

# T\*: Progressive Block Scaling for Masked Diffusion Language Models Through Trajectory Aware Reinforcement Learning

Hanchen Xia<sup>†\*</sup>, Baoyou Chen<sup>†◇\*</sup>, Yutang Ge<sup>‡</sup>, Guojiang Zhao<sup>§</sup>,  
Siyu Zhu<sup>†♣◇\*</sup>

<sup>†</sup>Shanghai Academy of AI for Science,

<sup>♣</sup> Shanghai Innovation Institute, <sup>◇</sup>Fudan University,

<sup>‡</sup>School of Mathematical Sciences, Shanghai Jiao Tong University,

<sup>§</sup>Carnegie Mellon University,

{xiahanchen, chenbaoyou}@sais.org.cn

## Abstract

We present T\*, a simple TRACERL-based curriculum for progressive block-size scaling in masked diffusion language models (MDMs). Starting from an AR-initialized small-block MDM, T\* gradually increases the block size while re-optimizing the denoising policy at each stage, enabling higher-parallelism decoding with limited degradation on math reasoning benchmarks. Across two SDAR scales and three benchmarks, T\* consistently outperforms direct large-block TRACERL and is substantially more stable during training. Our schedule analysis suggests that the learned policy does not simply revert to a strictly left-to-right order; instead, it retains block-size-specific non-monotone updates while improving accuracy.

## 1 Introduction

Before the current wave of large language models (LLMs), bidirectional Transformers trained with masked language modeling were a widely adopted backbone for NLP systems, with BERT and its optimized variants as canonical examples (Devlin et al., 2019; Liu et al., 2019). Today, autoregressive (AR) modeling via next-token prediction dominates both scaling practice and deployed systems (Brown et al., 2020; Touvron et al., 2023).

In parallel, diffusion language models have begun to emerge as viable alternatives or complements to the autoregressive decoding paradigm. Masked diffusion models stochastically mask a subset of tokens under a ratio-parameterized corruption process and optimize cross-entropy on masked positions to recover the original sequence (Sahoo et al., 2024). For scalability, recent work initializes diffusion LMs from pretrained autoregressive LLMs and trains them with random-mask diffusion objectives (Ye et al., 2025; Cheng et al., 2025).

At inference time, they often adopt blockwise decoding that denoises tokens within each block while generating blocks autoregressively to preserve global coherence (Arriola et al., 2025). In this setting, the block size is a control parameter that interpolates between stronger AR-like causality and higher-parallel masked updates.

Within each block, the denoising schedule is typically determined by model confidence. Given a prompt  $Q$  and the current partially denoised sequence  $x^{(s)}$ , let  $\mathcal{M}^{(s)}$  denote the set of masked positions at denoising step  $s$ . For each  $i \in \mathcal{M}^{(s)}$ , the model predicts a token distribution over the vocabulary  $\mathcal{V}$ . A common heuristic defines the confidence score

$$c_i^{(s)} = \max_{v \in \mathcal{V}} p_\theta(x_i = v \mid x^{(s)}, Q), \quad i \in \mathcal{M}^{(s)},$$
$$U^{(s)} = \{i \in \mathcal{M}^{(s)} : c_i^{(s)} \geq \eta\}, \quad (1)$$

where  $\eta \in (0, 1)$  is a confidence threshold that controls how many tokens are finalized at each step. Then materializes tokens in  $U^{(s)}$  (e.g., via argmax or sampling), while leaving the rest masked for subsequent refinement.

When examining the SDAR series models across scales (1.7B–30B) and block sizes (4–64), we find that math-centric reasoning becomes increasingly sensitive to larger blocks: accuracy generally degrades as block size  $B$  grows, with more pronounced drops for smaller models, which is also reported by Cheng et al. (2025). We consider the standard *absorbing-state* corruption used in masked diffusion LMs: once a token is replaced by the special  $[/MASK]$  symbol in the forward corruption process, subsequent corruption steps keep that position as  $[/MASK]$  rather than replacing it with another token. Under this corruption process, maximizing the ELBO yields a denoising objective; equivalently, the negative-ELBO reduces to a

\*Corresponding author.

\*Equally contributed to this work

reweighted cross-entropy over masked positions:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim q(x_t | x_0), t \sim \mathcal{U}(0,1)} \left[ -\frac{1}{t} \sum_{\ell=1}^L \mathbf{1}_{\{x_{t,\ell} = \text{[ /MASK ]}\}} \cdot \log p_{\theta}(x_{0,\ell} | x_t) \right], \quad (2)$$

Here,  $x_0$  denotes the clean target response and  $x_t$  its corrupted version at masking ratio  $t$ . For a block of size  $B$ , the expected number of  $\text{[ /MASK ]}$  positions is  $tB$ , so larger blocks contain more masked tokens to resolve within each denoising stage. Since we scale block sizes in powers of two ( $B = 2^n$ ), the number of jointly unresolved positions per stage increases with the stage index  $n$ . Standard supervised fine-tuning (SFT) data specify only the final token targets  $x_0$ —that is, which response tokens should ultimately be produced for a prompt—but do not specify an intermediate *unmasking schedule*, namely which subset of positions should be finalized at each denoising step.

In this work, we propose  $\mathbf{T}^*$ , an easy-to-implement yet effective strategy for progressive block-size scaling that increases block size with minimal performance degradation.  $\mathbf{T}^*$  offers a practical route for masked diffusion models (MDMs) to preserve the strong reasoning capability inherited from AR-initialized small-block models while moving toward higher-parallel decoding. Further analysis suggests that  $\mathbf{T}^*$  can induce an alternative decoding schedule, rather than reverting to the canonical left-to-right schedule.

## 2 Methodology

### 2.1 Trajectory-aware RL

We adopt TRACERL as our trajectory-aware reinforcement learning backbone, and build our method on top of it (Wang et al., 2025). TraceRL views diffusion decoding as a multi-step denoising trajectory and performs policy optimization on the same trajectory used at inference. Given a prompt  $Q$ , a diffusion LM produces a trajectory  $\tau = \tau(1) \cup \dots \cup \tau(T)$ , where  $T$  is the number of denoising steps and  $\tau(t)$  denotes the set of tokens decoded (unmasked) at step  $t$ . For brevity, we denote the trajectory prefix by  $\tau_{<t} := \tau(1:t-1)$  and suppress the dependence on  $Q$  when it is clear.

We treat each newly finalized token as an action. Concretely, at denoising step  $t$ , the policy samples token values for a subset of masked positions that are finalized at this step; we denote an action by  $o = (i, \hat{x}_i)$ , where  $i$  is the finalized position and  $\hat{x}_i$

is the sampled token. Accordingly,  $\pi_{\theta}(o | \tau_{<t}, Q)$  denotes the probability assigned to choosing  $\hat{x}_i$  at position  $i$  given the current trajectory prefix and prompt. TraceRL applies a PPO-like objective over all decoded tokens along the trajectory:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}} \left[ \sum_{t=1}^T \frac{1}{|\tau(t)|} \sum_{o \in \tau(t)} C_{\epsilon}(\rho_t(o), A(o)) \right] - \beta \text{KL}(\pi_{\theta} \parallel \pi_{\theta_{\text{old}}}), \quad (3)$$

where  $C_{\epsilon}(r, A) = \min\{rA, \text{clip}(r, 1-\epsilon, 1+\epsilon)A\}$  is the clipped surrogate and

$$\rho_t(o) = \frac{\pi_{\theta}(o | \tau_{<t}, Q)}{\pi_{\theta_{\text{old}}}(o | \tau_{<t}, Q)}. \quad (4)$$

In the simplest verifiable-reward setting, a single sequence-level reward (e.g., correctness of the final answer) is broadcast to the trajectory and used to form the advantages in Eq. 3.

To enable finer credit assignment over denoising steps, TRACERL aggregates token-level rewards (and value predictions) into step-level quantities by averaging within each denoising step, and computes step-wise advantages via TD/GAE (Schulman et al., 2015). These step advantages are then assigned back to all tokens decoded at the corresponding step, so that learning signals propagate through the entire denoising trajectory rather than only the final output (Lightman et al., 2023b).

### 2.2 Progressive Block Scaling

We propose  $\mathbf{T}^*$ , a stage-wise curriculum that alternates between policy adaptation at a fixed block size and block expansion to the next stage. The partitions below are applied only to the *response* segment; prompt tokens remain fixed context.

Let the response contain  $L$  token positions indexed by  $\{1, \dots, L\}$ . For a block size  $B$ , the *standard* partition is

$$\mathcal{P}_B^{(0)} = \{I_m^{(0)}\}_{m=0}^{M-1}, \quad (5)$$

$$I_m^{(0)} = \{mB + 1, \dots, \min((m+1)B, L)\},$$

where  $M = \lceil L/B \rceil$ .

To reduce sensitivity to fixed block boundaries, we also use a *shifted* partition with offset  $\Delta = B/2$  (all block sizes in our experiments are even):

$$\mathcal{P}_B^{(\Delta)} = \{I_m^{(\Delta)}\}_{m=0}^{M'}, \quad (6)$$

---

**Algorithm 1:  $T^*$ : Progressive Block Scaling (stage-wise)**

---

**Input:** Base model  $\theta_0$ , dataset  $\mathcal{D}$ , initial block size  $B_0$ , target block size  $\hat{B}$ , updates per stage  $K_B$   
**Output:** Optimized model  $\theta$

// Initialization  
1  $\theta \leftarrow \theta_0$ ;  $B \leftarrow B_0$ ;  
2 **while**  $B \leq \hat{B}$  **do**  
    // One stage at fixed block size  $B$   
    3  $\Delta \leftarrow B/2$ ;  
    4 Construct  $\mathcal{P}_B^{(0)}$  and  $\mathcal{P}_B^{(\Delta)}$ ;  
    5 **for**  $k \leftarrow 1$  **to**  $K_B$  **do**  
        6 Sample a rollout batch  $d$  from  $\mathcal{D}$ ;  
        7  $d_1, d_2 \leftarrow \text{SPLIT}(d)$ ;  
        // Update under the standard partition  
        8  $\theta \leftarrow \text{TRACERL}(\theta, d_1, B, \mathcal{P}_B^{(0)})$ ;  
        // Update under the shifted partition  
        9  $\theta \leftarrow \text{TRACERL}(\theta, d_2, B, \mathcal{P}_B^{(\Delta)})$ ;  
    // Expand to the next stage  
    10  $B \leftarrow 2B$ ;  
11 **return**  $\theta$

---

where

$$\begin{aligned} I_0^{(\Delta)} &= \{1, \dots, \min(\Delta, L)\}, \\ I_m^{(\Delta)} &= \{(m-1)B + \Delta + 1, \dots, \\ &\quad \min(mB + \Delta, L)\} \quad (m \geq 1), \end{aligned} \quad (7)$$

and  $M' = \max\{0, \lceil (L - \Delta)/B \rceil\}$ . Thus, the first shifted block is a prefix of length  $\Delta$ , and every subsequent boundary is moved to the right by  $\Delta$  tokens relative to the standard partition. The shifted variant changes only the block partition used by blockwise diffusion/TRACERL; it does not modify the underlying tokens or rewards.

At a fixed block size  $B$ , one *update step* consists of sampling a rollout batch and applying TRACERL once under the standard partition and once under the shifted partition. A *stage* consists of  $K_B$  such update steps at the same block size. Only after finishing the stage do we expand to  $2B$ . We therefore use the terms *update step* and *stage* throughout, rather than *epoch*, to avoid ambiguity. Algorithm 1 summarizes the procedure.

## 3 Experiments

### 3.1 Setup

We conduct experiments with the masked diffusion models **SDAR-1.7B-Chat** and **SDAR-4B-Chat**. These models are trained via block-diffusion strategy using different block sizes  $B \in \{4, 8, 16, 32\}$ . Our training dataset consists of 8K high-quality mathematical problems with

difficult levels 3-5 from Open1math. In each step, we randomly sample 128 problems from the dataset and generate 16 responses per problem using the static sampling strategy. Training process is performed on an 8-GPU H200 cluster using the AdamW optimizer with a learning rate of  $1 \times 10^{-6}$ . To prevent the policy from collapsing or drifting far from the base model, we apply a KL-divergence penalty with  $\beta = 0.01$ .

**Baselines.** For each target block size  $B \in \{8, 16, 32\}$  of **SDAR**, we directly apply 30 TRACERL policy-update steps at that block size.

**Evaluation.** We evaluate on MATH500 (Hendrycks et al., 2021; Lightman et al., 2023a), GSM8K (Cobbe et al., 2021), and AIME24 (Art of Problem Solving, 2024a,b) and report Pass@3. Each checkpoint is decoded with its target block size, i.e., the same block size used in the corresponding training stage.

**Checkpoint selection.** For each run, we periodically evaluate intermediate checkpoints on MATH500 and select a single checkpoint per run using MATH500 Pass@3. We then report that same checkpoint on MATH500, GSM8K, and AIME24. This protocol follows common practice in RL post-training, where one benchmark is often used for model selection; however, because MATH500 also appears in our headline results, the reported MATH500 scores should be interpreted as model-selection-aware rather than as a strictly held-out estimate. We therefore focus on cross-benchmark consistency, especially on GSM8K and AIME24, when comparing methods.

### 3.2 General Performance

Figure 1 plots MATH500 validation accuracy throughout training for the 1.7B model. While  $T^*$  remains relatively stable across stages, the direct TRACERL baseline exhibits abrupt collapses: a sharp drop occurs during the  $B=8$  stage (from  $\sim 56\%$  to the low-40% range), and another collapse appears near the end of the  $B=16$  stage (down to  $\sim 30\%$ ). We find this instability is highly sensitive to initialization at the target block size: applying TRACERL directly on the **SDAR-1.7B-Chat-b8** checkpoint collapses, whereas continuing TRACERL at  $B=8$  starting from a TRACERL-trained  $B=4$  diffusion policy (our stage transition) remains stable. A plausible explanation is that larger-block **SDAR** check-

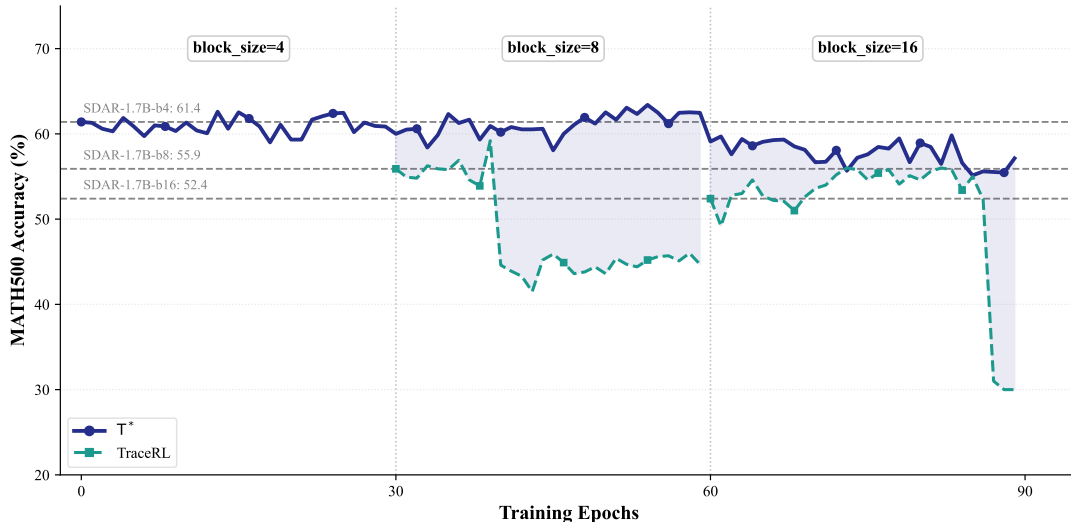


Figure 1: **Validation accuracy during block scaling (1.7B; x-axis = policy update steps).** MATH500 validation accuracy over training update steps for  $\mathbf{T}^*$  and a direct TRACERL baseline (dashed). A *stage* is a contiguous segment of  $K_B$  updates at a fixed block size  $B$ . Vertical dotted lines indicate stage transitions ( $B=4 \rightarrow 8 \rightarrow 16$ ). Horizontal dashed lines show the accuracies of the original **SDAR** checkpoints trained at each block size.

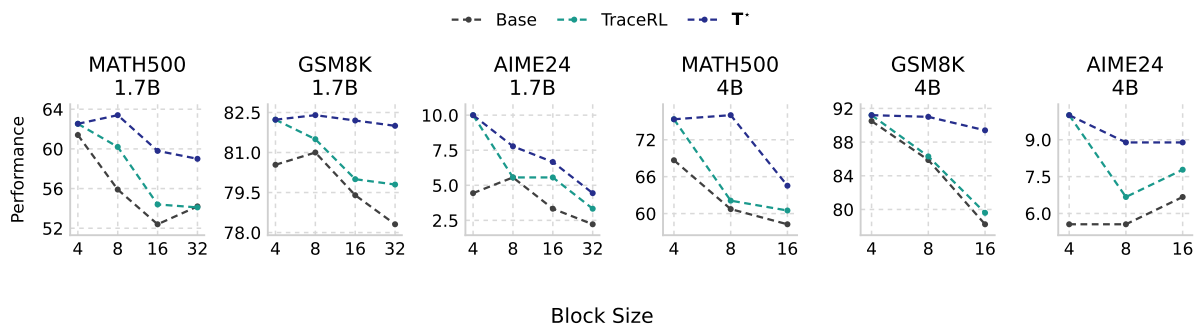


Figure 2: **Performance vs. block size across model scales.** Performance on MATH500, GSM8K, and AIME24 as a function of block size  $B$  for **SDAR** models at 1.7B (left) and 4B (right). **Base** denotes the original **SDAR-Chat-bB** checkpoint. **TRACERL** denotes applying TRACERL directly on the Base checkpoint at the same block size  $B$ .  $\mathbf{T}^*$  denotes our progressive curriculum that warm-starts from a small-block policy and increases  $B$  stage-by-stage (Alg. 1).

points operate under weaker conditioning contexts (cf. Eq. 2) and thus start from a lower-confidence regime, yielding noisier rollouts and higher-variance advantage estimates; when such advantages are broadcast to many tokens per denoising step, ratio-based updates can trigger likelihood drift and collapse, consistent with the Lazy Likelihood-Displacement “death spiral” analysis for GRPO-style training (Deng et al., 2025; Gao et al., 2025).

Figure 2 shows that, across different model sizes,  $\mathbf{T}^*$  consistently matches or exceeds the performance of the base models and TRACERL at the same block size on MATH500, GSM8K, and AIME24. When we expand the block size, the base

models generally show a downward trend, while  $\mathbf{T}^*$  remains more stable and achieves the strongest results at most evaluated block sizes; TRACERL often improves over the base model at smaller blocks but is typically below  $\mathbf{T}^*$  at larger blocks. All scores below are reported for the single checkpoint selected by the protocol in Sec. 3.1. The exact scores can be found in Table 2 and Appendix A.1.

### 3.3 Schedule

We compute LOCALSTRICT (Gong et al., 2025). Let  $\pi = (\pi_1, \dots, \pi_n)$  denote the linearized unmasking order obtained by sorting token positions by their first-unmask step (ties broken by smaller positions). LOCALSTRICT is defined as the frac-

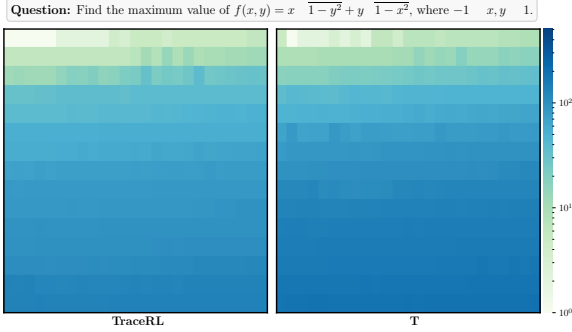


Figure 3: Decoding schedule under TRACERL vs.  $\mathbf{T}^*$ . More results can be found in Appendix A.2

tion of events that decode the leftmost remaining masked position:

$$\text{LOCALSTRICT} = \frac{1}{n} \sum_{k=1}^n \mathbb{1} \left[ \pi_k = \min_{j \geq k} \pi_j \right]. \quad (8)$$

Higher values indicate a schedule closer to the canonical left-to-right order  $S_0$ , while lower values reflect more non-monotone masked updates.

Model	LocalStrict	Accuracy	TPF
<b>Qwen3-1.7B</b>	1.000	70.2	1.0
<b>Qwen2.5-1.5B</b>	1.000	55.0	1.0
<b>SDAR-1.7B-b32</b>	0.743	54.2	3.74
+ TRACERL	0.704	54.1	3.67
+ $\mathbf{T}^*$	0.730	59.0	3.80
<b>SDAR-1.7B-b16</b>	0.766	52.4	3.38
+ TRACERL	0.824	54.4	3.41
+ $\mathbf{T}^*$	0.804	59.8	3.38
<b>SDAR-1.7B-b8</b>	0.915	55.9	2.91
+ TRACERL	0.984	60.2	2.84
+ $\mathbf{T}^*$	0.854	63.4	2.95

Table 1: **LocalStrict vs. accuracy on MATH500 with a decoding-efficiency proxy.** LocalStrict is computed by Eq. 8; higher values indicate a decoding order closer to canonical left-to-right. TPF denotes *tokens per forward*, i.e., the average number of tokens finalized per model forward pass during decoding (higher implies higher within-block parallelism).

**Decoding efficiency proxy (TPF).** To illustrate the efficiency benefit of block-size scaling, we report *tokens per forward* (TPF), i.e., the average number of output tokens finalized per model forward pass during decoding (higher is better). Autoregressive baselines have  $\text{TPF} \approx 1$  since they generate one token per forward step, whereas block-wise diffusion can finalize multiple tokens within a block in parallel. As the block size increases from

$B=8$  to  $B=16$  and  $B=32$ , the base SDAR-1.7B model shows a clear increase in TPF ( $2.91 \rightarrow 3.38 \rightarrow 3.74$ ), corresponding to  $\sim 16\%$  and  $\sim 29\%$  higher TPF, respectively. Equivalently, for a fixed output length, this reduces the required number of forward passes by  $\sim 14\%$  (from  $B=8$  to  $B=16$ ) and  $\sim 22\%$  (from  $B=8$  to  $B=32$ ). We stress that TPF is a forward-pass-level proxy rather than a direct end-to-end latency measurement, but it is still informative about the parallelism advantage of larger blocks in forward-pass-limited regimes. Importantly, applying TRACERL or  $\mathbf{T}^*$  does not negate this trend: the resulting models retain similar TPF at the same block size, indicating that the reasoning gains from RL-based training are compatible with the parallelism benefits of larger blocks.

Figure 3 visualizes token-level first-unmask step indices under TRACERL and  $\mathbf{T}^*$ . Table 1 reports LOCALSTRICT and accuracy under different block sizes. Overall, both methods retain largely monotone unmasking behavior (i.e., LOCALSTRICT remains high), but neither collapses to a strictly deterministic left-to-right schedule; instead, the learned step-wise schedules differ under the target block size (see Appendix A.2 for more examples).

## 4 Conclusion

Experiments show that  $\mathbf{T}^*$  stably scales block size with limited performance degradation, providing a practical recipe for transferring reasoning ability from AR-initialized small-block checkpoints to larger-block diffusion decoding. We further analyze the instability of direct TRACERL at larger block sizes and relate it to cumulative likelihood drift under noisy rollouts. Finally, our schedule analysis suggests that the learned denoising policy under a target block size is not simply a reversion to the canonical left-to-right order.

Recent work encourages non-linear reasoning via explicit external scaffolds such as tree-/graph-structured search over intermediate thoughts (Yao et al., 2023; Besta et al., 2024; Yao et al., 2024). In contrast, our results suggest that RL can also reshape the model’s internal token-finalization order without introducing an external search procedure, making internal schedule learning a complementary direction.

## Limitations

The limitations of this work can be summarized as:

- $T^*$  mitigates but does not fully eliminate degradation under block expansion; we suspect residual drops are partly due to the lack of a high-quality “cold-start” stage.
- We did not scale to very large blocks (e.g.,  $B=64$  or  $128$ ) in our  $T^*$  curriculum, because the inference engine becomes unstable at large block sizes.

## Acknowledgments

This work was supported in part by the Shanghai Municipal Commission of Economy and Informatization (No. 2025-GZL-RGZN-BTBX-01011), the Natural Science Foundation of Shanghai (No. 24ZR1407200), and the Shanghai Oriental Talents Project (No. QNKJ2024060).

## References

- Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. [Block diffusion: Interpolating between autoregressive and diffusion language models](#). In *International Conference on Learning Representations*.
- Art of Problem Solving. 2024a. [2024 AIME i](#). AoPS Wiki. Accessed: 2026-01-05.
- Art of Problem Solving. 2024b. [2024 AIME ii](#). AoPS Wiki. Accessed: 2026-01-05.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. [Graph of thoughts: Solving elaborate problems with large language models](#). In *Thirty-Eighth AAAI Conference on Artificial Intelligence, Vancouver, Canada*, pages 17682–17690.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#).
- Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. 2025. [SDAR: A synergistic diffusion–autoregression paradigm for scalable sequence generation](#).
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. 2025. [On grpo collapse in search-r1: The lazy likelihood-displacement death spiral](#). *arXiv preprint arXiv:2512.04220*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. [Soft adaptive policy optimization](#). *arXiv preprint arXiv:2511.20347*.
- Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatuo Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. [Diffucoder: Understanding and improving masked diffusion models for code generation](#).
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Advances in Neural Information Processing Systems*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. [Let’s verify step by step](#). *arXiv preprint arXiv:2305.20050*. Defines the nonstandard MATH-500 evaluation split released in the PRM800K repository.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. [Let’s verify step by step](#).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#).
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024.

Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Revolutionizing reinforcement learning framework for diffusion large language models.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Yao Yao, Zuchao Li, and Hai Zhao. 2024. GoT: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2901–2921, Mexico City, Mexico. Association for Computational Linguistics.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models.

## A Appendix

### A.1 Full Results

Base model	Method	MATH500 $\uparrow$	GSM8K $\uparrow$	AIME24 $\uparrow$
<b>SDAR-1.7B-Chat-b4</b>	–	61.40	80.54	4.44
	TRACERL	62.53	82.23	10.00
<b>SDAR-1.7B-Chat-b8</b>	–	55.90	81.00	5.56
	TRACERL	60.20	81.50	5.56
	<b>T*</b>	63.40	82.40	7.78
<b>SDAR-1.7B-Chat-b16</b>	–	52.40	79.40	3.33
	TRACERL	54.40	80.00	6.66
	<b>T*</b>	59.80	82.20	6.66
<b>SDAR-1.7B-Chat-b32</b>	–	54.20	78.31	2.22
	TRACERL	54.10	79.80	3.33
	<b>T*</b>	59.00	82.00	4.44
<b>SDAR-4B-Chat-b4</b>	–	68.67	90.50	5.56
	TRACERL	75.33	91.20	10.00
<b>SDAR-4B-Chat-b8</b>	–	60.73	85.87	5.56
	TRACERL	62.10	86.30	6.67
	<b>T*</b>	76.00	91.00	8.89
<b>SDAR-4B-Chat-b16</b>	–	58.26	78.24	6.67
	TRACERL	60.50	79.60	7.78
	<b>T*</b>	64.53	89.40	8.89

Table 2: **Reasoning performance under different block sizes.** “–” denotes the original SDAR--Chat-b $B$  checkpoint, TRACERL applies trajectory-aware RL at the same block size  $B$ , and **T\*** denotes our progressive block-size scaling.

## A.2 Case Study

**Question:** Find the product  $CD$  of the integers  $C$  and  $D$  for which  $\frac{C}{x-3} + \frac{D}{x+8} = \frac{4x-23}{x^2+5x-24}$  for all real values of  $x$  except  $-8$  and  $3$ .



Figure 4: **Case study: decoding schedule under TRACERL vs. T\*.** We visualize the token-level first-unmask step index (heatmaps; darker means decoded later) and the corresponding model solutions for a representative algebra problem, evaluated with block sizes  $B \in \{8, 16, 32\}$ . The top row shows a model trained with direct TRACERL at the same block size, and the bottom row shows the model obtained by T\*.