BLOCK DIFFUSION: INTERPOLATING BETWEEN AUTOREGRESSIVE AND DIFFUSION LANGUAGE MODELS

Marianne Arriola[†] [*] **Subham Sekhar Sahoo**[†] **Aaron Gokaslan**[†] **Zhihan Yang**[†]

Zhixuan Qi[†] **Jiaqi Han**[¶] **Justin T Chiu**[‡] **Volodymyr Kuleshov**[†]

ABSTRACT

Diffusion language models offer unique benefits over autoregressive models due to their potential for parallelized generation and controllability, yet they lag in likelihood modeling and are limited to fixed-length generation. In this work, we introduce a class of block diffusion language models that interpolate between discrete denoising diffusion and autoregressive models. Block diffusion overcomes key limitations of both approaches by supporting flexible-length generation and improving inference efficiency with KV caching and parallel token sampling. We propose a recipe for building effective block diffusion models that includes an efficient training algorithm, estimators of gradient variance, and data-driven noise schedules to minimize the variance. Block diffusion sets a new state-of-the-art performance among diffusion models on language modeling benchmarks and enables generation of arbitrary-length sequences. We provide the code[1], along with the model weights and blog post on the project page:

https://mariannearriola.github.io/bd3-lms

1 INTRODUCTION

Diffusion models are widely used to generate images (Ho et al., 2020; Dhariwal & Nichol, 2021) and videos (Ho et al., 2022; Gupta et al., 2023), and are becoming increasingly effective at generating discrete data such as text (Lou et al., 2024; Sahoo et al., 2024a) or biological sequences (Avdeyev et al., 2023; Goel et al., 2024). Compared to autoregressive models, diffusion models have the potential to accelerate generation and improve the controllability of model outputs (Schiff et al., 2024; Nisonoff et al., 2024; Li et al., 2024).

Discrete diffusion models currently face at least three limitations. First, in applications such as chat systems, models must generate output sequences of arbitrary length (e.g., a response to a user's question). However, most recent diffusion architectures only generate fixed-length vectors (Austin et al., 2021; Lou et al., 2024). Second, discrete diffusion uses bidirectional context during generation and therefore cannot reuse previous computations with KV caching, which makes inference less efficient (Israel et al., 2025). Third, the quality of discrete diffusion models, as measured by standard metrics such as perplexity, lags behind autoregressive approaches and further limits their applicability (Gulrajani & Hashimoto, 2024; Sahoo et al., 2024a).

This paper makes progress towards addressing these limitations by introducing Block Discrete Denoising Diffusion Language Models (BD3-LMs), which interpolate between discrete diffusion and autoregressive models. Specifically, block diffusion models (also known as semi-autoregressive models) define an autoregressive probability distribution over blocks of discrete random variables (Si et al., 2022; 2023); the conditional probability of a block given previous blocks is specified by a denoising discrete diffusion model (Austin et al., 2021; Sahoo et al., 2024a).

Developing effective BD3-LMs involves two challenges. First, efficiently computing the training objective for a block diffusion model is not possible using one standard forward pass of a neural

---

[*] Correspondence to Marianne Arriola: `ma2238@cornell.edu`

[†] Cornell Tech, NY, USA. [¶] Stanford University, CA, USA. [‡] Cohere, NY, USA.

[1] Code: https://github.com/kuleshov-group/bd3-lms

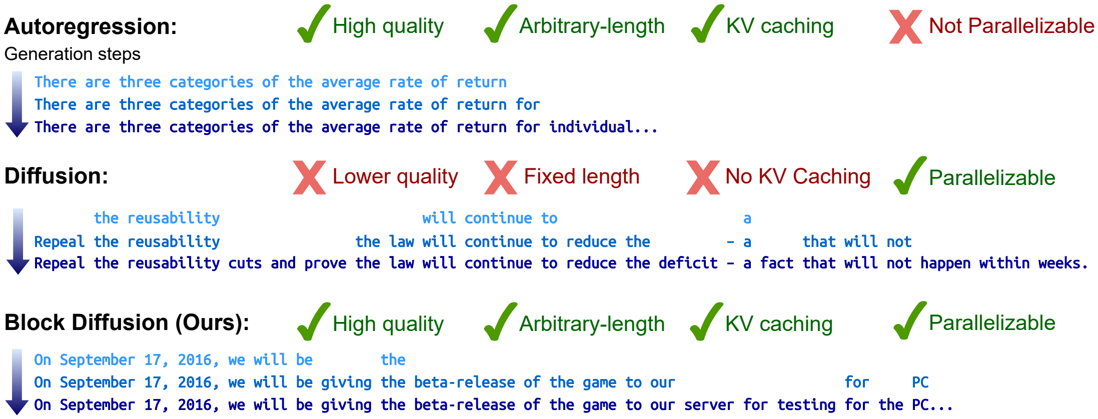
Autoregression:
✓ High quality
✓ Arbitrary-length
✓ KV caching
✗ Not Parallelizable
Generation steps
There are three categories of the average rate of return
There are three categories of the average rate of return for
There are three categories of the average rate of return for individual...
Diffusion:
✗ Lower quality
✗ Fixed length
✗ No KV Caching
✓ Parallelizable
the reusability will continue to a
Repeal the reusability the law will continue to reduce the - a that will not
Repeal the reusability cuts and prove the law will continue to reduce the deficit - a fact that will not happen within weeks.
Block Diffusion (Ours):
✓ High quality
✓ Arbitrary-length
✓ KV caching
✓ Parallelizable
On September 17, 2016, we will be the
On September 17, 2016, we will be giving the beta-release of the game to our for PC
On September 17, 2016, we will be giving the beta-release of the game to our server for testing for the PC...

Figure 1: Block diffusion models blocks of tokens autoregressively and performs diffusion within each block. By combining strength from autoregressive and diffusion models, block diffusion overcomes the limitations of both approaches by supporting variable-length, higher-quality generation and faster inference with KV caching and parallel sampling.

network and requires developing specialized algorithms. Second, training is hampered by the high variance of the gradients of the diffusion objective, causing BD3-LMs to under-perform autoregression even with a block size of one (when both models should be equivalent). We derive estimators of gradient variance, and demonstrate that it is a key contributor to the gap in perplexity between autoregression and diffusion. We then propose custom noise processes that minimize gradient variance and make progress towards closing the perplexity gap.

We evaluate BD3-LMs on language modeling benchmarks, and demonstrate that they are able to generate sequences of arbitrary length, including lengths that exceed their training context. In addition, BD3-LMs achieve new state-of-the-art perplexities among discrete diffusion models. Compared to alternative semi-autogressive formulations that perform Gaussian diffusion over embeddings (Han et al., 2022; 2023), our discrete approach features tractable likelihood estimates and yields samples with improved generative perplexity using an order of magnitude fewer generation steps. In summary, our work makes the following contributions:

- We introduce block discrete diffusion language models, which are autoregressive over blocks of tokens; conditionals over each block are based on discrete diffusion. Unlike prior diffusion models, block diffusion supports variable-length generation and KV caching.
- We introduce custom training algorithms for block-autoregressive models that enable efficiently leveraging the entire batch of tokens provided to the model.
- We identify gradient variance as a limiting factor of the performance of diffusion models, and we propose custom data-driven noise schedules that reduce gradient variance.
- Our results establish a new state-of-the-art perplexity for discrete diffusion and make progress toward closing the gap to autoregressive models.

2 BACKGROUND: LANGUAGE MODELING PARADIGMS

Notation We consider scalar discrete random variables with V categories as 'one-hot' column vectors in the space $\mathcal{V} = \{\mathbf{x} \in \{0,1\}^V : \sum_i \mathbf{x}_i = 1\} \subset \Delta^V$ for the simplex Δ^V. Let the V-th category denote a special [MASK] token, where $\mathbf{m} \in \mathcal{V}$ is its one-hot vector. We define $\mathbf{x}^{1:L}$ as a sequence of L tokens, where $\mathbf{x}^\ell \in \mathcal{V}$ for all tokens $\ell \in \{1, \ldots, L\}$, and use $\mathcal{V}^L$ to denote the set of all such sequences. Throughout the work, we simplify notation and refer to the token sequence as $\mathbf{x}$ and an individual token as $\mathbf{x}^\ell$. Finally, let $\text{Cat}(\cdot; p)$ be a categorical distribution with probability $p \in \Delta^V$.

2.1 AUTOREGRESSIVE MODELS

Consider a sequence of L tokens $\mathbf{x} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^L\right]$ drawn from the data distribution $q(\mathbf{x})$. Autoregressive (AR) models define a factorized distribution of the form

$$\log p_\theta(\mathbf{x}) = \sum_{\ell=1}^{L} \log p_\theta(\mathbf{x}^\ell \mid \mathbf{x}^{<\ell}), \tag{1}$$

where each $p_\theta(\mathbf{x}^\ell \mid \mathbf{x}^{<\ell})$ is parameterized directly with a neural network. As a result, AR models may be trained efficiently via next token prediction. However, AR models take L steps to generate L tokens due to the sequential dependencies.

2.2 DISCRETE DENOISING DIFFUSION PROBABILISTIC MODELS

Diffusion models fit a model $p_\theta(\mathbf{x})$ to reverse a forward corruption process q (Sohl-Dickstein et al., 2015; Ho et al., 2020; Sahoo et al., 2024b). This process starts with clean data $\mathbf{x}$ and defines latent variables $\mathbf{x}_t = \left[\mathbf{x}_t^1, \ldots, \mathbf{x}_t^L\right]$ for $t \in [0, 1]$, which represent progressively noisier versions of $\mathbf{x}$. Given a discretization into T steps, we define $s(j) = (j-1)/T$ and $t(j) = j/T$. For brevity, we drop j from $t(j)$ and $s(j)$ below; in general, s denotes the time step preceding t.

The D3PM framework (Austin et al., 2021) defines q as a Markov forward process acting independently on each token $\mathbf{x}^\ell$: $q(\mathbf{x}_t^\ell \mid \mathbf{x}_s^\ell) = \text{Cat}(\mathbf{x}_t^\ell; Q_t \mathbf{x}_s^\ell)$ where $Q_t \in \mathbb{R}^{V \times V}$ is the diffusion matrix. The matrix Q_t can model various transformations, including masking, random token changes, and related word substitutions.

An ideal diffusion model p_θ is the reverse of the process q. The D3PM framework defines p_θ as

$$p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) = \prod_{\ell=1}^{L} p_\theta(\mathbf{x}_s^\ell \mid \mathbf{x}_t) = \sum_{\mathbf{x}} \left[\prod_{\ell=1}^{L} q(\mathbf{x}_s^\ell \mid \mathbf{x}_t^\ell, \mathbf{x}^\ell) p_\theta(\mathbf{x}^\ell \mid \mathbf{x}_t)\right], \tag{2}$$

where the denoising base model $p_\theta(\mathbf{x}^\ell \mid \mathbf{x}_t)$ predicts clean token $\mathbf{x}^\ell$ given the noisy sequence $\mathbf{x}_t$, and the reverse posterior $q(\mathbf{x}_s^\ell \mid \mathbf{x}_t^\ell, \mathbf{x})$ is defined following Austin et al. (2021) in Suppl. B.3.

The diffusion model p_θ is trained using variational inference. Let $\text{KL}[\cdot]$ denote the Kullback–Leibler divergence. Then, the Negative ELBO (NELBO) is given by (Sohl-Dickstein et al., 2015):

$$\mathcal{L}(\mathbf{x}; \theta) = \mathbb{E}_q \left[-\log p_\theta(\mathbf{x} \mid \mathbf{x}_{t(0)}) + \sum_{j=1}^{T} D_{\text{KL}}[q(\mathbf{x}_{s(j)} \mid \mathbf{x}_{t(j)}, \mathbf{x}) \| p_\theta(\mathbf{x}_{s(j)} \mid \mathbf{x}_{t(j)})] + D_{\text{KL}}[q(\mathbf{x}_{t(T)} \mid \mathbf{x}) \| p_\theta(\mathbf{x}_{t(T)})] \right] \tag{3}$$

This formalism extends to continuous time via Markov chain (CTMC) theory and admits score-based generalizations (Song & Ermon, 2019; Lou et al., 2024; Sun et al., 2022). Further simplifications (Sahoo et al., 2024a; Shi et al., 2024; Ou et al., 2025) tighten the ELBO and enhance performance.

3 BLOCK DIFFUSION LANGUAGE MODELING

We explore a class of Block Discrete Denoising Diffusion Language Models (BD3-LMs) that interpolate between autoregressive and diffusion models by defining an autoregressive distribution over blocks of tokens and performing diffusion within each block. We provide a block diffusion objective for maximum likelihood estimation and efficient training and sampling algorithms. We show that for a block size of one, the diffusion objective suffers from high variance despite being equivalent to the autoregressive likelihood in expectation. We identify high training variance as a limitation of diffusion models and propose data-driven noise schedules that reduce the variance of the gradient updates during training.

3.1 BLOCK DIFFUSION DISTRIBUTIONS AND MODEL ARCHITECTURES

We propose to combine the language modeling paradigms in Sec. 2 by autoregressively modeling blocks of tokens and performing diffusion within each block. We group tokens in $\mathbf{x}$ into B blocks of length L' with $B = L/L'$ (we assume that B is an integer). We denote each block $\mathbf{x}^{(b-1)L':bL'}$ from token at positions $(b-1)L'$ to bL' for blocks $b \in \{1, \dots, B\}$ as $\mathbf{x}^b$ for simplicity. Our likelihood factorizes over blocks as

$$\log p_\theta(\mathbf{x}) = \sum_{b=1}^{B} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}^{<b}), \tag{4}$$

and each $p_\theta(\mathbf{x}^b \mid \mathbf{x}^{<b})$ is modeled using discrete diffusion over a block of L' tokens. Specifically, we define a reverse diffusion process as in (2), but restricted to block b:

$$p_\theta(\mathbf{x}_s^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) = \sum_{\mathbf{x}^b} q(\mathbf{x}_s^b \mid \mathbf{x}_t^b, \mathbf{x}^b) p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \tag{5}$$

We obtain a principled learning objective by applying the NELBO in (3) to each term in (4) to obtain

$$-\log p_\theta(\mathbf{x}) \leq \mathcal{L}_{\text{BD}}(\mathbf{x}; \theta) := \sum_{b=1}^{B} \mathcal{L}(\mathbf{x}^b, \mathbf{x}^{<b}; \theta), \tag{6}$$

where each $\mathcal{L}(\mathbf{x}^b, \mathbf{x}^{<b}; \theta)$ is an instance of (3) applied to $\log p_\theta(\mathbf{x}^b \mid \mathbf{x}^{<b})$. Since the model is conditioned on $\mathbf{x}^{<b}$, we make the dependence on $\mathbf{x}^{<b}, \theta$ explicit in $\mathcal{L}$. We denote the sum of these terms $\mathcal{L}_{\text{BD}}(\mathbf{x}; \theta)$ (itself a valid NELBO).

Model Architecture Crucially, we parameterize the B base denoiser models $p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b})$ using a single neural network $\mathbf{x}_\theta$. The neural network $\mathbf{x}_\theta$ outputs not only the probabilities $p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b})$, but also computational artifacts for efficient training. This will enable us to compute the loss $\mathcal{L}_{\text{BD}}(\mathbf{x}; \theta)$ in parallel for all B blocks in a memory-efficient manner. Specifically, we parameterize $\mathbf{x}_\theta$ using a transformer (Vaswani et al., 2017) with a block causal attention mask. The transformer $\mathbf{x}_\theta$ is applied to L tokens, and tokens in block b attend to tokens in blocks 1 to b. When $\mathbf{x}_\theta$ is trained, $\mathbf{x}_\theta^b(\mathbf{x}_t^b, \mathbf{x}^{<b})$ yields L' predictions for denoised tokens in block b based on noised $\mathbf{x}_t^b$ and clean $\mathbf{x}^{<b}$.

In autoregressive generation, it is normal to cache keys and values for previously generated tokens to avoid recomputing them at each step. Similarly, we use $\mathbf{K}^b, \mathbf{V}^b$ to denote the keys and values at block b, and we define $\mathbf{x}_\theta$ to support these as input and output. The full signature of $\mathbf{x}_\theta$ is

$$\mathbf{x}_{\text{logits}}^b, \mathbf{K}^b, \mathbf{V}^b \leftarrow \mathbf{x}_\theta^b(\mathbf{x}_t^b, \mathbf{K}^{1:b-1}, \mathbf{V}^{1:b-1}) := \mathbf{x}_\theta^b(\mathbf{x}_t^b, \mathbf{x}^{<b}), \tag{7}$$

where $\mathbf{x}_{\text{logits}}^b$ are the predictions for the clean $\mathbf{x}^b$, and $\mathbf{K}^b, \mathbf{V}^b$ is the key-value cache in the forward pass of $\mathbf{x}_\theta$, and $\mathbf{K}^{1:b-1}, \mathbf{V}^{1:b-1}$ are keys and values cached on a forward pass of $\mathbf{x}_\theta$ over $\mathbf{x}^{<b}$ (hence the inputs $\mathbf{x}^{<b}$ and $\mathbf{K}^{1:b-1}, \mathbf{V}^{1:b-1}$ are equivalent).

3.2 EFFICIENT TRAINING AND SAMPLING ALGORITHMS

Ideally, we wish to compute the loss $\mathcal{L}_{\text{BD}}(\mathbf{x}; \theta)$ in one forward pass of $\mathbf{x}_\theta$. However, observe that denoising $\mathbf{x}_t^b$ requires a forward pass on this noisy input, while denoising the next blocks requires running $\mathbf{x}_\theta$ on the clean version $\mathbf{x}^b$. Thus every block has to go through the model at least twice.

Training Based on this observation, we propose a training algorithm with these minimal computational requirements (Alg. 1). Specifically, we precompute keys and values $\mathbf{K}^{1:B}, \mathbf{V}^{1:B}$ for the full sequence $\mathbf{x}$ in a first forward pass $(\emptyset, \mathbf{K}^{1:B}, \mathbf{V}^{1:B}) \leftarrow \mathbf{x}_\theta(\mathbf{x})$. We then compute denoised predictions for each block using $\mathbf{x}_\theta^b(\mathbf{x}_t^b, \mathbf{K}^{1:b\text{-}1}, \mathbf{V}^{1:b\text{-}1})$. Each token passes through $\mathbf{x}_\theta$ twice.

Vectorized Training Naively, Alg. 1 would apply $\mathbf{x}_\theta^b(\mathbf{x}_t^b, \mathbf{K}^{1:b\text{-}1}, \mathbf{V}^{1:b\text{-}1})$ in a loop B times. We propose a vectorized implementation that computes $\mathcal{L}_{\text{BD}}(\mathbf{x}; \theta)$ in one forward pass on the concatenation $\mathbf{x}_{\text{noisy}} \oplus \mathbf{x}$ of clean data $\mathbf{x}$ with noisy data $\mathbf{x}_{\text{noisy}} = \mathbf{x}_{t_1}^1 \oplus \cdots \oplus \mathbf{x}_{t_B}^B$ obtained by applying a noise level t_b to each block $\mathbf{x}^b$. We mask $\mathbf{x}_{\text{noisy}} \oplus \mathbf{x}$ such that noisy tokens attend to other noisy tokens in their block and to all clean tokens in preceding blocks (see Suppl. B.6). Our method keeps the overhead of training BD3-LMs tractable and combines with pretraining to further reduce costs.

Sampling We sample one block at a time, conditioned on previously sampled blocks (Alg 2). We may use any sampling procedure $\text{SAMPLE}(\mathbf{x}_\theta^b, \mathbf{K}^{1:b\text{-}1}, \mathbf{V}^{1:b\text{-}1})$ to sample from the conditional distribution $p_\theta(\mathbf{x}_s^b|\mathbf{x}_t^b, \mathbf{x}^{<b})$, where the context conditioning is generated using cross-attention with pre-computed keys and values $\mathbf{K}^{1:b-1}, \mathbf{V}^{1:b-1}$. Similar to AR models, caching the keys and values saves computation instead of recalculating them when sampling a new block.

Notably, our block diffusion decoding algorithm enables us to sample sequences of arbitrary length, whereas diffusion models are restricted to fixed-length generation. Further, our sampler admits parallel generation within each block, whereas AR samplers are constrained to generate token-by-token.

Algorithm 1 Block Diffusion Training

```
Input: datapoint $\mathbf{x}$, # of blocks $B$, forward noise process $q_t(\cdot|\mathbf{x})$, model $\mathbf{x}_\theta$, loss $\mathcal{L}_{\text{BD}}$
repeat
    Sample $t_1, \dots, t_B \sim \mathcal{U}[0,1]$
    $\forall b \in \{1, ..., B\} : \mathbf{x}_{t_b}^b \sim q_{t_b}(\cdot|\mathbf{x}^b)$
    $\emptyset, \mathbf{K}^{1:B}, \mathbf{V}^{1:B} \leftarrow \mathbf{x}_\theta(\mathbf{x})$    ▷ KV cache
    $\forall b$: $\mathbf{x}_{\text{logit}}^b, \emptyset, \emptyset \leftarrow \mathbf{x}_\theta^b(\mathbf{x}_{t_b}^b, \mathbf{K}^{1:b\text{-}1}, \mathbf{V}^{1:b\text{-}1})$
    Let $\mathbf{x}_{\text{logit}} \leftarrow \mathbf{x}_{\text{logit}}^1 \oplus \cdots \oplus \mathbf{x}_{\text{logit}}^B$
    Take gradient step on $\nabla_\theta \mathcal{L}_{\text{BD}}(\mathbf{x}_{\text{logit}}; \theta)$
until converged
```

Algorithm 2 Block Diffusion Sampling

```
Input: # blocks $B$, model $\mathbf{x}_\theta$, diffusion sampling algorithm SAMPLE
$\mathbf{x}, \mathbf{K}, \mathbf{V} \leftarrow \emptyset$    ▷ output & KV cache
for $b = 1$ to $B$ do
    $\mathbf{x}^b \leftarrow \text{SAMPLE}(\mathbf{x}_\theta^b, \mathbf{K}^{1:b\text{-}1}, \mathbf{V}^{1:b\text{-}1})$
    $\emptyset, \mathbf{K}^b, \mathbf{V}^b \leftarrow \mathbf{x}_\theta^b(\mathbf{x}^b)$
    $\mathbf{x} \leftarrow \mathbf{x}^{1:b-1} \oplus \mathbf{x}^b$
    $(\mathbf{K}, \mathbf{V}) \leftarrow (\mathbf{K}^{1:b-1} \oplus \mathbf{K}^b, \mathbf{V}^{1:b-1} \oplus \mathbf{V}^b)$
end for
return $\mathbf{x}$
```

4 UNDERSTANDING LIKELIHOOD GAPS BETWEEN DIFFUSION & AR MODELS

4.1 MASKED BD3-LMS

The most effective diffusion language models leverage a masking noise process (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024a), where tokens are gradually replaced with a special mask token. Here, we introduce masked BD3-LMs, a special class of block diffusion models based on the masked diffusion language modeling framework (Sahoo et al., 2024a; Shi et al., 2024; Ou et al., 2025).

More formally, we adopt a per-token noise process $q(\mathbf{x}_t^\ell|\mathbf{x}^\ell) = \text{Cat}(\mathbf{x}_t^\ell; \alpha_t \mathbf{x}^\ell + (1-\alpha_t)\mathbf{m})$ for tokens $\ell \in \{1, \dots, L\}$ where $\mathbf{m}$ is a one-hot encoding of the mask token, and $\alpha_t \in [0,1]$ is a strictly decreasing function in t, with $\alpha_0 = 1$ and $\alpha_1 = 0$. We employ the linear schedule where the probability of masking a token at time t is $1 - \alpha_t$. We adopt the simplified objective from Sahoo et al. (2024a); Shi et al. (2024); Ou et al. (2025) (the full derivation is provided in Suppl. B.3):

$$-\log p_\theta(\mathbf{x}) \leq \mathcal{L}_{\text{BD}}(\mathbf{x}; \theta) := \sum_{b=1}^{B} \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{\alpha_t'}{1-\alpha_t} \log p_\theta(\mathbf{x}^b|\mathbf{x}_t^b, \mathbf{x}^{<b}) \tag{8}$$

where α_t' is the instantaneous rate of change of α_t under the continuous-time extension of (3) that takes $T \to \infty$. The NELBO is tight for $L' = 1$ but becomes a looser approximation of the true log-likelihood for $L' \to L$ (see Suppl. B.5).

4.2 CASE STUDY: SINGLE TOKEN GENERATION

Our block diffusion parameterization (8) is equivalent in expectation to the autoregressive NLL (1) in the limiting case where $L' = 1$ (see Suppl. B.4). Surprisingly, we find a two point perplexity gap between our block diffusion model for $L' = 1$ and AR when training both models on the LM1B dataset.

Table 1: Test perplexities for single-token generation (PPL; ↓) across 16B tokens on LM1B.

	PPL (↓)
AR	**22.88**
+ random batch size	24.37
BD3-LM $L' = 1$	$\leq$ 25.56
+ tuned schedule	**22.88**

Although the objectives are equivalent in expectation, we show that the remaining perplexity gap is a result of high training variance. Whereas AR is trained using the cross-entropy of L tokens, our block diffusion model for $L' = 1$ only computes the cross-entropy for masked tokens $\mathbf{x}_t^\ell = \mathbf{m}\ \forall \ell \in \{1, \dots L\}$

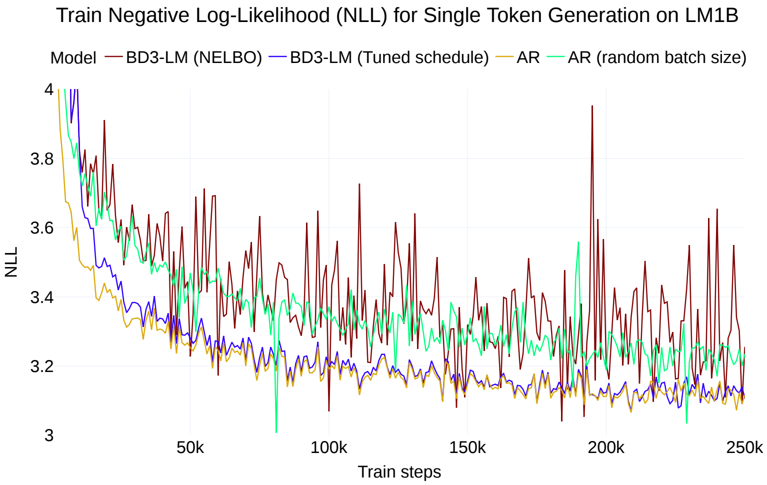

Figure 2: Train NLLs for modeling the per-token likelihood on LM1B. Training under the discrete diffusion NELBO, where half of the tokens in a batch are masked on average, has similar training variance to an AR model with a random batch size.

so that $\mathbb{E}_{t\sim\mathcal{U}[0,1]}q(\mathbf{x}_t^\ell = \mathbf{m}|\mathbf{x}^\ell) = 0.5$. Thus, training on the diffusion objective involves estimating loss gradients with 2x fewer tokens and is responsible for higher training variance compared to AR.

To close the likelihood gap, we train a BD3-LM for $L' = 1$ by designing the forward process to fully mask tokens, i.e. $q(\mathbf{x}_t^\ell = \mathbf{m}|\mathbf{x}^\ell) = 1$. Under this schedule, the diffusion objective becomes *equivalent* to the AR objective (Suppl. B.4). In Table 1, we show that training under the block diffusion objective yields the same perplexity as AR training. Empirically, we see that this reduces the variance of the training loss in Figure 2. We verify that tuning the noise schedule reduces the variance of the objective by measuring $\text{Var}_{\mathbf{x},t}\left[\mathcal{L}_{\text{BD}}(\mathbf{x};\theta)\right]$ after training on 328M tokens: while training on the NELBO results in a variance of 1.52, training under full masking reduces the variance to 0.11.

4.3 DIFFUSION GAP FROM HIGH VARIANCE TRAINING

Next, we formally describe the issue of gradient variance in training diffusion models. Given our empirical observations for single-token generation, we propose an estimator for gradient variance that we use to minimize the variance of diffusion model training for $L' \geq 1$. While the NELBO is invariant to the choice of noise schedule (Suppl. B.3), this invariance does not hold for our Monte Carlo estimator of the loss used during training. As a result, the variance of the estimator and its gradients are dependent on the schedule. First, we express the estimator of the NELBO with a batch size K. We denote a batch of sequences as $\mathbf{X} = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(K)}\right]$, with each $\mathbf{x}^{(k)} \overset{\text{iid}}{\sim} q(\mathbf{x})$. We obtain the batch NELBO estimator below, where $t(k,b)$ is sampled in sequence k and block b:

$$\mathcal{L}_{\text{BD}}(\mathbf{X};\theta) := l(\mathbf{X};\theta) = \frac{1}{K}\sum_{k=1}^{K}\sum_{b=1}^{B}\frac{\alpha'_{t(k,b)}}{1-\alpha_{t(k,b)}}\log p_\theta\left(\mathbf{x}^{(k),b} \mid \mathbf{x}^{(k),b}_{t(k,b)}, \mathbf{x}^{(k),<b}\right) \tag{9}$$

The variance of the gradient estimator over M batches for each batch $\mathbf{X}^m \; \forall m \in \{1,\dots,M\}$ is:

$$\text{Var}_{\mathbf{X},t}\left[\nabla_\theta l(\mathbf{X};\theta)\right] \approx \frac{1}{M-1}\sum_{m=1}^{M}\left\|\nabla_\theta l(\mathbf{X}^m;\theta) - \frac{1}{M}\sum_{m=1}^{M}\nabla_\theta l(\mathbf{X}^m;\theta)\right\|_2^2 \tag{10}$$

5 LOW-VARIANCE NOISE SCHEDULES FOR BD3-LMS

5.1 INTUITION: AVOID EXTREME MASK RATES

We aim to identify schedules that minimize the variance of the gradient estimator and make training most efficient. In a masked setting, we want to mask random numbers of tokens, so that the model learns to undo varying levels of noise, which is important during sampling. However, if we mask very few tokens, reconstructing them is easy and does not provide useful learning signal. If we mask everything, the optimal reconstruction are the marginals of each token in the data distribution, which is easy to learn, and again is not useful. These extreme masking rates lead to poor high-variance gradients: we want to learn how to clip them via a simple and effective new class of schedules.

5.2 CLIPPED SCHEDULES FOR LOW-VARIANCE GRADIENTS

We propose a class of "clipped" noise schedules that sample mask rates $1 - \alpha_t \sim \mathcal{U}[\beta, \omega]$ for $0 \leq \beta, \omega \leq 1$. We argue that from the perspective of deriving Monte Carlo gradient estimates, these schedules are equivalent to a continuous schedule where the mask probability is approximately 0 before the specified range such that $1 - \alpha_{<\beta} \approx \epsilon$ and approximately 1 after the specified range $1 - \alpha_{>\omega} \approx 1 - \epsilon$. Consequently, α'_t is linear within the range: $\alpha'_t \approx 1/(\beta - \omega)$.

5.3 DATA-DRIVEN CLIPPED SCHEDULES ACROSS BLOCK SIZES

As the optimal mask rates may differ depending on the block size L', we adaptively learn the schedule during training. While Kingma et al. (2021) perform variance minimization by isolating a variance term using their squared diffusion loss, this strategy is not directly applicable to our variance estimator in Equation 10 since we seek to reduce variance across random batches in addition to random t_b.

Instead, we optimize parameters β, ω to directly minimize training variance. To limit the computational burden of the optimization, we use the variance of the estimator of the diffusion ELBO as a proxy for the gradient estimator to optimize β, ω: $\min_{\beta,\omega} \text{Var}_{\mathbf{X},t} \left[\mathcal{L}(\mathbf{X}; \theta, \beta, \omega)\right]$. We provide experimental details on the optimization procedure in Sec. 6.

In Table 2, we show that variance of the diffusion NELBO is correlated with test perplexity. Under a range of "clipped" noise rate distributions, we find that there exists a unique distribution for each block size $L' \in \{4, 16, 128\}$ that minimizes both the variance of the NELBO and the test perplexity.

Table 2: Perplexities (PPLs; ↓) and variances of the NELBO $\text{Var}_{\mathbf{X},t} \left[\mathcal{L}_{\text{BD}}(\mathbf{X}; \theta)\right]$ (Var. NELBO; ↓). Models are trained on LM1B using a linear schedule for 65B tokens, then finetuned for 10B tokens.

	$\mathcal{U}[0, .5]$		$\mathcal{U}[.3, .8]$		$\mathcal{U}[.5, 1]$		$\mathcal{U}[0, 1]$	
L'	PPL	Var. NELBO	PPL	Var. NELBO	PPL	Var. NELBO	PPL	Var. NELBO
128	**31.72**	**1.03**	31.78	1.35	31.92	1.83	31.78	3.80
16	31.27	7.90	**31.19**	**3.62**	31.29	3.63	31.33	7.39
4	29.23	32.68	29.37	10.39	**29.16**	**8.28**	29.23	23.65

6 EXPERIMENTS

Table 3: Test perplexities (PPL; ↓) of models trained for 65B tokens on LM1B. Best diffusion value is bolded.

	PPL (↓)
Autoregressive	
Transformer-X Base (Dai et al., 2019)	23.5
Transformer (Sahoo et al., 2024a)	22.83
Diffusion	
D3PM (absorb) (Austin et al., 2021)	$\leq$ 82.34
SEDD (Lou et al., 2024)	$\leq$ 32.68
MDLM (Sahoo et al., 2024a)	$\leq$ 31.78
Block diffusion (Ours)	
BD3-LMs $L' = 16$	$\leq$ 30.60
$L' = 8$	$\leq$ 29.83
$L' = 4$	$\leq$ **28.23**

We evaluate BD3-LMs across standard language modeling benchmarks and demonstrate their ability to generate arbitrary-length sequences unconditionally. We pre-train a base BD3-LM using the maximum block size $L' = L$ for 850K gradient steps and fine-tune under varying L' for 150K gradient steps on the One Billion Words dataset (LM1B; Chelba et al. (2014)) and OpenWebText (OWT; Gokaslan et al. (2019)). Details on training and inference are provided in Suppl C.

To reduce the variance of training on the diffusion NELBO, we adaptively learn the range of masking rates by optimizing parameters β, ω as described in Section 5.3. In practice, we do so using a grid search during every validation step (after ∼5K gradient updates) to identify β, ω: $\min_{\beta,\omega} \text{Var}_{\mathbf{X},t}\left[\mathcal{L}(\mathbf{X};\theta,\beta,\omega)\right]$. During evaluation, we report likelihood under uniformly sampled mask rates (8) as in Austin et al. (2021); Sahoo et al. (2024a).

6.1 LIKELIHOOD EVALUATION

On LM1B, BD3-LMs outperform all prior diffusion methods in Table 3. Compared to MDLM (Sahoo et al., 2024a), BD3-LMs achieve up to 12% improvement in perplexity. We observe a similar trend on OpenWebText in Table 4.

We also evaluate the ability of BD3-LMs to generalize to unseen datasets in a zero-shot setting, following the benchmark from Radford et al. (2019). We evaluate the likelihood of models trained with OWT on datasets Penn Tree Bank (PTB; (Marcus et al., 1993)), Wikitext (Merity et al., 2016), LM1B, Lambada (Paperno et al., 2016), AG News (Zhang et al., 2015), and Scientific Papers (Pubmed and Arxiv subsets; (Cohan et al., 2018)). In Table 5, BD3-LM achieves the best zero-shot perplexity on Pubmed, surpassing AR, and the best perplexity among diffusion models on Wikitext, LM1B, and AG News.

Table 4: Test perplexities (PPL; ↓) on OWT for models trained for 524B tokens. Best diffusion value is bolded.

	PPL (↓)
AR (Sahoo et al., 2024a)	17.54
SEDD (Lou et al., 2024)	$\leq$ 24.10
MDLM (Sahoo et al., 2024a)	$\leq$ 22.98
BD3-LMs $L' = 16$	$\leq$ 22.27
$L' = 8$	$\leq$ 21.68
$L' = 4$	$\leq$ **20.73**

Table 5: 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	**81.07**	**25.32**	**51.14**	52.13	**52.11**	48.59	41.22
SEDD	96.33	35.98	68.14	48.93	67.82	45.39	40.03
MDLM	90.96	33.22	64.94	**48.29**	62.78	43.13	**37.89**
BD3-LM $L' = 4$	96.81	31.31	60.88	50.03	61.67	**42.52**	39.20

6.2 SAMPLE QUALITY AND VARIABLE-LENGTH SEQUENCE GENERATION

One key drawback of many existing diffusion language models (e.g,. Austin et al. (2021); Lou et al. (2024)) is that they cannot generate full-length sequences that are longer than the length of the output context chosen at training time. The OWT dataset is useful for examining this limitation, as it contains many documents that are longer than the training context length of 1024 tokens.

We record generation length statistics of 500 variable-length samples in Table 6. We continue sampling tokens until an end-of-sequence token [EOS] is generated or sample quality significantly degrades (as measured by likelihood and sample entropy). BD3-LMs generate sequences up to $\approx$10$\times$ longer than those of SEDD (Lou et al., 2024), which is restricted to the training context size.

Table 6: Generation length statistics from sampling 500 documents from models trained on OWT.

	Median # tokens	Max # tokens
OWT train set	717	131K
AR	4408	131K
SEDD	1021	1024
BD3-LM $L' = 16$	798	9982

We also examine the sample quality of BD3-LMs through quantitative and qualitative analyses. In Table 7, we generate sequences of lengths $L = 1024, 2048$ and measure their generative perplexity under GPT2-Large. To sample $L = 2048$ tokens from MDLM, we use their block-wise decoding technique (which does not feature block diffusion training as in BD3-LMs).

We also compare to SSD-LM (Han et al., 2022), an alternative block diffusion formulation. Unlike our discrete diffusion framework, SSD-LM uses Gaussian diffusion and does not support likelihood estimation. Further, BD3-LM adopts an efficient sampler from masked diffusion, where the number of generation steps (NFEs) is upper-bounded by L since tokens are never remasked (Sahoo et al., 2024a; Ou et al., 2025). For SSD-LM, we compare sample quality using $T = 1$k diffusion steps per block, matching their experimental setting, and $T = 25$ where NFEs are comparable across methods.

Table 7: Generative perplexity (Gen. PPL; ↓) and number of function evaluations (NFEs; ↓) of 300 samples of lengths $L = 1024, 2048$. All models are trained on OWT (AR, SEDD, MDLM are trained on 524B tokens, SSD-LM is pre-trained on 122B tokens). Best diffusion value is bolded. We provide further inference details on reporting generative perplexity and NFEs in Supp. C.5.

	$L = 1024$		$L = 2048$	
Model	Gen. PPL	NFEs	Gen. PPL	NFEs
AR	14.1	1K	13.2	2K
Diffusion				
SEDD	52.0	1K	–	–
MDLM	46.8	1K	41.3	2K
Block Diffusion				
SSD-LM $L' = 25$	37.2	40K	35.3	80K
	281.3	1K	281.9	2K
BD3-LMs $L' = 16$	33.4	1K	31.5	2K
$L' = 8$	30.4	1K	28.2	2K
$L' = 4$	**25.7**	1K	**23.6**	2K

BD3-LMs achieve the best generative perplexities compared to previous diffusion methods. Relative to SSD-LM, our discrete approach yields samples with improved generative perplexity using an order of magnitude fewer generation steps. We also qualitatively examine samples taken from BD3-LM and baselines (AR, MDLM) trained on the OWT dataset; we report samples in Suppl. D. We observe that BD3-LM samples have higher coherence than MDLM samples and approach the quality of AR.

6.3 ABLATIONS

We assess the impact of the design choices in our proposed block diffusion recipes, namely 1) selection of the noise schedule and 2) the efficiency improvement of the proposed training algorithm relative to a naive implementation.

SELECTING NOISE SCHEDULES TO REDUCE TRAINING VARIANCE

Compared to the linear schedule used in Lou et al. (2024); Sahoo et al. (2024a), training under "clipped" noise schedules is the most effective for reducing the training variance which correlates with test perplexity. In Table 8, the ideal "clipped" masking rates, which are optimized during training, are specific to the block size and further motivate our optimization.

Relative to other standard noise schedules (Chang et al., 2022), "clipped" masking achieves the best training variance and test perplexity. Since heavier masking is effective for the smaller block size $L' = 4$, we compare with logarithmic and square root schedules that also encourage heavy masking. As lighter masking is optimal for block size $L' = 16$, we compare with square and cosine schedules.

EFFICIENCY OF TRAINING ALGORITHM

In the BD3-LM training algorithm (Sec. 3.2), we compute $\mathbf{x}_{\text{logit}}$ using two options. We may perform two forward passes through the network (precomputing keys and values for the full sequence $\mathbf{x}$, then computing denoised predictions), or combine these passes by concatenating the two inputs into the same attention kernel.

We find that performing this operation in a single forward pass is often more efficient as we reduce memory bandwidth bottlenecks by leveraging efficient, pre-

Table 8: Effect of the noise schedule on likelihood estimation. We finetune BD3-LMs on 3B tokens from LM1B and evaluate on a linear schedule. For clipped schedules, we compare optimal clipping for $L' = 4, 16$.

Noise schedule	PPL	Var. NELBO
L' = 4		
Clipped		
$t \sim \mathcal{U}[0.45, 0.95]$	**29.21**	**6.24**
$t \sim \mathcal{U}[0.3, 0.8]$	29.38	10.33
Linear $t \sim \mathcal{U}[0, 1]$	30.18	23.45
Logarithmic	30.36	23.53
Square root	31.41	26.43
L' = 16		
Clipped		
$t \sim \mathcal{U}[0.45, 0.95]$	31.42	3.60
$t \sim \mathcal{U}[0.3, 0.8]$	**31.12**	**3.58**
Linear $t \sim \mathcal{U}[0, 1]$	31.72	7.62
Square	31.43	13.03
Cosine	31.41	13.00

exisiting FlashAttention kernels Dao et al. (2022). Instead of paying the cost of 2 passes through the network, we only pay the cost of a more expensive attention operation. Empirically we see that this approach has 5-10% speed-up during training relative to performing two forward passes.

7 DISCUSSION AND PRIOR WORK

Comparison to D3PM Block diffusion builds off D3PM (Austin et al., 2021) and applies it to each autoregressive conditional. We improve over D3PM in three ways: (1) we extend D3PM beyond fixed sequence lengths; (2) we study the perplexity gap of D3PM and AR models, identify gradient variance as a contributor, and design variance-minimizing schedules; (3) we improve over the perplexity of D3PM models. While (1) involves modifying D3PM to make it block-wise, (2) is applicable to vanilla D3PM.

Comparison to MDLM BD3-LMs further make use of the perplexity-enhancing improvements in MDLM (Sahoo et al., 2024a). We also build upon MDLM: (1) while Sahoo et al. (2024a) point out that their NELBO is invariant to the noise schedule, we show that the noise schedule has a significant effect on gradient variance; (2) we push the state-of-the-art in perplexity beyond MDLM. Note that our perplexity improvements stem not only from block diffusion, but also from optimized schedules, and could enhance standard MDLM models.

Comparison to Autoregressive-Diffusion Models Han et al. (2022) introduced an alternative block diffusion formulation that performs Gaussian diffusion over word embeddings, as in Li et al. (2022). Our approach instead applies discrete noise as in Austin et al. (2021), and features notable improvements: (1) tractable likelihood estimates enabling principled evaluation; (2) faster generation, as our number of model calls is bounded by the number of generated tokens, while SSD-LM performs orders of magnitude more calls; (3) significantly improved performance, measured by perplexity relative to existing models as well as generative perplexity relative to samples from both SSD-LM. AR-Diffusion (Wu et al., 2023) is a variant of SSD-LM that uses a noise schedule which encourages left-to-right generation. However, they sacrifice parallelism by assigning unique, per-token timesteps.

Comparison to Jacobi Decoding Jacobi decoding (Santilli et al., 2023) is an AR inference technique that iteratively refines a random sequence and supports parallel generation of token blocks. However, Santilli et al. (2023) preserve causal masking from AR and use uniform noise, whereas BD3-LMs may leverage more context by attending to tokens within a block and use masking which has been shown to be superior in language modeling (Austin et al., 2021; Lou et al., 2024). Consistency LLMs (Kou et al., 2024) extend Jacobi decoding to include a fine-tuning objective. In contrast, BD3-LMs may leverage clean conditional context $\mathbf{x}^{<b}$ to enhance predictions.

Limitations Training BD3-LMs requires a custom procedure that is more expensive than regular diffusion training. We propose a vectorized implementation that keeps training speed within <2x of diffusion training speed; in our experiments, we also pre-train with a standard diffusion loss to further reduce the speed gap. Additionally, BD3-LMs generate blocks sequentially, and hence may face the same speed and controllability constraints as AR especially when blocks are small. The optimal block size for control is task specific, and optimal blocks for speed depend on the parallelization capabilities of inferencing hardware (e.g,. FLOPS vs. memory throughput) and serving batch size.

8 CONCLUSION

This work explores block diffusion and is motivated by two problems with existing discrete diffusion: the need to generate arbitrary-length sequences and the perplexity gap to autoregressive models. We introduce BD3-LMs, which represent a block-wise extension of the D3PM framework (Austin et al., 2021), and leverage a specialized training algorithm and custom noise schedules that further improve performance. We observe that in addition to being able to generate long-form documents, these models also improve perplexity, setting a new state-of-the-art among discrete diffusion models. BD3-LMs are subject to inherent limitations of generative models, including hallucinations (Achiam et al., 2023), copyright infringement (Gokaslan et al., 2024), limited controllability (Schiff et al., 2024; Wang et al., 2023) and harmful outputs (Bai et al., 2022), which require further research.

ACKNOWLEDGMENTS AND DISCLOSURE OF FUNDING

This work was partially funded by the National Science Foundation under awards DGE-1922551, CAREER awards 2046760 and 2145577, and by the National Institute of Health under award MIRA R35GM151243. Marianne Arriola is supported by a NSF Graduate Research Fellowship under award DGE-2139899 and a Hopper-Dean/Bowers CIS Deans Excellence Fellowship.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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.

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

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.

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.

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.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Shrey Goel, Vishrut Thoutam, Edgar Mariano Marroquin, Aaron Gokaslan, Arash Firouzbakht, Sophia Vincoff, Volodymyr Kuleshov, Huong T Kratochvil, and Pranam Chatterjee. Memdlm: De novo membrane protein design with masked discrete diffusion protein language models. *arXiv preprint arXiv:2410.16735*, 2024.

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

Aaron Gokaslan, A Feder Cooper, Jasmine Collins, Landan Seguin, Austin Jacobson, Mihir Patel, Jonathan Frankle, Cory Stephenson, and Volodymyr Kuleshov. Commoncanvas: Open diffusion models trained on creative-commons images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8250–8260, 2024.

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

Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models, 2023. URL https://arxiv.org/abs/2312.06662.

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.

Xiaochuang Han, Sachin Kumar, Yulia Tsvetkov, and Marjan Ghazvininejad. David helps goliath: Inference-time collaboration between small specialized and large general diffusion lms. *arXiv preprint arXiv:2305.14771*, 2023.

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

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

Daniel Israel, Aditya Grover, and Guy Van den Broeck. Enabling autoregressive models to fill in masked tokens. *arXiv preprint arXiv:2502.06901*, 2025.

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

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. CLLMs: Consistency large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=8uzBOVmh8H.

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.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024.

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.

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.

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

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

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.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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*, 2024a. URL `https://openreview.net/forum?id=L4uaAR4ArM`.

Subham Sekhar Sahoo, Aaron Gokaslan, Christopher De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL `https://openreview.net/forum?id=loMa99A4p8`.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.689. URL `https://aclanthology.org/2023.acl-long.689`.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. *arXiv preprint arXiv:2412.10193*, 2024.

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`.

Phillip Si, Allan Bishop, and Volodymyr Kuleshov. Autoregressive quantile flows for predictive uncertainty estimation. In *International Conference on Learning Representations*, 2022.

Phillip Si, Zeyi Chen, Subham Sekhar Sahoo, Yair Schiff, and Volodymyr Kuleshov. Semi-autoregressive energy flows: exploring likelihood-free training of normalizing flows. In *International Conference on Machine Learning*, pp. 31732–31753. PMLR, 2023.

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.

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

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.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. InfoDiffusion: Representation learning using information maximizing diffusion models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 36336–36354. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/wang23ah.html`.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, yelong shen, Jian Jiao, Juntao Li, zhongyu wei, Jian Guo, Nan Duan, and Weizhu Chen. AR-diffusion: Auto-regressive diffusion model for text generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=0EG6qUQ4xE`.

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

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

CONTENTS

1 Introduction 1

2 Background: Language Modeling Paradigms 2

- 2.1 Autoregressive Models . . . 3
- 2.2 Discrete Denoising Diffusion Probabilistic Models . . . 3

3 Block Diffusion Language Modeling 3

- 3.1 Block Diffusion Distributions and Model Architectures . . . 3
- 3.2 Efficient Training and Sampling Algorithms . . . 4

4 Understanding Likelihood Gaps Between Diffusion & AR Models 5

- 4.1 Masked BD3-LMs . . . 5
- 4.2 Case Study: Single Token Generation . . . 5
- 4.3 Diffusion Gap from High Variance Training . . . 6

5 Low-Variance Noise Schedules for BD3-LMs 6

- 5.1 Intuition: Avoid Extreme Mask Rates . . . 6
- 5.2 Clipped Schedules for Low-Variance Gradients . . . 7
- 5.3 Data-Driven Clipped Schedules Across Block Sizes . . . 7

6 Experiments 7

- 6.1 Likelihood Evaluation . . . 8
- 6.2 Sample Quality and Variable-Length Sequence Generation . . . 8
- 6.3 Ablations . . . 9

7 Discussion and Prior Work 10

8 Conclusion 10

A Block Diffusion NELBO 16

B Masked BD3-LMs 16

- B.1 Forward Process . . . 17
- B.2 Reverse Process . . . 17
- B.3 Simplified NELBO for Masked Diffusion Processes . . . 17
- B.4 Recovering the NLL from the NELBO for Single Token Generation . . . 18
- B.5 Tightness of the NELBO . . . 19
- B.6 Specialized Attention Masks . . . 19

C Experimental Details 20

- C.1 Datasets . . . 20

C.2 Architecture 20
C.3 Training 21
C.4 Likelihood Evaluation 21
C.5 Inference 21

D Samples **23**

A BLOCK DIFFUSION NELBO

Below, we provide the Negative ELBO (NELBO) for the block diffusion parameterization. Recall that the sequence $\mathbf{x}^{1:L} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^L\right]$ is factorized over B blocks, which we refer to as $\mathbf{x}$ for simplicity, drawn from the data distribution $q(\mathbf{x})$. Specifically, we will factorize the likelihood over B blocks of length L', then perform diffusion in each block over T discretization steps. Let $D_{\text{KL}}[\cdot]$ to denote the Kullback–Leibler divergence. We derive the NELBO as follows:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}) &= -\sum_{b=1}^{B} \log p_\theta(\mathbf{x}^b|\mathbf{x}^{<b}) \\
&= -\sum_{b=1}^{B} \log \mathbb{E}_q \frac{p_\theta(\mathbf{x}^b_{0:T}|\mathbf{x}^{<b})}{q(\mathbf{x}^b_{1:T}|\mathbf{x}^b)} \\
&= -\sum_{b=1}^{B} \log \mathbb{E}_q \frac{p_\theta(\mathbf{x}^b_T|\mathbf{x}^{<b}) \prod_{i=1}^{T} p_\theta(\mathbf{x}^b_s|\mathbf{x}^b_t, \mathbf{x}^{<b})}{\prod_{t=1}^{T} q(\mathbf{x}^b_t|, \mathbf{x}^b_s)} \\
&\leq \sum_{b=1}^{B} \Bigg[\underbrace{-\mathbb{E}_q \log p_\theta(\mathbf{x}^b|\mathbf{x}^b_1, \mathbf{x}^{<b})}_{\mathcal{L}_{\text{recons}}} \\
&\qquad + \underbrace{\mathbb{E}_{t\in\left\{\frac{2}{T},\ldots,\frac{T-1}{T},1\right\}} \mathbb{E}_q T \mathrm{D}_{KL}\left(q(\mathbf{x}^b_s|\mathbf{x}^b_t, \mathbf{x}^b) \parallel p_\theta(\mathbf{x}^b_s|\mathbf{x}^b_t, \mathbf{x}^{<b})\right)}_{\mathcal{L}_{\text{diffusion}}} \\
&\qquad + \underbrace{\mathrm{D}_{KL}\left(q(\mathbf{x}^b_{t=1}|\mathbf{x}^b) \parallel p_\theta(\mathbf{x}^b_{t=1})\right)}_{\mathcal{L}_{\text{prior}}} \Bigg]
\end{aligned} \tag{11}$$

For simplicity, we will denote $s = (t-1)/T$.

B MASKED BD3-LMS

We explore a specific class of block diffusion models that builds upon the masked diffusion language modeling framework. In particular, we focus on masking diffusion processes introduced by Austin et al. (2021) and derive a simplified NELBO under this framework as proposed by (Sahoo et al., 2024a; Shi et al., 2024; Ou et al., 2025).

First, we define the diffusion matrix Q_t for states $i \in \{1, \ldots, V\}$. Consider the noise schedule function $\alpha_t \in [0, 1]$, which is a strictly decreasing function in t satisfying $\alpha_0 = 1$ and $\alpha_1 = 0$. Denote the mask index as $m = V$. The diffusion matrix is defined by Austin et al. (2021) as:

$$[Q_t]_{ij} = \begin{cases} 1 & \text{if } i = j = m \\ \alpha_t & \text{if } i = j \neq m \\ 1 - \alpha_t & \text{if } j = m, i \neq m \end{cases} \tag{12}$$

The diffusion matrix for the forward marginal $Q_{t|s}$ is:

$$[Q_{t|s}]_{ij} = \begin{cases} 1 & \text{if } i = j = m \\ \alpha_{t|s} & \text{if } i = j \neq m \\ 1 - \alpha_{t|s} & \text{if } j = m, i \neq m \end{cases} \tag{13}$$

where $\alpha_{t|s} = \alpha_t / \alpha_s$.

B.1 FORWARD PROCESS

Under the D3PM framework (Austin et al., 2021), the forward noise process applied independently for each token $\ell \in \{1, \dots L\}$ is defined using diffusion matrices $Q_t \in \mathbb{R}^{V \times V}$ as

$$q(\mathbf{x}_t^\ell | \mathbf{x}^\ell) = \text{Cat}\left(\mathbf{x}_t^\ell; \overline{Q}_t \mathbf{x}^\ell\right), \quad \text{with} \quad \overline{Q}_t = Q_{t(1)} Q_{t(2)} \dots Q_{t(i)} \tag{14}$$

B.2 REVERSE PROCESS

Let $Q_{t|s}$ denote the diffusion matrix for the forward marginal. We obtain the reverse posterior $q(\mathbf{x}_s^\ell \mid \mathbf{x}_t^\ell, \mathbf{x}^\ell)$ using the diffusion matrices:

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

where $\odot$ denotes the Hadmard product between two vectors.

B.3 SIMPLIFIED NELBO FOR MASKED DIFFUSION PROCESSES

Following Sahoo et al. (2024a); Shi et al. (2024); Ou et al. (2025), we simplify the NELBO in the case of masked diffusion processes. Below, we provide the outline of the NELBO derivation; see the full derivation in Sahoo et al. (2024a); Shi et al. (2024); Ou et al. (2025).

We will first focus on simplifying the diffusion loss term $\mathcal{L}_{\text{diffusion}}$ in Eq. 11. We employ the SUBS-parameterization proposed in Sahoo et al. (2024b) which simplifies the denoising model p_θ for masked diffusion. In particular, we enforce the following constraints on the design of p_θ by leveraging the fact that there only exists two possible states in the diffusion process $\mathbf{x}_t^\ell \in \{\mathbf{x}^\ell, \mathbf{m}\}\ \forall \ell \in \{1, \dots, L\}$.

1. **Zero Masking Probabilities**. We set $p_\theta(\mathbf{x}^\ell = \mathbf{m} | \mathbf{x}_t^\ell) = 0$ (as the clean sequence $\mathbf{x}$ doesn't contain masks).
2. **Carry-Over Unmasking**. The true posterior for the case where $\mathbf{x}_t^\ell \neq \mathbf{m}$ is $q(\mathbf{x}_s^\ell = \mathbf{x}_t^\ell | \mathbf{x}_t^\ell \neq \mathbf{m}) = 1$ (if a token is unmasked in the reverse process, it is never remasked). Thus, we simplify the denoising model by setting $p_\theta(\mathbf{x}_s^\ell = \mathbf{x}_t^\ell | \mathbf{x}_t^\ell \neq \mathbf{m}) = 1$.

As a result, we will only approximate the posterior $p_\theta(\mathbf{x}_s^\ell = \mathbf{x}^\ell | \mathbf{x}_t^\ell = \mathbf{m})$. Let $\mathbf{x}^{b,\ell}$ denote a token in the ℓ-th position in block $b \in \{1, \dots, B\}$. The diffusion loss term becomes:

$$\begin{aligned}
\mathcal{L}_{\text{diffusion}} &= \sum_{b=1}^{B} \mathbb{E}_t \mathbb{E}_q T \left[\text{D}_{\text{KL}} \left[q(\mathbf{x}_s^b | \mathbf{x}_t^b) \| p_\theta(\mathbf{x}_s^b | \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \right] \\
&= \sum_{b=1}^{B} \mathbb{E}_t \mathbb{E}_q T \left[\sum_{\ell=1}^{L'} \text{D}_{\text{KL}} \left[q(\mathbf{x}_s^{b,\ell} | \mathbf{x}_t^{b,\ell}, \mathbf{x}^\ell) \| p_\theta(\mathbf{x}_s^{b,\ell} | \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \right]
\end{aligned}$$

D_{KL} is simply the discrete-time diffusion loss for the block b; hence, from Sahoo et al. (2024a) (Suppl. B.1), we get:

$$\begin{aligned}
&= \sum_{b=1}^{B} \mathbb{E}_t \mathbb{E}_q T \left[\sum_{\ell=1}^{L'} \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \log p_\theta(\mathbf{x}^{b,\ell} \mid \mathbf{x}_t^{b,\ell}, \mathbf{x}^{<b}) \right] \\
&= \sum_{b=1}^{B} \mathbb{E}_t \mathbb{E}_q T \left[\frac{\alpha_s - \alpha_t}{1 - \alpha_t} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right]
\end{aligned} \tag{16}$$

Lastly, we obtain a tighter approximation of the likelihood by taking the diffusion steps $T \to \infty$ (Sahoo et al., 2024a), for which $T(\alpha_s - \alpha_t) = \alpha_t'$:

$$\mathcal{L}_{\text{diffusion}} = \sum_{b=1}^{B} \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \left[\frac{\alpha_t'}{1 - \alpha_t} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \tag{17}$$

For the continuous time case, Sahoo et al. (2024a) (Suppl. A.2.4) show the reconstruction loss reduces to 0 as $\mathbf{x}_1^b \sim \lim_{T\to\infty} \text{Cat}(.; q(\mathbf{x}_1^b|\mathbf{x}^b)) \implies \mathbf{x}_1^b \sim \text{Cat}(.; \mathbf{x}^b)$. Using this, we obtain:

$$\begin{aligned}\mathcal{L}_{\text{recons}} &= -\log p_\theta(\mathbf{x}^b|\mathbf{x}_1^b, \mathbf{x}^{<b}) \\ &= -\log p_\theta(\mathbf{x}^b|\mathbf{x}_1^b = \mathbf{x}^b, \mathbf{x}^{<b}) \\ &= 0 \end{aligned} \tag{18}$$

The prior loss $\mathcal{L}_{\text{prior}} = \text{D}_{KL}\left(q(\mathbf{x}_T^b|\mathbf{x}^b) \parallel p_\theta(\mathbf{x}_T^b)\right)$ also reduces to 0 under the SUBS-parameterization as $q(\mathbf{x}_T^b|\mathbf{x}^b) = \text{Cat}(.; \mathbf{m})$ and $p_\theta(\mathbf{x}_T^b) = \text{Cat}(.; \mathbf{m})$ (see Sahoo et al. (2024a), Suppl. A.2.4).

Finally, we obtain a simple objective that is a weighted average of cross-entropy terms:

$$\mathcal{L}_{\text{BD}}(\mathbf{x}; \theta) = \sum_{b=1}^{B} \mathbb{E}_{t\sim[0,1]} \mathbb{E}_q \left[\frac{\alpha_t'}{1-\alpha_t} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \tag{19}$$

The above NELBO is invariant to the choice of noise schedule α_t, see Sahoo et al. (2024a) (Suppl. E.1.1).

B.4 RECOVERING THE NLL FROM THE NELBO FOR SINGLE TOKEN GENERATION

Consider the block diffuson NELBO for a block size of 1 where $L' = 1, B = L$. The block diffusion NELBO is equivalent to the AR NLL when modeling a single token:

$$\begin{aligned} -\log p(\mathbf{x}) &\leq \sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \mathbb{E}_q \left[\frac{\alpha_t'}{1-\alpha_t} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \\ &\because \alpha_t' = -1 \text{ and } \alpha_t = 1-t, \\ &= -\sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \mathbb{E}_q \left[\frac{1}{t} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \\ &= -\sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \frac{1}{t} \mathbb{E}_q \left[\log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \right] \\ &\text{Expanding } \mathbb{E}_q[.], \\ &= -\sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \frac{1}{t} \Big[q(\mathbf{x}_t^b = \mathbf{m}|\mathbf{x}^b) \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{m}, \mathbf{x}^{<b}) \\ &\qquad + q(\mathbf{x}_t^b = \mathbf{x}^b|\mathbf{x}^b) \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{x}^b, \mathbf{x}^{<b}) \Big] \end{aligned} \tag{20}$$

Recall that our denoising model employs the SUBS-parameterization proposed in Sahoo et al. (2024b). The "carry-over unmasking" property ensures that $\log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{x}^b, \mathbf{x}^{<b}) = 0$, as an unmasked token is simply copied over from from the input of the denoising model to the output. Hence, (20) reduces to following:

$$\begin{aligned} -\log p_\theta(\mathbf{x}) &\leq -\sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \frac{1}{t} q(\mathbf{x}_t^b = \mathbf{m}|\mathbf{x}^b) \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{m}, \mathbf{x}^{<b}) \\ &\because q(\mathbf{x}_t^b = \mathbf{m}|\mathbf{x}^b) = t, \text{ we get:} \\ &= -\sum_{b=1}^{L} \mathbb{E}_{t\sim[0,1]} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{m}, \mathbf{x}^{<b}) \\ &= -\sum_{b=1}^{L} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{m}, \mathbf{x}^{<b}) \end{aligned} \tag{21}$$

For single-token generation ($L' = 1$) we recover the autoregressive NLL.

B.5 TIGHTNESS OF THE NELBO

For block sizes $1 \leq K \leq L$, we show that -log $p(\mathbf{x}) \leq \mathcal{L}_K \leq \mathcal{L}_{K+1}$. Consider $K = 1$, where we recover the autoregressive NLL (see Suppl B.4):

$$\begin{aligned}\mathcal{L}_1 &= \sum_{b=1}^{L} \log \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{\alpha_t'}{1-\alpha_t} p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \\ &= -\sum_{b=1}^{L} \log p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b = \mathbf{m}, \mathbf{x}^{<b}) \end{aligned} \tag{22}$$

Consider the ELBO for block size $K = 2$:

$$\mathcal{L}_2 = \sum_{b=1}^{L/2} \log \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{\alpha_t'}{1-\alpha_t} p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \tag{23}$$

We show that $\mathcal{L}_1 \leq \mathcal{L}_2$, and this holds for all $1 \leq K \leq L$ by induction. Let $\mathbf{x}^{b,i}$ correspond to the token in position $\ell \in [1, L']$ of block b. We derive the below inequality:

$$\begin{aligned}-\sum_{b=1}^{L} \log p_\theta(\mathbf{x}_t^b = \mathbf{m} | \mathbf{x}^{<b}) &= -\sum_{b=1}^{L/2} \log \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{1}{1-\alpha_t} p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \\ &= -\sum_{b=1}^{L/2} \log \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \prod_{i=1}^{2} \frac{1}{1-\alpha_t} p_\theta(\mathbf{x}^{b,\ell} \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \\ &= -\sum_{b=1}^{L/2} \log \prod_{i=1}^{2} \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{1}{1-\alpha_t} p_\theta(\mathbf{x}^{b,\ell} \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \\ &\leq -\sum_{b=1}^{L/2} \sum_{i=1}^{2} \log \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{1}{1-\alpha_t} p_\theta(\mathbf{x}^{b,\ell} \mid \mathbf{x}_t^b, \mathbf{x}^{<b}) \end{aligned} \tag{24}$$

B.6 SPECIALIZED ATTENTION MASKS

We aim to model conditional probabilities $p_\theta(\mathbf{x}^b \mid \mathbf{x}_t^b, \mathbf{x}^{<b})$ for blocks $b \in [1, B]$ simultaneously by designing an efficient training algorithm with our transformer backbone. However, modeling all B conditonal terms requires processing both the noised sequence $\mathbf{x}_t^b$ and the conditional context $\mathbf{x}^{<b}$.

Rather than calling the denoising network B times, we process both sequences simultaneously by concatenating them $\mathbf{x}_{full} \leftarrow \mathbf{x}_t \oplus \mathbf{x}$ as input to a transformer. We update this sequence $\mathbf{x}_{full}$ of length $2L$ using a custom attention mask $\mathcal{M}(L, B) \in \{0,1\}^{2L \times 2L}$ for efficient training.

This attention mask is comprised of 4 $L \times L$ smaller attention masks:

$$\textsc{Mask}(L, B) = \begin{bmatrix} \mathcal{M}_{BD} & \mathcal{M}_{OBC} \\ \mathbf{0} & \mathcal{M}_{BC} \end{bmatrix}$$

where $\mathcal{M}_{BD}$ and $\mathcal{M}_{OBC}$ are used to update the representation of $\mathbf{x}_t$ and $\mathcal{M}_{BC}$ is used to update the representation of $\mathbf{x}$. We define these masks as follows:

- $\mathcal{M}_{BD}$ (Block-diagonal mask): Self-attention mask within noised blocks $\mathbf{x}_t^b$

$$\mathcal{M}_{BD} = (m_{i,j})_{L \times L} = \begin{cases} 1 & \text{if } i, j \text{ are in the same block} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathcal{M}_{OBC}$ (Offest block-causal mask): Cross-attention mask for conditional context $\mathbf{x}^{<b}$

$$\mathcal{M}_{OBC} = (m_{i,j})_{L \times L} = \begin{cases} 1 & \text{if } j \text{ belongs in a block before } i \\ 0 & \text{otherwise} \end{cases}$$

- $\mathcal{M}_{BC}$ (Block-causal mask): Attention mask for updating $\mathbf{x}^b$

$$\mathcal{M}_{BC} = (m_{i,j})_{L' \times L} = \begin{cases} 1 & \text{if } j \text{ belongs in the same block as } i \text{, or a block before } i \\ 0 & \text{otherwise} \end{cases}$$

We visualize an example attention mask for $L = 6$ and block size $L' = 2$ in Figure 3.

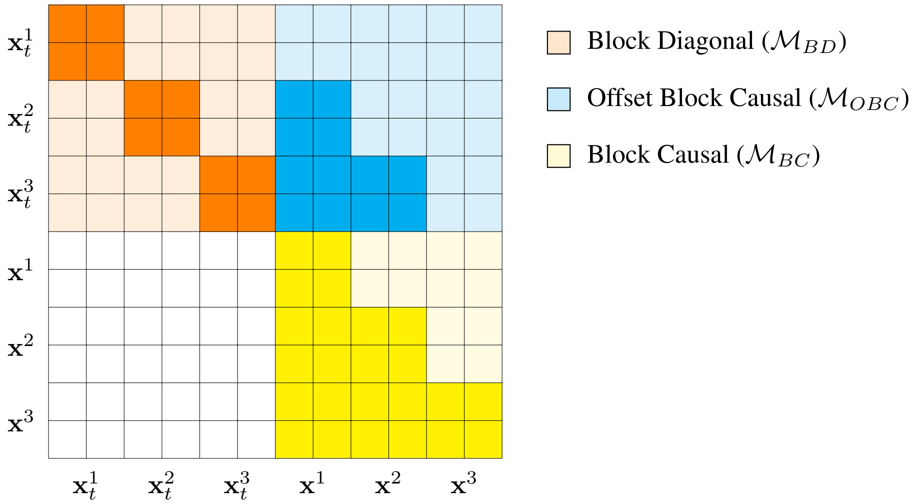

Figure 3: Example of Specialized Attention Mask

C EXPERIMENTAL DETAILS

We closely follow the same training and evaluation setup as used by Sahoo et al. (2024a).

C.1 DATASETS

We conduct experiments on two datasets: The One Billion Word Dataset (LM1B; Chelba et al. (2014)) and OpenWebText (OWT; Gokaslan et al. (2019)). Models trained on LM1B use the `bert-base-uncased` tokenizer and a context length of 128. We report perplexities on the test split of LM1B. Models trained on OWT use the `GPT2` tokenizer Radford et al. (2019) and a context length of 1024. Since OWT does not have a validation split, we leave the last 100k documents for validation.

In preparing LM1B examples, Sahoo et al. (2024a) pad each example to fit in the context length. Since most examples consist of only a single sentence, block diffusion modeling for larger block sizes $L' > 4$ would not be useful for training. Instead, we concatenate and wrap sequences to a length of 128. As a result, we retrain our autoregressive baseline, SEDD, and MDLM on LM1B with wrapping.

Similarly for OWT, we do not pad or truncate sequences, but concatenate them and wrap them to a length of 1024 similar to LM1B. For unconditional generation experiments in Section 6.2, we wish to generate sequences longer than the context length seen during training. However, Sahoo et al. (2024a) inject beginning-of-sequence and end-of-sequence tokens ([BOS], [EOS] respectively) at the beginning and end of the training context. Thus, baselines from Sahoo et al. (2024a) will generate sequences that match the training context size. To examine model generations across varying lengths in Section 6.2, we retrain our AR, SEDD, and MDLM baselines without injecting [BOS] and [EOS] tokens in the examples. We also adopt this preprocessing convention for training all BD3-LMs on OWT.

C.2 ARCHITECTURE

The model architecture augments the diffusion transformer (Peebles & Xie, 2023) with rotary positional embeddings (Su et al., 2021). We parameterize our autoregressive baselines, SEDD, MDLM, and BD3-LMs with a transformer architecture from Sahoo et al. (2024a) that uses 12 layers, a hidden dimension of 768, and 128 attention heads. We do not include timestep conditioning as Sahoo et al. (2024a) show it does not affect performance. We use the AdamW optimizer with a batch size of 512 and constant learning rate warmup from 0 to 3e-4 for 2.5k gradient updates.

C.3 TRAINING

We train a base BD3-LM using the maximum context length $L' = L$ for 850K gradient steps and fine-tune under varying L' using the noise schedule optimization for 150K gradient steps on the One Billion Words dataset (LM1B) and OpenWebText (OWT). This translates to 65B tokens and 73 epochs on LM1B, 524B tokens and 60 epochs on OWT. We use 3090, A5000, A6000, and A100 GPUs. Fine-tuning BD3-LMs on LM1B takes 1.5 days on 4 A5000s and OWT takes 4.5 days on 8 A100s.

C.4 LIKELIHOOD EVALUATION

We use a single Monte Carlo estimate for sampling t to evaluate the likelihood of a token block. We adopt a low-discrepancy sampler proposed in Kingma et al. (2021) that reduces the variance of this estimate by ensuring the time steps are more evenly spaced across the interval [0,1] following Sahoo et al. (2024a). In particular, we sample the time step for each block $b \in \{1, \dots, B\}$ and sequence $k \in \{1, \dots, K\}$ from a different partition of the uniform interval $t(k, b) \sim \mathcal{U}[\frac{(k-1)B+b-1}{KB}, \frac{(k-1)B+b}{KB}]$.

This low-discrepancy sampler is used for evaluation. For training, each time step may be sampled from a “clipped" range $t \sim \mathcal{U}[\beta, \omega]$. During training, we uniformly sample $t \in [0, 1]$ under the low-discrepancy sampler, then apply a linear interpolation: $t' = \beta + (\omega - \beta) \cdot t$.

When reporting zero-shot likelihoods on benchmark datasets from Radford et al. (2019) using models trained on OWT, we wrap all sequences to 1024 tokens and do not add [EOS] between sequences following Sahoo et al. (2024a).

C.5 INFERENCE

Generative Perplexity We report generative perplexity under GPT2-Large from models trained on OWT using a context length of 1024 tokens. Since GPT2-Large uses a context size of 1024, we compute the generative perplexity for samples longer than 1024 tokens using a sliding window with a stride length of 512 tokens.

Nucleus Sampling Following SSD-LM (Han et al., 2022), we employ nucleus sampling for BD3-LMs and our baselines. For SSD-LM, we use their default hyperparameters $p = 0.95$ for block size $L' = 25$. For BD3-LMs, AR and MDLM, we use $p = 0.9$. For SEDD, we find that $p = 0.99$ works best.

Number of Diffusion Steps In Table 7, BD3-LMs and MDLM use $T = 5$k diffusion steps. BD3-LMs and MDLM use efficient sampling by caching the output of the denoising network as proposed by Sahoo et al. (2024a); Ou et al. (2025), which ensures that the number of generation steps does not exceed the sample length L. Put simply, once a token is unmasked, it is never remasked as a result of the simplified denoising model (Suppl. B.3). We use MDLM's block-wise decoding algorithm for generating variable-length sequences, however these models are not trained with block diffusion.

SSD-LM (first row in Table 7) and SEDD use $T = 1$k diffusion steps. Since block diffusion performs T diffusion steps for each block $b \in 1, \dots, B$, SSD-LM undergoes BT generation steps. Thus to fairly compare with SSD-LM, we also report generative perplexity for $T = 25$ diffusion steps so that the number of generation steps does not exceed the sequence length (second row in Table 7).

Improved Categorical Sampling of Diffusion Models We employ two improvements to the Gumbel-based categorical sampling in SEDD, MDLM, and BD3-LMs as proposed by Zheng et al. (2024).

First, we use the corrected Gumbel-based categorical sampling from Zheng et al. (2024) by sampling 64-bit Gumbel variables. Reducing the precision to 32-bit has been shown to significantly truncate the Gumbel variables, lowering the temperature and decreasing the sentence entropy.

Second, Zheng et al. (2024) show that the MDLM sampling time scales with the diffusion steps T, even though the number of generation steps is bounded by the sequence length. For sample length L and vocabulary size V, the sampler requires sampling $\mathcal{O}(TLV)$ uniform variables and performing logarithmic operations on them.

We adopt the first-hitting sampler proposed by Zheng et al. (2024) which is theoretically equivalent to MDLM sampling while achieving up to a 20× speedup when $T \gg L$. In particular, the first-hitting sampler by leveraging two observations: (1) the transition probability is independent of the denoising network, (2) the transition probability is the same for all masked tokens for a given t. Thus, the first timestep where a token is unmasked can be analytically sampled as follows (assuming a linear schedule where $\alpha_t = 1 - t$):

$$t_{n-1} = t_n u^{1/n}, \tag{25}$$

where $n \in \{L, \ldots, 1\}$ denotes the number of masked tokens, $u_n \sim \mathcal{U}[0, 1]$ and t_{n-1} corresponds to the first timestep where $n - 1$ tokens are masked. This ensures that the sampling speed is within a reasonable range when $T \gg L$.

Variable-Length Sequence Generation For arbitrary-length sequence generation using BD3-LMs and AR in Table 6, we continue to sample tokens until the following stopping criteria are met:

1. an [EOS] token is sampled
2. the average entropy of the the last 256-token chunk is below 4

where criterion 2 are necessary to prevent run-on samples from compounding errors (for example, a sequence of repeating tokens). We find that degenerate samples with low entropy result in significantly low perplexities under `GPT2` and lower the reported generative perplexity. Thus, when a sample meets criterion 2, we regenerate the sample when reporting generative perplexity in Table 7.

D SAMPLES

<|endoftext|>'s architect, lawyer and San Giovanni concerto art critic Paolo Capacotti, gained attention from fellow gallery members and even invited him to present a retrospective, publishing issues and newspaper interviews.[10] On 6 September, Kissi and his assistants agreed to move to Angelo's Marcus Collection,[10] which included Giorgio Avolivo Arth and Moscolliso (later owned by the artist Belzina Massingolo) and Pan Giazzoglio Romeam-Guessle. The businessman, Giovanni Paletti, an outstanding collector, owned the museum and the painting. The level of criminal activity around the museum has continued to increase, which is part of several attempts to counter centennial rumors including the possibility that museum staff and visitors are tortured and even exposed to del Cavello for the only full year of Francesco Belzina's life (1999).[4] On the evening of 22 October 2005 it was reported that earlier that evening, guards had come on duty and began flinging an electric field with umbrellas from the balcony. As the fire continued, some of the guards sparked an apparent spat from the window of the cathedral. They remained idly watched by a pile of trash left after a piano key by Pietro Jolla, who died on 21 October 2005.[10] Just before 3:00 to 3pm on Monday, 27 October 2005, strong winds brought the trash on to the residence that opened on 17 October. Some ruined books and statues were hurled in front from every direction of the window. Some claimed that a customer Jacques Monet had beaten the hand of photographer Franco Campetti and in some cases had stuck a broken candle in the doorway of the museum. Andr Romeam-Guessle responded by laughing when he spoke. Giancio Giuliano, the artistic director of the Museum, even tried to told journalists and press that 'the patient in the trisomy machine [sic] carried some corpses four hours into the museum, but the whole time it was the guy who stroked the young man who who broke him'. In 2008, Giuliano told the same press that the hours of the destruction are truly ""wrong for their morality"" and further stated that 'We are never satisfied with our decision. We made an informed decision to build the museum after destruction.[5] Deaths [edit] A little after 12:00 am 17 October 2005, Giuliano and his partner Monica Concerta, noticed that the trash was being thrown by passers-by. Captain Iamienowska leaned over to his film camera and said, in a joking manner, that Iannorello, the chair of the Muscéei, was a thief that director Frank Nolan said ""he would later be arrested."" When Iamienowska arrived, the people in question were interviewed by Captain Anderson Tulaqyuk, a co-man who was initially lying on the scene and whom Iamienowska said was able to stop them from passing in the vicinity. Iguano proceeded to collect the trash and the police arrived, and closed the door of the museum.[6] During the war, the statue structure was partially removed and its cannons damaged. On the eve of the war, the U.S. Army and Canadian Air Company, who once owned a lot on the Coopers of Paris near Leopold Street, sheltered the POWs, who were briefly overworked. This following years led to condemnation of the organization and artists, including John DiDione,[7] returned to the compound to sell the museum items (including many of them originally stolen over the years—more than a million returned); on 12 October, the sculptor Jack Uellein installed a temporary dome on Del Cavella.[10] Among the changes over the years, Diena featured the lion in the middle of the other two halves of the dome and the lion in the back of the Viduccio.[11] In March 2008, Arvnzo Cissino announced that seven studio studios were built between the summer of 1985 and the last one was completed on November 17, 2005.[10] In the fourth quarter of the mid-2000s, a number of new structures were constructed by various museums, but most notably by Arvnzo Cissino, the brothers-soldier, as portrayed by architect Duvois Mayaven in the illusion of the El Valleratas Humanities and Cultural Center, reconstructed by architect Duvois Benjamin. Just as the very significant additions to the current management masts of the del Museum are such private three-bedroom rooms as Tradello's cabin, as well as the architectural aspect from the museum's south wall. Also new are the rockings performed by MC Urquhart, Aiba Ruiz and March Flez, during a music festival held in the US in 2006. Urquhart was later credited with establishing record label Empire Records as an influence on his music. In<|endoftext>|

Figure 4: Sample from MDLM (Sahoo et al., 2024a) of length $L = 1024$ and $T = 5\text{k}$ diffusion steps. The generative perplexity of this sample under GPT2-Large is 69.26 and its entropy is 5.62.

<|endoftext|>Report: High Evolution Studio v7.1.4 Oversightly and Submitted to Flytv's Windows Maintenance Team Posted, July 16, 2014 – Jonathan Martinez Like me, creators of Flying.tv, I am continually struck by the enthusiasm and support of programmers and enthusiasts who have gone crazy for it…for a hobby that makes us all live better lives. I'm extremely happy we have been involved with it so long. There are many issues with this project, and I want to share them on this blog and quickly fix them. As of late today, there's been nothing posted about the project and Programmer lives in the AMC. Still, I can't sure he will continue working, or at least show he won't. So, at about 14:59:23 in the afternoon, I spoke with the developer yesterday night on Github and we have a reply and I write. Initially, the security developer was concerned that the developer needed to relax to chat a bit. This is a really bad way to silence anyone with #librVPN I'm never seen bad VPN porn in person. Thank you so much Really? I know…it gets pretty annoying these days but we did this to have fun. We didn't have a plan for this actually, but nonetheless, I think we should have. Here are some things we can tell you guys. We know you wouldn't like to do this. At least it wasn't such a thing. We really want to ask you this. Let us do this to make it better for everyone involved. Fix the problems. Let us let you know we'd appreciate your support to do this. Great for you, and we'll now get your email about this post. And, if you click the /home link, click Open your terminal and download and install your new game package package. Have good to you guys. Link for this document is "Support Website Development Pack" Description: "From disk to virtual cloud." In Virtualcloud you can enter and open the dialog box. In the dialog box are a list of deauthuments (default destination) which means that you must have a proof-of-concept certificate for you, and the ability to sign one. This means that the client's servers cannot break into or or decrypt your ports and files. If you have the correct password (eg: "private-no-privacy-password") then you can sign one in the middle of those emails. If not, this means that the server can do absolutely nothing here. In your words is if you believe that your password is indeed a false one (or something for that matter and the client doesn't want to do anything), that might mean that they will immediately break and shutdown your server if you try to fix it." One of the main points in the document is that you use the default response id as valid to sign in, so that your server can use the non-network access to send you messages. This is a good idea, as it allows your server running around to send a message. The fact is that there are many available ways to do this, and after that it seems only the one works in some limited circumstances (e.g.: for servers to exploit the stfates of traffic). To continue with the problem, we want to make sure that this behavior is enough to prevent other installers from taking over, and we want them to keep your server shut down so that you plug in no access to other systems and your server is completely vulnerable. The original version says you should not do this, but we did, again, check this out. Here are the main issues: all passwords must be stored on the server encryption system passwords cannot be ignored this allows for clients with different system passwords and with information about other clients this holds back all non-secure emails and can potentially be stolen during the installation case will complain that multiple users have uploaded the same empty file always ask a client to ask for confirmation of the connection 'wonders' are pretty important We want you to follow this post as technical and conceptually as possible, but I think it's worth noting that the security team were very worried before the project a little on this…and were worried actually was a bad idea. So, we are now moving to the next point of security and privacy, as we are releasing the YPC packages on the WinRT servers. Of course, users already have Windows 5 or MD5 tools installed, but WinRT is a new base. However, now that the platform has been released, we want you to download the latest packages from them. Yes. Just download the files for you In case of Edge Plus, we're no longer working on a NMS download as we've just released mainly the YPC update (see RC1.0). This package is below. urm .NET The first version of the library of this branch of the library is version 0.6.15 (the 2nd version of the library which we released right now). The .NET library handles the common API of the database, and objects. Slices as components To put the whole problem into a picture, here's how we get started. It takes a lot of effort because we have to include all the required APIs and for it to run well, it's really hard, but we all do it every day: to set up your own apps. To give you a little more information on how to use Windows PowerShell in App Engine to create your Windows apps, you are going to see an example from our .NET Core Library, which is helpful for starting up your App Engine app. Here our ScriptPistry for PowerShell version 4 is available. The exact name of the project is listed in the project example, as well as a listing of the version and development information that you'll find in the .NET source package. Here's our example package and the name of the project for the PowerShell connection and the regular query. Now that you've gotten a few files and are willing to go to the next step, let's start the project. We will start with the .NET libraries, let's start on the projects ourselves. Choose the root project component package name, because it's used later in the source package. The project name for this is available in the project file followed by this part. To start the project itself, go back to Visual Studio. The Visual Studio team recommend following the guide to Create a folder in the project folder for the project. They don't like to have to create a normal folder when starting a project because you are going to the registry or anything. This is the easiest to create a separate folder. In the folder folder, choose the folder of the Visual Studio project and the target project. In directory of the folder project, move into the folder that is created. This is self-contained and will need the permission to rename the folder after the project name. If you want all the files available, it's free by choosing the target project folder which is a directory. Go to the created folder project. Create a new file in the root folder which is the root folder of the project. The only real way to launch this file from Visual Studio then as a shortcut. Let's do this in the way we're going to start this project. Here we create a new text editor. To name this editor, we need to use a prefix there within the project project name. The default command options in the Visual Studio IDE are – Add the name to the selected editor. Once you have created your text editor, you can start the PowerShell command. csm.PowerShell We launch the Microsoft PowerShell CLI using the command similar to "csm". In the command prompt, you can see the workspace content table and the start of the installation. To activate the Windows Project CLI, enter the command prompt: csmPowerShell start <uva!> Build the project component The file should be generated. To build the component using the components, you need to create the Project folder.The project folder in the folder named is 'Project'. The folder must have a similar name. Enter the new project in the command. Here you can see all the project files in these folder. Next, we have created the Project file component. The file is named 'project.component' and a 'project name' and a project path can be a directory. This is the named folder. Here we have created the Project Item component. In the folder created in the command prompt: csm.ProjectItem start<General Goal> Open the component Your browser should open the components.component.component, click "Open File", open the Settings folder. In the text box, click 'Test', and here you can see the components folder with folder and file. You now have opened this folder and you can quickly open it in Windows. You'll see something like the Windows Project Settings button. Now, a closer look of the folder will look blank: PowerShell Start Project Now, you really need to make sure everything works. Right now, put the name of this component in the File/Project.Exps inside it and select 'Project'. Open it in the File Explorer window, and enter it in the Project field you have created in the project folder. Folder/Projects It's now time to create the components in the Projects folder. In the Folder Settings window, click "Create new folder" and you can see that they have the same name as I mentioned. Then you need to select the File/Projects.ts folder in order to start building the component. Now it's time to build the component component. This is where you click "Select the Component Step". Click "Projects Step", and when this step completes, you should see a tooltip with the component that you are building using the PowerShell Core version of your application. It should be as an XML file or as a container that it will be using. To open the project, change the template to put "Scripts" in the project name and add the following: sourceProjectPrepare $BuildName = $Scripts.BuildName_Get + myModule.BuildInstall You can save this to the project folder, click in the project select, and click the button Apply in the context menu, right-click the project. Now you need to replace the empty main folder with the component "Folder.cs" Click "Now" button In the same dialog that you have created before, replace the component selected in the project menu and click close the project With the button click, we need to create getComponent.cs. To access it, you need to import "GetComponent.cs.ts". And then you need to change the URL (http://developer.com/GetModule.) instead of your web browser URL (http://developer.com/GetModule?). Now pass the components to your module called the getComponent.GetModule. Let's call function this: GetModule: Change the element of the varGetComponent variable, and add the #testCase variable so we can pass it into this Module object. With that, we can start to create the project files for the PowerShell Core application. Create a New Project With Name Create a New Project With Name Image Type Create a Project to CreateTag. BuildBuild Yourtags/buildtags.ts file: Change the default to the BuildNumber of projects to 1 and you need to select the myModule.BuildInstall and c:
package:1 Finally, we will look inside the project folder for the SPC React project. Creating sample React project For now, we have created two package called React.ps , each of which uses the same "test" package. react.ps calls<|endoftext|>

Figure 5: Sample from BD3-LM for block size $L' = 16$ of length $L = 2652$ under $T = 5\text{k}$ diffusion steps (trained with a context length of $L = 1024$). The generative perplexity of this sample under GPT2-Large is 26.0, and its entropy is 5.4.

<|endoftext|>, but Wilson took over the program and turned it around. “He’s done a tremendous job,” Caldwell said. “He’s done a fantastic job.” The offense has always had an abundance of weapons, but it became evident that they weren’t going to have a weapon to actually go after players from the slot. Now they’re in two different weapons sets. The top group features Dez Bryant and Mohamed Sanu, and the bottom group features an assortment of weapons and pass rushers. The job has become far more complex. The other players can make plays on the ball and get those targets at a higher rate. Sanu is more of a classic, get to the quarterback and leave the corner open. Dontari Poe got the job done this year and became one of the more effective players at the position, even in the passing game. However, Dallas has got to figure out how to get their franchise wideouts to contribute on the field. That can be tough. Adding Poe can help get the receiving corps going. C.J. Spiller is a two-time Pro Bowler, but if the Cowboys want to upgrade their receiving corps, he’s going to have to step up in a big way. “We’ve got to be a little more aggressive with the type of weapons that we have,” Caldwell said. “I think that’s part of the reason why our last two games, especially when you’re playing in Washington, D.C., you’ve got to be aggressive, make sure you’re hitting at every catch. When you are, you’re giving up a lot of yards.” Part of that means taking the quarterback out of the equation and having him beat coverage a lot more. In the NFC West, you want your offensive weapons to do a better job of running through coverage. The biggest threat that Dallas has is a QB in Ben Roethlisberger. Roethlisberger is far and away the best quarterback in the league, but a lot of the credit has to go to his receiver group. Martavis Bryant and Antonio Brown are both big-time receivers, and last year they were in the top 10 of yards per catch and receiving yards in the league. That production will never be sustainable, but if you’re going to be an elite offense, it’s going to take a lot of catching up. Roethlisberger is an All-Pro receiver, and he’s not the most dynamic option. But it would take something like Bryant or Brown at a better position, and at a slightly lower price, to make him the most productive receiver on the offense. The truth is that Roethlisberger isn’t going to be great. He may only have 18 games left in his career, but he’s been doing it since he was a rookie in 1991. But that’s not the worst thing in the world. Roethlisberger’s ability to hit guys on the outside with good movement, vision and running ability is what the Cowboys need in order to keep up with the competition. If he keeps getting better, he could become the best receiver in the league. Follow @walterfootball for updates. Like our Facebook page for more Cowboys news, commentary and conversation.The owner of 1H10 Tree in Charlotte Gardens is taking legal action against the city. Derek Jarman says he’s been forced to evict his neighbour, Bob, after he took to social media to threaten to burn down his neighbor’s house. "I’m incredibly furious with the city," Jarman told 7.30. "I’ve been trying to keep my eyes on the prize." Tree in Charlotte Gardens saying it had seen ’9,000+ people’ enjoying a great weekend The company that owns 1H10 named Bob after a bee and said the tree was frequently targeted because of its unusual location. Bob said he had his concerns about the tree when he was contacted in October. He said they had had ’an ongoing conversation about my neighbor. He called, hung up and he was very threatening’ in the 30 days before they turned the tree over to him. A neighbour posted the following online message on 8 October. "I am shocked about the serious problems you are having with your neighbour that has caused you all (sic). You and the 2 of you are making money at the expense of the good people of Charlotte Gardens." Bob says he was furious and said he’d just got off the phone with the city manager. "I told her, ’no, I’m going to bring a lawsuit’, and I called the solicitor and tried to get my phone, just hoping the solicitor would help me out. I called again, and I asked if I could go to court and to try and get an injunction. "They told me ’you cannot’, and they said, ’we can’t, we can’t’ because you’re sending people to the police’." Tree in Charlotte Gardens (Facebook) He also said he’d threatened the city attorney if he didn’t stop the building from burning down. The internet user tweeted: "I’s on the tree, but after I said ’threw this away, here’s a spot to burn’, the building started to burn." Tree in Charlotte Gardens (Facebook) Bob said the neighbour had threatened to burn down the tree, the windows, the living room and his entire backyard. "It was more than a threat," he said. "He was a very strong person. He’s already damaged so many people in this building. It’s not going to go away." Tree in Charlotte Gardens (Facebook) Jarman says he tried to talk the building owner out of the move, but the building owner’s behaviour had "deleted him." "I’m going to stop him by letter telling him not to come to my house any more," he said. "I have three kids, and if Bob is going to be in my house, I need to make sure I have someone who can go in there and protect me. "My son does a really good job of protecting me, and I’m not going to let that get in the way of that." Tree in Charlotte Gardens (Facebook) Jarman said Bob had pulled him up on social media, calling him a "white nut" and saying: "For God’s sake, stop calling me a white nut. "I should have shut him up on Facebook." He said he sent Bob the letter and thanked him for the support. "He should have done it because he’s a real artist and he’s a real artist," he said. He declined to name the architect of the new tree, but says the firm is the same one that designs buildings. ’The building is burning down’, neighbour says Bob’s neighbour, Michael Banks, says the fire is an insult to his daughter. "There are two black women that live next door to me and they told me ’you can’t do that’, and then the fire went up and then the building burnt down," he said. "You can’t burn down a house if you don’t burn down the house." Coun-Pete Lawrence, the Northumberland MP for Wood Green, says he has concerns about Bob’s neighbours. "It’s a very, very sad commentary on the state of society and democracy in general," he said. "It’s interesting in a community that’s 50,000-plus people, you’ve got your regular residents and well-meaning neighbours who are apparently oblivious to the destruction of their own home. "To me, that’s appalling and it is probably a shocking amount of devastation that it’s left behind. "I would expect there to be outrage as well." Bob Jarman fears for his life after the tree was torched Bob says he has told the Northumberland Council that he had already received $1,000 in legal action from the building owner, when he told them about the incident. The building owner has declined to comment on the situation. The builder is currently assessing its legal options. "We’ve got to sort this out and have an understanding with the builder, Mr Banks," he said. "We’ve got to make sure we can’t get into into a legal battle with that person and make that person change his mind. "We don’t want to do anything to cause a scene or anybody in the street to be upset." Bob Jarman hopes to have an understanding with the builder on its legal options, who have refused to comment. Topics: state-parliament, smoking-and-doubt, black-wales-6168, united-kingdom, england First postedWhen the other guys are away playing, do a short commercial to get you fired up for the next work day. Once you make it home, get a few junkies for them. They’ll be very happy to have you, for at least a day. They might not be so happy after a couple of days. Have a bunch of friends and get ready to keep it going. What are you waiting for? Make this long, one-off<|endoftext>

Figure 6: Sample from an AR model (Sahoo et al., 2024a) with length $L = 2003$ (trained with a context length of $L = 1024$). The generative perplexity of this sample under GPT2-Large is 10.6 and its entropy is 5.5.