Simple and Effective Masked Diffusion Language Models

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

While diffusion models excel at generating high-quality images, prior work reports a significant performance gap between diffusion and autoregressive (AR) methods on language modeling. In this work, we show that simple masked discrete diffusion is more performant than previously thought. We apply an effective training recipe that improves the performance of masked diffusion models and derive a simplified, Rao-Blackwellized objective that results in additional improvements. Our objective has a simple form—it is a mixture of classical masked language modeling losses—and can be used to train encoder-only language models that admit efficient samplers, including ones that can generate arbitrary lengths of text semi-autoregressively like a traditional language model. On language modeling benchmarks, a range of masked diffusion models trained with modern engineering practices achieves a new state-of-the-art among diffusion models, and approaches AR perplexity.

1 Introduction

In this work we describe (1) a simple masked diffusion language modeling (MDLM) framework with a well-engineered implementation that outperforms all existing diffusion models across language modeling benchmarks (LM1B [4], OWT [11], DNA [33]), and that significantly improves the performance of existing baselines [1, 17]. Our MDLM framework implements (2a) a substitution-based parameterization (SUBS) of the reverse unmasking diffusion process; SUBS allows us to derive (2b) a simple, continuous-time, Rao-Blackwellized objective that improves tightness and variance of the ELBO, further increasing performance. We complement MDLM with (3) fast samplers that support semi-autoregressive (SAR) generation and outperform previous SAR models.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Table 1: Test perplexities (PPL; ↓) on LM1B. †Reported in He et al. [17]. Best diffusion value is bolded.

		Parameters	PPL (↓)
Ar	Transformer-X Base [8]	0.46B	23.5
	OmniNet$_T$ [40]	100M	21.5
Dif	BERT-Mouth [41]	110M	$\leq$142.89
	D3PM (absorb) [1]	70M	$\leq$77.50
	Diffusion-LM [20]†	80M	$\leq$118.62
	DiffusionBert [17]	110M	$\leq$63.78
	SEDD [21] (33B tokens)	110M	$\leq$ 32.79
Ar (Retrained)	Transformer (33B tokens)	110M	22.32
	Transformer (330B tokens)		20.86
Dif (Ours)	MDLM (33B tokens)	110M	$\leq$27.04
	MDLM (330B tokens)		$\leq$**23.00**

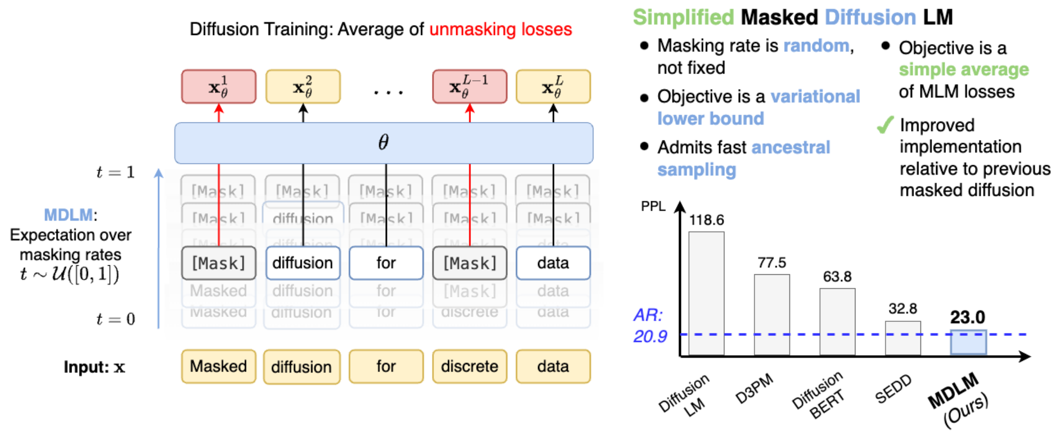

Figure 1: *(Left)* Our proposed masked diffusion language model (MDLM) is trained using a weighted average of masked cross entropy losses. (*Top Right*) In comparison to masked language models (MLM), MDLM's objective correspond to a principled variational lower bound, and supports generation via ancestral sampling. (*Bottom Right*) Perplexity (PPL) on One Billion Words benchmark.

2 Background

2.1 Diffusion Models

Diffusion models are trained to iteratively undo a forward corruption process q that takes clean data $\mathbf{x}$ drawn from the data distribution $q(\mathbf{x})$ and defines latent variables $\mathbf{z}_t$ for $t \in [0,1]$ that represent progressively noisy versions of $\mathbf{x}$ [18, 34, 36]. The standard forward process for continuous $\mathbf{x}$ is

$$\mathbf{z}_t = \sqrt{\alpha_t} \cdot \mathbf{x} + \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon} \tag{1}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $(\alpha_t)_{t\in[0,1]}$ is a noise schedule, monotonically decreasing in t. The parameterized reverse diffusion model p_θ over $\mathbf{x}$ and $\mathbf{z}_t$ is trained to maximize a variational lower bound on log-likelihood (ELBO). Given a number of discretization steps T, defining $s(i) = (i-1)/T$ and $t(i) = i/T$, and using $D_{\mathrm{KL}}[\cdot]$ to denote the Kullback–Leibler divergence, the ELBO equals [34]:

$$\mathbb{E}_q \Bigg[\underbrace{\log p_\theta(\mathbf{x}|\mathbf{z}_{t(0)})}_{\mathcal{L}_{\text{recons}}} - \underbrace{\sum_{i=1}^{T} D_{\mathrm{KL}}[q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x}) \| p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})]}_{\mathcal{L}_{\text{diffusion}}} \Bigg] \\ \underbrace{- D_{\mathrm{KL}}[q(\mathbf{z}_{t(T)}|\mathbf{x}) \| p_\theta(\mathbf{z}_{t(T)})]}_{\mathcal{L}_{\text{prior}}} \tag{2}$$

For brevity, we drop i from $t(i)$ and $s(i)$ below; in general, s will denote the time step before t.

3 Simple Masked Diffusion Models

While previous work on discrete diffusion supports general forward processes (e.g., general Q_t in D3PM), absorbing state (i.e., masking) diffusion consistently achieves the best performance [1, 21]. In this work, instead of supporting general noise processes, we focus on masking and derive tight Rao-Blackwellized objectives that outperform general approaches and do not require CTMC theory. We denote our overall approach as masked diffusion (MDLM in the context of language models).

Notation. We denote scalar discrete random variables with K categories as 'one-hot' column vectors and define $\mathcal{V} \in \{\mathbf{x} \in \{0,1\}^K : \sum_{i=1}^{K} \mathbf{x}_i = 1\}$ as the set of all such vectors. Define $\text{Cat}(\cdot; \boldsymbol{\pi})$ as the categorical distribution over K classes with probabilities given by $\boldsymbol{\pi} \in \Delta^K$, where Δ^K denotes the K-simplex. We also assume that the K-th category corresponds to a special [MASK] token and let $\mathbf{m} \in \mathcal{V}$ be the one-hot vector for this mask, i.e., $\mathbf{m}_K = 1$. Additionally, let $\mathbf{1} = \{1\}^K$ and $\langle \mathbf{a}, \mathbf{b} \rangle$ and $\mathbf{a} \odot \mathbf{b}$ respectively denote the dot and Hadamard products between two vectors $\mathbf{a}$ and $\mathbf{b}$.

3.1 Interpolating Discrete Diffusion

We restrict our attention to forward processes q that interpolate between clean data $\mathbf{x} \in \mathcal{V}$ and a target distribution $\text{Cat}(.;\boldsymbol{\pi})$, forming a direct extension of Gaussian diffusion in (1) given as:

$$q(\mathbf{z}_t|\mathbf{x}) = \text{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x} + (1-\alpha_t)\boldsymbol{\pi}), \tag{3}$$

where $\alpha_t \in [0,1]$ is a strictly decreasing function in t, with $\alpha_0 = 1$ and $\alpha_1 = 0$. This implies transition probabilities $q(\mathbf{z}_t|\mathbf{z}_s) = \text{Cat}(\mathbf{z}_t; \alpha_{t|s}\mathbf{z}_t + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t)$ where $\alpha_{t|s} = \alpha_t / \alpha_s$ and $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ is given as:

$$\text{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{z}_t + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t] \odot [\alpha_s \mathbf{x} + (1-\alpha_s)\boldsymbol{\pi}]}{\alpha_t \mathbf{z}_t^\top \mathbf{x} + (1-\alpha_t)\mathbf{z}_t^\top \boldsymbol{\pi}}\right) \tag{4}$$

See Suppl. 14 for details. While (3) and (4) represent a special case of the more general diffusion processes proposed in D3PM [1], we show below that they yield a simplified variational lower bound objective and admit straightforward continuous time extensions.

3.2 Masked Diffusion

Forward Masking Process In masked (i.e., absorbing state) diffusion, we set $\boldsymbol{\pi} = \mathbf{m}$. At each noising step, the input $\mathbf{x}$ transitions to a 'masked' state $\mathbf{m}$ with a probability increasing in t. If an input transitions to $\mathbf{m}$ at any time t', it will remain in this state for all $t > t' : q(\mathbf{z}_t \mid \mathbf{z}_{t'} = \mathbf{m}) = \text{Cat}(\mathbf{z}_t; \mathbf{m})$. The marginal of the forward process (3) is given by $q(\mathbf{z}_t|\mathbf{x}) = \alpha_t \mathbf{x} + (1-\alpha_t)\mathbf{m}$. Using properties of the masking process, the posterior $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ simplifies (4); see Suppl. A:

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \begin{cases} \text{Cat}(\mathbf{z}_s; \mathbf{z}_t) & \mathbf{z}_t \neq \mathbf{m}, \\ \text{Cat}\left(\mathbf{z}_s; \frac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}}{1-\alpha_t}\right) & \mathbf{z}_t = \mathbf{m}. \end{cases} \tag{5}$$

Reverse Unmasking Process: SUBS Parameterization The reverse process inverts the noise process defined by q. We consider both a finite number of steps T, as well as a continuous time model corresponding to $T \to \infty$. We begin with the discrete-time case for which the generative model is expressed as $p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{z}_1) p_\theta(\mathbf{x}|\mathbf{z}_0) \prod_{i=1}^{T} p_\theta(\mathbf{z}_s|\mathbf{z}_t) \mathrm{d}\mathbf{z}_{0:T}$. We introduce a model $\mathbf{x}_\theta(\mathbf{z}_t, t) : \mathcal{V} \times [0,1] \to \Delta^K$ that approximates $\mathbf{x}$ with a neural network. The specific parameterization for $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ that we use is

$$p_\theta(\mathbf{z}_s|\mathbf{z}_t) = \begin{cases} \text{Cat}(\mathbf{z}_s; \mathbf{z}_t), & \mathbf{z}_t \neq \mathbf{m}, \\ \text{Cat}\left(\mathbf{z}_s; \frac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}_\theta(\mathbf{z}_t, t)}{1-\alpha_t}\right). & \mathbf{z}_t = \mathbf{m}. \end{cases} \tag{6}$$

In order for $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ to be a valid probability, $\mathbf{x}_\theta(\mathbf{z}_t, t)$ must satisfy two requirements. We implement these as substitutions to the output of $\mathbf{x}_\theta(\mathbf{z}_t, t)$, hence we call our parameterization SUBS.

Zero Masking Probabilities First, notice that by definition, $\langle \mathbf{x}, \mathbf{m} \rangle = 0$. For this reason, we design the denoising network such that $\langle \mathbf{x}_\theta(\mathbf{z}_t, t), \mathbf{m} \rangle = 0$, i.e., we substitute the logit index corresponding to the [MASK] token with $-\infty$. This property enables the simplified expression of (6) (Suppl. A.3.2) and ensures that case 2 in (6) is a valid probability.

Carry-Over Unmasking Second, if $\mathbf{z}_t$ is unmasked, then we desire $\mathbf{x}_\theta(\mathbf{z}_t, t) = \mathbf{z}_t$, i.e., unmasked latents are 'carried over'. We accomplish this by substituting the output of our network to simply copy unmasked inputs. This ensures that case 1 in (6) always holds, and furthermore reduces $\mathcal{L}_{\text{recons}}$ to 0.

3.3 Rao-Blackwellized Likelihood Bounds

Recall from (2) that the diffusion traning objective has the form $\mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{diffusion}} + \mathcal{L}_{\text{prior}}$. For the simplified reverse process in (6), the discrete-time diffusion loss for finite T simplifies to (Suppl. B.1):

$$\mathcal{L}_{\text{diffusion}} = \sum_{i=1}^{T} \mathbb{E}_q \left[\frac{\alpha_{t(i)} - \alpha_{s(i)}}{1 - \alpha_{t(i)}} \log \langle \mathbf{x}_\theta(\mathbf{z}_{t(i)}), \mathbf{x} \rangle \right]. \tag{7}$$

Note that this objective is simpler and more well-behaved than the expression one would obtain for $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_s|\mathbf{z}_t,\mathbf{x})\|p_\theta(\mathbf{z}_s|\mathbf{z}_t))$ under the parameterization induced by using $p_\theta(\mathbf{z}_s|\mathbf{z}_t)=q(\mathbf{z}_s|\mathbf{z}_t,\mathbf{x}=\mathbf{x}_\theta(\mathbf{z}_t,t))$ from (4), which is similar to what is used by D3PM [1] (see Suppl. 27):

$$\left[\frac{\alpha_s-\alpha_t}{1-\alpha_t}\log\frac{\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t)}{(1-\alpha_t)\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle}\right.$$
$$\left.+\frac{1-\alpha_s}{1-\alpha_t}\log\frac{(1-\alpha_s)(\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t))}{(1-\alpha_t)(\alpha_s\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_s))}\right]\langle\mathbf{z}_t,\mathbf{m}\rangle. \tag{8}$$

We refer to the process of obtaining (7) in lieu of (8) as a form of Rao-Blackwellization.

3.4 Continuous-Time Likelihood Bounds

Previous works have shown empirically and mathematically that increasing the number of steps T yields a tighter approximation to the ELBO [19]. Following a similar argument, we form an continuous extension of (7) by taking $T\to\infty$ (see Suppl. B.2), which yields

$$\mathcal{L}^{\infty}_{\text{diffusion}}=\mathbb{E}_q\int_{t=0}^{t=1}\frac{\alpha'_t}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle\mathrm{d}t \tag{9}$$

3.5 Masked Diffusion Language Models

Next, we apply masked diffusion to language modeling over sequences $\mathbf{x}^{1:L}$ of L tokens, with $\mathbf{x}^\ell$ denoting the ℓ-th token. We make the assumption that the forward noising process is applied independently across a sequence and that, conditioned on a sequence of latents $\mathbf{z}_t^{1:L}$, the denoising process factorizes independently across tokens, i.e., $p_\theta(\mathbf{z}_s^{1:L}\mid\mathbf{z}_t^{1:L})=\prod_{\ell=1}^{L}p_\theta(\mathbf{z}_s^\ell\mid\mathbf{z}_t^{1:L})$. Thus, we use a single model to compute $\mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L},t)$ for each ℓ from a masked sequence $\mathbf{z}_t$, optimizing:

$$\mathcal{L}^{\infty}_{\text{diffusion}}=\mathbb{E}_q\int_{t=0}^{t=1}\frac{\alpha'_t}{1-\alpha_t}\sum_{\ell}\log\langle\mathbf{x}_\theta^\ell(\mathbf{z}_t),\mathbf{x}^\ell\rangle\mathrm{d}t \tag{10}$$

Interestingly, our objective has a simple form: it is the weighted average of masked language modeling (MLM) losses [9]. Thus our work establishes a connection between generative diffusion models and encoder-only BERT models. Our objective enables principled selection of a (randomized) masking rate, and also endows BERT-style models with principled generation capabilities, see Sec. 6.

4 Inference and Sampling in Masked Diffusion Language Models

4.1 Efficient Ancestral Sampling

To generate a sequence of length L, the reverse diffusion process starts with the sequence $\mathbf{z}_{t=1}^{1:L}$ where $\mathbf{z}_{t=1}^\ell=\mathbf{m},\forall\ell\in\{1,\dots,L\}$. Then the subsequent latents, $\mathbf{z}_t^{1:L}$ are generated by discretizing the reverse diffusion process with some finite T. Given $\mathbf{z}_t^{1:L}$, we construct $\mathbf{z}_s^{1:L}$ by sampling each token $\mathbf{z}_s^\ell$ independently from the distribution $p_\theta(\mathbf{z}_s^\ell|\mathbf{z}_t^{1:L})$ given in (6).

4.2 Semi-Autoregressive Masked Diffusion Language Models

Our method also admits an effective semi-autoregressive (SAR) decoding method that allows the model to generate sequences of arbitrary length. Let $\tilde{\mathbf{x}}^{1:L}$ represent the output from sampling a sequence of L tokens using the reverse diffusion process described above. To generate additional $L'<L$ tokens, we propose a generation algorithm in which the latter $L-L'$ tokens $\tilde{\mathbf{x}}^{L':L-L'}$ are used as a prefix for an additional round of generation. Given the carry-over unmasking described in Sec. 3.2, these prefix tokens will simply be copied over at each decoding step. The remaining tokens are generated as above with $\mathbf{z}_s^\ell\sim p_\theta(\mathbf{z}_s^\ell\mid\mathbf{z}_t^{L':L+L'})$ for all $\ell\in\{L+1,\dots L+L'\}$, with $\mathbf{z}_1^{L':L-L'}$ initialized to $\tilde{\mathbf{x}}^{L':L-L'}$ as opposed to being initialized as masked tokens $\mathbf{m}$. At the end of this process, we have produced $L+L'$ tokens $\mathrm{concat}[\tilde{\mathbf{x}}^{1:L},\tilde{\mathbf{x}}^{L+1:L+L'}]$, where $\mathrm{concat}[\cdot]$ denotes concatenation along the sequence length dimension. This process can repeat indefinitely, with the prefix shifted for every new round of generation.

5 Experiments

The experiment setup is described in Suppl. C.1

5.1 Masked Diffusion Language Models

Likelihood Evaluation On LM1B, MDLM outperforms all previous diffusion methods (Table 1). Compared to the SEDD baseline reported by Lou et al. [21], trained for 66B tokens, MDLM, which we train for the same amount, achieves a 17% improvement on the perplexity bound. Finally, MDLM gets within 14% of an AR baseline and continues to improve with more training. We see the same trend for models trained on OWT, a larger dataset, shown in Table 9 – MDLM outperforms prior diffusion methods, closing the gap towards AR models. Results on OWT time step conditioning are in Table 6, Suppl. C.5 where we find that models trained with and without time conditioning attain similar perplexities. Additionally, Figure 2 demonstrates the reduced variance we achieve from our objective, when compared to previous masked diffusion models, such as SEDD [21].

Zero-Shot Likelihood Evaluation We also explore models' ability to generalize by taking models trained on OWT and evaluating how well they model unseen datasets. MDLM consistently outperforms the SEDD diffusion parameterization on all datasets. In some cases, e.g., for Lambada and Scientific Papers, MDLM attains better perplexity than AR. Details in Suppl. C.6.

Downstream Task Evaluation In Table 8, we find that BERT fine-tuned with MDLM to be a generative model results in strong perplexities while preserving performance on downstream tasks.

Semi-Autoregressive Modeling To test the SAR decoding algorithm presented in Sec. 4.2, we compare to SSD-LM [16]. In Table 11, we find that in addition to achieving better generative perplexity, MDLM enables ∼25-30x faster SAR decoding relative to SSD-LM (details in Suppl. C.10).

5.2 Masked Diffusion DNA Models

Table 2: Genomic Benchmarks. Top-1 accuracy (↑) across 5-fold cross-validation (CV) for a pre-trained AR Mamba, and pre-trained Caduceus model fine-tuned with different diffusion parameterizations. Best values per task are bolded and second best are italicized. Error bars indicate difference between maximum and minimum values across 5 random seeds used for CV.

Model / Fine-Tuning (Params)	Mamba / AR (465K)	Caduceus / MLM (467K)	Caduceus / Plaid (507k)	Caduceus / SEDD (467k)	Caduceus / MDLM (467k)
Mouse Enhancers	0.763 {±0.008}	**0.810** {±0.016}	0.745 {±0.079}	0.784 {±0.058}	*0.795* {±0.029}
Coding vs. Intergenomic	0.897 {±0.004}	**0.913** {±0.003}	*0.908* {±0.003}	**0.913** {±0.005}	**0.913** {±0.003}
Human vs. Worm	0.967 {±0.002}	*0.970* {±0.002}	**0.971** {±0.001}	*0.970* {±0.003}	*0.970* {±0.003}
Human Enhancers Cohn	0.734 {±0.027}	0.737 {±0.001}	*0.743* {±0.010}	**0.746** {±0.015}	*0.743* {±0.016}
Human Enhancer Ensembl	0.856 {±0.003}	**0.907** {±0.000}	0.885 {±0.003}	*0.905* {±0.006}	0.899 {±0.004}
Human Regulatory	0.861 {±0.008}	**0.874** {±0.003}	*0.868* {±0.010}	0.828 {±0.037}	*0.868* {±0.004}
Human OCR Ensembl	0.806 {±0.005}	*0.821* {±0.000}	0.820 {±0.004}	0.816 {±0.008}	**0.823** {±0.008}
Human NonTATA Promoters	0.926 {±0.008}	*0.935* {±0.014}	*0.935* {±0l007}	*0.935* {±0.014}	**0.940** {±0.007}

We also explore the use of our generative formulation in conjunction with Structured State Space models [14]. Namely, we build on the recently proposed Caduceus [33] model, which uses as a backbone the data-dependent SSM Mamba block [13]. We pre-train the encoder-only Caduceus [33], which is an MLM, on the HG38 human reference genome [7] and perform fine-tuning using our diffusion parameterization. We use a context length of 1024 tokens and follow Schiff et al. [33] for the experimental setup, other than learning rate which was reduced to 1e-3. See Suppl. I.4 for full experimental details. We assess both generative performance using perplexity and downstream performance on Genomics Benchmarks [12] across language diffusion paradigms and AR models.

Generative Performance We fine-tune the Caduceus MLM across diffusion parameterizations and compare perplexities against AR models. We report perplexity values in Table 3. MDLM outperforms all other diffusion language modeling schemes.

Downstream Task Fine-tuning We perform downstream evaluation with the Genomics Benchmarks [12], a recently proposed benchmark with eight regulatory element classification tasks. As shown in Table 2, our generative fine-tuning paradigm preserves or improves upon downstream

Table 3: Test perplexities (PPL; ↓) of generative fine-tuning of the Caduceus MLM [33] on the HG38 reference genome. Best diffusion model values are bolded. Error bars indicate the difference between the maximum and minimum values across 5 random seeds used for fine-tuning. † denotes retrained models.

		Params	PPL (↓)
AR†	Mamba	465K	$3.067 \pm .0104$
	HyenaDNA	433K	$3.153 \pm .001$
Dif†	Plaid	507K	$\leq 3.240 \pm .005$
	SEDD	467K	$\leq 3.216 \pm .003$
Dif (Ours)	MDLM	467K	$\leq \mathbf{3.199} \pm .010$

performance from MLM pre-training. Absorbing-state diffusion methods outperform Plaid across tasks except for the simplest task Human vs. Worm, where all methods have roughly the same performance. For tasks where the input is a biased subsample of the full genome, we observe that the correlation between perplexity and downstream performance is weaker; see Suppl. I.4.

6 Conclusion

Conclusion In this work, we explore masked diffusion. With a well-engineered implementation that supports a simple variational objective, we attain state-of-the-art diffusion perplexities on language benchmarks and demonstrate how to efficiently convert BERT-style encoders into generative models.

References

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[2] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.

[3] 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.

[4] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014.

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

[6] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018. doi: 10.18653/v1/n18-2097. URL http://dx.doi.org/10.18653/v1/n18-2097.

[7] Genome Reference Consortium. Genome reference consortium human build 37 (grch37. *Database (GenBank or RefSeq)*, 2009.

[8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

[11] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[12] Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

[13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[14] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[15] Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.

[17] Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*, 2022.

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

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

[20] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

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

[22] Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Vincent Mallet and Jean-Philippe Vert. Reverse-complement equivariant networks for dna sequences. *Advances in neural information processing systems*, 34:13511–13523, 2021.

[24] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[25] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[26] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36, 2024.

[27] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1144.

[28] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

[29] Jacob Portes, Alex Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. Mosaicbert: A bidirectional encoder optimized for fast pretraining, 2024.

[30] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

[31] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[32] 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. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

[33] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234*, 2024.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

[35] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[37] Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*, 2022.

[38] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[39] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

[40] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip M Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. Omninet: Omnidirectional representations from transformers. In *International Conference on Machine Learning*, pp. 10193–10202. PMLR, 2021.

[41] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[42] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

[43] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[44] Hannah Zhou, Avanti Shrikumar, and Anshul Kundaje. Towards a better understanding of reverse-complement equivariance for deep learning models in genomics. In *Machine Learning in Computational Biology*, pp. 1–33. PMLR, 2022.

A Discrete time ELBO

This section is organized as follows: First, we derive the expressions for the true posterior and the approximate posterior as outlined in Suppl. A.2. We then simplify these expressions specifically for the case of absorbing state diffusion in Suppl. A.3. Finally, we derive the expression for the ELBO for absorbing state diffusion in Suppl. A.3.3.

A.1 Discrete Diffusion Models

Applications of diffusion modeling to discrete data can be broken into two broad categories. First are works that embed discrete structures in continuous space and then perform the Gaussian diffusion defined above on these continuous representations [5, 10, 15, 16, 20, 22, 37]. More related to our method are works that define a diffusion process directly on discrete structures. D3PM [1] introduces a framework with a Markov forward process $q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \text{Cat}(\mathbf{z}_t; Q_t \mathbf{z}_{t-1})$ defined by the multiplication of matrices Q_t over T discrete time steps. This process induces marginals

$$q(\mathbf{z}_t|\mathbf{x}) = \text{Cat}(\mathbf{z}_t; \bar{Q}_t \mathbf{x}) = \text{Cat}(\mathbf{z}_t; Q_t \cdot Q_{t-1} \cdots Q_1 \mathbf{x}) \tag{11}$$

that represent the discrete-state form of (1). Extending this formalism to continuous time (as in (1)) relies on continuous time Markov chain (CTMC) theory [3]. The CTMC framework in turns leads to generalizations of the score matching perspective on diffusion modeling [35] to discrete data [21, 39]. Notably, SEDD [21] connects score-based approaches with ELBO maximization, enabling performant likelihood-based training of score-based models.

A.2 Generic case

A.2.1 $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$

Given the state transition matrix Q_t, prior $\boldsymbol{\pi}$, and the latent variables $\mathbf{z}_s$ and $\mathbf{z}_t$, where $s < t$, the forward process defined in (11) has the following posterior [1]:

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \text{Cat}\left(\mathbf{z}_s; \frac{Q_{t|s}\mathbf{z}_t \odot Q_s^\top \mathbf{x}}{\mathbf{z}_t^\top Q_t^\top \mathbf{x}}\right) \tag{12}$$

$$Q_{t|s} = \alpha_{t|s}\mathbf{I}_n + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \tag{13}$$

which we simplify to the following:

$$\begin{aligned}
&q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})\\
&= \text{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{I}_n + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top]\mathbf{z}_t \odot [\alpha_s \mathbf{I}_n + (1-\alpha_s)\mathbf{1}\boldsymbol{\pi}^\top]^\top \mathbf{x}}{\mathbf{z}_t^\top [\alpha_t \mathbf{I}_n + (1-\alpha_t)\mathbf{1}\boldsymbol{\pi}^\top]^\top \mathbf{x}}\right)\\
&= \text{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{z}_t + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t] \odot [\alpha_s \mathbf{x} + (1-\alpha_s)\boldsymbol{\pi}]}{\mathbf{z}_t^\top [\alpha_t \mathbf{x} + (1-\alpha_t)\boldsymbol{\pi}\mathbf{1}^\top \mathbf{x}]}\right)\\
&\text{Using the property } \mathbf{1}^\top \mathbf{x} = 1 \text{ we get,}\\
&= \text{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{z}_t + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t] \odot [\alpha_s \mathbf{x} + (1-\alpha_s)\boldsymbol{\pi}]}{\alpha_t \mathbf{z}_t^\top \mathbf{x} + (1-\alpha_t)\mathbf{z}_t^\top \boldsymbol{\pi}}\right).
\end{aligned} \tag{14}$$

A.2.2 $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$

Austin et al. [1] approximate the reverse process in the following manner:

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \mathbf{x}_\theta(\mathbf{z}_t, t)) = \text{Cat}\left(\mathbf{x}_s; \frac{Q_{t|s}\mathbf{x}_t \odot Q_s^\top \mathbf{x}_\theta(\mathbf{z}_t, t)}{\mathbf{x}_t^\top Q_t^\top \mathbf{x}_\theta(\mathbf{z}_t, t)}\right). \tag{15}$$

where $\mathbf{x}_\theta(\mathbf{z}_t, t): \mathcal{V} \times [0,1] \rightarrow \Delta^K$ is an approximation for $\mathbf{x}$.

A.3 Absorbing state

For the absorbing state diffusion process we have $\boldsymbol{\pi} = \mathbf{m}$.

A.3.1 $q(\mathbf{z}_s|\mathbf{z}_t,\mathbf{x})$

Since, $\mathbf{z}_t \in \{\mathbf{x},\mathbf{m}\}$, takes only 2 values we consider the separate cases: $\mathbf{z}_t = \mathbf{x}$ and $\mathbf{z}_t = \mathbf{m}$.

Case 1. Consider the case $\mathbf{z}_t = \mathbf{x}$ i.e. $\mathbf{z}_t$ is unmasked. From (14), we have the following:

$$\begin{aligned}
&q(\mathbf{z}_s|\mathbf{z}_t = \mathbf{x},\mathbf{x}) \\
&= \mathrm{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{x} + (1-\alpha_{t|s})\mathbf{1}\mathbf{m}^\top \mathbf{x}] \odot [\alpha_s \mathbf{x} + (1-\alpha_s)\mathbf{m}]}{\alpha_t \mathbf{x}^\top \mathbf{x} + (1-\alpha_t)\mathbf{x}^\top \mathbf{m}}\right) \\
&= \mathrm{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{x}] \odot [\alpha_s \mathbf{x} + (1-\alpha_s)\mathbf{m}]}{\alpha_t}\right) && \text{since } \mathbf{x}^\top \mathbf{m} = 0 \\
&= \mathrm{Cat}\left(\mathbf{z}_s; \frac{\alpha_t \mathbf{x}}{\alpha_t}\right) && \text{since } \mathbf{x}^\top \mathbf{m} = 0 \text{ and } \alpha_t = \alpha_{t|s}\alpha_s \\
&= \mathrm{Cat}(\mathbf{z}_s; \mathbf{x}) && \text{since } \alpha_t = \alpha_{t|s}\alpha_s
\end{aligned} \tag{16}$$

Thus, we have the following:

$$q(\mathbf{z}_s|\mathbf{z}_t = \mathbf{x},\mathbf{x}) = \mathrm{Cat}(\mathbf{z}_s; \mathbf{x}). \tag{17}$$

Case 2. Consider the case $\mathbf{z}_t = \mathbf{m}$. By substituting $\mathbf{z}_t = \mathbf{m}$ and $\boldsymbol{\pi} = \mathbf{m}$ in (14), $q(\mathbf{z}_s|\mathbf{z}_t,\mathbf{x})$ simplifies to the following:

$$\begin{aligned}
q(\mathbf{z}_s|\mathbf{z}_t = \mathbf{m},\mathbf{x}) &= \mathrm{Cat}\left(\frac{(\alpha_{t|s}\mathbf{m} + (1-\alpha_{t|s})\mathbf{1}) \odot (\alpha_s \mathbf{x} + (1-\alpha_s)\mathbf{m})}{(1-\alpha_t)}\right) \\
&= \mathrm{Cat}\left(\frac{(\alpha_{t|s}(1-\alpha_s)\mathbf{m} + (1-\alpha_{t|s})(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x})}{(1-\alpha_t)}\right) \\
&= \mathrm{Cat}\left(\mathbf{z}_s; \frac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}}{1-\alpha_t}\right)
\end{aligned} \tag{18}$$

Note that the above categorical distribution is non-zero for $\mathbf{z}_s \in \{\mathbf{x},\mathbf{m}\}$ and zero for every other value. The non-zero values are specified as follows:

$$q(\mathbf{z}_s = \mathbf{x}|\mathbf{z}_t = \mathbf{m},\mathbf{x}) = \frac{\alpha_s - \alpha_t}{1-\alpha_t} \tag{19}$$

$$q(\mathbf{z}_s = \mathbf{m}|\mathbf{z}_t = \mathbf{m},\mathbf{x}) = \frac{1-\alpha_s}{1-\alpha_t} \tag{20}$$

A.3.2 $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$

For the absorbing state diffusion process with $\boldsymbol{\pi} = \mathbf{m}$, we want to simplify the (15). For this reason, we consider 2 cases: first, when $\mathbf{z}_t \neq \mathbf{m}$ (**case 1**), second, when $\mathbf{z}_t \neq \mathbf{m}$ (**case 2**).

Case 1. Consider the case when $\mathbf{z}_t \neq \mathbf{m}$. (15) simplifies to the following:

$$\begin{aligned}
p_\theta(\mathbf{z}_s|\mathbf{z}_t \neq \mathbf{m}) &= \mathrm{Cat}\left(\mathbf{x}_s; \frac{Q_{t|s}\mathbf{z}_t \odot Q_s^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}{\mathbf{z}_t^\top Q_t^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right) && (21)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{Q_{t|s}\mathbf{z}_t \odot Q_s^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}{[Q_t \mathbf{z}_t]^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{z}_t] \odot [\alpha_s \mathbf{I}_n + (1-\alpha_s)\mathbf{m}\mathbf{1}^\top]\mathbf{x}_\theta(\mathbf{z}_t,t)}{[\alpha_t \mathbf{z}_t]^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{z}_t] \odot [\alpha_s \mathbf{x}_\theta(\mathbf{z}_t,t) + (1-\alpha_s)\mathbf{m}\langle \mathbf{1}, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle]}{\alpha_t \langle \mathbf{z}_t, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle}\right)\\
&\text{since } \langle \mathbf{1}, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle = 1 \text{, we have the following:}\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{z}_t] \odot [\alpha_s \mathbf{x}_\theta(\mathbf{z}_t,t) + (1-\alpha_s)\mathbf{m}]}{\alpha_t \langle \mathbf{z}_t, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle}\right)\\
&\text{since } \mathbf{z}_t \odot \mathbf{m} = \mathbf{0} \text{, we have the following:}\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{\alpha_t \mathbf{z}_t \odot \mathbf{x}_\theta(\mathbf{z}_t,t)}{\alpha_t \langle \mathbf{z}_t, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle}\right) && (22)
\end{aligned}$$

Case 2. Consider the case when $\mathbf{z}_t = \mathbf{m}$. (15) simplifies to the following:

$$\begin{aligned}
p_\theta(\mathbf{x}_s|\mathbf{z}_t = \mathbf{m}) &= \mathrm{Cat}\left(\mathbf{x}_s; \frac{Q_{t|s}\mathbf{m} \odot Q_s^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}{\mathbf{m}^\top Q_t \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{Q_{t|s}\mathbf{m} \odot Q_s^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}{[Q_t^\top \mathbf{m}]^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{m} + (1-\alpha_{t|s})\mathbf{1}] \odot [\alpha_s \mathbf{I}_n + (1-\alpha_s)\mathbf{m}\mathbf{1}^\top]\mathbf{x}_\theta(\mathbf{z}_t,t)}{[\alpha_t \mathbf{m} + (1-\alpha_t)\mathbf{1}]^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{m} + (1-\alpha_{t|s})\mathbf{1}] \odot [\alpha_s \mathbf{x}_\theta(\mathbf{z}_t,t) + (1-\alpha_s)\mathbf{m}\langle \mathbf{1}, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle]}{\alpha_t \langle \mathbf{m}, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle + (1-\alpha_t)\langle \mathbf{1}, \mathbf{x}_\theta(\mathbf{z}_t,t)\rangle}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{[\alpha_{t|s}\mathbf{m} + (1-\alpha_{t|s})\mathbf{1}] \odot [\alpha_s \mathbf{x}_\theta(\mathbf{z}_t,t) + (1-\alpha_s)\mathbf{m}]}{\alpha_t \langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{m}\rangle + (1-\alpha_t)}\right)\\
&= \mathrm{Cat}\left(\mathbf{x}_s; \frac{\alpha_t \mathbf{m} \odot \mathbf{x}_\theta(\mathbf{z}_t,t) + (\alpha_s - \alpha_t)\mathbf{x}_\theta(\mathbf{z}_t,t) + (1-\alpha_s)\mathbf{m}}{\alpha_t \langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{m}\rangle + (1-\alpha_t)}\right) && (23)
\end{aligned}$$

Note that the above categorical distribution, we can obtain the values for $p_\theta(\mathbf{x}_s = \mathbf{x}|\mathbf{x}_t = \mathbf{m})$ and $p_\theta(\mathbf{x}_s = \mathbf{m}|\mathbf{x}_t = \mathbf{m})$ which are as follows:

$$p_\theta(\mathbf{x}_s = \mathbf{x}|\mathbf{x}_t = \mathbf{m}) = \frac{(\alpha_s - \alpha_t)\langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{x}\rangle}{\alpha_t \langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{m}\rangle + (1-\alpha_t)} \tag{24}$$

$$p_\theta(\mathbf{x}_s = \mathbf{m}|\mathbf{x}_t = \mathbf{m}) = \frac{\alpha_s \langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{m}\rangle + (1-\alpha_s)}{\alpha_t \langle \mathbf{x}_\theta(\mathbf{z}_t,t), \mathbf{m}\rangle + (1-\alpha_t)} \tag{25}$$

As a sanity check, we can verify that (24) reduces to (19), and (25) reduces to (20) if our denoising network can reconstruct $\mathbf{x}$ perfectly, i.e., $\mathbf{x}_\theta(\mathbf{z}_t,t) = \mathbf{x}$.

A.3.3 Diffusion Loss

For a given T, Let $\mathcal{L}_T = \mathbb{E}_{t \in \{1,\dots,T\}} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})} T \mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}) \| p_\theta(\mathbf{x}_s|\mathbf{x}_t))$ denote the diffusion loss. We break down the computation of $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}) \| p_\theta(\mathbf{x}_s|\mathbf{x}_t))$ into 2 cases: $\mathbf{z}_t = \mathbf{x}$ (**case 1**) and $\mathbf{z}_t = \mathbf{m}$ (**case 2**).

Case 1. consider the case $\mathbf{z}_t = \mathbf{x}$. Let's simplify $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_s|\mathbf{z}_t = \mathbf{x},\mathbf{x}) \| p_\theta(\mathbf{z}_s|\mathbf{z}_t = \mathbf{x}))$.

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_s|\mathbf{z}_t=\mathbf{x},\mathbf{x})\|p_\theta(\mathbf{z}_s|\mathbf{z}_t=\mathbf{x}))\\
&=\sum_{\mathbf{z}_s} q(\mathbf{z}_s|\mathbf{z}_t=\mathbf{x},\mathbf{x})\log\frac{q(\mathbf{z}_s|\mathbf{z}_t=\mathbf{x},\mathbf{x})}{p_\theta(\mathbf{z}_s|\mathbf{z}_t=\mathbf{x})}\\
&\text{Since } q(\mathbf{z}_s|\mathbf{z}_t,\mathbf{x}) \text{ is 1 only for } \mathbf{z}_s=\mathbf{x} \text{ we get,}\\
&=\log\frac{1}{p_\theta(\mathbf{z}_s=\mathbf{x}|\mathbf{z}_t=\mathbf{x})}\\
&=\log 1 && \text{From (21)}\\
&=0
\end{aligned}
\tag{26}
$$

Case 2. Consider the case $\mathbf{z}_t=\mathbf{m}$. Let's simplify $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))$.

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))\\
&=\sum_{\mathbf{x}_s} q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\log\frac{q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})}{p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m})}\\
&=\sum_{\mathbf{x}_s\in\{\mathbf{x},\mathbf{m}\}} q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\log\frac{q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})}{p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m})}\\
&=\underbrace{q(\mathbf{x}_s=\mathbf{x}|\mathbf{x}_t=\mathbf{m},\mathbf{x})\log\frac{q(\mathbf{x}_s=\mathbf{x}|\mathbf{x}_t=\mathbf{m},\mathbf{x})}{p_\theta(\mathbf{x}_s=\mathbf{x}|\mathbf{x}_t=\mathbf{m})}}_{\text{Simplify using (19) and (24)}}\\
&+\underbrace{q(\mathbf{x}_s=\mathbf{m}|\mathbf{x}_t=\mathbf{m},\mathbf{x})\log\frac{q(\mathbf{x}_s=\mathbf{m}|\mathbf{x}_t=\mathbf{m},\mathbf{x})}{p_\theta(\mathbf{x}_s=\mathbf{m}|\mathbf{x}_t=\mathbf{m})}}_{\text{Simplify using (20) and (25)}}\\
&=\frac{\alpha_s-\alpha_t}{1-\alpha_t}\log\frac{\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t)}{(1-\alpha_t)\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle}\\
&+\frac{1-\alpha_s}{1-\alpha_t}\log\frac{(1-\alpha_s)(\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t))}{(1-\alpha_t)(\alpha_s\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_s))}
\end{aligned}
\tag{27}
$$

Thus, $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t))$ can be written in the following manner where $\langle\mathbf{z}_t,\mathbf{x}\rangle$ evaluates to 1 if $\mathbf{z}_t=\mathbf{x}$ and $\langle\mathbf{z}_t,\mathbf{m}\rangle$ evaluates to 1 if $\mathbf{z}_t=\mathbf{m}$:

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t))\\
&=\underbrace{\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{x},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{x}))}_{=0\text{, from (26)}}\langle\mathbf{z}_t,\mathbf{x}\rangle+\underbrace{\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))}_{\text{Given by (27)}}\langle\mathbf{z}_t,\mathbf{m}\rangle
\end{aligned}
\tag{28}
$$

Thus, we derive the diffusion loss, $\mathcal{L}_T$, in the following manner:

$$
\begin{aligned}
\mathcal{L}_T&=\mathbb{E}_{t\in\{1,\dots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}T\,\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t))\\
&=\mathbb{E}_{t\in\{1,\dots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}T\Bigg[\frac{\alpha_s-\alpha_t}{1-\alpha_t}\log\frac{\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t)}{(1-\alpha_t)\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle}\\
&\qquad+\frac{1-\alpha_s}{1-\alpha_t}\log\frac{(1-\alpha_s)(\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t))}{(1-\alpha_t)(\alpha_s\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_s))}\Bigg]\langle\mathbf{z}_t,\mathbf{m}\rangle
\end{aligned}
\tag{29}
$$

Note that $\mathcal{L}_T$ is 0 if $\mathbf{z}_t$ is an unmasked token i.e. $\mathbf{z}_t=\mathbf{x}$.

B MDLM: Rao-Blackwelization using SUBS parameterization

In this section we show how SUBS parameterization can simplify the functional form of the ELBO as defined in (29).

B.1 ELBO

The SUBS parameterization, as described in Sec. 3.2, simplifies $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))$ ((27)) to the following:

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))\\
&=\frac{\alpha_s-\alpha_t}{1-\alpha_t}\log\frac{\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t)}{(1-\alpha_t)\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle}\\
&+\frac{1-\alpha_s}{1-\alpha_t}\log\frac{(1-\alpha_s)(\alpha_t\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_t))}{(1-\alpha_t)(\alpha_s\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle+(1-\alpha_s))}\\
&\text{Since SUBS sets } \langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{m}\rangle=0\text{, the above equation simplifies to the following:}\\
&=\frac{\alpha_s-\alpha_t}{1-\alpha_t}\log\frac{(1-\alpha_t)}{(1-\alpha_t)\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle}\\
&=\frac{\alpha_t-\alpha_s}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle
\end{aligned} \tag{30}$$

Using this, we obtain the following expression for the diffusion loss, $\mathcal{L}_T$:

$$\begin{aligned}
\mathcal{L}_T&=T\mathbb{E}_{t\in\{1,\ldots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}\mathrm{D}_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m},\mathbf{x})\|p_\theta(\mathbf{x}_s|\mathbf{x}_t=\mathbf{m}))\langle\mathbf{z}_t,\mathbf{m}\rangle\\
&=T\mathbb{E}_{t\in\{1,\ldots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}\frac{\alpha_t-\alpha_s}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle\langle\mathbf{z}_t,\mathbf{m}\rangle\\
&\text{When } \mathbf{z}_t=\mathbf{m},\ \log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle=0\text{; hence, the term } \langle\mathbf{z}_t,\mathbf{m}\rangle \text{ can be safely dropped to obtain:}\\
&=T\mathbb{E}_{t\in\{1,\ldots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}\frac{\alpha_t-\alpha_s}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle
\end{aligned} \tag{31}$$

B.2 Continous Time ELBO

To derive the continuous-time diffusion loss, $\mathcal{L}^{\infty}_{\text{diffusion}}$, we consider the limiting case $\lim_{T\to\infty}\mathcal{L}_T$:

$$\begin{aligned}
\mathcal{L}^{\infty}_{\text{diffusion}}&=\lim_{T\to\infty}\mathcal{L}_T\\
&=\mathbb{E}_{t\in\{1,\ldots,T\}}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}\left[\lim_{T\to\infty}T\frac{\alpha_t-\alpha_s}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle\right]\\
&\text{Using } \lim_{T\to\infty}T(\alpha_s-\alpha_t)=\alpha'_t\text{, we obtain:}\\
&=\mathbb{E}_{t\sim[0,1]}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})}\left[\frac{\alpha'_t}{1-\alpha_t}\log\langle\mathbf{x}_\theta(\mathbf{z}_t,t),\mathbf{x}\rangle\right]
\end{aligned} \tag{32}$$

C Additional Experiments

C.1 Experimental Setup

We evaluate MDLM as a generative model of language and as a representation model via fine-tuning on downstream tasks.

For language modeling likelihood evaluation, we conduct experiments on two datasets: The One Billion Words Dataset (LM1B; [4]) and OpenWebText (OWT; [11]). We use the `bert-base-uncased` tokenizer for One Billion Words, and report perplexities on the test split. Models have a context size of 128. For OWT, which does not have a pre-defined split, we reserve the last 100K documents as a held-out validation set and report perplexities on this set. We use the `GPT2` tokenizer [31] for OWT. Models have a context size of 1,024. We utilize the transformer architecture from Lou et al. [21], which augments the diffusion transformer [28] with rotary embeddings [38]. MDLM was trained for 1M or 10M steps (corresponding to 33B, 330B tokens, respectively) on LM1B and 1M steps on OWT (which corresponds to 262B tokens). The corresponding AR baseline was trained for half the number of steps to ensure similar number of tokens seen (details in Suppl. F). Full hyperparameters are given in Suppl. I.1. On OWT, we train with and without time step conditioning.

For representation learning, we pre-train models on the C4 dataset [32], then fine-tune and evaluate models on the GLUE benchmark [42]. Models have a context size of 128. We use the `bert-base-uncased` tokenizer for the representation learning experiments. We utilize the MosaicBERT architecture from Portes et al. [29], an extension of the original BERT architecture [9]. We pre-train a bidirectional MosaicBERT using an MLM objective for 37B tokens of C4, as well as a causal variant on the same data. We further fine-tune MosaicBERT model using the MDLM for 327M tokens, less than 1% of the pre-training data. We provide the full hyperparameters in Suppl. I.3.

C.2 LM1B perplexity

Table 4: Test perplexities (PPL; ↓) on LM1B. †Reported in He et al. [17]. Best diffusion value is bolded.

		Parameters	PPL (↓)
Autoregressive	Transformer-X Base [8]	0.46B	23.5
	OmniNet$_T$ [40]	100M	21.5
Diffusion	BERT-Mouth [41]	110M	$\leq$142.89
	D3PM (absorb) [1]	70M	$\leq$77.50
	Diffusion-LM [20]†	80M	$\leq$118.62
	DiffusionBert [17]	110M	$\leq$63.78
	SEDD [21] (33B tokens)	110M	$\leq$ 32.79
Autoregressive (Retrained)	Transformer (33B tokens)	110M	22.32
	Transformer (330B tokens)		20.86
Diffusion (Ours)	MDLM (33B tokens)	110M	$\leq$27.04
	MDLM (330B tokens)		$\leq$**23.00**

C.3 LM1B ablations

We assess the importance of our continuous-time framework by performing ablation on diffusion steps T. In Table 5, we compare NLL and PPL under continuous and discrete T in MDLM. We find that NLL consistently decreases as $T \to \infty$.

Table 5: Discrete vs continuous time evaluation for MDLM on LM1B. MDLM was trained with $T = \infty$ and a smaller model containing 70M non-embedding parameters for 200K steps. We report test perplexity for a discrete T.

Method	NLL	PPL
MDLM$_{T=\infty}$	$\leq \mathbf{3.61 \pm 0.001}$	$\leq$**37.25**
MDLM$_{T=10}$	$\leq 4.14 \pm 0.003$	$\leq$62.83
MDLM$_{T=100}$	$\leq 3.66 \pm 0.002$	$\leq$39.04
MDLM$_{T=1000}$	$\leq 3.62 \pm 0.000$	$\leq$37.38

C.4 Train NLL curves on OWT

In Figure 2, we show that MDLM achieves lower variance loss during training compared to a previous diffusion language model, SEDD. Training is performed over 1M steps on OWT (which corresponds to 524B tokens).

C.5 Time-conditioning ablation on OWT

In Table 6, we assess the importance of time conditioning in MDLM on OWT. We observe that time-conditioning has minimal impact on perplexity. Training is performed over 1M steps on OWT (which corresponds to 524B tokens).

Figure 2: Train negative log-likelihood (NLL) curves across 1M gradient steps (524B tokens) on OpenWebText [11]. NLL is logged every 1K steps without value smoothing.

Table 6: Ablation on time-conditioning in MDLM on OWT.

Method	PPL
MDLM w/ time-conditioning	23.21
MDLM w/o time-conditioning	**23.05**

C.6 Zero shot evaluations

We also explore models' ability to generalize by taking models trained on OWT and evaluating how well they model unseen datasets. We compare the perplexities of our MDLM with a SEDD parameterization and an AR Transformer language model. Our zero-shot datasets include the validation splits of Penn Tree Bank (PTB; [24]), Wikitext [25], LM1B, Lambada [27], AG News [43], and Scientific Papers (Pubmed and Arxiv subsets; [6]). Full experimental details are available in Suppl. I.1.

MDLM consistently outperforms the SEDD diffusion parameterization. In some cases, e.g., for Lambada and Scientific Papers, MDLM attains better perplexity than AR. We hypothesize that these datasets are farther from OWT, and that diffusion models may be more robust to out-of-domain evaluation due to the unmasking-based objective.

Table 7: Zero-shot validation perplexities (↓) of models trained for 524B tokens on OWT. All perplexities for diffusion models are upper bounds.

	PTB	Wikitext	LM1B	Lambada	AG News	Pubmed	Arxiv
AR (Retrained)	**82.05**	**25.75**	**51.25**	51.28	**52.09**	49.01	41.73
SEDD (Retrained)	100.09	34.28	68.20	49.86	62.09	44.53	38.48
MDLM (Ours)	95.26	32.83	67.01	**47.52**	61.15	**41.89**	**37.37**

Table 8: GLUE evaluation results. Evaluation measures (↑) are F1 score for QQP and MRPC, Spearman correlations for STS-B, and accuracy for the rest. For MNLI, we report match/mismatch accuracies.

	MNLI (m/mm)	QQP	QNLI	SST-2	COLA	STS-B	MRPC	RTE	Avg
AR	80.94/80.78	86.98	86.16	90.14	33.43	84.32	83.88	47.29	74.88
BERT	84.43/85.35	88.41	**90.46**	**92.20**	54.81	**88.41**	89.16	61.37	81.62
+MDLM-FT	**84.76/85.07**	**88.49**	90.30	**92.20**	**57.69**	87.48	**90.53**	**62.09**	**82.06**

Table 9: Test perplexities (PPL; ↓) on OWT for models trained for 262B tokens. † denotes retrained models.

	PPL (↓)
AR†	17.54
SEDD†	≤24.10
MDLM (Ours)	≤**23.21**

Table 10: Test perplexities (PPL; ↓) for MDLM ablations on LM1B. All the models were trained for 200K steps. Standard deviation is measured over 5 seeds during evaluation.

	PPL
MDLM	$\mathbf{33.59 \pm .11}$
w/o Continuous time	$33.70 \pm .07$
& carry-over	$35.57 \pm .15$
& zero masking	$35.31 \pm .16$

C.7 Glue Evaluation

C.8 OWT perplexity

C.9 Ablation Analysis

In Table 10, we can see the effect of our streamlined masked diffusion implementation. The improvements described in Sec. **??** allow us to greatly reduce perplexity of previously discounted models, such as D3PM (see the bottom row of this table, which is mathematically equivalent to the D3PM formulation). While most works assumed that D3PM achieves mediocre log-likelihoods, we show that is is incorrect: our re-implementation almost matches state-of-the-art score-based methods. This introduces a new strong baseline that opens new research opportunities. Additionally, in Table 10, we ablate different components of MDLM. We observe that the perplexity for MDLM trained with a discrete $T = 1000$ marginally worsens by 0.1 compared to MDLM trained in continuous time. Additionally, removing the "carry over" operation from the SUBS parameterization increases the perplexity by 2 points. However, further removing the "zero masking" operation does not lead to any meaningful change in perplexity.

We provide further ablations for the continuous time formulation in the Appendix, showing in Table 5 that for a pre-trained model, at inference, increasing T yields better likelihoods.

C.10 SEMI-AR

To test the SAR decoding algorithm presented in Sec. 4.2, we compare to SSD-LM [16] a diffusion model that was designed to generate blocks of text autoregressively. We generate 200 sequences of length 2048 tokens on a single 3090 GPU and evaluate generative perplexity under a pre-trained GPT-2 [31] model. The SSD-LM sequences are generated using blocks of 25 tokens (as implemented in their pre-trained model) and the MDLM sequences are generated using $L' = 512$. In Table 11, we find that in addition to achieving better generative perplexity, MDLM enables ~25-30x faster SAR decoding relative to SSD-LM.

Table 11: Semi-AR generative perplexity (Gen. PPL; $\downarrow$) for sequences of 2048 tokens.

	Gen. PPL ($\downarrow$)	Sec/Seq ($\downarrow$)
SSD-LM	35.43	2473.9
MDLM (Ours)	**27.18**	**89.3**

C.11 Generative Performance

Table 12: Test perplexities (PPL; $\downarrow$) of generative fine-tuning of the Caduceus MLM [33] on the HG38 reference genome. Best diffusion model values are bolded. Error bars indicate the difference between the maximum and minimum values across 5 random seeds used for fine-tuning. † denotes retrained models.

		Params	PPL ($\downarrow$)
AR†	Mamba	465K	$3.067 \pm .0104$
	HyenaDNA	433K	$3.153 \pm .001$
Dif†	Plaid	507K	$\leq 3.240 \pm .005$
	SEDD	467K	$\leq 3.216 \pm .003$
Dif (Ours)	MDLM	467K	$\leq \mathbf{3.199} \pm .010$

D Noise schedule parameterization

As described in Sec. 3.4, the ELBO is invariant to the functional form of α_t. To demonstrate this, we evaluate MDLM, initially trained using a log-linear schedule on OWT, by replacing the noise schedule with various other noise schedules as mentioned below. Following prior works [1, 21, 34], we parameterize $\alpha_t = e^{-\sigma(t)}$, where $\sigma(t):[0,1] \to \mathbb{R}^+$. Various functional forms of $\sigma(t)$ are listed below:

Log Linear [1, 21, 34] The log linear schedule is given as:

$$\sigma(t) = -\log t \tag{33}$$

Cosine Squared schedule [16] The Cosine Squared schedule is given as:

$$\sigma(t) = -\log \cos^2\left(\frac{\pi}{2}(1-t)\right) \tag{34}$$

Cosine schedule The Cosine schedule is given as:

$$\sigma(t) = -\log \cos^2\left(\frac{\pi}{2}(1-t)\right) \tag{35}$$

Linear The Linear schedule is given as:

$$\sigma(t) = \sigma_{\max}(1-t) \tag{36}$$

where $\sigma_{\max}$ is a very large number. In our experiments we set it to 10^8.

In Table 13 we demonstrate empirically that noise schedules with different functional forms evaluate to the same Likelihood which is consistent with our theory in Sec. 3.4. However, different schedules lead to different per data point variance.

E Likelihood Evaluation

How you do it Say that it incurs lower variance by referencing to the Ablattions table The variance is low because of the low discrepancy sampler

Table 13: Likelihood in bits per dimension (BPD) for different noise schedules on OWT dataset, is reported along with the mean and variance associated with each noise schedule per data point. We empirically observe that noise schedules with different functional forms yield the same likelihood, consistent with our theory in Sec. 3.4; however, different schedules result in different variances. Notably, the log-linear schedule exhibits the lowest variance among all the noise schedules considered.

$\sigma(t)$	Mean	Variance per datapoint
Log Linear (33)	3.30	1.81
Cosine (35)	3.30	3.30
Cosine Squared (34)	3.30	3.30
Linear (36)	3.30	7.57

F Avg. Number of Tokens seen

Given `training_steps`, `batch_size`, `context_length`, the number of tokens seen by the AR model is given as:

$$\texttt{training\_steps} \times \texttt{batch\_size} \times \texttt{context\_length}.$$

However, this expression doesn't hold true for a diffusion model, since at each training step, the model sees masked input. Let p_m be the probability of a token being masked at a timestep t. Then the diffusion model sees the following number of tokens in expection:

$$\begin{aligned}
&\mathbb{E}_t[\texttt{training_steps} \times \texttt{batch_size} \times \texttt{context_length} \times p_m] \\
&= \texttt{training_steps} \times \texttt{batch_size} \times \texttt{context_length} \times \mathbb{E}_t[p_m] \\
&\text{For log-linear schedule used in our experiments } p_m = t\text{; thus,} \\
&= \texttt{training_steps} \times \texttt{batch_size} \times \texttt{context_length} \times 0.5
\end{aligned} \tag{37}$$

G Low discrepancy sampler

To reduce variance during training we use a low-discrepancy sampler, similar to that proposed in Kingma et al. [19]. Specifically, when processing a minibatch of N samples, instead of independently sampling N from a uniform distribution, we partition the unit interval and sample the time step for each sequence $i \in \{1,...,N\}$ from a different portion of the interval $t_i \sim U[\frac{i-1}{N}, \frac{i}{N}]$. This ensures that our sampled timesteps are more evenly spaced across the interval $[0,1]$, reducing the variance of the ELBO.

H Faster sampling with caching

In Figure 14 we compare the wall clock times of variaous methods: AR, SEDD, MDLM with caching, and MDLM without caching for generating 64 samples on a single GPU. We observe that MDLM without caching yields samples that consistently get better generative perplexity than SEDD. For $T = \{5k, 10k\}$, both SEDD and MDLM get better generative perplexity than the AR model.

Table 14: Wall clock time reported in seconds.

	$T=5k$	$T=10k$
MDLM	4215.9	7675.4
+ caching	2407.3	3626.6
Speedup	1.75x	2.12x

I Experimental details

I.1 Language Modeling

For our forward noise process, we use a log-linear noise schedule similar to Lou et al. [21].

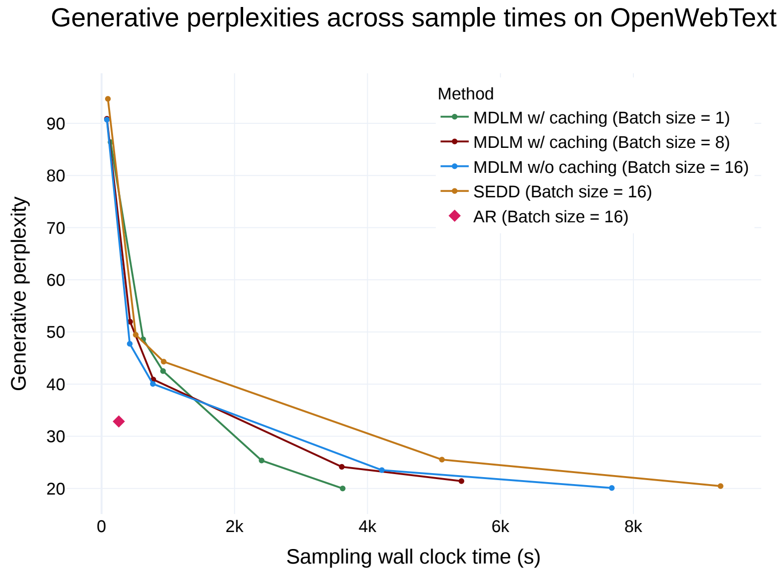

Figure 3: Generative perplexities across wall clock time for generating 64 samples on OWT using a single 32GB A5000 GPU are compared by varying $T \in \{100, 500, 1000, 5000, 10000\}$ in the reverse diffusion process. The samples are generated in mini-batches with a batch size of 16 for AR, SEDD, and MDLM without caching, as it is the largest batch size that fits on this GPU. For MDLM with caching, we vary the batch size.

We detokenize the One Billion Words dataset following Lou et al. [21], whose code can be found here. We tokenize the One Billion Words dataset with the `bert-base-uncased` tokenizer, following He et al. [17]. We pad and truncate sequences to a length of 128.

We tokenize OpenWebText with the `GPT2` tokenizer. We do not pad or truncate sequences – we concatenate and wrap them to a length of 1,024. When wrapping, we add the `eos` token in-between concatenated. We additionally set the first and last token of every batch to be `eos`. Since OpenWebText does not have a validation split, we leave the last 100k docs as validation.

We parameterize our autoregressive baselines, SEDD, and MDLM with the transformer architecture from Lou et al. [21]. We use 12 layers, a hidden dimension of 768, 12 attention heads, and a timestep embedding of 128 when applicable. Word embeddings are not tied between the input and output.

We use the AdamW optimizer with a batch size of 512, constant learning rate warmup from 0 to a learning rate of 3e-4 for 2,500 steps. We use a constant learning rate for 1M, 5M, or 10M steps on One Billion Words, and 1M steps for OpenWebText. We use a dropout rate of 0.1.

I.2 Zeroshot Likelihood

We evaluate zeroshot likelihoods by taking the models trained on OpenWebText and evaluating likelihoods on the validation splits of 7 datasets: Penn Tree Bank (PTB; Marcus et al. [24]), Wikitext [25], One Billion Word Language Model Benchmark (LM1B; Chelba et al. [4]), Lambada [27], AG News [43], and Scientific Papers (Pubmed and Arxiv subsets; Cohan et al. [6]). We detokenize the datasets following Lou et al. [21]. For the AG News and Scientific Papers (Pubmed and Arxiv), we apply both the Wikitext and One Billion Words detokenizers. Since the zeroshot datasets have different conventions for sequence segmentation, we wrap sequences to 1024 and do not add `eos` tokens in between sequences.

I.3 Representation Learning

Following Devlin et al. [9], we evaluate on all GLUE tasks [42], but exclude WNLI.

We pre-train a MosaicBERT model on C4 [32] for 70k steps, corresponding to 36B tokens. We pad and truncate the data to 128 tokens using the `bert-base-uncased` tokenizer.

MosaicBERT [29] has a similar architecture to `bert-base-uncased` and has 137M parameters, 12 layers, 12 attention heads, a hidden dimension of 768, an intermediate size of 3072, and ALiBi attention bias [30].

For pre-training, we use the following hyperparameters: A global batch size of 4096 with gradient accumulation, a learning rate of 5e-4, linear decay to 0.02x of the learning rate with a warmup of 0.06x of the full training duration, and the decoupled AdamW optimizer with 1e-5 weight decay and betas 0.9 and 0.98.

For diffusion fine-tuning we use AdamW with a warmup of 2,500 steps from a learning rate of 0 to 5e-5, betas 0.95 and 0.999, and batch size 512. We train for 5k steps total, corresponding to 32M tokens.

For GLUE evaluation, we use the HuggingFace script found here. We use the default parameters for all datasets, except for a batch size of 16, which we found helped with smaller datasets. This includes the default of 3 epochs for all datasets and learning rate of 2e-5.

I.4 Diffusion DNA Models

Dataset We pre-train the Caduceus MLM [33] on the HG38 human reference genome [7]. Following Schiff et al. [33], we use character- / base pair-level tokenization. The dataset is based on the splits used in Avsec et al. [2]: the training split comprises of 35 billion tokens covering the human genome. This consists of 34,021 segments extended to a maximum length of 1,048,576 (220 segments). We maintain a constant 2^{20} tokens per batch. For the Genomics Benchmark tasks, we use 5-fold cross-validation where we split the training set into 90/10 train/validation splits.

Architecture The Caduceus MLM uses as a backbone a bi-directional variant of the data-dependent SSM Mamba block proposed in Gu et al. [14]. This architecture is ideal as it contains inductive biases that preserve reverse complement (RC) equviariance, respecting the inherent symmetry of double-stranded DNA molecules [23, 33, 44].

Training details All models are pre-trained on 10B tokens (10K steps) and fine-tuned on a generative objective for an additional 50B tokens (50K steps). We use a global batch size of 1024 for a context length of 1024 tokens. Downstream task fine-tuning is performed for 16K steps (1B tokens).

For performing Caduceus MLM pre-training, we follow Schiff et al. [33] for the model size configuration, and hyperparameter selection. For pre-training, we use a fixed 15% mask rate as done in Devlin et al. [9]. Of the 'masked' tokens, 80% are replaced with [MASK] , 10% are replaced with a random token from the vocabulary, and 10% are left unchanged.

For fine-tuning all Mamba-based models (including Caduceus) on diffusion objectives, we lower the learning rate from 8e-3 to 1e-3. For fine-tuning HyenaDNA [26], we lower the learning rate from 6e-4 to 5e-5. Similar to Gu et al. [14], Schiff et al. [33], we found that Mamba-based models were robust to higher learning rates. We exclude timestep embeddings for all Diffusion DNA experiments, as we show it has minimal impact on generative performance (see Table 6, Suppl. C.5).

We perform downstream task fine-tuning on the final hidden state embedding from pre-training. We perform mean pooling across the sequence length, which may vary from 200 to approximately 2,000 bps. We report the mean and $\pm$ on max/min classification accuracy over 5-fold cross-validation (CV) using different random seeds, with early stopping on validation accuracy. For each task, we do a hyperparameter sweep over batch size and learning rate and report the values of the 5-fold CV for the best configuration.

Genomic Benchmark Task Distributions We use a subset of the Genomic Benchmark tasks with an emphasis on tasks from Human data. The positive samples for each dataset were generated by selecting samples that were annotated, either computationally or experimentally, in previous work (e.g enhancers, promoters, open chromatin regions (OCR)) [12]. These annotations each correspond to subsets of the genome of varying sizes that may exhibit different distributions of DNA than those observed globally over the reference genome. Due to this, the observed dataset may have a different distribution than the data used for pre-training and calculating perplexity. This might in turn lead to a case where perplexity and downstream performance may not necessarily correlate.