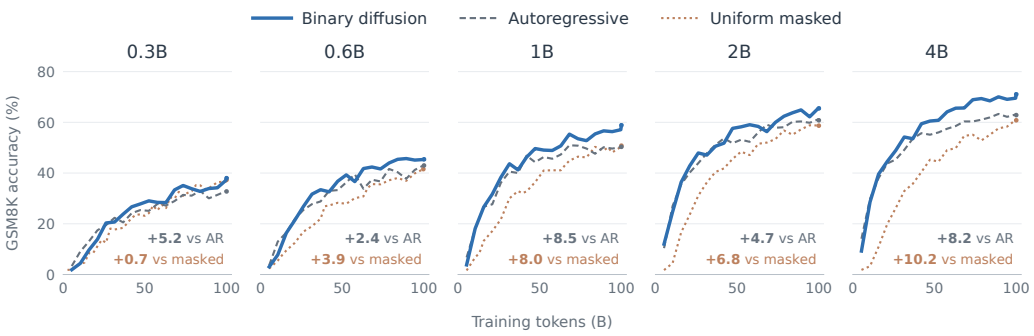


Figure 4: Binary, autoregressive, and uniform-masked training on GSM8K. Each column shows one model size on a shared accuracy scale.

4 Experiments

4.1 Experiment setup

Training. Our implementation is built on-top of `torchtitan` (Liang et al., 2025) and use float8 training for all our experiments. Binary and masked models share the same Qwen-style transformer configuration and vocabulary-head dimensions; the autoregressive baseline uses causal attention (Yang et al., 2025). We train 0.3B, 0.6B, 1B, 2B, and 4B models on the same math mixture for 100B target tokens. All models use sequence length 2048 and AdamW with learning rate 2×10^{-4} and cosine learning rate schedule. Following the two-stage design of latent diffusion, we train BinaryDLM's backbone from random initialization using a fixed codebook learned from pretrained Qwen3-4B. Thus, our proposed approach, masked diffusion and AR baseline are comparable and share the same downstream training token budget. Note that the training token budget only count noisy tokens in training samples and excludes clean tokens for block-diffusion training. Hence, the training FLOP of (binary/masked) diffusion will be larger than AR although they have the same training tokens budget.

For detailed hyper-parameters, architecture, and data mixture we refer reader to Appendix A.

Evaluation. Prior work on binary diffusion mostly uses generative perplexity for evaluation. However, perplexity rankings can disagree with downstream accuracy and favor repetition. Unlike prior binary diffusion studies, which focus on small-scale experiments and generative perplexity, we evaluate on downstream task accuracy across multiple scales and training budgets. We therefore use GSM8K accuracy throughout, as it is simple enough for evaluating small-scale models. We report five-shot accuracy using flexible answer extraction.

Inference. The main results uses a 256-token output budget and diffusion blocks of $B = 8$. Binary decoding uses at most $2B$ steps per block; for masked diffusion, we use B steps per blocks for best performance. The autoregressive baseline decodes greedily. For binary diffusion we set the temperature $\tau = 0.8$, the maximum number of anchor tokens will increase linearly during denoising and set to $4 \times$ step e.g., at denoising step 2 we can anchor maximum 8 tokens. Note that we don't use classifier-free guidance for our binary diffusion.

4.2 Comparing with masked diffusion across scales

Figure 4 compares BinaryDLM with autoregressive (AR) and uniform-masked diffusion baselines across model sizes from 0.3B to 4B parameters and training budgets of up to 100B target tokens. All models are trained on the same math mixture, allowing us to examine how the relative performance of the three modeling approaches changes with both model scale and training progress. Across scales, BinaryDLM remains competitive throughout training and achieves the highest final GSM8K accuracy at every model size.

At 100B training tokens, BinaryDLM improves over the AR baseline by 5.2, 2.4, 8.5, 4.7, and 8.2 percentage points for the 0.3B, 0.6B, 1B, 2B, and 4B models, respectively. Compared with uniform-

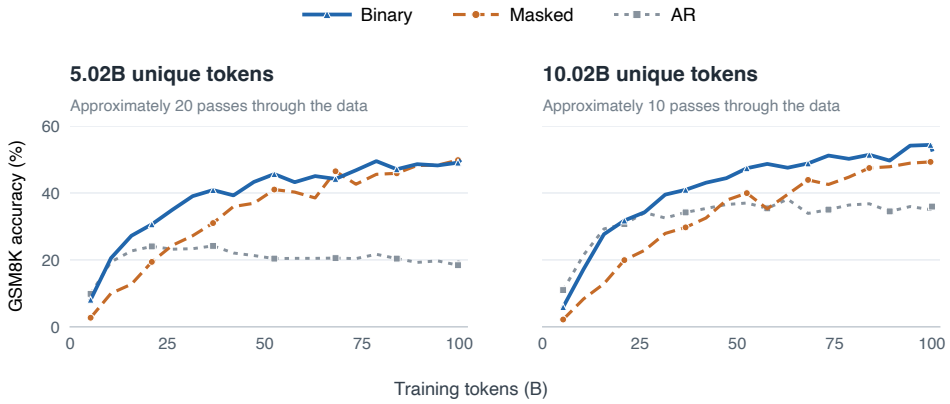


Figure 5: Data-constrained training of 1B-parameter models on GSM8K. We reuse 5.02B or 10.02B unique tokens for 100B total training tokens, corresponding to approximately 20 or 10 passes through the data. Both panels use the same accuracy scale.

masked diffusion, the corresponding improvements are 0.7, 3.9, 8.0, 6.8, and 10.2 points. The relatively small gap at 0.3B grows substantially at larger model sizes, suggesting that the advantages of preserving partial token information become more pronounced as model capacity increases. In particular, the 4B model improves over masked diffusion by 10.2 points despite both methods being trained on the same target-token budget. These results indicate that the benefit of BinaryDLM is not restricted to a particular model size or training regime, but persists across the full range of scales considered here.

4.3 Data-Constrained Training

Prior work reports that diffusion and autoregressive models behave differently when training repeatedly reuses a fixed dataset (Prabhudesai et al., 2025; Ni et al., 2025). Specifically, they found that masked diffusion language model are significantly stronger than autoregressive when the training corpus is limited and we can repeat to even hundreds epochs. Thus, we ask if the proposed method can achieve good performance under these conditions. We conduct ablation study on model size of $1B$ and total target tokens of $100B$. We then sample a small subset of our training data above where each corpus has 5.02B and 10.02B unique tokens. These settings correspond to approximately 20 and 10 passes through the data, respectively.

At both unique-data budgets, autoregressive accuracy peaks earlier around epochs $4-6$ consistent with past finding, while the diffusion models continue improving and finish higher as in Figure 5. Binary and masked diffusion reach similar final accuracy with 5B unique tokens, while binary finishes higher with 10B. Thus, under this data constrain setup, binary also benefits from increasing epochs and performs on par or slightly better than masked diffusion.

4.4 Block Size

We conduct ablation study to see if our proposed method robust across block size. More specific, we use $1B$ models with block size of $B = 16$ and $B = 32$. For inference, we uses 32 and 64 steps per block for binary diffusion; masked diffusion uses 16 and 32. Other configuration e.g., number generated tokens, temperature are similar to standard protocol. Larger blocks therefore lose quality at the same nominal iteration budget within each family, however, our proposed method still outperform masked diffusion in this setup as in Figure 6.

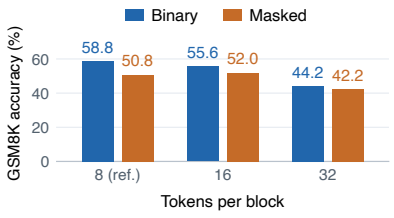


Figure 6: Varying block size ablation between binary diffusion and masked diffusion.

Table 1: Benchmark results for BinaryDLM-8B and other diffusion models (%).

Model	Objective	B	GSM8K	MATH 500	GPQA	Human Eval	MBPP	AIME 2024	AIME 2025	LCB v5	LCB v6
LLaDA-8B-Instruct[a]	Masked	–	79.9	39.2	33.3	49.4	41.0	–	–	–	–
LLaDA-MoE[b]	Masked	–	–	–	–	–	–	3.3	0.0	12.9	–
Dream-7B-Instruct[c]	Masked	–	81.0	43.0	33.0	55.5	58.8	–	–	–	–
iLLaDA-8B-Instruct[d]	Masked	–	89.0	–	–	65.9	58.0	–	–	–	–
SDAR-8B-Chat[e]	Block	4	91.3	78.6	40.2	78.7	72.0	10.0	10.0	–	16.6
LLaDA2.0-mini[f] (16B)	Block	32	94.2	–	48.0	86.6	81.5	–	36.7	–	–
Fast-dLLM v2-7B[g]	Block	32	83.7	–	31.9	63.4	63.0	–	–	–	–
Efficient-DLM-8B[h]	Block	32	88.3	–	–	68.3	61.0	–	–	–	–
TDAR-8B[i] + BACD + TCCF	Block	16/1	–	–	–	–	–	42.9	35.8	42.6	–
DiRL-8B-Instruct[j]	Block	–	93.1	85.1	–	–	–	21.5	22.9	–	–
TraDo-8B-Instruct[k]	Block	4	92.3	78.5	–	–	–	13.3	–	–	–
TraDo-8B-Thinking[k]	Block	4	94.2	87.4	–	–	–	35.5	–	–	–
WeDLM-8B-Instruct[l]	Causal diff.	–	92.3	–	45.0	80.5	70.5	–	–	–	–
NLD-8B[m]	AR+diff.	–	93.0	85.8	43.9	78.7	83.9	46.7	26.7	–	–
LLaDA MoE v2[n] 30B-A3B-Instruct	Masked	64	–	–	–	–	81.0	30.0	20.0	–	27.8
Binary DLM-8B (ours)	Binary block	64	93.2	91.1	53.2	81.2	76.9	48.9	50.0	46.6	44.3

B: inference block length; –: unreported or inapplicable.
Sources: [a]Nie et al. (2025); [b]Zhu et al. (2025a); [c]Ye et al. (2025); [d]Nie et al. (2026); [e]Cheng et al. (2025); [f]Bie et al. (2025); [g]Wu et al. (2025); [h]Fu et al. (2025); [i]Lu et al. (2026); [j]Zhu et al. (2025b); [k]Wang et al. (2025a); [l]Liu et al. (2025); [m]Fu et al. (2026); [n]Zhu et al. (2026). LLaDA and Dream scores follow Fu et al. (2026); LLaDA-MoE scores follow Lu et al. (2026).

4.5 Adapting a Pretrained 8B Model

Recent diffusion language models are often initialized from pretrained autoregressive (AR) models to improve performance and reduce training cost. Following this paradigm, we adapt Qwen3-8B (Yang et al., 2025) to investigate whether BinaryDLM transfers effectively to a strong pretrained model. Unlike our from-scratch experiments, which focus primarily on mathematical reasoning, we continue training the 8B model on a broader mixture of math, code, science, and instruction-following data.

The adapted model retains the core BinaryDLM formulation but uses a different training recipe. In particular, we adopt a curriculum in which the diffusion block size gradually increases from $B = 2$ to $B = 64$, together with a multi-stage training procedure. We first perform continued pretraining for 700B tokens on a mixture of open-source datasets, using a sequence length of 4096, a global batch size of 1024, and a learning rate of 3×10^{-5}. We then conduct supervised fine-tuning (SFT) for 400B tokens with a sequence length of 16384, a global batch size of 256, and a learning rate of 2×10^{-5}. Finally, we perform an additional 50B tokens of long-context training with a sequence length of 65536 and a global batch size of 32.

For evaluation, we use the block size of 64 and maximum number of denoising steps 200, maximum generated blocks are set to 950. We report the performance of the resulting model in Table 1 across a broad set of benchmarks. BinaryDLM-8B is competitive with state-of-the-art diffusion language models of similar scale, including models that employ RL-based post-training, while operating with a substantially larger diffusion block size.

4.6 Decoding Ablations

In this section, we conduct ablation study on the performance of 8B binary diffusion with different inference configurations.

Token anchoring. We vary the anchor-budget increment in the adapted BinaryDLM-8B decoder with a fixed budget of 200 denoising steps per block. This parameter controls how quickly the upper budget for clean anchor tokens grows during denoising (Section 3.3); anchors are reselected at every update. Intuitively, a low value of anchor-budget limit clean tokens emerges during training and

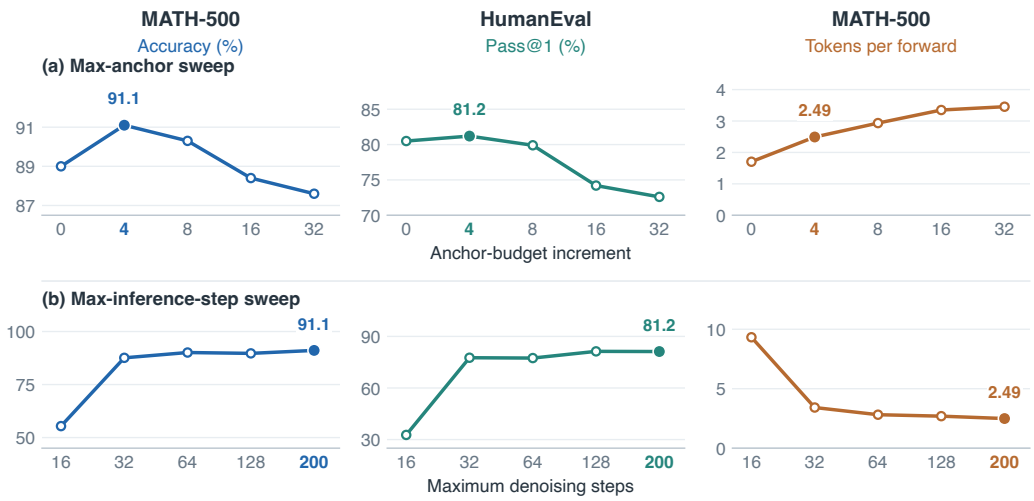


Figure 7: Decoding ablations for adapted BinaryDLM-8B. (a) Max-anchor sweep, we change the growing budget of token anchoring while keeping other hyperparameters (b) Max-inference-step sweep. Tokens per forward (TPF) are measured on all samples of MATH-500 evaluation.

blur the denoising trajectory of other tokens within blocks. At the same time, setting this value too high will exposure the model to the risk of having multiple "noisy" clean tokens containing frequent phrases e.g., the, to, of. We report the performance in Figure 7a. Among the tested values $\{0, 4, 8, 16, 32\}$, the reference increment of 4 has the highest reported scores on both MATH-500 (91.1%) and HumanEval (81.2%), exceeding zero increment by 2.1 and 0.7 percentage points, respectively. Larger increments reduce performance: at 32, the scores fall to 87.6% and 72.6%, with a larger decline on code generation. These results favor a moderate rate of anchoring over rapidly expanding the clean-token budget in this decoding setting.

Denoising steps. We also vary the maximum denoising budget from 16 to 200 steps per block, fixing the anchor-budget increment at 4 (Figure 7b). Increasing the budget from 16 to 32 steps raises MATH-500 accuracy from 55.4% to 87.6% and HumanEval pass@1 from 32.7% to 77.6%. Beyond 32 steps, gains are smaller. The results show a large benefit from moving beyond the smallest tested budget, followed by diminishing quality gains from additional denoising steps.

It is worth noticing this is the "maximum" steps per blocks, most of the time the model will "early stop" much faster than our budget, we therefore report the tokens per forwards on average for MATH-500 benchmarks for reference.

5 Conclusion and Future Work

In this work, we introduced BinaryDLM, which combines learned binary codes and token-level diffusion forcing to preserve emerging token information. Experiments across model sizes show substantial gains over the masked-diffusion baseline and validate the effectiveness of each component on real downstream benchmarks. Additionally, our adapted Qwen3-8B outperforms or is comparable with state-of-the-art masked diffusion models of the same size. The controlled from-scratch comparison focuses on math reasoning; extending it to general language and code generation is a natural next step. As binary embeddings are also widely adopted in the visual domain, a promising direction is incorporating our proposed method into unified, multimodal models.

AI use statement

Generative AI tools were used to assist with literature search and summarization of related work, background research, software implementation and code editing, and revision of the manuscript for clarity and readability. All AI-assisted research, code, and text were reviewed and verified by the

authors. The authors take full responsibility for the content of the paper, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

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.

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. CoBit: Language modeling with bitstream diffusion. *arXiv preprint arXiv:2605.07013*, 2026.

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. URL https://arxiv.org/abs/2512.15745v2.

Chen-Hao Chao, Wei-Fang Sun, Hanwen Liang, Chun-Yi Lee, and Rahul G. Krishnan. Beyond masked and unmasked: Discrete diffusion models via partial masking. *arXiv preprint arXiv:2505.18495*, 2025.

Chen-Hao Chao, Wei-Fang Sun, Junwei Quan, Chun-Yi Lee, and Rahul G. Krishnan. MDM-Prime-v2: Binary encoding and index shuffling enable scaling of diffusion language models. *arXiv preprint arXiv:2603.16077*, 2026. URL https://arxiv.org/abs/2603.16077v4.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *arXiv preprint arXiv:2407.01392*, 2024.

Sijin Chen, Yinuo Ren, Heyang Zhao, Ziheng Cheng, Quanquan Gu, and Lexing Ying. Multi-Mask Diffusion Language Models for Few-Step Generation. *arXiv preprint arXiv:2607.19686*, 2026. URL https://arxiv.org/abs/2607.19686v3.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

DiffusionGemma Team. DiffusionGemma Technical Report. *arXiv preprint arXiv:2608.00146*, 2026. URL https://arxiv.org/abs/2608.00146v1.

Chao Feng, Zhiyang Xu, Bowei Chen, Yuanjun Xiong, Xiyao Wang, Jui-Hsien Wang, Richard Zhang, Zhe Lin, Andrew Owens, and Yijun Li. On the diffusibility of high-dimensional latents. *arXiv preprint arXiv:2609.28473*, 2026. URL https://arxiv.org/abs/2609.28473.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*, 2025.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukić, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-Labs-Diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. *arXiv preprint arXiv:2510.17206*, 2025.

Keyue Jiang, Yuxiang Wang, Yanan Zhao, Xiang Yu, Qifang Zhao, Bohan Tang, Baojian Zhou, Yanghua Xiao, Lin Qu, and Xiaoxiao Xu. Semantic DLM+: Improving Diffusion Language Models through Bias-variance Trade-off in Transition Kernel Design. *arXiv preprint arXiv:2606.15327*, 2026. URL https://arxiv.org/abs/2606.15327v1.

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC-Math: A 133 billion-token-scale high quality math pretraining dataset. *arXiv preprint arXiv:2508.15096*, 2025. URL https://arxiv.org/abs/2508.15096.

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

Vitaliy Kinakh and Slava Voloshynovskiy. Binary diffusion probabilistic model. *arXiv preprint arXiv:2501.13915*, 2025.

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

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, et al. Torchtitan: One-stop pytorch native solution for production ready llm pretraining. In *International Conference on Learning Representations*, volume 2025, pp. 92161–92181, 2025.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. WeDLM: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025. URL https://arxiv.org/abs/2512.22737v1.

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

Yi Lu, Deyang Kong, Jianing Wang, Linsen Guo, Xue Wang, Qi Guo, Tao Gui, Xuanjing Huang, Wei Ye, Shikun Zhang, and Wei Wang. Adaptive test-time compute allocation for block diffusion language models in complex reasoning. *arXiv preprint arXiv:2602.09555*, 2026.

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*, 2025.

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

Shen Nie, Qiyang Min, Shaoxuan Xu, Zihao Huang, Yuxuan Song, Yong Shan, Yankai Lin, Wayne Xin Zhao, Chongxuan Li, and Ji-Rong Wen. Improved large language diffusion models. *arXiv preprint arXiv:2606.25331*, 2026.

NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. Technical report, NVIDIA, 2025. URL https://arxiv.org/abs/2512.20848.

NVIDIA et al. NVIDIA Nemotron Nano 2: An accurate and efficient hybrid Mamba-Transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025. URL https://arxiv.org/abs/2508.14444.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*, 2025.

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

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

Johannes Schusterbauer, Ming Gui, Yusong Li, Pingchuan Ma, Felix Krause, and Björn Ommer. Denoising, fast and slow: Difficulty-aware adaptive sampling for image generation. *arXiv preprint arXiv:2604.19141*, 2026.

Weitian Wang, Lianlei Shan, Shubham Rai, Cecilia De La Parra, and Akash Kumar. x-Prediction Flow: Efficient continuous decoding for masked diffusion language models. *arXiv preprint arXiv:2606.29066*, 2026.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025a.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. OctoThinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b. URL https://arxiv.org/abs/2506.20512.

Christopher Wewer, Bart Pogodzinski, Bernt Schiele, and Jan Eric Lenssen. Spatial reasoning with denoising models. *arXiv preprint arXiv:2502.21075*, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026. URL https://arxiv.org/abs/2605.18530v2.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. URL https://arxiv.org/abs/2508.15487.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023. URL https://arxiv.org/abs/2309.12284.

Yifan Zhang. StackMathQA: A curated collection of 2 million mathematical questions and answers sourced from stack exchange. Technical report, Math-AI, 2024. URL https://stackmathqa.github.io/StackMathQA.pdf.

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. *arXiv preprint arXiv:2406.07548*, 2024.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/39b77b5e422b4e070e2811b73ea9bcf7-Paper-Conference.pdf.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. *arXiv preprint arXiv:2510.03206*, 2025. URL https://arxiv.org/abs/2510.03206.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. LLaDA-MoE: A sparse MoE diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025a. URL https://arxiv.org/abs/2509.24389.

Fengqi Zhu, Shaoxuan Xu, Jingyang Ou, Zebin You, Yipeng Xing, Huabin Liu, Xiaolu Zhang, Jun Zhou, Zhenzhong Lan, Yankai Lin, Wayne Xin Zhao, Jianguo Li, Chongxuan Li, and Ji-Rong Wen. LLaDA MoE v2: Scaling mixture-of-experts diffusion language models. *arXiv preprint arXiv:2608.03457*, 2026. URL https://arxiv.org/abs/2608.03457v1.

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyang He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. DiRL: An efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*, 2025b.