Efficient-DLM

From Autoregressive to Diffusion Language Models, and Beyond in Speed

Paper by Yonggan Fu, Lexington Whalen, et al.

Research group paper discussion

arXiv:2512.14067v2 | Revised 29 April 2026

Reading prepared 14 October 2026

Match training structure to block decoding, then learn reliable parallel generation.

Parallel generation needs both reuse and reliable predictions

Research question

Can a pretrained AR model become a faster dLM while retaining its task abilities?

- AR decoding reuses KV states, but usually emits one token per forward pass.
- Fully bidirectional denoising can predict many positions, but complicates KV reuse.
- Small batches expose a practical opportunity: more accepted tokens per pass can amortize weight reads.

AR initialization $\longrightarrow$ block denoising $\longrightarrow$ confidence-based parallel decoding

Paper, Sec. 1, PDF pp. 1–2. Small-batch motivation; not a claim about all serving regimes.

The recipe addresses two training–inference gaps

Across blocks: context state

Training on corrupted prefixes differs from decoding after the prefix is complete.

Use clean preceding blocks.
Retain block causality and enable native KV caching.

Within a block: mask positions

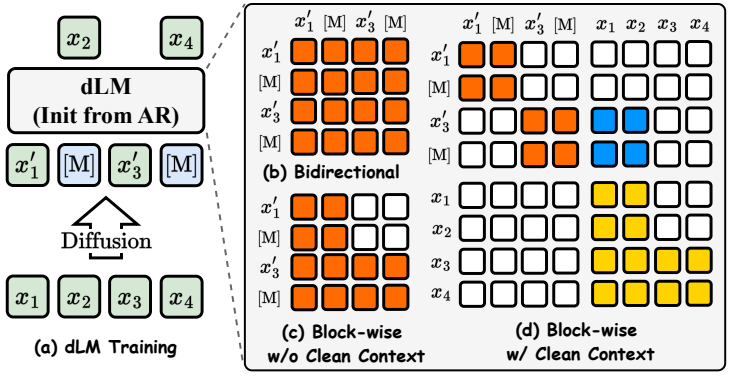
Uniform training masks differ from the masks left by confidence-based decoding.

Bias masks toward later positions.
Strengthen this bias as corruption decreases.

Keep the AR initialization; adapt its attention, prediction alignment, and masking distribution.

Paper, Secs. 2–3, PDF pp. 2–7. Block diffusion and confidence decoding build on prior work.

Clean-prefix block attention restricts where answers are visible



Noisy queries see their own block and earlier clean blocks only.

The clean copy of the target block is hidden from its denoising queries.

Paper, Fig. 2(a–d), PDF p. 3; original panels cropped, labels retained. Orange: noisy; blue: clean prefix; yellow: clean branch.

The denoising objective conditions each block on a clean prefix

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\,\mathbb{E}_{\tilde{\mathbf{x}}_t^b\sim q(\cdot|\mathbf{x}^b)}\left[-\frac{1}{t}\sum_{b=1}^{B}\log p_\theta(\mathbf{x}^b \mid \tilde{\mathbf{x}}_t^b, \mathbf{x}^{<b})\right].$$

- $\mathbf{x}$ has length L; B blocks have length $L' = L/B$.
- $\tilde{\mathbf{x}}_t^b$: current corrupted block; $\mathbf{x}^{<b}$: clean preceding blocks.
- q: masking process; t: noise level; $1/t$: loss weighting.
- Initialize θ from AR weights, then continue pretraining with this objective.

Prediction alignment

Remove the AR token shift: recover a masked token at its own position.

Paper, Eq. (1), Secs. 2.1–2.2, PDF pp. 3–4. Block likelihood is the paper's denoising notation.

Completed blocks remain cacheable during later decoding

Teaching example: denoise the second block

`[A B] | [C MASK] | [E F]`

Visible: `A, B, C` Hidden answers: `D, E, F`

- Training concatenates noisy and clean streams with a special attention mask.
- At inference, finish one block, retain its prefix KV states, and advance.
- Within the active block, accept multiple sufficiently confident predictions per step.

Clean means unmasked. Generated prefix tokens may still contain errors.

Teaching illustration derived from Fig. 2(d) and Sec. 2.1, PDF p. 3; confidence decoding: Appendix A, p. 13.

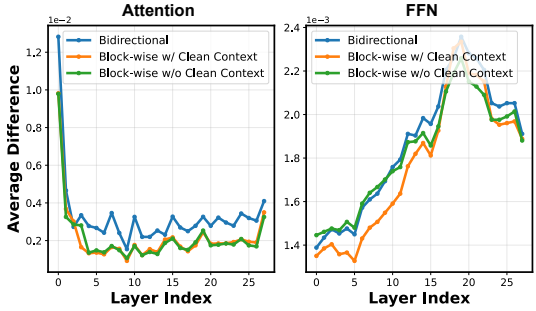
Clean context recovers much of the accuracy lost in conversion

Attention / budget	Prefix	Shift	Avg. (%)
Original AR	clean	yes	41.79
Bidirectional, 50B	noisy	yes	18.10
Bidirectional, 50B	noisy	no	19.29
Block, 50B	noisy	yes	28.23
Block, 100B	noisy	yes	27.73
Block, 50B	clean	yes	37.69
Block, 50B	clean	no	**38.41**

- Clean prefix: +9.46 pp; removing shift afterward: +0.72 pp.
- Doubling noisy-prefix tokens does not replace a clean-prefix objective.

Paper, Table 1, PDF p. 4. Qwen2.5 1.5B; block size 16; six generation tasks. pp = percentage points.

Block attention is associated with smaller weight changes

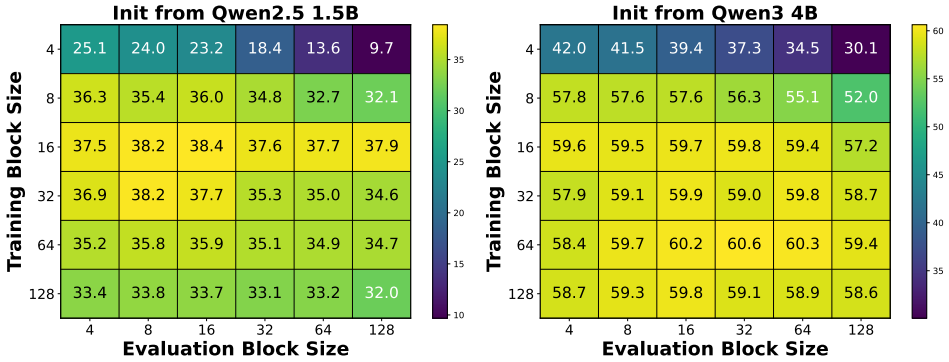


(e) Weight Change after Continuous Pretraining

- Full bidirectionality is associated with larger attention-weight changes.
- Clean context reduces FFN drift relative to a noisy-prefix block model.

Paper, Fig. 2(e), PDF p. 3. Layer index vs. average weight difference; y-axis scales differ. Association, not an isolated causal test of drift.

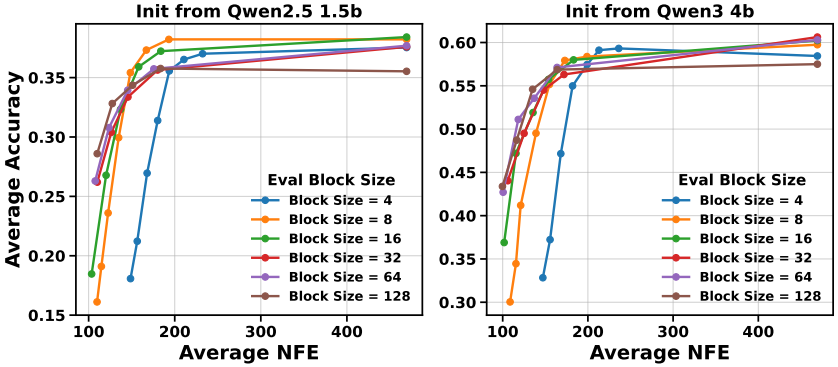
A useful training block size balances context and corruption



- Rows: training block size; columns: evaluation size; cells: six-task accuracy.
- The study selects 16 for 1.5B and 64 for 4B; good rows transfer across columns.
- Balance richer bidirectional context against heavier corruption.

Paper, Sec. 2.3, Figs. 3–4, PDF pp. 4–5. Budgets: 50B tokens (1.5B), 25B (4B).

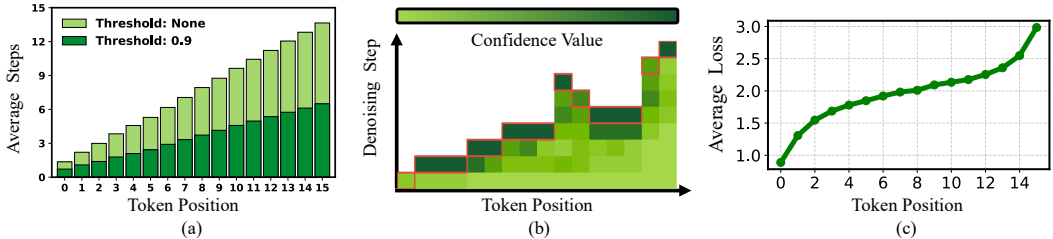
Larger evaluation blocks help when decoding is more aggressive



At low NFE, more candidate positions can support better parallel choices.
At high NFE, moderate block sizes give similar accuracy.

Paper, Fig. 5, PDF p. 5. x: average model evaluations (NFE); y: six-task accuracy. NFE is not wall-clock throughput.

Confidence decoding leaves later positions masked for longer



- Left: later positions need more steps, with or without parallel decoding.
- Middle: confidence rises as neighboring tokens become available.
- Right: later masked positions have higher average training loss.

Paper, Fig. 6, PDF p. 6. Qwen2.5 1.5B, block size 16; GSM8K decoding; loss averaged over blocks in 200 samples.

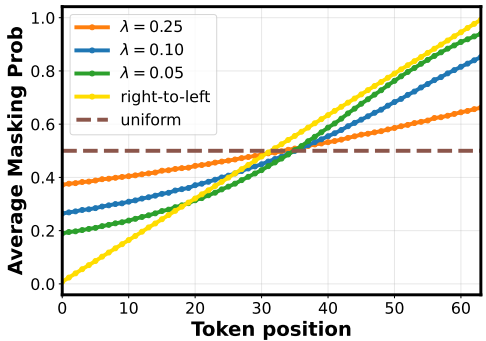
Position-dependent masking tilts a fixed-size mask set

$$w_i(t) = \exp[\beta(1-t)i]$$

$$k = \mathrm{round}(tL'), \qquad \lambda = \frac{\ln 2}{\beta L'}$$

Normalize weights; sample k distinct positions with Gumbel-top-k.

- High noise: nearly uniform weights.
- Low noise: favor later positions.
- Smaller λ: stronger tilt.



$w_i(t)$ is an unnormalized sampling weight, not an independent mask probability.

Paper, Eq. (2), Fig. 7, Secs. 3.2–3.3, PDF pp. 6–7. Final models use $\lambda = 0.1$.

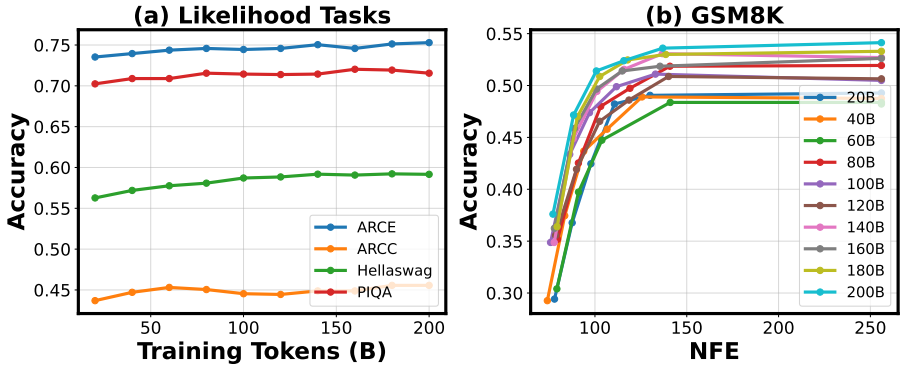
A moderate positional bias helps; an extreme suffix mask fails

Masking	TPF 1	TPF 2.8	TPF 4	TPF 5.6
Uniform	60.27	57.12	51.11	33.99
$\lambda = 0.25$	60.41	58.56	50.97	34.55
$\lambda = 0.10$	**62.02**	**58.79**	**53.75**	**38.37**
$\lambda = 0.05$	60.51	57.68	53.06	37.38
Rightmost k only	38.21	27.55	18.60	14.13

- $\lambda = 0.1$: $+1.75$ pp at TPF 1; $+4.38$ pp at TPF 5.6.
- Stronger bias is not monotonically better; preserve context diversity.
- TPF 5.6 remains a low-accuracy operating point despite its improvement.

Paper, Table 2, PDF p. 7. Qwen3 4B, 25B tokens, training block 64; six-task mean accuracy (%).

Longer training can improve the parallel-decoding frontier



- Likelihood estimation improves overall; fewer evaluations can become sufficient.
- Check the full accuracy–NFE curve: individual tasks and checkpoints fluctuate.

Paper, Fig. 8(a–b), PDF p. 7; Qwen2.5 1.5B up to 200B tokens. HumanEval fluctuations appear in Fig. 8(d).

Final models use 300B–500B additional training tokens

Model	Tokens	AR initialization	Train block	Eval block
Efficient-DLM 1.5B	300B	Qwen2.5 1.5B	16	16
Efficient-DLM 4B	300B	Qwen3 4B	64	32
Efficient-DLM 8B	500B	Qwen3 8B	64	32

- All: clean-prefix block attention, no shift, $\lambda = 0.1$; training on 128 H100s.
- Quality: 12 tasks = 4 coding + 2 math + MMLU + 5 commonsense tasks.
- TPF and throughput: averaged over the six generation tasks, on one H100, batch 1.

Paper, Sec. 5 and Table 3, PDF p. 8; Appendix A, p. 13. Final budget differs from the early accuracy-recovery observation.

Fast 8B vs. Qwen3 8B: 3.07× throughput, −0.93 pp

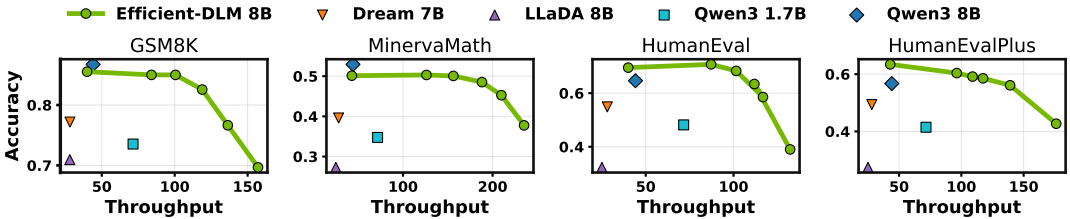
Model	TPF	Tokens/s	Avg. (%)
Qwen3 4B	1.00	47.13	67.97
Qwen3 8B	1.00	42.51	71.58
Dream 7B	1.00	28.11	65.30
Efficient-DLM 8B	1.00	39.99	71.62
Efficient-DLM 8B	2.74	109.78	70.93
Efficient-DLM 8B	3.27	**130.71**	**70.65**

Versus Qwen3 4B: 2.77× throughput and +2.68 pp.

At TPF 1, accuracy is retained, but throughput is slightly below Qwen3 8B.

Paper, Table 3, PDF p. 8: H100, batch 1. Ratios recomputed from this table. Dream ratio is 4.65×; Sec. 5.1 says 4.50×. Table 6 gives a different Qwen3 4B TPS, so it is not mixed here.

One set of weights supplies several accuracy–speed choices



- Change the confidence threshold, not the model weights.
- Higher throughput eventually incurs a substantial quality drop.
- Inspect task-specific curves: the 12-task mean also includes unchanged likelihood scores.

Paper, Fig. 9 and Sec. 5.2, PDF p. 9; aggregation protocol: Appendix A, p. 13. High-accuracy performance is not best on every task.

Attention and clean context deliver the largest ablation gain

Cumulative Qwen3 4B conversion	Avg. (%)	Gain (pp)
Bidirectional baseline, 25B tokens	44.59	—
+ Block attention and clean prefix	59.01	+14.42
+ Remove token shift	60.27	+1.26
+ Position-dependent masking	62.02	+1.75
+ Extend training to 300B tokens	64.05	+2.03

- Fixing the training structure precedes gains from masking and scale.
- These are cumulative increments; individual tasks do not all improve at each step.

Paper, Table 5, PDF p. 10. Six coding/math tasks. Adding position bias lowers both math scores in this ablation but raises the overall mean.

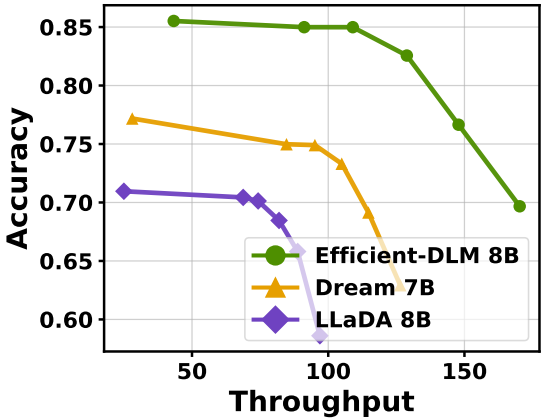
Text embeddings also benefit, with task-dependent gains

Model	Category avg.	Pair classification	Retrieval
Qwen2.5 1.5B	29.54	24.59	20.69
Efficient-DLM 1.5B	37.25	56.76	18.67
Qwen3 4B	30.79	33.94	19.46
Efficient-DLM 4B	40.70	65.59	20.17

- Zero-shot MTEB: 15 datasets, six categories; last-layer mean pooling.
- Qwen uses causal attention; Efficient-DLM uses fully bidirectional attention here.
- Mean gains: $+7.71$ / $+9.91$ score points; 1.5B retrieval decreases.

Paper, Table 4, PDF p. 9; Appendix A, p. 13. Avg. equally weights six category scores; metrics vary by task. No embedding fine-tuning.

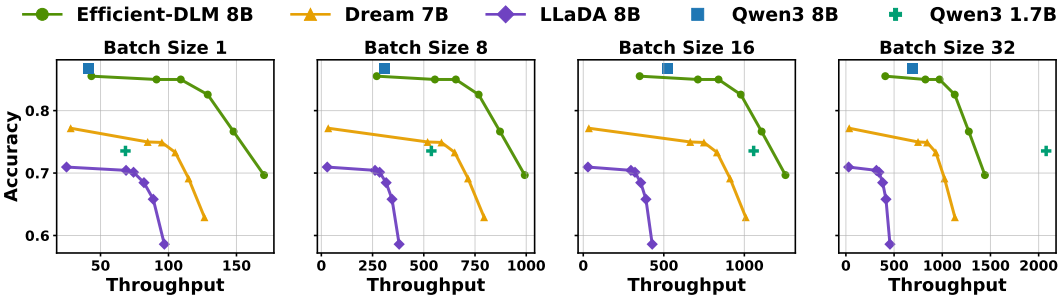
Efficient-DLM also beats accelerated dLMs on GSM8K



- Main-table Dream and LLaDA use TPF 1.
- This comparison adds Fast-dLLM dual caching and parallel decoding to both.
- Efficient-DLM retains a better curve on this task.

Paper, Appendix C, Fig. 10, PDF p. 13. GSM8K-only result; confidence thresholds are varied.

The relative benefit narrows as serving batches grow



- The paper reports favorable trade-offs through batch 16 on GSM8K.
- At batch 32, Qwen3 1.7B exceeds Efficient-DLM 8B in throughput.
- Small-batch, memory-bound serving is the clearest use case in this study.

Paper, Appendix C, Fig. 11, PDF pp. 13–14. Panel x-axis scales differ; compare accuracy as well as throughput.

Align training with the intended decoder

1. **Preserve useful structure.** Block causality and clean prefixes retain AR abilities and make caching natural.
2. **Train on plausible residual masks.** Moderate positional bias improves the parallel-decoding trade-off.
3. **Evaluate the entire frontier.** TPF 1 quality, aggressive decoding, and batch scaling answer different questions.

What constrains the result

Final training costs 300B–500B tokens; no matched continued-AR training control is reported. Small-model retention and large-batch speed remain weaker points.

Synthesis of Secs. 2–5 and Appendix C. Cost and comparison interpretation based on the reported protocol.

Paper identity and resources

Efficient-DLM: From Autoregressive to Diffusion Language Models, and Beyond in Speed

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan (Celine) Lin, and Pavlo Molchanov.

- Version used: arXiv:2512.14067v2, revised 29 April 2026.
- Model links from the paper: Efficient-DLM-4B, Efficient-DLM-8B.
- Figures reproduced from the authors' same-version TeX assets; cropped panels are identified.
- This is an independent reading presentation. Backup details follow.

Backup: retention differs across model sizes

Model	TPF	Tokens/s	Avg. (%)
Qwen2.5 1.5B	1.00	73.03	54.47
Efficient-DLM 1.5B	1.00	68.52	52.09
Efficient-DLM 1.5B	2.33	158.89	52.04
Efficient-DLM 1.5B	2.69	184.48	51.77
Qwen3 4B	1.00	47.13	67.97
Efficient-DLM 4B	1.00	44.13	67.54
Efficient-DLM 4B	2.52	119.33	67.39
Efficient-DLM 4B	3.01	130.24	67.18

1.5B at TPF 2.33: 2.18× throughput, −2.43 pp vs. its AR initializer.

Paper, Table 3, PDF p. 8. H100, batch 1; accuracy over 12 tasks, throughput over six generation tasks.

Backup: quality and speed averages cover different tasks

Generation task	Shots	Maximum output tokens
GSM8K	8	256
Minerva Math	4	512
HumanEval / HumanEval Plus	0	512
MBPP / MBPP Plus	3	512

$$\mathrm{Avg}_{12} = \frac{4\,\mathrm{Coding} + 2\,\mathrm{Math} + \mathrm{MMLU} + 5\,\mathrm{CR}}{12}.$$

- TPF / TPS: six generation tasks only.
- MMLU and commonsense scores stay fixed across reported decoding thresholds.
- AR evaluation uses lm-evaluation-harness; Dream / LLaDA use official code.

Paper, Table 3, PDF p. 8; Appendix A, p. 13. Weighted-average expression describes the reported task aggregation.

Backup: Gumbel-top-k samples a fixed-size mask set

$$g_i = -\log(-\log u_i), \quad u_i \sim \mathcal{U}(0,1), \qquad S = \mathrm{TopK}_i\{\log w_i(t) + g_i\}.$$

- $|S| = \mathrm{round}(tL')$: the count is fixed at a given noise level.
- Weight ratios describe preference, not marginal inclusion probabilities.

$$\frac{w_{i+\Delta}(t)}{w_i(t)} = 2^{(1-t)\Delta/(\lambda L')}.$$

Teaching example

$L' = 16$, $\lambda = 0.25$, $t = 0.5$: select eight positions.
Positions eight tokens apart have a weight ratio of two.

Derivation / sampling explanation from Eq. (2) and its Gumbel-top-k description, PDF p. 6. Not an additional experiment.

Backup: learning rate and LoRA both affect adaptation

Qwen3 4B, 25B tokens

Initial LR	Avg. (%)
10^{-4}	50.95
3×10^{-5}	56.59
10^{-5}	**60.63**
3×10^{-6}	60.11
10^{-6}	54.56

Too large: greater drift.
Too small: insufficient adaptation.

Qwen2.5 1.5B conversion

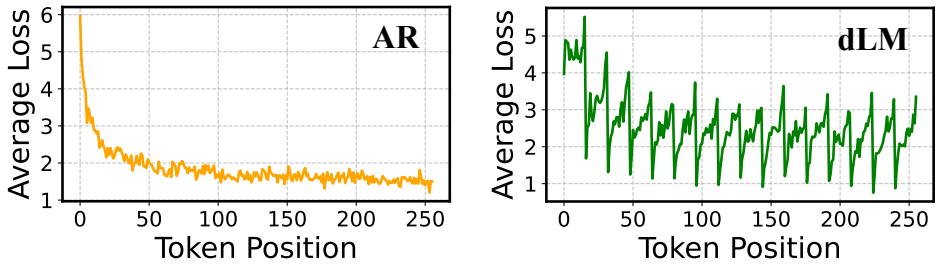
Update	Avg. (%)
LoRA rank 16	27.93
LoRA rank 64	30.78
Full model	38.41

LoRA also trains embeddings, normalization, and output head.

Rank 64 trails full training by 7.63 pp.

Paper, Appendices D and F, Tables 7–8, PDF pp. 14–15. Both report six-task generation accuracy.

Backup: block position creates a repeated pattern of difficulty



- AR: high loss at sequence start, then a flatter positional profile.
- Block dLM: loss rises within blocks and falls again at block boundaries.
- The repeated tail difficulty supports studying position-aware masking.

Paper, Appendix E, Fig. 12, PDF p. 14. First 256 training positions; diffusion Qwen2.5 1.5B uses block size 16.