Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models

Ntountounakis Georgios, Markoulidakis Georgios, Vitalis Petros,
Makras Ilias, Kritharidis Konstantinos, Kordas Nikolaos

Pattern Recognition, ECE
National Technical University of Athens

January 2026

Table of Contents

Table of Contents

Introduction to the Problem-Motivation

Two main approaches for Language Models:

Autoregressive (AR):

- Token-by-token generation
- High quality
- KV caching
- Variable length

Diffusion:

- Parallel generation
- Better controllability
- Fixed length (limitation)
- Lower quality (Perplexity Gap)

Question

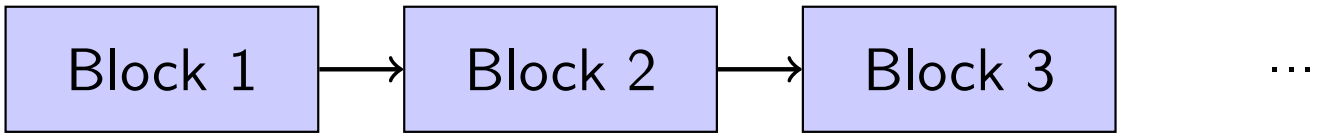
Can we combine the advantages of both approaches?

Core Idea: Block Diffusion

Key Innovation

Split sequence into **blocks** and apply:

- Autoregression **between** blocks
- Diffusion **within** each block



Diffusion within block(parallel)

Autoregressive over blocks

Parameterization: Block size L' controls the trade-off

- $L' = 1 \rightarrow$ Pure AR
- $L' = L \rightarrow$ Pure Diffusion

Block Diffusion Training Algorithm

Challenge: To compute the loss $\mathcal{L}_{BD}$ for block b:

- **Noisy Pass:** The model must process $\mathbf{x}_t^b$ to learn denoising.
- **Clean Pass:** The model needs the clean $\mathbf{x}^b$ to provide context for subsequent blocks.

Naive Solution: Every block must pass through $\mathbf{x}_\theta$ at least *twice*.

Algorithm 1 Block Diffusion Training

Input: $\mathbf{x}$, B, q_t, $\mathbf{x}_\theta$, $\mathcal{L}_{BD}$

1: **repeat**
2: Sample $t_1, \ldots, t_B \sim U[0, 1]$
3: $\mathbf{x}_{t_b}^b \sim q_{t_b}(\cdot|\mathbf{x}^b)$
4: $\emptyset, K_{1:B}, V_{1:B} \leftarrow \mathbf{x}_\theta(\mathbf{x})$ ▷ *Clean Pass*
5: $\mathbf{x}_{\text{logit}}^b \leftarrow \mathbf{x}_\theta^b(\mathbf{x}_{t_b}^b, K_{1:b-1}, V_{1:b-1})$ ▷ *Noisy Pass*
6: Grad step $\nabla_\theta \mathcal{L}_{BD}$
7: **until** converged

Why this design?

- **Indep.** t_b**:** Reduces variance
- **KV caching:** 5× speedup
- **Vectorized:** FlexAttention parallelism

Block Diffusion Sampling Algorithm

Challenge: Standard diffusion models cannot generate variable-length sequences.

Algorithm 2 Block Diffusion Sampling

Input: # blocks B, model $\mathbf{x}_\theta$, diffusion sampling algorithm SAMPLE

1: $\mathbf{x}, K, V \leftarrow \emptyset$ ▷ *Initialize output & cache*
2: **for** $b = 1$ to B **do**
3: $\quad \mathbf{x}^b \leftarrow \text{SAMPLE}(\mathbf{x}_\theta^b, K_{1:b-1}, V_{1:b-1})$ ▷ *Parallel block production*
4: $\quad (K, V) \leftarrow (K \oplus K_b, V \oplus V_b)$ ▷ *Extend cache for new block*
5: **end for**
6: **return** $\mathbf{x}$

Why this design?

- **Autoregressive over blocks:** Generate arbitrary length sequences
- **Diffusion within blocks:** Parallel token generation $\rightarrow$ faster text generation
- **KV caching:** Reuse computation from previous blocks $\rightarrow$ inference efficiency
- **Result:** 40$\times$ fewer steps than SSD-LM with better quality

The Problem: Perplexity Gap

Question: Why do diffusion models have worse perplexity?

Observation

For $L' = 1$ (block size $= 1$), the diffusion objective is equivalent to the AR objective in expectation.

But in practice:

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

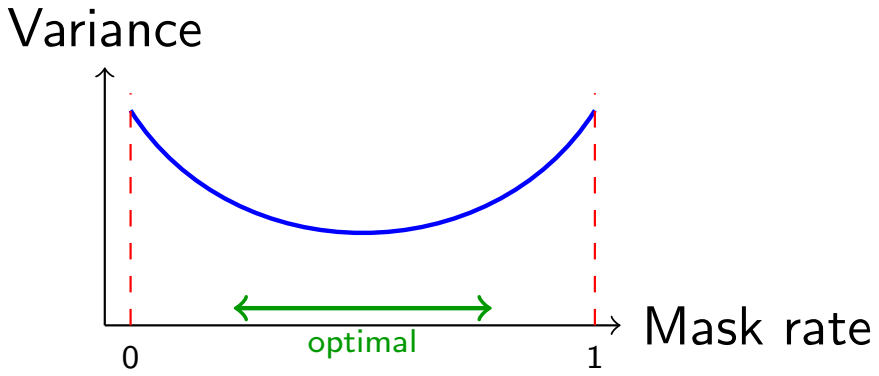
Masking $\Rightarrow$ diffusion training estimates loss gradients with 2x fewer tokens.
Conclusion: The problem is not the objective, but the **variance**!

The Solution: Clipped Noise Schedules

Goal: Design schedules that minimize gradient variance (Section 5)

Why extreme mask rates are bad:

- Too few masked (near 0): Easy task, no learning signal
- All masked (near 1): Predict marginals, doesn't help generation
- Both cause **high variance gradients**



The Solution: Clipped Noise Schedules

Clipped Schedules

$$1 - \alpha_t \sim U[\beta, \omega]$$

- β: Min mask rate, ω: Max mask rate
- Avoids extremes near 0 or 1

Examples:

- $U[0, 1]$: Standard linear
- $U[0.3, 0.8]$: Clipped
- $U[0.45, 0.95]$: Heavy masking (best for $L' = 4$)

Key Finding

There exists a unique distribution for each block size, that minimizes both the variance of the NELBO and the PPL.

Main Experimental Results

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

	PPL (↓)
AR	17.54
SEDD	$\leq$ 24.10
MDLM	$\leq$ 22.98
BD3-LMs $L' = 16$	$\leq$ 22.27
$L' = 8$	$\leq$ 21.68
$L' = 4$	$\leq$ **20.73**

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

	Median # tokens	Max # tokens
OWT train set	717	131K
AR	4008	131K
SEDD	1021	1024
BD3-LM $L' = 16$	798	9982

Table of Contents

Implementation Phases

Phase	Description	Weeks
1	Paper & codebase study	1-2
2	Environment & data setup	2-3
3	Initial reproduction	3-5
4	Final reproduction	5-6
5	Extension experiments	6-7
6	Analysis & Writing	7-8

Current Status

Phase 3 (Initial reproduction): running a small-scale setup (tiny model + reduced dataset) to validate the full pipeline.

Reproduction

Goal: Reproduce the paper's training/inference pipeline in a *controlled* setting.

Training datasets and tokenizers

Training dataset	**Tokenizer**
LM1B	`bert-base-uncased`
OWT	`GPT2`

Reproduction (experimental setup)

Experimental Setup (Small-scale)

Configuration	Choice
Training Samples	500
Training Steps	800
Evaluation Samples	100
Model Size	`tiny`
Context Length	128

What we will report:

- Training loss curves and evaluation perplexity (PPL)
- Sampling sanity checks: coherence vs. speed (block size trade-off)

Expected outcome: correct trends and stable training (not necessarily paper-level absolute PPL due to limited resources)

Potential Extensions

Extension 1: Diffusion Optimization

- Optimize elements of the diffusion process to improve quality and/or performance.

Extension 2: Adaptive Block Size

- Adaptively change the block size according to the need for fine-grained detail of each block.

Extension 3: Training / Fine-tuning on new, resource-friendly datasets

- **AG News**: news-domain text for domain adaptation
- **LAMBADA**: long-range dependency evaluation
- **Penn Treebank (`ptb`)**: classic language modeling benchmark
- **BabyLM Corpus**: child-language data for acquisition modeling

Experimental Results (Reproduction)

Table 3 Initial Reproduction: Test Perplexities

	PPL (↓)
Autoregressive	
Transformer	1221
Diffusion	
SEDD	$\leq$ 1403
MDLM	$\leq$ 1370
Block diffusion (Ours)	
BD3-LMs $L' = 16$	$\leq$ 1345
$L' = 8$	$\leq$ 1210
$L' = 4$	$\leq$ **1176**

Table of Contents

Computational Resources & Data

Computational Resources:

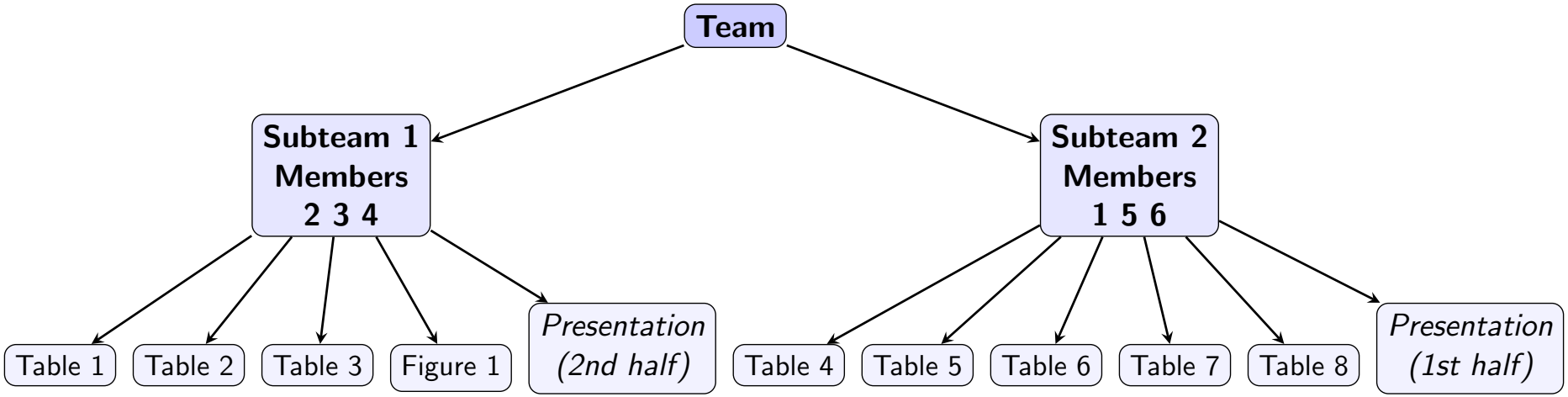
- Google Colab (Primary environment)
- Kaggle Notebooks
- our GPUs

Datasets:

- LM1B
- OWT

Table of Contents

Team Organization and Structure



Coordination:

- Weekly team meetings
- Subteams internal coordination
- GitHub repo
- Discord server

Member Roles

Phase	Description	Members
1	Paper & codebase study	Everyone
2	Environment & data setup	Georgios M., Petros
3	Initial reproduction	Georgios M., Petros, Ilias
4	Final reproduction	Georgios Nt., Konstantinos, Nikolaos
5	Extension experiments	Everyone
6	Analysis & Writing	Everyone

Thank You!

Questions?

Paper: Block Diffusion (ICLR 2025)
Code: `https://github.com/kuleshov-group/bd3lms`
Our Repo: `https://github.com/ntua-el21050/bd3lms`
Authors: Arriola, Gokaslan, Chiu, Yang, Qi, Han, Sahoo, Kuleshov