Auto-Regressive Masked Diffusion Models (ARMD)

Mahdi Karami
Ali Ghodsi

Outline

1) Background

Bridging the Modeling Dilemma

ARM:
Slow Sampling

Sequential Generation:
Factorization

Inefficient to Scale?

DLM:
Inefficient Training

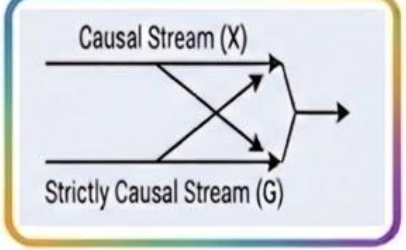
Parallel sampling
Lagging Efficiency

Goal: Unify efficiency & fast sampling

2) Method:

Strictly Causal Attention & Two-Stream Attention

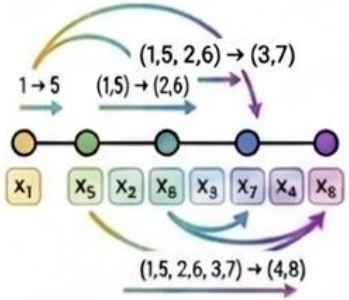
- Strictly Causal Attention
  uses SC mask to restrict attention



- Two-Stream Architecture
  implements deep SC efficiently via shared parameters/two streams

3) Sample Generation

Strided Block-Parallel (SBP) Generation



4) Results

Key Performance & Efficiency Results

- Significantly Fewer Training Steps
- SOTA Parallel Sampling Time
- Top Perplexity on LM1B dataset
- Top Zero-shot PPL

The Modeling Dilemma

Autoregressive Models (ARMs):

✚ Demonstrate strong performance and training efficiency.
− Inherent sequential Generation: slow; struggles with bidirectional context or non-sequential reasoning

Diffusion Language Models (DLMs):

✚ Supports parallel generation and non-sequential reasoning
− Lag behind ARMs in performance and training efficiency.

Motivation

The Goal:

- Bridge the gap by unifying the training efficiency of ARMs with the parallel generation capabilities of DLMs.
- Create a model that supports efficient, autoregressive-style training and fast parallel sampling.

✨ARMD

✚ Reframes masked diffusion as a ***block-wise causal model***.

✚ Computes all conditional probabilities for multiple denoising steps in a ***single, parallel forward pass*****.**

✚ **Result:** Achieves SOTA performance with significantly fewer training steps, setting a new SOTA for parallel text generation.

Background

- **ARMs:** factorize the joint probability of a sequence into a product of univariate conditionals using the chain rule

$$\log p(\boldsymbol{x}_{1:N}) = \sum_{t=1}^{N} \log p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}),$$

Parallelized Training: Thanks to causal masking, ARMs can evaluate the full log-likelihood for a sequence in ***a single forward pass*****.**

- **Masked diffusion models (MDMs):** Require *many network calls per each training example* to evaluate different denoising steps.

$$\mathcal{L}_{\text{diff}} = \sum_{t=1}^{T} \mathbb{E}_q \left[\frac{\alpha^t - \alpha^{t-1}}{1 - \alpha^t} \sum_{\{j \mid \boldsymbol{z}_j^t = \boldsymbol{m}\}} \log p_\theta(\boldsymbol{z}_j^{t-1} \mid \boldsymbol{z}_{1:N}^t) \right]$$

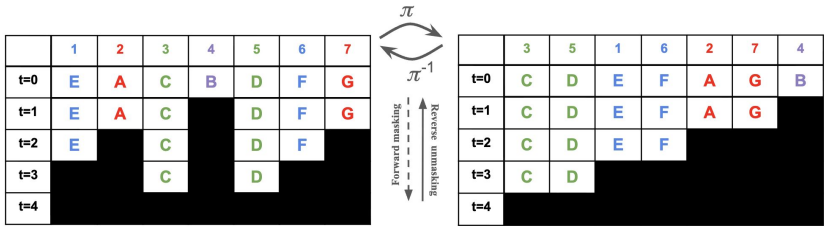
Method

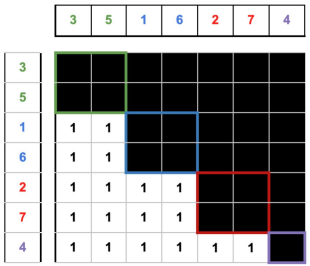
Strictly Causal Masked Diffusion Models

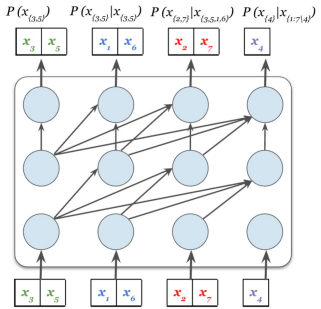
- Sorting tokens by their "unmasking" time, reveals a **block-triangular strictly causal structure**. The MDM loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_q \Big[\sum_{t=1}^{T-1} \gamma(t) \sum_{n \in \mathcal{B}(t)} - \log p_\theta(\boldsymbol{x}_n \mid [\boldsymbol{x}_i \mid \mathcal{B}(i) \leq T - t]) \Big]$$

- This MDM loss can be computed in **a single, parallel forward pass**, adopting a *Strictly Causal Model,* $y_n = p(x_n|\text{previous blocks}) = f^{sc}(x_{<\mathcal{B}(n)})$







Strictly Causal Attention

- Unlike ARMs, which is based on causal attention, ARMD uses ***Strictly Causal Attentions***.

- We apply an SC attention mask, Msc, that restricts attention only to keys/values from preceding blocks.
- We define a modified query as a SC function: $\hat{\boldsymbol{q}}_n = f_q^{sc}\left(\{\boldsymbol{q}_i \mid \mathcal{B}(i) < \mathcal{B}(n)\}\right)$

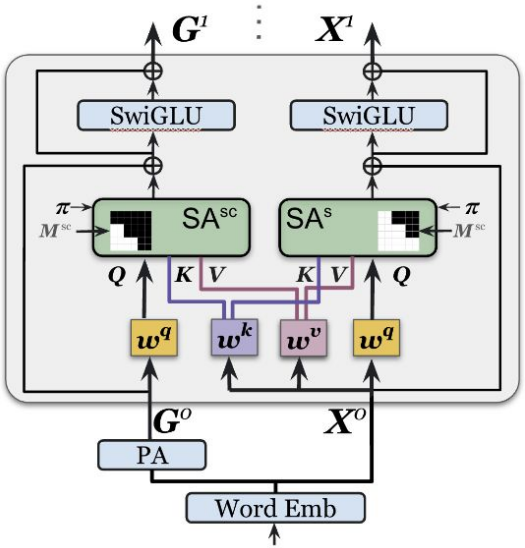
$$\boldsymbol{y}_n = \mathrm{SA}^{sc}\left([\boldsymbol{x}_i \mid \mathcal{B}(i) < \mathcal{B}(n)]\right)[n] = \sum_{i:\mathcal{B}(i)<\mathcal{B}(n)} \mathrm{Softmax}_i\left(\hat{\boldsymbol{q}}_n^\top \boldsymbol{K}^\top + M^{sc}_{(n,i)}\right)\boldsymbol{v}_i,$$

Two-Stream Strictly Causal-Attention

To implement *a deep SC architecture efficiently*, ARMD uses a Two-Stream architecture, *with parameter sharing:*

1. ***Causal Stream (X)*:** Standard self-attention for a causal Keys/Values context.
2. ***Strictly Causal Stream (G):*** Uses a modified SC query and SC attention mask.

$$\boldsymbol{X}^l = \mathrm{SA}_\theta^c(\boldsymbol{X}^{l-1}; \boldsymbol{M}^c) = \mathrm{Attn}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{M}^c\right),$$
$$\boldsymbol{G}^l = \mathrm{SA}_\theta^{sc}(\boldsymbol{G}^{l-1}, \boldsymbol{X}^{l-1}; \boldsymbol{M}^{sc}) = \mathrm{Attn}\left(\hat{\boldsymbol{Q}}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{M}^{sc}\right)$$

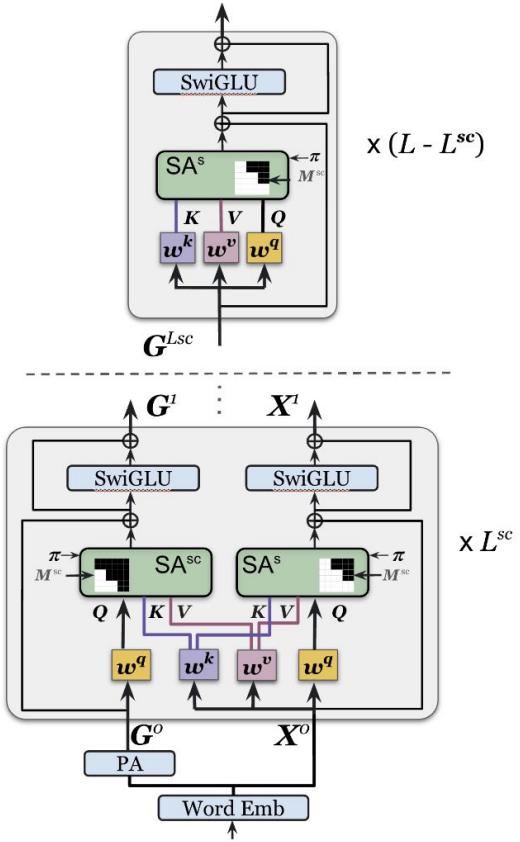


Two-Stream Strictly Causal-Attention

The final model is composed of

1. an L^{2s}-layer two-stream stack and
2. a subsequent $(L - L^{2s})$-layer standard causal attention.

- Composing a strictly causal layer with any number of causal ones ($f^c \circ f^{sc}$) yields a strictly causal network.



Strided Block-Parallel (SBP) Generation

- To satisfy approximate *conditional independence*

$$p(x_2, x_6 \mid context) \approx p(x_2 \mid context)p(x_6 \mid context)$$

We *maximizes the physical distance* between tokens simultaneously generated in parallel

- ***Partition*** *into S equal-sized streams*
- **interleave** streams: *grouping tokens by their relative position*

For example: N = 8, S = 2:

streams: S1 = $[x_1, x_2, x_3, x_4]$, *S2 =* $[x_5, x_6, x_7, x_8]$

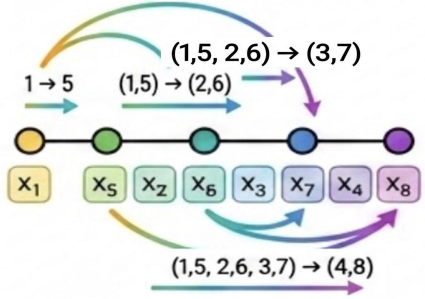
→ *permutation* $\pi = [1, 5, 2, 6, 3, 7, 4, 8]$

1. The _ _ _ _ _ _ _
2. The _ _ _ jumps _ _ _
3. The quick _ _ jumps over _ _
4. The quick brown _ jumps over the _
5. The quick brown fox jumps over the dog

Strided Block-Parallel (SBP) Generation

1. Partition

The target sequence is evenly split into **S parallel**, interleaved streams.



2. Anchoring

The **"head" (first token)** of each stream is generated sequentially to establish context.

3. Extension

Subsequent tokens are **generated in simultaneous parallel strides** across all streams.

1. The _ _ _ _ _ _ _
2. The _ _ _ jumps _ _ _
3. The quick _ _ jumps over _ _
4. The quick brown _ jumps over the _
5. The quick brown fox jumps over the dog

4. Acceleration

Speeding up sampling by factor S with maximum distance between parallel tokens,

Results

Zero-shot language modeling perplexity (↓): pretrained on the OpenWebText dataset

Method	LAMBADA	WikiText2	PTB	WikiText103	1BW
small size models					
GPT-2*	45.04	42.43	138.43	41.60	75.20
D3PM*	93.47	77.28	200.82	75.16	138.92
PLAID (600K)*	57.28	51.80	142.60	50.86	91.12
SEDD-Uniform (400K)*	65.40	50.27	140.12	49.60	101.37
SEDD (400K)*	50.92	41.84	114.24	40.62	79.29
RADD (400K)*	51.70	39.98	**107.85**	37.98	72.99
ARMD (180K)	**44.66**	36.25	130.31	35.66	53.18
ARMD (400K)	45.35	**35.64**	123.43	**35.15**	**52.94**
medium size models					
GPT-2*	**35.66**	31.80	123.14	31.39	55.72
SEDD (400K)*	42.77	31.04	87.12	29.98	61.19
RADD (400K)*	44.10	30.60	**82.08**	29.29	60.32
ARMD (120K)	40.62	27.57	105.09	27.09	45.49
ARMD (300K)	39.08	**26.06**	97.75	**25.55**	**43.91**

Test perplexities (ppl; ↓) on One Billion Words (LM1B) dataset

Method	ppl (↓)
Autoregressive	
Transformer-XBase*	23.5
Transformer*	22.83
Diffusion	
D3PM-absorb*	82.34
Diffusion-LM*	118.62
DiffusionBert*	63.78
SEDD (1M)*	32.68
MDLM (1M)*	31.78
BD3-LMs L'= 16 (1M)*	30.60
BD3-LMs L'= 4 (1M)*	28.23
ARMD (300K)	23.64
ARMD (1M)	**22.36**

Sample Generation vs. Block Size *S*: Performance is reported as Perplexity(↓) + (Entropy). ARMD's average sampling time per sequence is *3.51s* for *S = 1, 1.78s* for *S = 2,* and *0.93s* for *S = 4* on one H100 GPU.

Model	T=1024 (Sequential)	T=512 (S=2)	T=256 (S=4)
AR Baseline*	∼36 (8.1)	-	-
SEDD*	∼106 (8.1)	∼110 (8.1)	-
MDLM*	∼103 (8.1)	∼113 (8.1)	-
ARMD (ρ=32, i_{SBP}=10K)	38.7 (8.0)	43.2 (8.0)	47.8 (8.1)
ARMD (ρ=32, i_{SBP}=50K)	39.1 (8.0)	40.1 (8.1)	43.2 (8.1)
ARMD (ρ=32, i_{SBP}=80K)	**36.5** (7.9)	**37.8** (8.0)	**39.4** (8.0)
ARMD (ρ=16, i_{SBP}=10K)	50.0 (8.1)	54.6 (8.1)	64.7 (8.2)
ARMD (ρ=16, i_{SBP}=50K)	45.7 (8.0)	46.1 (8.1)	50.1 (8.1)
ARMD (ρ=16, i_{SBP}=80K)	39.4 (8.0)	40.5 (8.1)	43.7 (8.1)

Summary

1) Background

Bridging the Modeling Dilemma

ARM:
Slow Sampling

Sequential Generation: Factorization

Inefficient to Scale?

DLM:
Inefficient Training

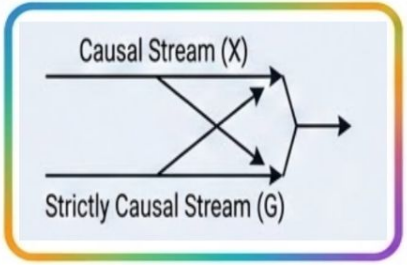
Parallel sampling
Lagging Efficiency

Goal: Unify efficiency & fast sampling

2) Method:

Strictly Causal Attention & Our Architecture: Two-Stream Attention

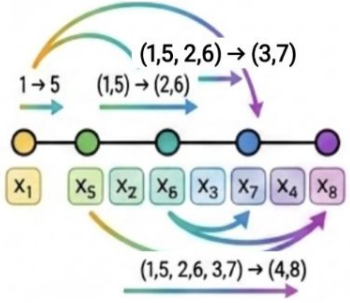
- Strictly Causal Attention
  uses SC mask to restrict attention



- Two-Stream Architecture
  implements deep SC efficiently via shared parameters/two streams

3) Sample Generation

Strided Block-Parallel (SBP) Generation



4) Results

Key Performance & Efficiency Results

- Significantly Fewer Training Steps (~180K/400K vs 1M)
- SOTA Parallel Sampling Time: ~1s vs ~3.5s
- Top Perplexity on LM1B dataset (~22.4)
- Top Zero-shot PPL (<26.1)