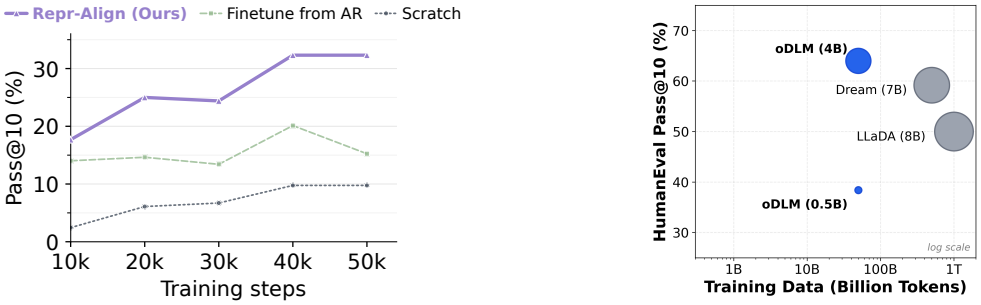
Don't Retrain—Align: Adapting Autoregressive LMs to Diffusion LMs via Representation Alignment

Anonymous authors
Paper under double-blind review

Abstract

Diffusion language models (DLMs) have recently demonstrated capabilities that complement standard autoregressive (AR) models, particularly in non-sequential generation and bidirectional editing. Although recent work has shown that pretrained autoregressive checkpoints can be converted into diffusion language models, existing recipes primarily transfer parameters through continued denoising training with objective- and attention-level modifications. We instead ask whether the internal representation geometry learned by next-token prediction can be explicitly preserved during AR→DLM conversion. We hypothesize that much of the semantic structure learned by AR pretraining can transfer across generation orders, and thus DLM training should be viewed as relearning the decoding path rather than relearning language representations. To investigate this, we introduce Repr-Align, a Representation Alignment objective that adapts a bidirectional masked diffusion model to reuse representations from a pretrained AR model of identical architecture. Concretely, we align the hidden states of the DLM to the frozen AR model at every layer using cosine similarity, while optimizing the standard masked denoising objective. This simple alignment—with no adapters and no architectural changes beyond the attention mask—yields up to 4× training acceleration in our setting and is particularly effective in low-data regimes. Our results suggest that linguistic representations are universal regardless of generation order, and representation alignment could be a new go-to technique for training diffusion language models. Code will be made publicly available.



(a) Adaptation speed. (b) Public DLM frontier.

Figure 1: Don't retrain—align. Left: Repr-Align consistently accelerates AR→DLM adaptation on HumanEval pass@10, outperforming both AR fine-tuning and scratch training throughout early conversion. Right: The resulting oDLM achieves a favorable HumanEval pass@10 versus training-data trade-off among public DLMs.

1 Introduction

The dominant paradigm in large-scale language modeling has long been autoregressive (AR) sequence modeling. By factoring the joint probability distribution as a product of conditional probabilities, models such as GPT and Qwen have demonstrated strong general-purpose generation capabilities (Shi et al., 2025; Achiam et al., 2023; Touvron et al., 2023).

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.

Recently, diffusion language models (DLMs) have emerged as an alternative formulation for text generation, spanning continuous diffusion over embeddings, discrete absorbing-state diffusion, likelihood-based diffusion LMs, masked diffusion LMs, and large-scale DLMs (Li et al., 2022; Austin et al., 2021a; Gulrajani & Hashimoto, 2023; Sahoo et al., 2024; Nie et al., 2025b; Ye et al., 2025). By framing generation as any-order decoding (Sohl-Dickstein et al., 2015; Ho et al., 2020; Yang et al., 2019; Ghazvininejad et al., 2019), DLMs naturally support non-left-to-right behaviors such as infilling and iterative refinement (Sahoo et al., 2024; Gulrajani & Hashimoto, 2023; Chang et al., 2022). Despite these advantages, scaling DLMs remains expensive. In theory DLMs learn $L!$ paths to generate a sequence vs. one left-to-right generation in ARs, and thus require L-times more compute. While several recent methods reduce this cost by initializing from pretrained AR checkpoints or converting AR models into DLMs (Gong et al., 2025a; Ye et al., 2025; Fu et al., 2025), existing conversion recipes largely reuse AR parameters through continued denoising training, attention-pattern modifications, or sampling conventions. They do not explicitly constrain the converted DLM to preserve the internal representation geometry of the AR model.

In this paper, we question the need to treat ARs and DLMs as two disjoint paradigms. We start from a simple view: the hard part of language generation is learning language representations—the semantic and syntactic structure of the data—not committing to a particular generation order. Autoregressive pretraining has already learned strong internal features that organize this structure. If so, training a diffusion language model should not require relearning language representations from scratch. Instead, the remaining work is mainly mechanical: adapt these existing features to an iterative any-order decoder. This reframes DLM training from representation learning to an alignment problem, where we reuse the AR backbone for the representations and train the diffusion mechanism to operate in the same feature space but with any-order generation.

To test this hypothesis, we apply Representation Alignment (Yu et al., 2025; Singh et al., 2025; Wu et al., 2025; Jiang et al., 2025) for the first time for a minimalist adaptation of a pretrained AR transformer into a masked diffusion language model (Figure 2). Our setup uses two models with identical architecture: (i) a pretrained AR model with causal attention, and (ii) the same architecture initialized from the AR weights but with bidirectional attention. During training, we randomly mask a sequence and optimize the DLM to predict the masked tokens. In parallel, we feed the clean sequence into the frozen AR model (teacher-forced under causal attention) and extract its hidden states at each layer. Because the two networks share the same layer structure and hidden sizes, we can directly align their intermediate representations via a layer-wise cosine-similarity loss, without introducing adapters or additional parameters. Intuitively, the AR model provides a stable representational anchor, and diffusion training is reduced to learning an any-order decoding mechanism that operates in that anchored feature space. The method is designed to change as little as possible, so that any gains can be attributed to the reuse of AR features rather than to architectural modifications or heavy fine-tuning.

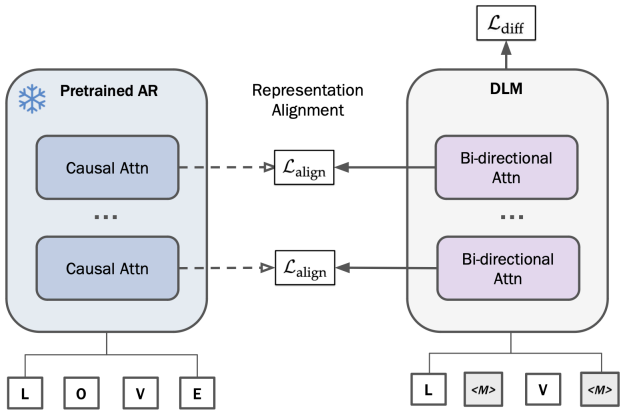


Figure 2: Overview of our method Repr-Align: we adapt a pretrained autoregressive (AR) transformer into a masked diffusion language model (DLM) by switching to bidirectional attention and training with a masked denoising objective, while anchoring layer-wise hidden states to a frozen AR backbone.

Our experiments support the reuse-through-alignment view: representation alignment turns AR→DLM conversion into a largely mechanical adaptation problem rather than relearning linguistic representations. On HumanEval, alignment improves conversion quality and the gains grow with model size (Figure 3), increasing pass@10 from 24.9 to 31.0 at 0.6B and

from 31.1 to 40.5 at 1.7B. Beyond quality, alignment enables substantially cheaper conversion through selective training (Figure 4) and remains effective under a 0.8B-token tiny subset (Figure 5). As a scale-up validation, we train a 4B oDLM using the same representation-preserving conversion recipe. Against Dream-7B, a strong public DLM that also builds on AR initialization, oDLM achieves a better HumanEval-family pass@10 trade-off (Figure 1b and Table 1): it improves HumanEval and HumanEval+ pass@10 by 2.39 and 2.40 points, respectively, despite using fewer parameters and a substantially lighter data-and-compute budget.

Contributions. Our contributions are summarized as follows:

- We identify representation preservation as a missing ingredient in AR→DLM conversion. Instead of merely initializing from an AR checkpoint, we explicitly anchor the DLM student to the frozen AR model's layer-wise hidden-state geometry during masked denoising training.
- We introduce a simple representation-preserving conversion recipe that requires no adapters or architectural changes beyond switching from causal to bidirectional attention. Across model scales, representation alignment improves conversion quality and sample efficiency, with larger gains at larger model sizes.
- We show that AR→DLM conversion is not inherently data- or parameter-update hungry. With representation alignment, training on a 0.8B-token subset can outperform training on the full 50B-token stream under the same step budget, and freezing embeddings and MLP blocks improves throughput by up to $\sim 2\times$ without degrading quality.
- As a scale-up validation, we train a 4B oDLM using the same recipe. Compared with Dream-7B, a strong public DLM that also leverages AR initialization, oDLM improves HumanEval and HumanEval+ pass@10 by 2.39 and 2.40 points, respectively, while using a smaller backbone and a substantially lighter data-and-compute budget.

2 Related Work

Diffusion language models. Diffusion language models span continuous diffusion over embeddings, discrete categorical diffusion, likelihood-based diffusion LMs, and masked diffusion LMs (Li et al., 2022; Austin et al., 2021a; Gulrajani & Hashimoto, 2023; Sahoo et al., 2024; Nie et al., 2025a). Recent large-scale systems such as LLaDA and Dream show that masked diffusion can support instruction following, reasoning, and code generation at billion-parameter scale (Nie et al., 2025b; Ye et al., 2025). These advances make DLMs a serious alternative to autoregressive generation, but competitive DLMs still require substantial diffusion-specific optimization. We provide a more detailed discussion of DLM formulations in Appendix A.1.

Adapting autoregressive models to diffusion language models. A growing line of work avoids training DLMs from scratch by converting pretrained AR checkpoints into denoising models (Gong et al., 2025a; Ye et al., 2025; Xie et al., 2025; Fu et al., 2025; Xue et al., 2025). These methods establish that AR checkpoints are strong initializations, but they primarily adapt the objective, masking process, attention pattern, or sampling convention. Our work instead explicitly preserves the AR model's internal representation geometry by aligning a bidirectional DLM student to a frozen same-architecture AR teacher, as formalized in Section 3.3. Appendices A.2 and A.3 give a fuller comparison with AR→DLM conversion, any-order generation, iterative masked decoding, and path-planning methods.

Representation Alignment for Generative Models. Representation alignment has recently accelerated diffusion training by matching generative-model hidden states to representations from strong pretrained encoders (Yu et al., 2025; Singh et al., 2025; Wu et al., 2025; Jiang et al., 2025). Our setting differs in both teacher and purpose: the teacher is not an external encoder, but the exact AR model being converted, with the same tokenizer, architecture,

hidden size, and initialization as the DLM student. Alignment therefore acts as representation preservation during mechanism adaptation, rather than feature import from another model; Equation (2) gives the exact objective. Appendix A.4 expands this distinction.

3 Method

We study the question: are representations learned by next-token prediction sufficient for masked denoising generation?

To isolate this factor, we keep the architecture fixed and change only what is necessary for diffusion-style denoising. Our method instantiates two transformers with identical parameterization and dimensionality: a pretrained autoregressive (AR) model with causal attention, and a masked diffusion model with bidirectional attention. We then train the diffusion model with the standard masked prediction objective, while adding a single layer-wise representation alignment loss to reuse the AR model's internal features. Alg. 1 summarizes the overall conversion procedure. No adapters or auxiliary modules are introduced. The four design choices below map directly to the experimental setup in Section 4.1 and the ablations in Section 4.3.

Algorithm 1 Repr-Align AR→DLM conversion with layer-wise representation alignment.

```
# f_AR: frozen AR teacher (causal attention), params theta_AR
# f_D : DLM student (bidirectional attention), params theta
# theta <- theta_AR; f_D differs only by attention mask (causal -> bidir)

theta = theta_AR
freeze(f_AR); f_AR.eval()

for x in data_stream:                          # x: [B, n] clean tokens
    r = Uniform(0, 1)                          # mask ratio (per sequence)
    M = sample_positions(x, r)
    x_tilde = x.clone(); x_tilde[M] = mask_id   # corrupted input

    # student denoising pass (bidirectional)
    logits, H_D = f_D(x_tilde, bidir=True, output_hidden_states=True)

    # teacher anchor pass (causal) on clean x
    with no_grad():
        _, H_AR = f_AR(x, causal=True, output_hidden_states=True)

    # masked denoising loss (shift logits by one token;)
    loss_diff = CE( shift_logits_1(logits)[M], x[M] )

    # layer-wise cosine alignment on hidden states
    loss_align = (1 - cos(H_D, H_AR)).mean()

    loss = loss_diff + lambda_align * loss_align
    step_optimizer(theta, loss)
```

3.1 Two models, same architecture

Let $x = (x_1, \ldots, x_n) \in \mathcal{V}^n$ be a token sequence, and let $\mathcal{V}$ include a special mask token $\langle \mathrm{M} \rangle$. We define: (i) an autoregressive transformer $f_{\mathrm{AR}}(\cdot; \theta_{\mathrm{AR}})$ with a causal attention mask, pretrained by next-token prediction; and (ii) a diffusion transformer $f_{\mathrm{D}}(\cdot; \theta)$ with bidirectional attention. Crucially, f_{AR} and f_{D} share the same layer structure and hidden size d; the only architectural difference is the attention mask. We keep θ_{AR} frozen throughout training. In practice, we initialize θ from θ_{AR} and switch the attention mask from causal to bidirectional, so that any gains can be attributed to mechanism adaptation rather than new capacity.

Let $h^{(\ell)}_{\mathrm{AR}}(x) \in \mathbb{R}^{n \times d}$ and $h^{(\ell)}_{\mathrm{D}}(x) \in \mathbb{R}^{n \times d}$ denote the hidden states at layer $\ell \in \{1, \ldots, L\}$ for the two models. This matched-conversion design is evaluated under the fixed-budget protocol in Sections 4.1 and 4.2.

3.2 Masked diffusion objective

We train f_{D} as a masked denoiser. We sample a mask set $M \subseteq \{1, \ldots, n\}$ and construct a corrupted input $\tilde{x}$ by replacing x_i with $\langle \mathrm{M} \rangle$ for $i \in M$. The diffusion model predicts the

masked tokens conditioned on $\tilde{x}$, and we optimize cross-entropy only on masked positions:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{x,M}\left[\sum_{i\in M} \text{CE}(p_\theta(\cdot \mid \tilde{x})_i,\ x_i)\right], \tag{1}$$

where $p_\theta(\cdot \mid \tilde{x})_i$ is the diffusion model's predicted distribution at position i. This denoising objective is the shared training loss used across the baseline and aligned models in Section 4.1.

Shift convention. To match the next-token indexing convention inherited from the AR checkpoint, we apply the standard shift operation used in AR→DLM adaptation: the student produces logits at positions i, which are shifted by one position before computing the denoising cross-entropy and during sampling. We use this same shift convention for both the baseline and aligned models; implementation details are provided in Appendix B.1.

3.3 Layer-wise representation alignment

To reuse AR features, we run a frozen autoregressive teacher on the clean sequence x under causal attention, and anchor the diffusion student—which consumes the corrupted $\tilde{x}$ under bidirectional attention—to the teacher's hidden states. Because the two networks share identical architecture and hidden dimensionality, we align their layer-wise representations directly without adapters.

Let $h_{\text{AR}}^{(\ell)}(x) \in \mathbb{R}^{n\times d}$ denote the teacher hidden states at layer ℓ given clean x, and $h_{\text{D}}^{(\ell)}(\tilde{x}) \in \mathbb{R}^{n\times d}$ denote the student hidden states given corrupted $\tilde{x}$. We minimize a cosine distance loss:

$$\mathcal{L}_{\text{align}}(\theta) = \frac{1}{L}\sum_{\ell=1}^{L} \mathbb{E}_{x,M}\left[\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\left(1 - \cos\left(h_{\text{D}}^{(\ell)}(\tilde{x})_i,\ \text{stopgrad}\left(h_{\text{AR}}^{(\ell)}(x)_i\right)\right)\right)\right], \tag{2}$$

where $\mathcal{I}$ is the aligned position set (default: all positions), and the teacher is run in evaluation mode with stop-gradient. Intuitively, the teacher provides a stable representational coordinate system induced by next-token pretraining, and the student learns a denoising mechanism that operates within this coordinate system. The choice of cosine distance, aligned layers, and alignment strength is ablated in Section 4.3, while the exact hidden-state tuple and masking positions used for alignment are specified in Appendix B.4.

3.4 Training objective

The full objective is a weighted sum:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{diff}}(\theta) + \lambda\,\mathcal{L}_{\text{align}}(\theta), \tag{3}$$

with a scalar λ (set to 10 by default) controlling the strength of the anchor. We optimize θ while keeping θ_{AR} fixed. Overall, this procedure changes as little as possible: the architecture is shared, the pretrained AR representations are preserved, and diffusion training is reduced to learning an any-order denoising mechanism that operates within an already-formed feature space. In addition to the standard masked diffusion model loss, we include the PAPL (Peng et al., 2026) loss from prior work as part of the default DLM training recipe, with weight 1. Sections 4.2 to 4.3 test whether the alignment term in Equation (3) improves conversion quality and how sensitive it is to λ.

4 Experiments

4.1 Experimental Setup

AR-to-DLM conversion setting. We study efficient adaptation from an autoregressive (AR) causal language model to a masked diffusion language model (DLM) using Qwen3 checkpoints (Yang et al., 2025) at three scales (0.6B, 1.7B, and 4B parameters). For each scale, we treat the original causal model as a frozen teacher and initialize the DLM student from

the same checkpoint, changing only the attention mask from causal to bidirectional. This setup instantiates the same-architecture construction in Section 3.1.

Training data. We train on the Nemotron-SFT-Code dataset (NVIDIA et al., 2025) (approximately 50B tokens and 70M sequences). This release is a large synthetically generated and curated SFT-style corpus covering STEM, academic, reasoning, and multilingual instruction data, and includes code-focused instruction data within the Nemotron-SFT-Code subset. The resulting training stream contains approximately 70M sequences and ∼50B tokens. To study data efficiency, we also consider a tiny subset constructed by uniformly subsampling to ∼0.8B tokens. For fixed-compute comparisons, we sample with replacement so that the number of optimization steps is matched. In both the full and tiny settings, training examples are sampled with replacement, so the optimization budget is controlled by steps (and thus total tokens processed) rather than epochs. The tiny-data comparison in Section 4.2 uses this matched-step construction.

Representation alignment. In addition to the denoising loss, we apply layer-wise representation alignment to anchor the student to the frozen AR teacher. The teacher consumes the clean sequence x (causal attention), while the student consumes the corrupted $\tilde{x}$ (bidirectional attention). We align the output of every transformer block at every token position using a cosine distance loss, with teacher features stop-gradiented and the teacher run in evaluation mode. We use $\lambda = 10$ as the default alignment weight, selected from the ablation in Table 2. We additionally enable the PAPL loss (Peng et al., 2026) in all runs with weight 1. Unless stated otherwise, we treat PAPL as part of the default DLM training recipe and include it in all methods (baseline and aligned) so comparisons isolate the effect of representation alignment. Section 4.3 varies the metric, layer set, and λ in this alignment term.

Optimization, batching, and evaluation. We optimize using AdamW with learning rate 3×10^{-4}, cosine decay, warmup ratio 0.001, weight decay 0.01, gradient clipping (max norm 1.0), and mixed-precision training. The maximum sequence length is 4096, and the global batch size is 96 sequences per optimization step. We evaluate code generation on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), and their EvalPlus variants (Liu et al., 2023). HumanEval uses the canonical docstring prompts, and all evaluations are zero-shot. For decoding, we use P2-self sampling (Peng et al., 2025) with 128 sampling steps, max_new_tokens = 128, temperature 0.8, and top-p 0.95. Full data preprocessing, optimization, decoding, and evaluation configurations are reported in Appendices B.2, B.6 and B.7.

4.2 Results

The experiments are organized around four questions induced by Section 3. First, does the alignment term in Equation (2) improve AR→DLM adaptation under a matched training budget? Second, do the gains scale with model size? Third, does representation preservation reduce the need to update all parameters? Fourth, does conversion remain effective when the adaptation data is sharply reduced? The following paragraphs answer these questions, respectively.

Repr-Align improves adaptation efficiency and quality. Figure 3 evaluates autoregressive-to-diffusion conversion on HumanEval under a fixed 200k-step budget. Starting from the same pretrained Qwen3 checkpoint, the baseline performs masked diffusion training (MDLM-style masking with the shift convention and PAPL), while our method additionally anchors the student to the frozen AR teacher via layer-wise cosine alignment. Repr-Align consistently improves pass@10 throughout training, indicating substantially better sample efficiency during adaptation. At the 200k-step cutoff, the aligned model achieves higher pass@10 than the baseline (a gain of 6.1 points at 0.6B and 9.4 points at 1.7B), demonstrating that anchoring pretrained representations is beneficial even when the student is trained with a bidirectional denoising objective. This validates the alignment hypothesis encoded by Equation (2) under the matched conversion setup in Section 3.1.

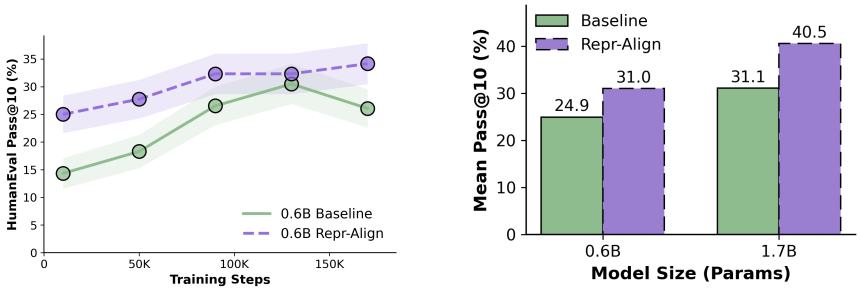


Figure 3: Repr-Align improves both adaptation speed and final quality. Left: HumanEval pass@10 vs. training steps for Qwen3-0.6B during AR→DLM conversion; adding representation alignment to the frozen AR teacher improves sample efficiency throughout training. Right: pass@10 results for 0.6B and 1.7B models; representation alignment provides larger gains at 1.7B than at 0.6B.

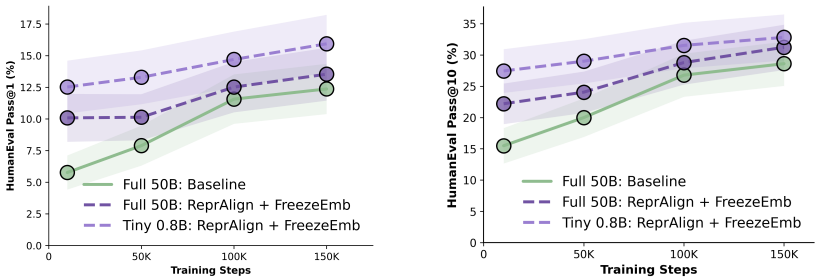
Alignment gains grow with model size. Figure 3 summarizes the same conversion procedure across model scales. For each size, we compare the last checkpoint after 200k steps for the baseline and aligned variants under identical evaluation. We observe that alignment benefits increase with model capacity: the absolute improvement from alignment is larger for the 1.7B model than for the 0.6B model. This scaling trend supports the view that AR pretraining learns a strong representational geometry that is increasingly valuable to preserve as model capacity grows, while the remaining adaptation primarily concerns the generation mechanism induced by the attention mask and denoising dynamics. This directly tests whether the preservation term in Equation (3) becomes more useful as the AR teacher's representation capacity increases.

oDLM is competitive with public diffusion language models. Figure 1b gives the high-level comparison that motivates the paper: after conversion, oDLM reaches the public frontier of diffusion language models on code-generation pass@10 while using pretrained AR representations rather than retraining a diffusion LM from scratch. Table 1 reports the full benchmark breakdown across HumanEval, HumanEval+, MBPP, and MBPP+ against recent public DLM systems, with code-focused diffusion LMs such as DiffuCoder and Dream-Coder providing closely related context for this evaluation setting (Nie et al., 2025b; Ye et al., 2025; Gong et al., 2025b; Xie et al., 2025). The strongest oDLM checkpoint is especially competitive on pass@10, which is the metric most directly tied to iterative diffusion sampling quality. This comparison evaluates the full objective in Equation (3) rather than the alignment term in isolation.

Table 1: Code generation results on HumanEval, HumanEval+, MBPP, and MBPP+ benchmarks, compared with recent public diffusion language models (Nie et al., 2025b; Ye et al., 2025).

	HumanEval		HumanEval+		MBPP		MBPP+	
Model	Pass@1	Pass@10	Pass@1	Pass@10	Pass@1	Pass@10	Pass@1	Pass@10
LLaDA (8B)	<u>35.4</u>	50.0	<u>30.5</u>	43.3	<u>38.8</u>	<u>53.4</u>	<u>52.6</u>	<u>69.1</u>
Dream (7B)	56.7	<u>59.2</u>	50.0	<u>53.7</u>	55.4	56.2	71.5	72.5
Mask DFM (1.3B)	9.1	17.6	7.9	13.4	6.2	25.0	–	–
Edit Flow (1.3B)	12.8	24.3	10.4	20.7	10.0	36.4	–	–
oDLM (0.6B)	17.87	35.98	16.40	31.10	15.70	37.00	22.99	49.47
oDLM (4B)	30.49	61.59	26.95	56.10	14.42	37.00	20.24	47.35

Freezing large blocks improves throughput with a mild quality gain. If AR pretraining already provides strong representations, and representation alignment preserves the teacher's internal feature space during conversion, then AR→DLM adaptation should not require updating the entire network. We test this by freezing large parameter blocks in the aligned student while keeping the rest of the training protocol fixed. Figure 4 shows a favorable performance–efficiency trade-off at the 1.7B scale: freezing token embeddings and, more aggressively, freezing both embeddings and MLP blocks yields a substantial increase in training throughput (up to $\sim 2\times$), while maintaining and even slightly improving HumanEval pass@10. This supports the view that, once representations are anchored, the remaining learning signal is concentrated in adapting the denoising mechanism under bidirectional attention, and freezing provides a practical knob to reduce conversion cost without sacrific-



(a) HumanEval pass@1. (b) HumanEval pass@10.

Figure 5: Alignment is not data-hungry: a tiny subset can improve conversion. The tiny run trains on a 0.8B-token random subsample instead of the full 50B-token stream. With representation alignment, the tiny subset yields consistently higher pass@1 (left) and pass@10 (right), supporting the view that AR→DLM conversion is primarily mechanism adaptation without massive training data.

ing quality. This result tests the mechanism-adaptation interpretation of Section 3.3; the freezing protocol is specified in Appendix B.9.

Alignment is especially data-efficient: tiny data can improve conversion. To test whether AR-to-DLM conversion is inherently data-hungry, we reduce the training corpus from the full 50B-token Nemotron-SFT-Code stream to a tiny 0.8B-token random subsample, while keeping the optimization budget fixed (same batch size, sequence length, and number of steps; tiny data is sampled with replacement). Figure 5 shows a clear and somewhat surprising outcome: with representation alignment, training on the tiny subset yields higher HumanEval pass@10 than training on the full data stream at the same training steps. This indicates that, once pretrained AR representations are preserved, the remaining learning problem is primarily mechanism adaptation (adapting bidirectional attention and denoising dynamics), which can be accomplished with remarkably little data under a fixed-step budget. This validates the low-data implication of anchoring the student to the teacher representation in Equation (2).

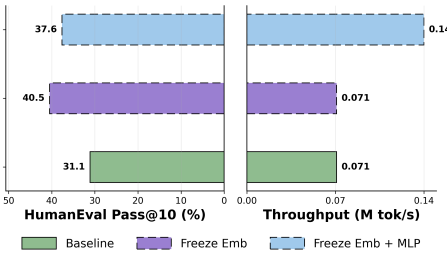


Figure 4: Freezing improves training efficiency with a mild performance gain (1.7B). All runs use representation alignment and share the same training protocol and budget; we only vary which parameter blocks are frozen. Freezing embeddings and MLP blocks increases throughput by up to ∼2× while slightly improving HumanEval pass@10.

Practical takeaway. Across scales and data regimes, our experiments support a simple recipe for training diffusion language models efficiently. Given a pretrained AR transformer, we can convert it into a competitive masked diffusion model by switching to bidirectional attention, training with an MDLM-style denoising objective with the shift convention, and anchoring intermediate representations to the frozen AR teacher with a layer-wise cosine loss. When compute is constrained, freezing token embeddings and even MLP blocks further improves efficiency with little or no degradation in quality. When data is constrained, alignment enables effective adaptation even on a tiny subset, suggesting that large diffusion pretraining costs are not intrinsic, but largely reflect relearning representations that AR pretraining already provides.

4.3 Ablation Studies

We ablate the main design choices in Equations (2) and (3): the distance used to match hidden states, the strength λ of the alignment term, and the set of layers included in the alignment loss. All ablations use the same AR→DLM conversion protocol unless otherwise stated: the student is initialized from the AR checkpoint, trained with bidirectional masked denoising, evaluated on HumanEval, and decoded with the same sampling configuration. This setup isolates whether the gain comes from preserving the pretrained representation

Table 2: Ablations of representation alignment on HumanEval.

Metric			Weight			Layers		
Variant	pass@1	pass@10	λ	pass@1	pass@10	Variant	pass@1	pass@10
L2	12.00	25.00	1	7.50	22.60	Lower	11.28	25.00
Cosine	18.00	31.00	5	9.90	26.80	Middle	15.73	31.10
			10	18.00	31.00	Upper	8.96	31.71
			20	11.30	29.30	All	18.00	31.00

geometry rather than from changes in architecture or decoding. Table 2 summarizes the main ablation outcomes. Cosine alignment is the best default metric; $\lambda = 10$ gives the strongest overall anchor; and all-layer alignment provides the best pass@1 while maintaining competitive pass@10.

Cosine alignment is better than matching hidden states by L2. We first compare two natural choices for the representation loss: an L2 loss on hidden states and a cosine-distance loss. This directly tests the cosine-distance choice in Equation (2). Cosine alignment improves HumanEval pass@1 from 12.0 to 18.0 and pass@10 from 25.0 to 31.0, suggesting that the useful signal in the AR teacher is primarily geometric rather than metric. Because transformer hidden-state norms vary across layers, tokens, and training states, an L2 objective can be dominated by scale; cosine alignment instead preserves representation direction. We therefore use cosine alignment as the default representation loss.

The alignment weight controls a trade-off between anchoring and adaptation. We next vary the alignment weight λ in Equation (3), which controls the trade-off between anchoring to the AR teacher and adapting to the denoising objective. The sweep shows a clear trade-off: weak alignment underuses the AR teacher, while excessive anchoring constrains the diffusion student. Performance rises from $\lambda = 1$ to $\lambda = 10$, reaching 18.0 pass@1 and 31.0 pass@10, but drops at $\lambda = 20$. This supports treating representation alignment as an auxiliary constraint rather than exact teacher imitation. We therefore set $\lambda = 10$ by default.

Where to align matters. Finally, we partition hidden states into lower, middle, and upper thirds to test where the AR teacher is most useful. Middle- and upper-layer anchors improve pass@10, but only all-layer alignment gives the strongest pass@1 while preserving competitive pass@10. The transferable signal therefore appears distributed across depth. We therefore use all-layer cosine alignment as the default.

5 Conclusion

In this work, we challenge the prevailing dichotomy between autoregressive and diffusion language models. Our empirical investigation substantiates that the rich semantic representations acquired during standard autoregressive pretraining are not specific to sequential decoding, but are broadly applicable to non-autoregressive generation. By introducing Repr-Align, we demonstrate that it is possible to inherit this semantic topology directly. We achieve state-of-the-art performance with a fraction of the standard pretraining cost.

Our findings have broader implications for discrete generation. Alternative frameworks such as uniform, latent, and simplex diffusion models have historically struggled to scale, often due to optimization difficulties. By anchoring new generation mechanisms to a robust AR prior, Repr-Align offers a recipe for adapting pretrained representations rather than training each paradigm ab initio. We hope this encourages a shift toward mechanism alignment as a practical way to unlock new forms of generation.

AI use statement

In this work, we used generative AI tools for manuscript formatting, LaTeX template conversion, grammar and style editing, and for assisting with plotting code. We have not used generative AI tools for generating research ideas, hypotheses, experimental designs, or the scientific claims and results reported in this paper. All AI-assisted edits were reviewed

and verified by the authors, who take full responsibility for the final content of this work, including text produced with the aid of generative AI.

Ethics statement

This work can lower the compute barrier for training competitive diffusion language models, which may make efficient generation research more accessible. Cheaper conversion of pretrained language models can also make code synthesis and other generative capabilities easier to scale for harmful or deceptive uses, so deployment should follow ordinary model-stewardship practices. Because our contribution is a training method rather than a released model or dataset, we do not propose separate release restrictions.

Reproducibility statement

All models, datasets, and evaluation protocols are standard and publicly available: Qwen3 decoder-only checkpoints, the code-training stream described in Appendix B.2, and HumanEval/MBPP with their EvalPlus variants. Appendix B (model conversion, data preprocessing, the masked diffusion objective, the representation-alignment loss, and optimization details) together with Appendices B.7 and B.8 specifies every hyperparameter needed to rerun training and evaluation, including the full ablation grid. Decoding uses P2-self sampling with the configuration given in Appendix B.6. Code will be made publicly available.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv, 2023. 1

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. arXiv, 2021a. 2, 3, 14

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021b. URL https://arxiv.org/abs/2108.07732. 6, 19

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models, 2022. 14

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11315–11325, 2022. 2, 16

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374. 6, 19

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, 2025. URL https://arxiv.org/abs/2512.14067. 2, 3, 15

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 6112–6121. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633/. 2, 16

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In International Conference on Learning Representations, 2025a. 2, 3, 15

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, 2025b. URL https://arxiv.org/abs/2506.20639. 7, 15

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. Neural Information Processing Systems, 2023. 2, 3, 14

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020. 2

Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. arXiv preprint arXiv:2505.02831, 2025. 2, 3, 16

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. 2, 3, 14

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Advances in Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=1qvx610Cu7. 6, 19

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In International Conference on Machine Learning, 2023. 14

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. International Conference on Learning Representations, 2025a. 3, 14

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, 2025b. URL https://arxiv.org/abs/2502.09992. 2, 3, 7, 14

NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, Andrew Tao, Ann Guan, Anna Shors, Anubhav Mandarwal, Arham Mehta, Arun Venkatesan, Ashton Sharabiani, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Banghua Zhu, Barnaby Simkin, Bilal Kartal, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Brian Yu, Bryan Catanzaro, Charles Wang, Charlie Truong, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christian Munley, Christopher Parisien, Dan Su, Daniel Afrimi, Daniel Korzekwa, Daniel Rohrer, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Dima Rekesh, Dina Yared, Dmytro Pykhtar, Dong Ahn,

Duncan Riach, Eileen Long, Elliott Ning, Eric Chung, Erick Galinkin, Evelina Bakhturina, Gargi Prasad, Gerald Shen, Haifeng Qian, Haim Elisha, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Hoo Chang Shin, Hua Huang, Iain Cunningham, Igor Gitman, Ivan Moshkov, Jaehun Jung, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jian Zhang, Jiaqi Zeng, Jimmy Zhang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jonathan Cohen, Joseph Jennings, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kezhi Kong, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Kushan Ahmadian, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Luis Vega, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Mark Cai, Markus Kliegl, Marta Stepniewska-Dziubinska, Matvei Novikov, Mehrzad Samadi, Meredith Price, Meriem Boubdir, Michael Boone, Michael Evans, Michal Bien, Michal Zawalski, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Namit Dhameja, Nave Assaf, Negar Habibi, Nidhi Bhatia, Nikki Pope, Nima Tajbakhsh, Nirmal Kumar Juluru, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pablo Ribalta, Padmavathy Subramanian, Parth Chadha, Pavlo Molchanov, Peter Dykas, Peter Jin, Piotr Bialecki, Piotr Januszewski, Pradeep Thalasta, Prashant Gaikwad, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi Mahabadi, Rajen Patel, Ran El-Yaniv, Ranjit Rajan, Ria Cheruvu, Rima Shahbazyan, Ritika Borkar, Ritu Gala, Roger Waleffe, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Sahil Jain, Samuel Kriman, Sanjeev Satheesh, Saori Kaji, Sarah Yurick, Saurav Muralidharan, Sean Narenthiran, Seonmyeong Bak, Sepehr Sameni, Seungju Han, Shanmugam Ramasamy, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shizhe Diao, Shreya Gopal, Shrimai Prabhumoye, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Siddhartha Jain, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Syeda Nahida Akter, Terry Kong, Tim Moon, Tomasz Hliwiak, Tomer Asida, Tony Wang, Tugrul Konuk, Twinkle Vashishth, Tyler Poon, Udi Karpas, Vahid Noroozi, Venkat Srinivasan, Vijay Korthikanti, Vikram Fugro, Vineeth Kalluru, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wonmin Byeon, Ximing Lu, Xin Dong, Yashaswi Karnati, Yejin Choi, Yian Zhang, Ying Lin, Yonggan Fu, Yoshi Suhara, Zhen Dong, Zhiyu Li, Zhongbo Zhu, and Zijia Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444. 6

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling, 2025. URL https://arxiv.org/abs/2502.03540. 6, 16

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Michael M. Bronstein, Anru Zhang, Joey Bose, and Alexander Tong. Planner aware path learning in diffusion language models training. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=lAlI5FuIf7. 5, 6, 16

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. 2, 3, 14

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329. 1

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. Neural Information Processing Systems, 2022. 16

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? arXiv preprint arXiv:2512.10794, 2025. 2, 3, 16

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei

(eds.), Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html. 2

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. arXiv, 2023. 1

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. arXiv preprint arXiv:2507.01467, 2025. 2, 3, 16

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream-coder 7b: An open diffusion language model for code, 2025. URL https://arxiv.org/abs/2509.01142. 3, 7, 15

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order gpt as masked diffusion model: Decoupling formulation and architecture, 2025. URL https://arxiv.org/abs/2506.19935. 3, 15

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388. 5

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In Advances in Neural Information Processing Systems, 2019. 2, 16

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487. 2, 3, 7, 14, 15

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In International Conference on Learning Representations, 2025. 2, 3, 15, 16

Appendix Contents

A Extended Related Work 14
A.1 Diffusion language models . . . 14
A.2 Autoregressive-to-diffusion language model conversion . . . 15
A.3 Any-order generation, iterative decoding, and path planning . . . 15
A.4 Representation alignment and feature preservation . . . 16

B Experimental Details 17
B.1 Model Conversion . . . 17
B.2 Data and Preprocessing . . . 17
B.3 Masked Diffusion Objective . . . 18
B.4 Representation Alignment . . . 18
B.5 Path-Planning Loss . . . 19
B.6 Optimization . . . 19
B.7 Decoding and Evaluation . . . 19
B.8 Ablation Protocols . . . 20
B.9 Freezing Protocol . . . 21

C Limitations 21

A Extended Related Work

This appendix provides a more detailed comparison with the closest lines of work. The purpose is not to survey all diffusion or language-generation methods exhaustively, but to clarify the specific position of our method: we study AR→DLM conversion through representation preservation. Existing conversion methods show that pretrained autoregressive checkpoints are useful initializations for diffusion language models, but they primarily adapt the objective, attention pattern, masking convention, or sampling procedure. Our method instead explicitly preserves the hidden-state geometry of the pretrained AR model by anchoring the DLM student to a frozen same-architecture AR teacher. This appendix supports the positioning claims made in Section 2 and the methodological distinction formalized in Section 3.3.

A.1 Diffusion language models

Diffusion language models have been developed through several distinct formulations. Early approaches introduced diffusion into language generation by applying continuous diffusion over word embeddings or latent representations (Li et al., 2022). In parallel, discrete diffusion models define corruption and denoising processes directly over categorical state spaces, including structured discrete transition kernels, continuous-time Markov formulations, ratio-estimation objectives, and likelihood-based language-modeling objectives (Austin et al., 2021a; Campbell et al., 2022; Lou et al., 2023; Gulrajani & Hashimoto, 2023). More recent masked diffusion language models simplify the discrete diffusion process by using absorbing mask corruption and masked-token denoising objectives, yielding strong likelihood and generation performance with a training objective closely related to masked language modeling (Sahoo et al., 2024; Nie et al., 2025a). Large-scale systems such as LLaDA and Dream further demonstrate that diffusion language models can support instruction following, reasoning, and general-purpose language generation at billion-parameter scale (Nie et al., 2025b; Ye et al., 2025).

Table 3: Comparison with closely related AR-to-DLM conversion and representation-alignment methods. Existing AR-to-DLM methods reuse pretrained AR parameters, but do not explicitly preserve the AR model's hidden-state geometry during denoising training. REPA-style methods use representation alignment, but typically align diffusion models to external encoders rather than to the same AR model being converted.

Method	Starts from AR LM	DLM target	Hidden alignment	Same-architecture AR teacher
Gong et al. (Gong et al., 2025a)	Yes	Yes	No	No
Dream (Ye et al., 2025)	Yes	Yes	No	No
Dream-Coder (Xie et al., 2025)	Yes	Yes	No	No
Efficient-DLM (Fu et al., 2025)	Yes	Yes	No	No
Any-Order GPT (Xue et al., 2025)	Yes	Yes	No	No
REPA (Yu et al., 2025)	No	Yes	Yes	No
Ours	Yes	Yes	Yes	Yes

Our work is orthogonal to the choice of diffusion parameterization. We use a masked denoising objective as the target conversion objective, but we do not propose a new discrete diffusion process, transition kernel, or likelihood estimator. Instead, we study how a pretrained AR model can be converted into a DLM without relearning its internal language representations from scratch. This changes the optimization problem from full diffusion pretraining to representation-preserving mechanism adaptation: the DLM must learn to denoise under bidirectional attention and any-order generation, while remaining anchored to the semantic coordinate system already formed by AR pretraining. The masked-denoising objective used in the main method is given in Section 3.2.

A.2 Autoregressive-to-diffusion language model conversion

The closest line of work studies how pretrained autoregressive language models can be adapted into diffusion language models. Gong et al. propose one of the first systematic AR→DLM adaptation recipes, showing that GPT- and LLaMA-style checkpoints can be converted into DiffuGPT and DiffuLLaMA through continued diffusion training (Gong et al., 2025a). A key ingredient in this conversion is reconciling the next-token indexing convention of AR models with the position-wise denoising convention of masked diffusion, for example through shift operations during training and sampling. Subsequent large-scale DLMs further exploit AR initialization. Dream uses AR-based initialization together with diffusion-specific training strategies for large-scale diffusion language modeling (Ye et al., 2025), while Dream-Coder adapts pretrained AR checkpoints to masked diffusion for code generation (Xie et al., 2025). Efficient-DLM studies the conversion problem through attention and objective design, emphasizing that preserving favorable properties of the AR checkpoint is important for both quality and inference efficiency (Fu et al., 2025). Relatedly, Any-Order GPT connects masked diffusion with any-order autoregressive generation in a decoder-only framework, further highlighting that the distinction between AR and DLMs is partly a distinction between generation mechanisms rather than model families (Xue et al., 2025). Code-oriented diffusion models such as DiffuCoder provide additional evidence that masked diffusion can be competitive in execution-based code-generation settings (Gong et al., 2025b).

These works establish that AR checkpoints are valuable starting points for DLM training. However, existing conversion methods primarily reuse AR parameters and then adapt the model through objective-level, attention-level, masking-level, or sampling-level changes. They do not explicitly constrain the converted DLM to preserve the internal representation geometry of the AR model throughout training. Our method adds this missing constraint. We keep a frozen AR teacher, initialize the DLM student from the same checkpoint, switch the student from causal to bidirectional attention, and align the student's hidden states to the teacher's layer-wise representations during denoising training. Thus, our method does not merely initialize from an AR model; it uses the AR model as a persistent representational anchor. The corresponding conversion procedure in our work is defined in Section 3.1 and evaluated under matched conditions in Section 4.2.

A.3 Any-order generation, iterative decoding, and path planning

Our work is also connected to prior studies of generation order. Autoregressive language models are usually trained with a fixed left-to-right factorization, but this factorization is

only one possible ordering of the joint distribution. XLNet introduced permutation language modeling, showing that autoregressive pretraining can incorporate bidirectional context by optimizing over multiple factorization orders (Yang et al., 2019). More generally, any-order autoregressive models study training and inference when the generation order is allowed to vary rather than being fixed in advance (Shih et al., 2022). These works support the view that generation order is a modeling mechanism rather than an intrinsic property of the data distribution.

Iterative masked decoding provides another precursor to masked diffusion generation. Mask-Predict generates sequences by repeatedly predicting masked positions and remasking low-confidence tokens, offering an early non-autoregressive sequence-generation procedure based on masked language models (Ghazvininejad et al., 2019). MaskGIT later demonstrated the effectiveness of confidence-based iterative masked generation in visual token models (Chang et al., 2022). Modern masked diffusion language models can be viewed as probabilistic successors to these iterative refinement methods: they define an explicit corruption process, train a denoiser over partially masked sequences, and sample by progressively resolving uncertainty across positions.

Recent work further shows that the choice of denoising path and token order can substantially affect DLM sampling and training. P2 studies path planning for masked diffusion sampling and shows that non-uniform remasking and update schedules can improve generation quality without changing the denoising model (Peng et al., 2025). PAPL extends this view to training by incorporating planner-aware path learning into diffusion language model optimization (Peng et al., 2026). In this paper, we use PAPL as part of the default DLM training recipe for both the baseline and representation-aligned models. Therefore, the comparisons isolate the contribution of representation alignment rather than path-planning improvements. Conceptually, this line of work reinforces our central premise: if generation order and denoising paths are mechanisms that can be changed after pretraining, then the key question is whether the underlying language representations can be preserved while the mechanism changes. The role of PAPL in our experimental protocol is specified in Section 4.1 and Appendix B.5; because PAPL is shared across baselines and aligned models, it does not define the comparison axis.

A.4 Representation alignment and feature preservation

Representation alignment has recently emerged as an effective way to accelerate generative model training. REPA aligns the hidden states of diffusion or flow-based transformers to representations from strong pretrained visual encoders, showing that generative training can be bottlenecked by slow representation learning rather than by denoising alone (Yu et al., 2025). Follow-up work studies which components of the target representation are most useful, whether global information or spatial structure matters most, and whether diffusion transformers can provide representation guidance without external encoders (Singh et al., 2025; Wu et al., 2025; Jiang et al., 2025). Together, these methods suggest that explicitly constraining intermediate representations can improve the efficiency and stability of generative training.

Our setting differs in both the teacher and the purpose of alignment. Prior representation-alignment methods typically align a generative model to an external encoder, often in the vision domain. The teacher and student may have different architectures, modalities, or training objectives, so alignment acts as a form of semantic feature distillation. In contrast, our teacher is the exact AR model being converted: it has the same tokenizer, transformer architecture, hidden dimensionality, and pretrained weights as the DLM student before conversion. The student differs only in the generation mechanism induced by bidirectional attention and masked denoising. Thus, alignment is not used to import features from an external model; it is used to preserve the internal coordinate system of the pretrained AR model while learning a new denoising mechanism.

This distinction is central to our interpretation. If AR pretraining has already learned useful semantic and syntactic representations, then AR→DLM conversion should not require relearning those representations from scratch. The remaining problem is to adapt the model so that these representations can support any-order denoising rather than left-to-right next-

token prediction. Layer-wise representation alignment directly implements this view: it anchors the DLM student to the frozen AR teacher while the diffusion loss trains the student to operate under masked bidirectional generation. The resulting method combines the practical benefits of AR initialization with an explicit constraint that preserves the AR model's hidden-state geometry throughout conversion. The same-architecture AR-teacher version of this idea is formalized in Section 3.3 and ablated in Section 4.3.

B Experimental Details

This appendix gives the implementation details for the experimental protocol summarized in Section 4.1 and used in Sections 4.2 and 4.3. Unless otherwise specified, all experiments use the same model initialization, data preprocessing, masked denoising objective, path-planning loss, optimization configuration, and decoding protocol. Ablations vary only the factor explicitly stated.

B.1 Model Conversion

We convert a pretrained autoregressive language model into a masked diffusion language model by changing the attention pattern from causal to bidirectional during masked diffusion training. The transformer architecture is otherwise unchanged: the model remains a decoder-only Qwen-style transformer with rotary position embeddings, RMSNorm, MLP blocks, and a tied input embedding/output language-modeling head. The original AR model is used as a frozen teacher, and the DLM student is initialized from the same checkpoint. This implements the two-model construction in Section 3.1.

In the implementation, the bidirectional attention mode is activated only for masked diffusion training. When the forward pass receives a mask ratio, the model sets is_causal=False; otherwise, it preserves the standard causal mode. Diffusion generation uses full-sequence non-causal forward passes and does not use KV caching.

Listing 1: Causal-to-bidirectional conversion.

```
# In Qwen*ForCausalLM.forward
if mask_ratio is not None:
    is_causal = False
else:
    is_causal = True

outputs = self.model(
    input_ids=input_ids,
    attention_mask=attention_mask,
    is_causal=is_causal,
    output_hidden_states=output_hidden_states,
)
```

Following AR-to-DLM adaptation, we use a one-token shift convention to reconcile next-token prediction with position-wise masked denoising. During training, labels are shifted left and hidden states are sliced accordingly. During sampling, logits are shifted so that diffusion predictions remain consistent with the inherited AR indexing convention. The same shift convention is used for both the baseline and representation-aligned models.

B.2 Data and Preprocessing

Training uses the Nemotron-SFT-Code corpus stored as local Parquet files. The training stream is loaded in streaming mode. Each example is read from the text field, appended with an EOS token, and tokenized with add_special_tokens=False. Examples longer than the maximum sequence length are split into chunks of length 4096, while shorter examples are kept and packed. When packing is enabled, multiple examples are concatenated into packed rows with reset position ids, so that sequence boundaries are preserved. No additional filtering is applied. This preprocessing supports the data regimes described in Section 4.1, including the tiny-data comparison in Section 4.2.

The main assets used in the experiments are licensed or governed as follows: the Qwen3 checkpoints are released under Apache 2.0; the Nemotron-SFT-Code training corpus is

Table 4: Data preprocessing configuration.

Setting	Value
Dataset format	Local Parquet files
Number of files	78
Number of rows	76,447,340
Text field	text
Tokenization	append EOS, add_special_tokens=False
Maximum sequence length	4096
Long examples	split into 4096-token chunks
Short examples	kept and packed
Packing	concatenate examples with reset position ids
Shuffle seed	42
Shuffle buffer	10000
Filtering	none

governed by NVIDIA's data-access terms for model training; HumanEval is released under the original MIT license from OpenAI; MBPP is released under CC BY 4.0 via the Google Research dataset release; and the EvalPlus HumanEval+/MBPP+ releases are Apache 2.0.

B.3 Masked Diffusion Objective

This section implements the main denoising loss in Equation (1). The DLM student is trained with a masked denoising objective. For each original sequence, we sample a mask ratio

$$r \sim \text{Uniform}\left(\frac{1}{500}, 1 - \frac{1}{500}\right). \tag{4}$$

Conditioned on this ratio, each token position is independently masked with probability r. Thus, the implementation uses Bernoulli masking rather than selecting exactly $\lfloor rn \rceil$ masked positions. The cross-entropy loss is computed only on masked positions, while unmasked positions and packed-sequence boundary positions are assigned IGNORE_INDEX.

Listing 2: Masked diffusion corruption.

```
mask_ratio = torch.rand(1).clamp(1 / 500, 1 - 1 / 500)
mask = torch.rand_like(input_ids.float()) < mask_ratio

labels = input_ids.clone()
input_ids[mask] = mask_token_id
labels[input_ids != mask_token_id] = IGNORE_INDEX
```

Let M denote the set of masked and shift-valid positions. The masked denoising loss is

$$\mathcal{L}_{\text{mdm}} = \frac{1}{|M|} \sum_{i \in M} \text{CE}(p_\theta(\cdot \mid \tilde{x})_i, x_i), \tag{5}$$

where $\tilde{x}$ is the corrupted input sequence. In implementation, token losses are summed over valid masked positions and normalized by the number of valid masked tokens.

B.4 Representation Alignment

For representation alignment, we construct a frozen teacher by deep-copying the initialized AR model. The teacher is placed in evaluation mode and all teacher parameters are frozen. During training, the teacher consumes the clean sequence under causal attention, while the student consumes the masked sequence under bidirectional attention. Teacher features are computed under torch.no_grad(). This section implements the alignment loss in Equation (2), including which hidden states and token positions are included.

We align the hidden-state tuple returned by the model, including the embedding hidden state, block outputs, and final post-normalization hidden state. By default, all returned hidden states are aligned. Layer ablations select contiguous thirds of this hidden-state tuple. Alignment is applied only on masked and shift-valid positions, matching the positions that contribute to the denoising loss.

For cosine alignment, hidden states are normalized along the feature dimension and the loss is computed as one minus the mean cosine similarity:

$$\mathcal{L}_{\text{align}} = 1 - \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \frac{1}{|M|} \sum_{i \in M} \left\langle \frac{h_{\text{D},i}}{\|h_{\text{D},i}\|_2}, \frac{\text{sg}(h_{\text{AR},i})}{\|\text{sg}(h_{\text{AR},i})\|_2} \right\rangle, \tag{6}$$

where $\mathcal{H}$ is the selected set of hidden states, M is the masked and shift-valid position set, and $\text{sg}(\cdot)$ denotes stop-gradient.

Listing 3: Layer-wise representation alignment.

```
with torch.no_grad():
    teacher_outputs = teacher(
        input_ids=clean_input_ids,
        is_causal=True,
        output_hidden_states=True,
    )

student_outputs = student(
    input_ids=masked_input_ids,
    is_causal=False,
    output_hidden_states=True,
)

loss_mask = labels != IGNORE_INDEX
repr_loss = cosine_distance(
    student_outputs.hidden_states,
    teacher_outputs.hidden_states,
    loss_mask=loss_mask,
)

loss = mdm_loss + path_loss + lambda_repr * repr_loss
```

The default alignment weight is $\lambda_{\text{repr}} = 10$. Unless otherwise specified, all aligned models use cosine alignment over all selected hidden states.

B.5 Path-Planning Loss

All DLM runs, including both the baseline and representation-aligned models, include the same path-planning auxiliary loss. We therefore treat this loss as part of the default DLM training recipe rather than as a separate method-specific component. This ensures that comparisons isolate the effect of Repr-Align. This term is included in both baseline and aligned models in Section 4.1, so the main comparisons isolate $\mathcal{L}_{\text{align}}$.

For Qwen3-based runs, the path-planning loss reweights the masked-token cross-entropy by the detached model confidence:

$$\mathcal{L}_{\text{path}} = \frac{1}{|M|} \sum_{i \in M} \exp(-\ell_i)_{\text{sg}}\, \ell_i, \tag{7}$$

where ℓ_i is the token-level cross-entropy loss at masked position i, and the confidence weight is stop-gradiented. The total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{mdm}} + \mathcal{L}_{\text{path}} + \lambda_{\text{repr}} \mathcal{L}_{\text{align}}. \tag{8}$$

For baseline DLM conversion, the representation-alignment term is omitted:

$$\mathcal{L}_{\text{baseline}} = \mathcal{L}_{\text{mdm}} + \mathcal{L}_{\text{path}}. \tag{9}$$

B.6 Optimization

All models are optimized with AdamW. Unless otherwise specified, we use the same optimization configuration across baseline and representation-aligned runs. This optimization protocol is the fixed-budget setting used in Section 4.2.

B.7 Decoding and Evaluation

We evaluate code generation on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), and the HumanEval+/MBPP+ variants from EvalPlus (Liu et al., 2023). HumanEval uses the canonical prompt from the benchmark. MBPP uses the task description

Table 5: Default optimization configuration.

Setting	Value
Optimizer	AdamW
Learning rate	3×10^{-4}
Adam betas	$(0.9, 0.95)$
Adam epsilon	10^{-8}
Weight decay	0.01
Parameter groups	single group for all trainable parameters
LR schedule	cosine decay
Warmup ratio	0.001
Minimum learning rate	3×10^{-6}
Gradient clipping	1.0
Precision	bfloat16
Maximum sequence length	4096
Per-device batch size	6
Gradient accumulation	1
Global batch size	96 sequences
Number of GPUs	16
Parallelism	DDP
Attention backend	FlashAttention-2
Gradient checkpointing	disabled
FSDP / DeepSpeed	disabled

Table 6: Default decoding and evaluation configuration.

Setting	Value
Benchmarks	HumanEval, HumanEval+, MBPP, MBPP+
Prompting	zero-shot
Samples per problem	10
Metric	execution-based pass@1/pass@10
Sampler	P2-self, alg=p2
Sampling steps	128
Maximum new tokens	128
Temperature	0.8
Top-k	200
Algorithm temperature	0.5

together with the first three tests and ends with a Python code block. All evaluations are zero-shot. This evaluation protocol underlies the main comparisons in Section 4.2 and the ablations in Section 4.3.

For each problem, we generate 10 samples and compute execution-based pass@1 and pass@10 using the standard code-evaluation pipeline. The same decoding configuration is used for all methods being compared.

The P2 sampler keeps prompt tokens fixed and iteratively remasks low-confidence variable positions. At each step, the model predicts the full sequence, samples variable positions, and then remasks a fraction of the lowest-confidence generated tokens according to the sampling schedule.

Listing 4: Evaluation command template.

```
accelerate launch --num_processes 4 eval.py \
  --model custom_coder \
  --model_args "pretrained=<checkpoint>,max_new_tokens=128,steps=128,temperature=0.8,alg=p2" \
  --tasks humaneval --num_fewshot 0 --batch_size 10 \
  --output_path evals_results/humaneval-ns0 --log_samples \
  --confirm_run_unsafe_code
```

B.8 Ablation Protocols

All ablations use the same AR→DLM conversion pipeline unless otherwise stated. In particular, the model is initialized from the same AR checkpoint, trained with the same masked denoising objective and path-planning loss, and evaluated with the same decoding configuration. Each ablation varies only the listed factor. These protocols correspond to the ablation

Table 7: Ablation protocols.

Ablation	Variants	Changed factor
Alignment metric	L2, cosine	representation loss
Alignment weight	$1, 5, 10, 20$	λ_{repr}
Aligned layers	lower, middle, upper, all	selected hidden states
Freezing	none, embeddings/LM head/MLP	trainable parameter blocks
Data size	full, reduced	training stream size

Table 8: Freezing configuration.

Variant	Frozen parameters	Trainable parameters
No freezing	none	all parameters
Embedding/MLP freezing	embed_tokens, lm_head, mlp	attention and normalization layers

results in Section 4.3.

For the alignment-metric ablation, L2 alignment uses mean-squared error between student and teacher hidden states, while cosine alignment normalizes hidden states along the feature dimension and matches their directions. For the layer ablation, we partition the hidden-state tuple into three contiguous groups and apply the representation loss only to one group at a time. The default setting aligns all hidden states with cosine distance and $\lambda_{\text{repr}} = 10$.

B.9 Freezing Protocol

To test whether AR→DLM conversion requires updating the full network, we freeze selected parameter blocks during representation-aligned training. Freezing is implemented by case-insensitive substring matching on parameter names. In the main freezing variant, parameters whose names contain embed_tokens, lm_head, or mlp are frozen. Attention layers and normalization layers remain trainable. This protocol supports the efficiency result in Section 4.2.

This freezing protocol is designed to preserve most of the pretrained representational and feed-forward computation while allowing the attention mechanism and normalization statistics to adapt to bidirectional denoising.

C Limitations

Our study is limited to same-architecture AR→DLM conversion on Qwen3 decoder-only checkpoints and code-generation benchmarks. The gains we report may not transfer unchanged to other model families, modalities, or downstream tasks, and the method still depends on access to a strong pretrained AR teacher and substantial training compute.