DREAMON: DIFFUSION LANGUAGE MODELS FOR CODE INFILLING BEYOND FIXED-SIZE CANVAS

Zirui Wu[1,4*], **Lin Zheng**[1*], **Zhihui Xie**[1], **Jiacheng Ye**[1], **Jiahui Gao**[1], **Shansan Gong**[1], **Yansong Feng**[4], **Zhenguo Li**[3], **Wei Bi**[2], **Guorui Zhou**[2], **Lingpeng Kong**[1†]
[1]The University of Hong Kong [2]Kuaishou Technology [3]Huawei Noah Ark Lab
[4]Peking University
ziruiwu@pku.edu.cn, lzheng2@cs.hku.hk, lpk@cs.hku.hk

ABSTRACT

Diffusion Language Models (DLMs) present a compelling alternative to autoregressive models, offering flexible, any-order infilling without specialized prompting design. However, their practical utility is blocked by a critical limitation: the requirement of a fixed-length masked sequence for generation. This constraint severely degrades code infilling performance when the predefined mask size mismatches the ideal completion length. To address this, we propose DREAMON, a novel diffusion framework that enables dynamic, variable-length generation. DREAMON augments the diffusion process with two length control states, allowing the model to autonomously expand or contract the output length based solely on its own predictions. We integrate this mechanism into existing DLMs with minimal modifications to the training objective and no architectural changes. Built upon Dream-Coder-7B and DiffuCoder-7B, DREAMON achieves infilling performance on par with state-of-the-art autoregressive models on HumanEval-Infilling and SantaCoder-FIM and matches oracle performance achieved with ground-truth length. Our work removes a fundamental barrier to the practical deployment of DLMs, significantly advancing their flexibility and applicability for variable-length generation. Our code is available at https://github.com/DreamLM/DreamOn.

1 INTRODUCTION

In recent years, autoregressive language models have achieved remarkable progress (Comanici et al., 2025; OpenAI, 2025; Guo et al., 2025; Qwen et al., 2025). They model language as generating text sequentially in a fixed left-to-right manner. While dominant, this paradigm is now being complemented by Diffusion Language Models (DLMs) (Hoogeboom et al., 2021; Austin et al., 2021; Zheng et al., 2023; Singh et al., 2023; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025; Ye et al., 2025; DeepMind, 2025; Labs et al., 2025), which have emerged as a promising alternative and are gaining significant attention.

DLMs operate through a multi-step denoising process, progressively refining a masked sequence to enable flexible, any-order generation (Austin et al., 2021; Hoogeboom et al., 2021). This property makes them inherently suited for infilling tasks—generating content to fill between a given prefix and suffix (Bavarian et al., 2022; Fried et al., 2023; Allal et al., 2023). In contrast, autoregressive models must resort to cumbersome workarounds for infilling, such as permuting the target span to the end of the sequence (Fried et al., 2023; Guo et al., 2024a; Hui et al., 2024; Seed et al., 2025). Such methods not only disrupt the natural contextual structure but also necessitate specialized prompting during training and infenence.

Despite the theoretical advantage, the practical application of DLMs is hindered by a critical bottleneck: the reliance on a pre-specified, fixed-length mask. Current DLMs (Ye et al., 2025; Nie et al., 2025; Xie et al., 2025; Gong et al., 2025b) require the input and output sequences to have

[*]Equal contribution.
[†]Corresponding author.

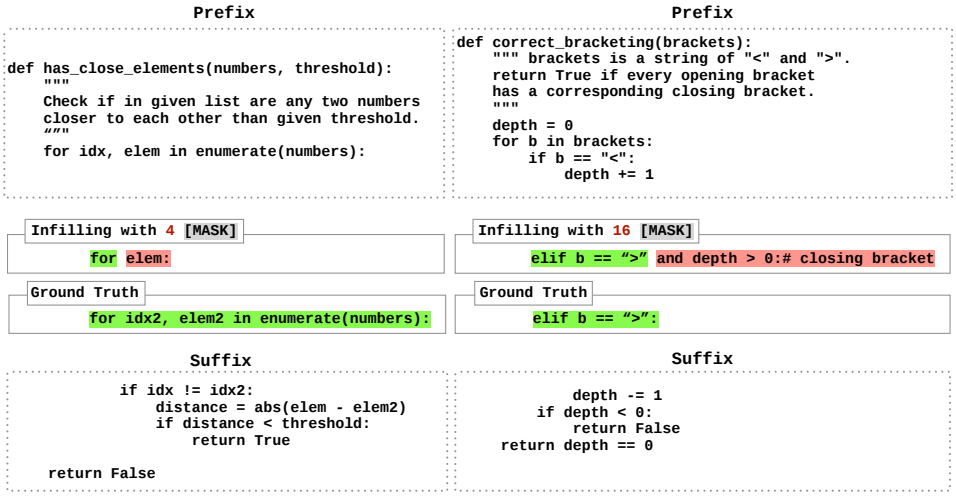


Figure 1: Example of DreamCoder-7B failing at code infilling due to the length mismatch between masked input and ground truth. Incorrect and correct code is marked in red and green. With too few masked tokens, diffusion models lack sufficient room for meaningful code infilling. Too many masks cause over-generation of unnecessary code snippet (e.g., `depth > 0` that is incorrect).

identical lengths, which prevents them from dynamically determining the length of the output. This limitation is especially damaging for code infilling, where solution lengths can vary significantly across examples. As shown in Figure 1, Dream-Coder-7B (Xie et al., 2025) produces incomplete or over-generated code when the mask length does not align with the ground truth. More critically, we observe an average performance drop of 38% on HumanEval-Infilling (Bavarian et al., 2022) when the predefined mask length does not align with the ground truth length (Table 2), highlighting the extreme sensitivity of current DLMs to this hyperparameter.

To address this bottleneck, we propose DREAMON, a discrete diffusion language modeling framework equipped with adaptive length adjustment (§3.1). DREAMON introduces dynamic length adaptation through two dedicated special tokens, `[expand]` and `[delete]`, requiring no architectural modifications. We augment the standard diffusion training process with auxiliary length-control states, allowing DREAMON to be trained with minimal deviation from conventional DLM objectives (§3.2). During inference, the model adaptively expands and contract the masked sequence solely on its predictions without external guidance (§3.3). Based on Dream-Coder-7B (Xie et al., 2025) and DiffuCoder-7B (Gong et al., 2025b), DREAMON achieves competitive infilling performance with state-of-the-art autoregressive models on HumanEval-Infilling (Bavarian et al., 2022) and SantaCoder-FIM (Allal et al., 2023) (§4), and approaches oracle-level performance achieved with ground truth length (§5.1).

- We alleviate the fixed-length bottleneck of diffusion language models (DLMs) by introducing DREAMON, which allows the model to dynamically expand or contract sequences during generation without any architectural changes.
- Our method achieves variable-length generation with two special states `[expand]` and `[delete]`, and supports scalable end-to-end learning of length adaptation through simple augmentation techniques with minimal deviation from standard diffusion objectives.
- On multiple infilling benchmarks, DREAMON delivers an average absolute performance boost of **26.4%** over diffusion baselines, matches the performance achieved with oracle length, and brings diffusion models close to or on par with state-of-the-art autoregressive models.

2 PRELIMINARY

Let $\mathbf{x}_0 = [\mathbf{x}_0^1, \ldots, \mathbf{x}_0^N]$ be a sequence of N discrete tokens sampled from the data distribution $q(\mathbf{x})$. Each token takes values from a vocabulary of size $V + 1$, consisting of V regular symbols plus an additional absorbing state `[mask]`. We represent each token $\mathbf{x}_{0,n}$, as well as the absorbing

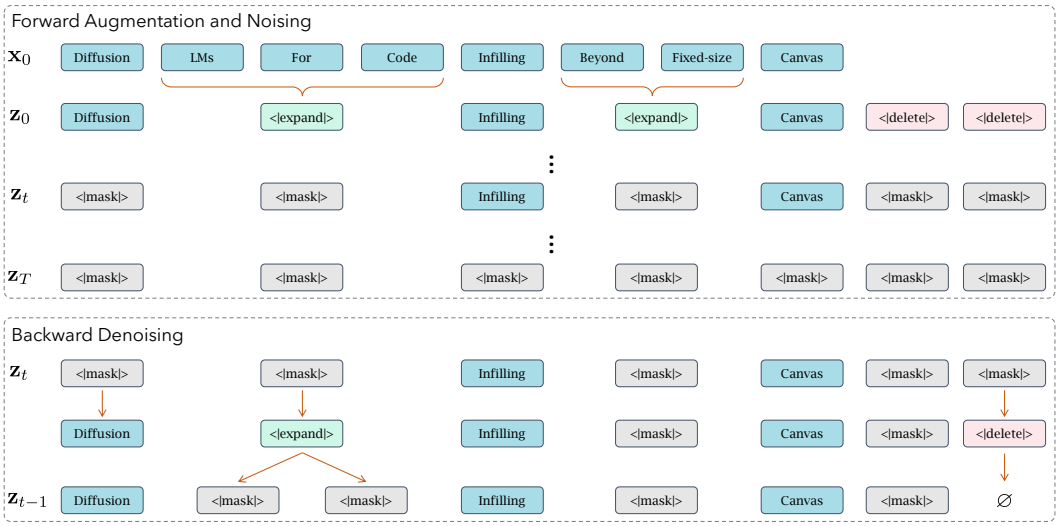


Figure 2: Overview of the augmented diffusion process. **Top**: the forward augmentation-and-noising procedure maps the input sequence $\mathbf{x}_0$ to an augmented latent $\mathbf{z}_0$ containing `[expand]` and `[delete]` states, and then applies a standard masked diffusion process over $\mathbf{z}_0$ to obtain $\mathbf{z}_t$ and eventually $\mathbf{z}_T$. **Bottom**: a single denoising step where `[mask]` positions in $\mathbf{z}_t$ can be predicted as either regular tokens or special states; `[expand]` deterministically expands into two `[mask]` tokens, while `[delete]` will remove the corresponding position, yielding a new sequence $\mathbf{z}_{t-1}$ with a different length from $\mathbf{z}_t$.

state `[mask]`, as one-hot vectors in $\{0,1\}^{V+1}$. Typical discrete-time masked diffusion models are defined as a class of latent variable models over such sequences with a forward and backward transition process. In the forward process q, each token is preserved with a certain probability or replaced by `[mask]` otherwise, giving $q(\mathbf{x}_t \mid \mathbf{x}_0) = \alpha_t \mathbf{x}_0 + (1-\alpha_t)\texttt{[mask]}$ with a predefined schedule α_t. As t increases, the schedule is designed such that the sequence converges to full mask tokens. The generative model reverses this process, starting from $\mathbf{x}_T$ and applying parameterized transitions $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ that approximate the true posterior $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$. This yields $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.

This class of generative models can be generalized to **continuous-time** parameterization by considering $t \in [0,1]$, which avoids the bias introduced by predefined discretization over time steps. We adopt the frameworks in Kingma et al. (2021); Campbell et al. (2022); Sahoo et al. (2024); Shi et al. (2024); Ou et al. (2025) and train p_θ with a weighted cross-entropy loss objective,

$$\mathcal{L}(\theta) = -\mathbb{E}_{\substack{\mathbf{x}_0 \sim q(\mathbf{x}) \\ t \sim \mathcal{U}(0,1) \\ \mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)}} \left[w(t) \sum_{n=1}^{N} \mathbf{1}_{[\mathbf{x}_t^n = \texttt{[mask]}]} \log p_\theta(\mathbf{x}_0^n \mid \mathbf{x}_t) \right], \tag{1}$$

where the indicator $\mathbf{1}_{[\mathbf{x}_t^n = \texttt{[mask]}]}$ implies the loss is only evaluated on masked positions, and $w(t) \in (0,1]$ is a time-dependent weighting term derived from the noise schedule α_t (Shi et al., 2024; Gong et al., 2025a). This objective provides a tractable variational upper bound on the negative log-likelihood and serves as an effective training target for large-scale diffusion language models.

3 METHOD

In this section, we present our formulation for extending masked diffusion models beyond fixed-length generation. We begin with an overview of our framework in §3.1, followed by training and inference procedures in sections 3.2 and 3.3, and practical implementation details in §3.4.

Algorithm 1 DREAMON Training

Require: Model parameters p_θ, merge rate scheduler $\mathcal{S}$;
1: **repeat**
2: Sample original data $\mathbf{x}_0 \sim q(\mathbf{x})$ and a time step $t \sim \mathcal{U}(0, 1)$;
3: Construct augmented sequence $\mathbf{z}_0$ from $\mathbf{x}_0$ with `[expand]` and `[delete]` with $\mathcal{S}$;
4: Sample masked sequence $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{z}_0)$;
5: Compute weighted loss $\mathcal{L}(\theta)$ via eq. (2);
6: Update parameters $\boldsymbol{\theta}$ via $\nabla_{\boldsymbol{\theta}}\mathcal{L}$;
7: **until** convergence

3.1 MASKED DIFFUSION WITH AUGMENTED STATES FOR LENGTH CONTROL

The key ingredient of DREAMON is to introduce two new special states `[expand]` and `[delete]` during the diffusion process. When a token transitions to the state `[expand]`, we will expand it into two `[mask]` tokens at the same position of the sequence; and whenever a `[delete]` state is yielded, the token is removed from the sequence. With proper predictions of these special states, the model acquires native length control.

Simulating Special Transitions via Data Augmentation. To train the model to predict `[expand]` and `[delete]`, we introduce an auxiliary augmented sequence $\mathbf{z}_0$ constructed from the original input $\mathbf{x}_0$. The augmentation merges random token spans into `[expand]` and inserts `[delete]` into the sequence. For merging, we first sample a time step $t \sim \text{Uniform}(0, 1)$ and compute a set of mask indices $\mathcal{M}_t$ according to the schedule α_t. Rather than masking tokens, we use $\mathcal{M}_t$ to gate merging such that only spans of consecutive mask indices in $\mathcal{M}_t$ will be replaced with `[expand]`, under a merging probability controlled by rate schedulers (§3.4). This pseudo-masking process provides finer control over the ratio of special to regular tokens, producing $\mathbf{z}_0$ with variable length and a balanced mix of regular and special states.

Masked Diffusion over Augmented $\mathbf{z}_0$. We impose the masked diffusion process $p_{\boldsymbol{\theta}}$ on $\mathbf{z}_0$. For these special states `[expand]` and `[delete]`, the forward diffusion process always maps them to `[mask]`, ensuring that all such tokens are masked and contribute to the learning signal. By construction, the prediction targets in the masked diffusion loss naturally include `[expand]` and `[delete]`. Consequently, the model is now trained to denoise not only regular tokens but also special sentinels, thereby learning length control behavior and enabling variable-length generation without any architecture changes.

3.2 TRAINING

Similarly to the masking state `[mask]` and any regular tokens, we treat `[expand]` and `[delete]` as sentinel tokens in the vocabulary, and the model is trained to predict them using the objective in Eq. (1). During training, however, we observe an imbalance: many `[mask]` positions correspond to `[delete]` targets, while far fewer correspond to `[expand]`, since each `[delete]` is transformed into a single `[mask]`, whereas multiple `[mask]` tokens are merged into one `[expand]`. As a result, `[delete]` tokens contribute disproportionately to the loss. To calibrate this, we introduce a loss weighting scheme that downscales the contribution of `[delete]` predictions so that their total weight is equivalent to that of a single `[mask]` prediction. The weighted training loss is then given by

$$\mathcal{L}(\theta) = -\mathbb{E}_{\substack{\mathbf{x}_0 \sim q(\mathbf{x}) \\ t \sim \mathcal{U}(0,1) \\ \mathbf{z_0} \sim q(\mathbf{z_0}|\mathbf{x_0}) \\ \mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{z}_0)}} \left[w(t) \sum_{n=1}^{N} \mathbf{1}_{[\mathbf{z}_t^n = \texttt{[mask]}]} \cdot w_n \cdot \log p_\theta(\mathbf{z}_0^n \mid \mathbf{z}_t) \right], \tag{2}$$

with the per-token weight w_n defined as

$$w_n = \frac{\mathcal{N}_{mask}}{\mathcal{N}_{mask} - \mathcal{N}_{delete} + 1} \times \begin{cases} 1, & \text{if } \mathbf{z}_0^n \neq \texttt{[delete]}, \\ \dfrac{1}{\mathcal{N}_{delete}}, & \text{if } \mathbf{z}_0^n = \texttt{[delete]}, \end{cases} \tag{3}$$

where $\mathcal{N}_{mask}$ and $\mathcal{N}_{delete}$ denote the number of `[mask]` and `[delete]` tokens in the sequence, respectively. The normalization factor ensures that the expected loss magnitude remains consistent across sequences with varying numbers of deletions.

3.3 INFERENCE

Our inference procedure, outlined in Algorithm 2, builds upon the standard masked diffusion denoising framework with key modifications to support variable-length generation. At each diffusion step, we simultaneously predict all masked positions and then selectively re-mask tokens based on prediction entropy, following Ye et al. (2025). However, the prior work employs fixed masking schedulers to determine how many tokens to unmask per step, which are ill-suited for dynamic-length modeling since they assume a pre-specified output length. Instead, we directly control the denoising trajectory by specifying n, the number of mask tokens to denoise at each step, enabling adaptive sequence length modeling. During denoising, predicted `[expand]` tokens are immediately expanded into two `[mask]` tokens, while generated `[delete]` tokens are removed from the sequence. To ensure stability and prevent unbounded growth, we enforce a maximum output sequence length L_{max}. The generation process terminates once all `[mask]` positions have been resolved.

Algorithm 2 Variable-Length Generation with DREAMON

Require: Trained model parameters $\boldsymbol{\theta}$, initial sequence length L, maximum length L_{max}, unmasking budget n per iteration, and sampling temperature τ;

for $l = 1, 2, \ldots, L$ **do**
  Initialize $\mathbf{z}^l \leftarrow$ `[mask]`;
end for
while `[mask]` in $\mathbf{z}$ **do**
  Compute token probabilities $p \leftarrow p_{\boldsymbol{\theta}}(\cdot \mid \mathbf{z})$;
  if $|\mathbf{z}| \geq L_{\text{max}}$ **then**
    Set the probability of `[expand]` to 0 and renormalize;
  end if
  Select up to n masked positions with highest confidence;
  for each selected position i **do**
    Draw $\widetilde{\mathbf{z}}^i \sim \text{Categorical}(p^i/\tau)$;
    if $\widetilde{\mathbf{z}}^i =$ `[expand]` **then**
      Replace $\mathbf{z}[i]$ with [`[mask]`, `[mask]`];
    else if $\widetilde{\mathbf{z}}^i =$ `[delete]` **then**
      Remove $\mathbf{z}[i]$ from the sequence;
    else
      Set $\mathbf{z}[i] \leftarrow \widetilde{\mathbf{z}}^i$;
    end if
    Update position indices if length has changed;
  end for
end while
Return $\mathbf{z}$.

3.4 PRACTICAL IMPLEMENTATIONS

Span Merging Schedulers. We design two empirical mask merging schedulers. (1) **Static scheduler**: merges adjacent `[mask]` tokens with a fixed probability p_{merge} and (2) **Dynamic inverse scheduler**: sets the merging probability inversely proportional to the number of `[mask]` tokens in the sequence. This scheduler merges less with more `[mask]` tokens to avoid merging too many tokens that might potentially influence the original performance of the base model. We find that a mixture of two schedulers during training yield the best performance as detailed in §5.2.

Broadcasting Deletion as Length Predictor. In practice, we observe that performance degrades slightly when there is a large discrepancy between the initial masked span and the true target length. This introduces inefficiency during inference, as the model must expend numerous forward passes

to adjust the sequence length via incremental expansions or contractions. To mitigate this, we introduce a training-free inference-time adaptation method that accelerates convergence. Specifically, whenever the model predicts a `[delete]`, we eliminate all subsequent tokens to its right if they are all `[mask]` tokens. This mechanism significantly reduces unnecessary computation and improves inference efficiency without sacrificing generation quality.

4 EXPERIMENTS

4.1 SETUP

We fine-tune Dream-7B (Ye et al., 2025), DiffuCoder-7B (Gong et al., 2025b), and DreamCoder-7B (Xie et al., 2025) on the education-instruction subset of OpenCoder SFT data (Huang et al., 2024), which contains about 110K Python instruction-solution pairs synthesized from high-quality educational data. Our experiments focus on code infilling, where the goal is to generate missing spans conditioned on surrounding prefix and suffix contexts. During training, we randomly split each solution into three segments: prefix, middle, and suffix. The instruction, prefix, and suffix are fixed as context, while diffusion is applied only to the middle segment.

For sequence contraction, we find it sufficient to append a random number of `[delete]` tokens (from 0 to 64) to the end of the middle segment during training. For sequence expansion, `[expand]` tokens are constructed with merging probability p_{merge} as 0.5, using a 1:1 mix of static and dynamic inverse schedulers. Models are trained for 10 epochs with batch size 128, maximum context length 1024, and learning rate $1\text{e}{-5}$ under a cosine decay schedule with 10% warmup steps. It takes approximately 5 hours to train with 8 H800 GPUs. The compute of DreamOn is only 0.15% compared with the compute for pretraining a base model (Ye et al., 2025). During inference we set temperature as 0.2 and top_p as 0.9. To prevent excessive growth, we cap mask expansion in DREAMON at $L_{max} = 128$. We also disable mask expansion in inference after expanding L_{max} times.

4.2 EVALUATION

Baselines. We compare against state-of-the-art *autoregressive* models pretrained with infilling objectives, specifically Deepseek-Coder-6.7B (Guo et al., 2024b), Qwen2.5-Coder-7B (Hui et al., 2024) and Seed-Coder-8B (Seed et al., 2025). For open-source *diffusion* language model baselines of similar scale, we evaluate LLaDA-8B (Nie et al., 2025), Dream-7B (Ye et al., 2025), DiffuCoder-7B (Gong et al., 2025b), and DreamCoder-7B (Xie et al., 2025).

Benchmarks. We evaluate models on HumanEval-Infilling (Bavarian et al., 2022) benchmarks, including single-line and multi-line subsets, and the Python subset of Santacoder-FIM (Allal et al., 2023). We use the official evaluation scripts to report pass@1 for HumanEval-Infilling and exact match for Santacoder-FIM. We evaluate autoregressive language models using their respective infilling templates used during pretraining. For all diffusion models, we set the mask length to 64 by default. DREAMON variants dynamically adjust this length as detailed in §3.4.

4.3 RESULTS

Table 1 shows that baseline diffusion models struggle with code infilling due to their fixed-length generation, lagging significantly behind autoregressive models. DREAMON effectively resolves this limitation, yielding an average absolute improvement of 26.4% over diffusion baselines and highlighting its effectiveness as a *model-agnostic* enhancement.

Notably, with DREAMON, DiffuCoder-7B and DreamCoder-7B not only match the performance of leading autoregressive models like Qwen2.5-Coder-7B, but also surpasses them in the more challenging multi-line infilling benchmark. This demonstrates that equipping diffusion models with our length-adaptive mechanism makes them highly competitive for infilling tasks.

Table 1: Pass@1 on HumanEval-Infilling and exact match on Santacoder-FIM, comparing open-source auto-regressive and diffusion model baselines.The best results across diffusion models are shown in bold, and the second best are underlined.

Models	HumanEval-Infilling (Pass@1)		SantaCoder (EM)
	Single-line	Multi-line	
Open-Weights AR Models			
Deepseek-Coder-6.7B	73.0	45.7	76.3
Seed-Coder-8B	89.7	59.3	77.2
Qwen2.5-Coder-7B	92.6	58.7	79.8
Open-Weights Diffusion Models			
LLaDA-8B	48.3	21.1	35.1
Dream-7B	48.2	21.9	60.3
+ DREAMON	88.6$_{+40.4}$	53.3$_{+31.4}$	73.8$_{+13.5}$
DiffuCoder-7B	53.7	45.0	58.0
+ DREAMON	**92.2**$_{+38.5}$	<u>63.1</u>$_{+18.1}$	<u>77.4</u>$_{+19.4}$
DreamCoder-7B	55.5	43.2	59.3
+ DREAMON	<u>92.1</u>$_{+36.6}$	**63.8**$_{+20.6}$	**79.0**$_{+19.7}$

Table 2: Infilling performance across different designs for diffusion language models. *Oracle*: performance with the oracle target length for reference. †: We use an AST parser to compute exact match to normalize huge syntactic differences between the model output and the ground truth.

Models	Initial Mask Length					Avg.	Oracle
	4	8	16	32	64		
Single-Line (Pass@1)							
Dream-Coder-7B	24.9	61.2	72.6	62.4	55.5	55.3	93.3
+ DREAMON	**88.7**	**90.6**	**91.0**	**91.6**	**92.1**	**90.8**	91.6
w/o Delete	87.8	77.9	71.2	62.3	37.8	67.4	93.3
w/o Expand	25.1	71.6	88.0	90.9	91.5	73.4	92.5
Multi-line (Pass@1)							
Dream-Coder-7B	5.5	14.7	27.1	39.4	43.2	26.0	69.0
+ DREAMON	**50.2**	**53.8**	**56.9**	**60.9**	**63.8**	**57.1**	66.6
w/o Delete	44.6	45.3	46.1	46.7	44.7	45.5	67.9
w/o Expand	5.5	16.5	30.7	48.2	61.3	32.4	63.2
SantaCoder-FIM (EM)							
Dream-Coder-7B	20.0	26.6	43.5	50.8	59.3	40.0	76.3†
+ DREAMON	**75.0**	**76.8**	**78.4**	**78.0**	**79.0**	**77.4**	82.0
w/o Delete	74.2	44.3	40.2	50.0	56.2	53.0	84.2
w/o Expand	22.5	55.0	74.7	77.8	78.0	61.6	78.6†

5 ANALYSIS

In this section, we conduct ablation studies to evaluate the effectiveness of DREAMON. All variants are fine-tuned from DreamCoder-7B and evaluated with initial mask lengths ranging from 4 to 64. We additionally evaluate the infilling results under an *oracle* setting, where the initial mask length matches the ground-truth solution length, providing an approximate upper bound for infilling performance of diffusion language models.

5.1 PERFORMANCE WITH DIFFERENT MASK LENGTHS

Performance Breakdown. As shown in Table 2, DreamCoder-7B without finetuning suffers significant performance degradation when using fixed mask lengths compared to the oracle-length performance, highlighting the strong dependence of infilling quality on accurate mask length. By contrast, DREAMON achieves near oracle-level performance across a wide range of initial mask lengths. Importantly, the performance gains stem from the combined use of both mask expansion and contraction mechanisms. DREAMON maintains stable performance on both single-line infilling and SantaCoder-FIM tasks regardless of the initial mask length. We provide two denoising trajectory examples in Appendix D.

Ablation on Length Control. To isolate the contributions of expansion and deletion mechanisms, we evaluate ablated variants of DREAMON: (1) **w/o Expand**, disabling mask expansion; and (2) **w/o Delete**, disabling mask deletion. Removing deletion leads to a sharp performance drop on longer mask lengths, as the model tends to over-generate and fill all given `[mask]` tokens. On the other hand, removing expansion severely harms performance on short lengths, as the model cannot dynamically extend mask sequences to accommodate more complex or longer completions. We also observe a slight performance decline on long masks without expansion, suggesting that even for longer masked inputs, expansion remains beneficial by allowing fine-grained length adjustments.

5.2 EXPANSION MECHANISM DESIGN

Mask merging strongly affects the number of `[expand]` tokens and the gap between initial and target sequence lengths. We study this in-depth and evaluate two merge rate schedulers: a static scheduler with fixed merge probability and a dynamic inverse scheduler with merge probability inversely proportional to the number of `[mask]` tokens. Using only the static scheduler enables effective expansion, achieving an 88.9% pass rate for length-4 masks. However, its performance is limited on longer masks.

The dynamic inverse scheduler merges less when more masks are present. It achieves higher performance on longer masks but struggles with large expansions, dropping to 82.5% on length-4 masks. We find a 1:1 mixture achieves the best overall results, offering a favorable balance across various mask lengths (Appendix C).

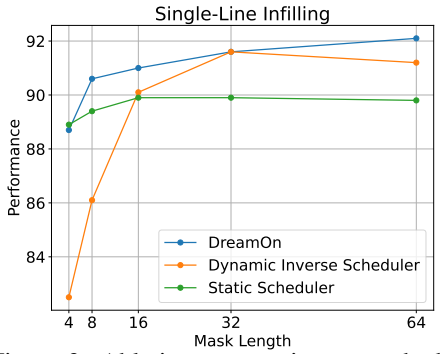


Figure 3: Ablation on merging rate scheduler design choices.

5.3 DELETION MECHANISM DESIGN

We ablate our design choices for handling `[delete]` tokens with the following experiments: (1) **w/o Loss Balancing**: train the model without down-weighing the loss on `[delete]` tokens, treating them equally with other tokens in the loss computation; (2) **w/o In-place Deletion**: Instead of removing `[delete]` tokens, keep them in the sequence, similar to generating padding placeholder tokens in standard diffusion language (Nie et al., 2025; Ye et al., 2025). To implement this, we randomly mask or preserve `[delete]` during training; and (3) **w/o Deletion Broadcasting**: disable the inference-time mechanism described in §3.4.

As shown in Table 2, removing loss balancing leads to a substantial performance drop to 84.6% average pass@1 rate, confirming that down-weighing `[delete]` loss is essential to prevent the model from overfitting to deletion signals. Keeping persistent `[delete]` tokens also performs poorly (average 85.3%), indicating that placeholder-like deletion tokens in the sequence disrupt positional coherence and degrade training. Disabling deletion broadcasting reduces performance by 0.6% on average, especially when the given mask length is much longer than the expected solution. The deletion broadcasting mechanism also accelerates generation by 2.1×.

Table 3: Ablation study for mask deletion mechanism implementations.

Models	Initial Mask Length					Avg.	Oracle
	4	8	16	32	64		
HumanEval-Infilling Single-Line (Pass@1)							
DREAMON	**88.7**	**90.6**	**91.0**	**91.6**	**92.1**	**90.8**	91.6
w/o Loss Balancing	75.8	82.5	87.0	87.2	90.4	84.6	88.6
w/o In-Place Deletion	85.9	85.7	88.5	84.8	78.0	84.6	93.1
w/o Deletion Broadcasting	**88.7**	90.5	90.0	90.2	91.4	90.2	91.6

5.4 EFFICIENCY ANALYSIS OF DELETION BROADCASTING

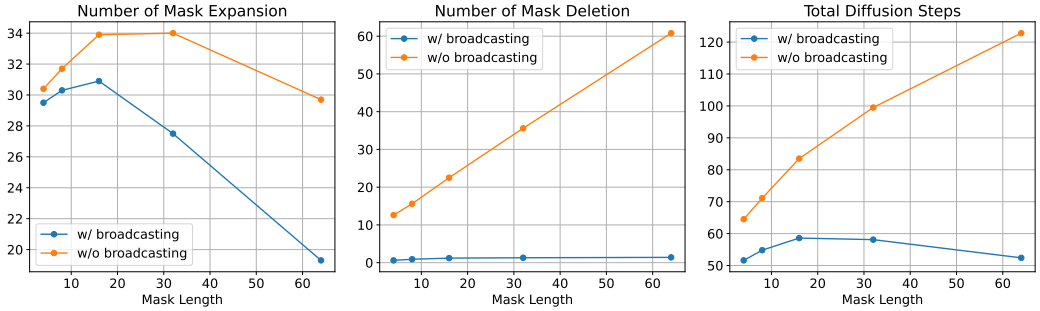


Figure 4: Average generation steps of DreamCoder + DREAMON on multi-line infilling subset.

The introduction of broadcasting dramatically enhances inference efficiency, primarily by transforming the deletion process from a token-by-token operation into a length prediction action. Without broadcasting, deletion steps scale almost linearly with the initial mask length because the model must iteratively predict and remove each excess mask token individually. In contrast, DREAMON with deletion broadcasting mechanism keep the number of mask deletion nearly constant around with roughly only 1 step on average. This optimization eliminates the computational bottleneck caused by large discrepancies between the initial masked span and the true target length, reducing total inference steps from as high as 122.8 (w/o broadcasting) to just 52.4 (w/ broadcasting) at mask length 64. Consequently, broadcasting not only slashes unnecessary forward passes but also stabilizes overall inference cost, making the generation process both faster and more robust to initialization variance without any impact on output quality.

6 RELATED WORK

Code Infilling with Autoregressive Models. Code infilling requires generating missing code segments conditioned on bidirectional context, a task inherently challenging for standard left-to-right autoregressive models. To address this, several approaches adapt architectures to better capture bidirectional dependencies Yang et al. (2019); Stern et al. (2019); Gu et al. (2019a); Chan et al. (2019); Welleck et al. (2019); Shen et al. (2020); Alon et al. (2020); Nguyen et al. (2023); Shen et al. (2023).

A widely adopted alternative preserves the standard left-to-right autoregressive architecture by relocating the target infill segment to the end of the input sequence, enabling the model to generate the missing code autoregressively (Raffel et al., 2020; Tay et al., 2023; 2022; Anil et al., 2023; Sun et al., 2024). This approach is compatible with decoder-only architectures (Bavarian et al., 2022) and has become the standard in modern code language models, including Codex (OpenAI et al., 2022), INCODER (Fried et al., 2023), SANTACODER (Allal et al., 2023), StarCoder (Li et al., 2023) and StarCoder 2 (Lozhkov et al., 2024), CODEGEN 2/2.5 (Nijkamp et al., 2023), Code-Llama (Roziere et al., 2023), DeepSeek-Coder (Guo et al., 2024a), CodeGemma (CodeGemma Team, 2024), Qwen-Coder (Bai et al., 2023; Hui et al., 2024), and Seed-Coder (Seed et al., 2025).

Discrete Diffusion Language Models. Discrete diffusion models have recently emerged as a compelling alternative to autoregressive models. Foundational work by Austin et al. (2021); Hoogeboom et al. (2021) introduced discrete diffusion processes for text data, enabling probabilistic modeling of token sequences through iterative and bidirectional denoising. Subsequent research has refined these approaches with continuous-time relaxations (Campbell et al., 2022), improved training objectives (Zheng et al., 2023; Lou et al., 2024), and generalized masked diffusion frameworks (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025). Scaling efforts have produced powerful models such as Plaid (Gulrajani & Hashimoto, 2023) and LLaDA (Nie et al., 2025). Adaptation techniques leveraging pretrained models, such as DiffuLLaMA (Gong et al., 2025a) and Dream (Ye et al., 2025), have narrowed the performance gap with state-of-the-art autoregressive language models.

Non-autoregressive Models with Length Control. Generating variable-length sequences remains a significant challenge for non-autoregressive models. Prior work has explored diverse strategies to address this, including learning separate length predictors (Gu et al., 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Zheng et al., 2023), marginalizing over latent alignments to contract sequence length (Chan et al., 2020), incorporating edit operations (Gu et al., 2019a;b; Stern et al., 2019; Johnson et al., 2021; Reid et al., 2023; Campbell et al., 2023; Patel et al., 2025; Havasi et al., 2025), and performing diffusion over sequence positions (Zhang et al., 2025; Kim et al., 2025).

Recent concurrent works also address this challenge. Edit flows (Havasi et al., 2025) present a discrete flow matching with edit operations over extended spaces for tractable and effective training; DDOT (Zhang et al., 2025) proposes to jointly denoise token states and positions for dynamic segment-length adjustment; FlexMDM (Kim et al., 2025) learns insertion and unmasking rates through a joint interpolant framework over both token states and positions, thereby enabling variable-length generation; and DAEDAL (Li et al., 2025) provides a training-free approach using inference-time prediction confidence scores to dynamically determine the response length.

In contrast, our method implements native length control in masked diffusion models with minimal additional training and no architectural modifications, directly adapting pretrained diffusion language models. This design preserves the simplicity of the original model, avoids complicated multi-stage inference pipelines, and yields substantial gains in flexibility and performance for variable-length generation.

7 CONCLUSIONS

In this work, we introduce DREAMON, a simple yet effective framework that enables dynamic length control through two special tokens (`[delete]` and `[expand]`) without architectural changes. By augmenting the diffusion process with auxiliary length-control states, DREAMON learns to expand or contract sequences based solely on model confidence. Our results show that DREAMON approaches oracle-length performance and achieves competitive results with state-of-the-art autoregressive models. We hope our work can pave the way for more practical and flexible DLMs beyond fixed-size canvas.

Limitations. Currently, we limit our evaluation to focus on code infilling tasks that require strong variable-length generation capabilities. Future work will extend the scope to broader applications to assess the generalizability of DREAMON. In addition, the training and inference procedures in DREAMON rely on heuristics to enable variable-length generation in a simple yet effective manner; developing a more principled formulation for flexible inference in masked diffusion models is an important direction for future research. Finally, our current design uses a single expansion state `[expand]` that deterministically expands into two `[mask]` tokens. This choice keeps the output space small and training stable, but requires multiple expansion steps when the target completion is much longer than the initial mask span. A promising direction for future work is to introduce a richer vocabulary with multiple expansion factors or to couple expansion with an explicit length-prediction head. They could reduce the number of denoising iterations, but would also enlarge the decision space and require careful rebalancing of the training objective to maintain well-behaved length adjustment dynamics. We leave the exploration of these richer expansion schemes to future work.

ACKNOWLEDGEMENTS

We acknowledge the open-source community for providing high-quality datasets and evaluation frameworks. This research was supported in part by the joint research scheme of the National Natural Science Foundation of China (NSFC) and the Research Grants Council (RGC) under grant number N_HKU714/21.

ETHICS STATEMENT

Our research focuses on developing a diffusion-based language modeling method capable of variable-length text generation. We did not collect any data involving human subjects, private information. And our study does not include any human evaluation. All datasets used in our experiments are publicly available benchmarks, and we strictly adhere to their respective usage licenses. Furthermore, our method does not present any foreseeable risks of misuse or societal harm.

REPRODUCIBILITY STATEMENT

We have taken deliberate steps to ensure the reproducibility of our work. Detailed descriptions of the experimental setups and hyperparameter configurations are provided in §4.

REFERENCES

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. SantaCoder: Don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023.

Uri Alon, Roy Sadaka, Omer Levy, and Eran Yahav. Structural language models of code. In *International conference on machine learning*, pp. 245–256. PMLR, 2020.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403*, 2023.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=h7-XixPCAL.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Andrew Campbell, William Harvey, Christian Dietrich Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=t6nA7x3GAC.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*, 2019.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence modelling via imputation and dynamic programming. In *International Conference on Machine Learning*, 2020.

Google LLC CodeGemma Team. Codegemma: Open code models based on gemma, 2024. URL https://storage.googleapis.com/deepmind-media/gemma/codegemma_report.pdf.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

DeepMind. Gemini diffusion. 2025. URL https://deepmind.google/models/gemini-diffusion/.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. InCoder: A generative model for code infilling and synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=hQwb-lbM6EL.

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 (EMNLP-IJCNLP)*, 2019. URL https://aclanthology.org/D19-1633/.

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. *International Conference on Learning Representations*, 2025a.

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

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 2019a. URL https://aclanthology.org/Q19-1042/.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 2019b.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024a.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024b.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=6nbpPqUCIi7.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, et al. Opencoder: The open cookbook for top-tier code large language models. *arXiv preprint arXiv:2411.04905*, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Daniel D. Johnson, Jacob Austin, Rianne van den Berg, and Daniel Tarlow. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. URL https://openreview.net/forum?id=cAsVBUe1Rnj.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. *arXiv preprint arXiv:2509.01025*, 2025.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1149/.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *arXiv preprint arXiv:2508.00819*, 2025.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=CNicRIVIPA.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, 2016.

Anh Nguyen, Nikos Karampatziakis, and Weizhu Chen. Meet in the middle: A new pre-training paradigm. *arXiv preprint arXiv:2303.07295*, 2023.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. CodeGen2: Lessons for training LLMs on programming and natural languages. *arXiv preprint arXiv:2305.02309*, 2023.

OpenAI. Openai o3 and o4-mini system card, 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf. System Card.

OpenAI, Mohammad Bavarian, Angela Jiang, Heewoo Jun, and Henrique Pondé. New GPT-3 Capabilities: Edit and Insert. *OpenAI blog*, 2022. URL https://openai.com/blog/gpt-3-edit-insert/.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *International Conference on Learning Representations*, 2025.

Dhruvesh Patel, Aishwarya Sahoo, Avinash Amballa, Tahira Naseem, Tim GJ Rudner, and Andrew McCallum. Insertion language models: Sequence generation with arbitrary-position insertions. *arXiv preprint arXiv:2505.05755*, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67, 2020.

Machel Reid, Vincent Josua Hellendoorn, and Graham Neubig. DiffusER: Diffusion via edit-based reconstruction. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nG9RF9z1yy3.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

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. URL https://openreview.net/forum?id=L4uaAR4ArM.

ByteDance Seed, Yuyu Zhang, Jing Su, Yifan Sun, Chenguang Xi, Xia Xiao, Shen Zheng, Anxiang Zhang, Kaibo Liu, Daoguang Zan, et al. Seed-coder: Let the code model curate data for itself. *arXiv preprint arXiv:2506.03524*, 2025.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. *arXiv preprint arXiv:2002.03079*, 2020.

Tianxiao Shen, Hao Peng, Ruoqi Shen, Yao Fu, Zaid Harchaoui, and Yejin Choi. Film: Fill-in language models for any-order generation. *arXiv preprint arXiv:2310.09930*, 2023.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, and Gust Verbruggen. CodeFusion: A pre-trained diffusion model for code generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.716/.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pp. 5976–5985. PMLR, 2019.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, et al. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734*, 2024.

Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q Tran, David R So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, et al. Transcending scaling laws with 0.1% extra compute. *arXiv preprint arXiv:2210.11399*, 2022.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6ruVLB727MC.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pp. 6716–6726. PMLR, 2019.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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.

Andrew Zhang, Anushka Sivakumar, Chiawei Tang, and Chris Thomas. Flexible-length text infilling for discrete diffusion models. *arXiv preprint arXiv:2506.13579*, 2025.

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

A The Use of Large Language Models

We employ large language models primarily for polishing written text—for example, to correct grammar and improve clarity. However, LLMs do not play a significant role in the core research activities, including idea generation, experimental design, or the substantive writing of the manuscript.

B Generalizability Beyond Code Infilling

To demonstrate that DreamOn is not limited to code infilling, we further evaluate the model's capability of commonsense narrative understanding on the ROCStories corpus Mostafazadeh et al. (2016). We fine-tune Dream-7B (Ye et al., 2025) on the ROCStories training split in two variants: with DreamOn, and an SFT baseline trained following the recipe in Ye et al. (2025). We consider two setups: (1) **Narrative Infilling**, where the model generates a missing sentence within the middle of a story, requiring bidirectional context understanding; and (2) **Prefix-guided Generation**, where the model is provided with the preceding story context and generates the final concluding sentence, serving as a proxy for general completion tasks.

Table 4 presents Rouge-L scores across varying initial mask lengths. In both infilling and prefix-guided settings, the baseline performance degrades significantly when there is a mismatch between the initialized mask length and the natural length of the missing content (e.g., at lengths 4 and 32). In contrast, DreamOn utilizes its length-adaptive logic—mediated by `[expand]` and `[delete]`-to achieve superior performance.

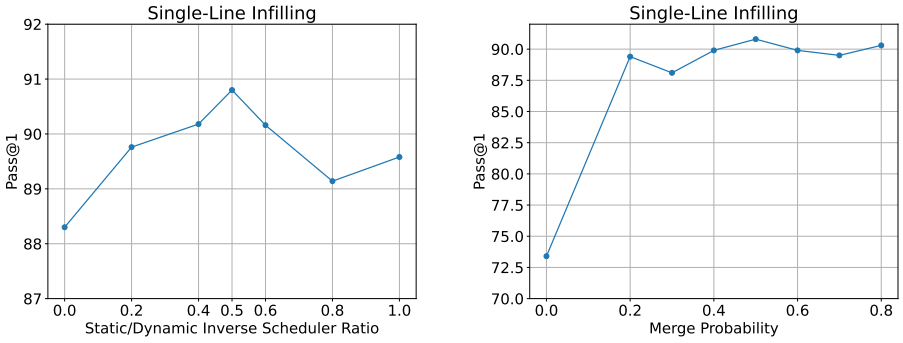
Crucially, our method exhibits high stability: the generation quality remains consistent regardless of the initial mask length. This confirms that the proposed length-adaptation mechanism successfully decouples generation quality from the initial mask length, demonstrating robust generalizability to creative, variable-length natural language tasks.

Table 4: Rouge-L scores on the ROCStories corpus across variable initial mask lengths.

Method	Initial Mask Length			
	4	8	16	32
Narrative Infilling				
Dream + SFT	19.2	29.8	26.5	18.9
DREAMON	**31.6**	**31.4**	**31.3**	**30.6**
Story Ending Generation				
Dream + SFT	16.3	**25.1**	22.4	16.7
DREAMON	**24.5**	24.6	**24.4**	**24.1**

C ABLATION FOR HYPERPARAMETERS

In this section,we provide the results on DreamCoder-7B with different training hyperparameters. As shown in Figure 5a, the Pass@1 score for single-line infilling reaches its peak—approximately 90.9%—when employing a balanced 1:1 mixture of static and dynamic inverse schedulers. This result highlights the substantial performance gain achieved through this synergistic combination. Similarly, the right panel reveals that the model attains its highest Pass@1 score of roughly 90.5% at a merge probability of 0.5. Guided by these findings, we adopt a 1:1 static/dynamic scheduler mix ratio and a merge probability of 0.5 in DREAMON configuration to maximize performance.



(a) Result with different scheduler merging ratio. (b) Result with different merging probability p_{merge}

Figure 5: Performance on single-line subset of HumanEvalInfilling-FIM with different hyperparameters during training. The performance is computed as the average pass@1 with mask length 4, 8, 16, 32 and 64.

D INFILLING EXAMPLES

A key advantage of DREAMON lies in its adaptive handling of sequence length variations during inference. This is achieved through two complementary states `[expand]` and `[delete]`. First, as depicted in Figure 6, DREAMON possesses the capability to expand mask sequences. This dynamic expansion allows the model to generate outputs longer than its initial input mask, effectively preventing truncation and enabling the generation of comprehensive sequences. Second, Figure 7 showcases the efficacy of the deletion broadcasting mechanism. This mechanism plays a crucial role

Figure 6: DREAMON adds mask tokens as needed.

in promoting rapid convergence to the optimal predicted sequence length by selectively removing redundant mask tokens, thereby streamlining the generation process and improving efficiency.

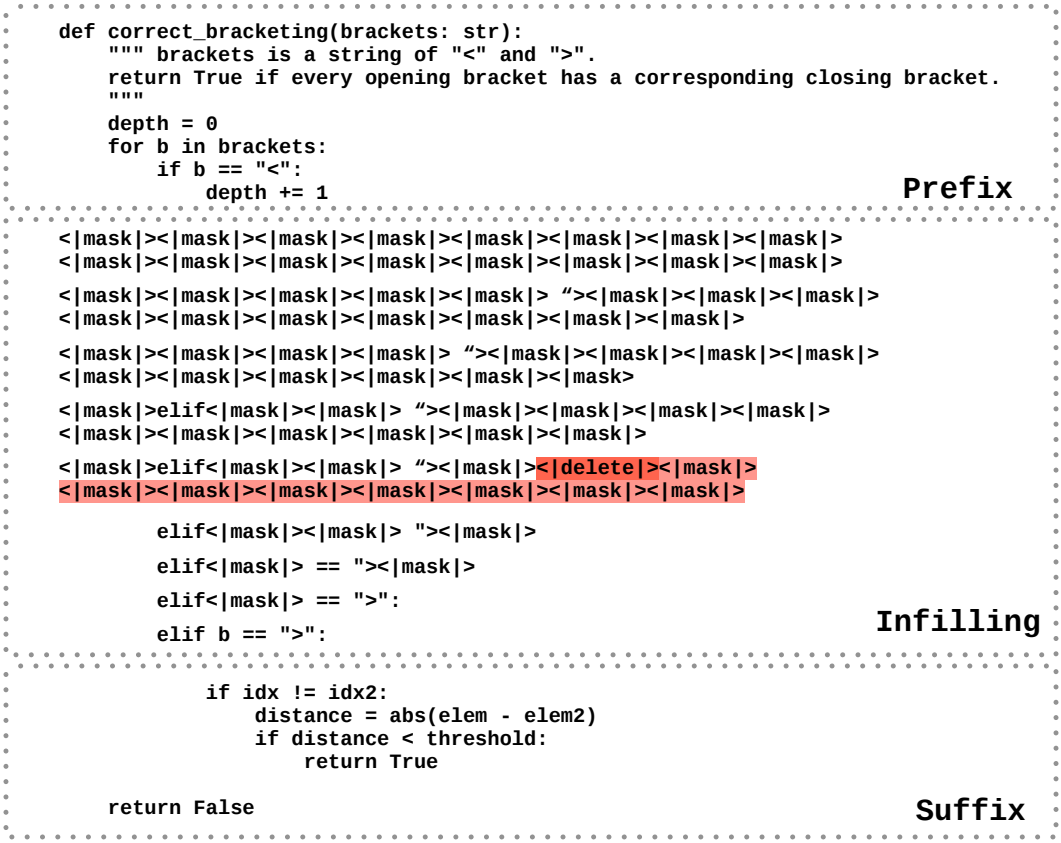


Figure 7: DREAMON deletes excess mask tokens with the deletion broadcasting mechanism.