

Recurrent, diffusion, and hybrid reasoning

State updates in recurrent and masked-diffusion reasoning.

Reasoning agents, feedback, and scientific evidence

An early commitment can block a later repair

Imagine a partial solution contains an early commitment that later evidence makes inconsistent. Continuing the sequence can leave the original commitment in place.

Two questions emerge: does the system need more processing of its state, or the ability to revise several output positions?

These diagnoses suggest different directions. Repeating computation and reopening a generated choice solve different problems.

Different ways to revisit computation

Reusing an internal block can supply more processing without a new parameter set for every step. Revisiting an output position can instead change an earlier choice.

Masked prediction makes the second route conceivable: hide uncertain positions, use the rest as context, and predict replacements. Several positions may be updated together.

Parallel revisions can conflict. A schedule, stopping rule and final constraint check become design requirements once outputs are revisable.

Choosing an update structure

Candidate	Reason to choose it	Cost or limitation
Autoregressive generation	A stable prefix supports the next decision	Early commitments can be costly to repair
Recurrent computation	A state benefits from additional processing	More iterations need not improve correctness
Masked denoising	Several output positions may need revision	Repeated passes and coordination have costs

Choose the update rule from the required revision. Compare representation, total work and sequential depth separately.

Pedagogical reconstruction

[Nie et al. \(2025\)](#)

[Jolicoeur-Martineau \(2025\)](#)

Masked diffusion predicts and revises token positions

A masked language diffusion model learns to recover clean tokens from partially masked sequences. At inference, a schedule progressively resolves uncertainty and may revisit earlier choices.

The conditioning context and the rule for selecting positions affect the generated trajectory. Arbitrary generation order does not automatically improve reasoning.

Count denoising passes and processed sequence positions; fewer sequential output tokens do not by themselves imply lower total compute.

Recurrence and hierarchy are related but distinct

A recurrent update applies a transformation repeatedly. A hierarchical system introduces multiple levels, timescales, or abstractions with different roles.

Two recurrent states can operate at different rates without representing a scientific strategy and procedure in a human-interpretable sense.

Interpret a model's architecture first in terms of its documented state transitions and training objective. Stronger semantic interpretations require evidence.

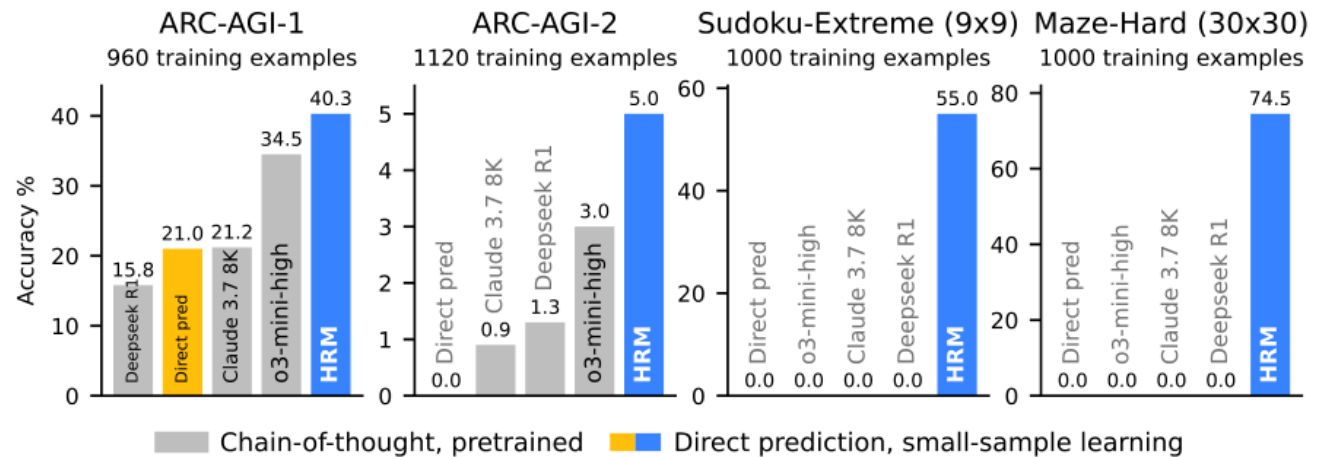
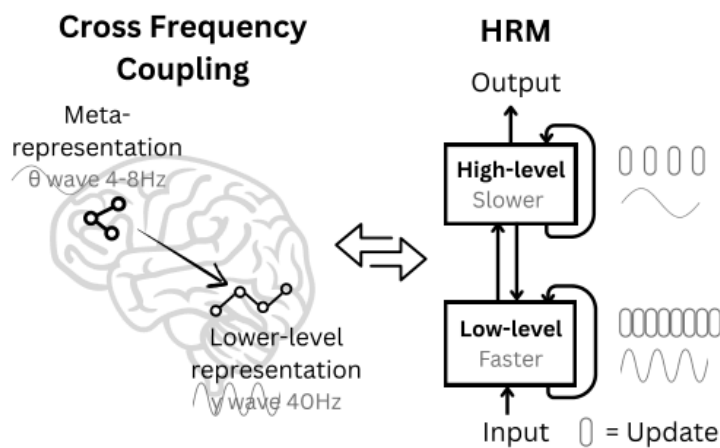
Hierarchical Reasoning Model: the method

HRM uses coupled recurrent components operating at different update rates, together with a training procedure designed for deep iterative computation.

The architecture is evaluated on structured reasoning tasks such as puzzles. Its state-update schedule and supervision are concrete computational choices.

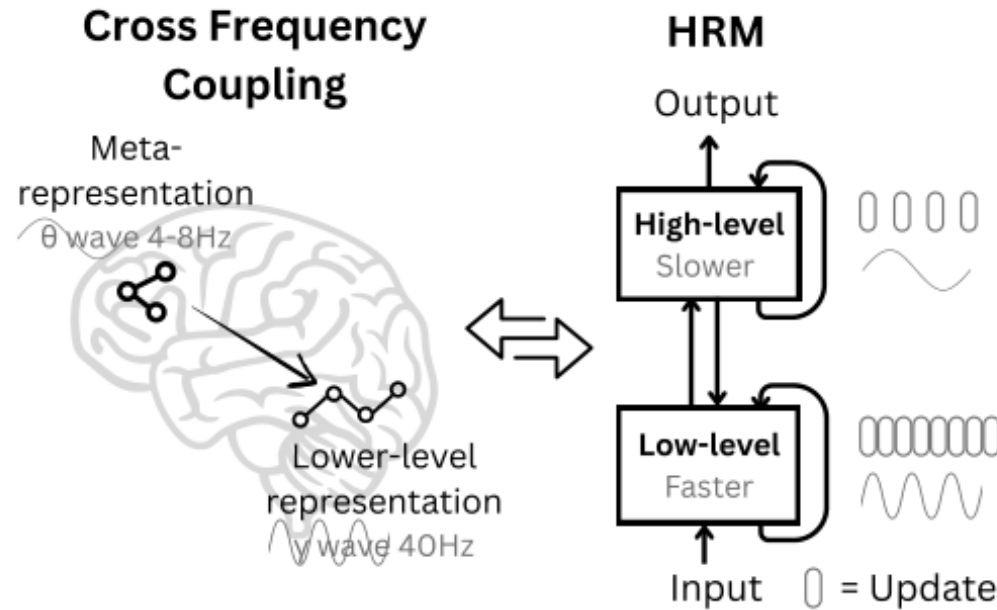
Calling the states high-level and low-level describes architectural roles; interpreting them as scientific strategy and procedure would require additional causal and behavioural evidence.

HRM: coupled recurrent components



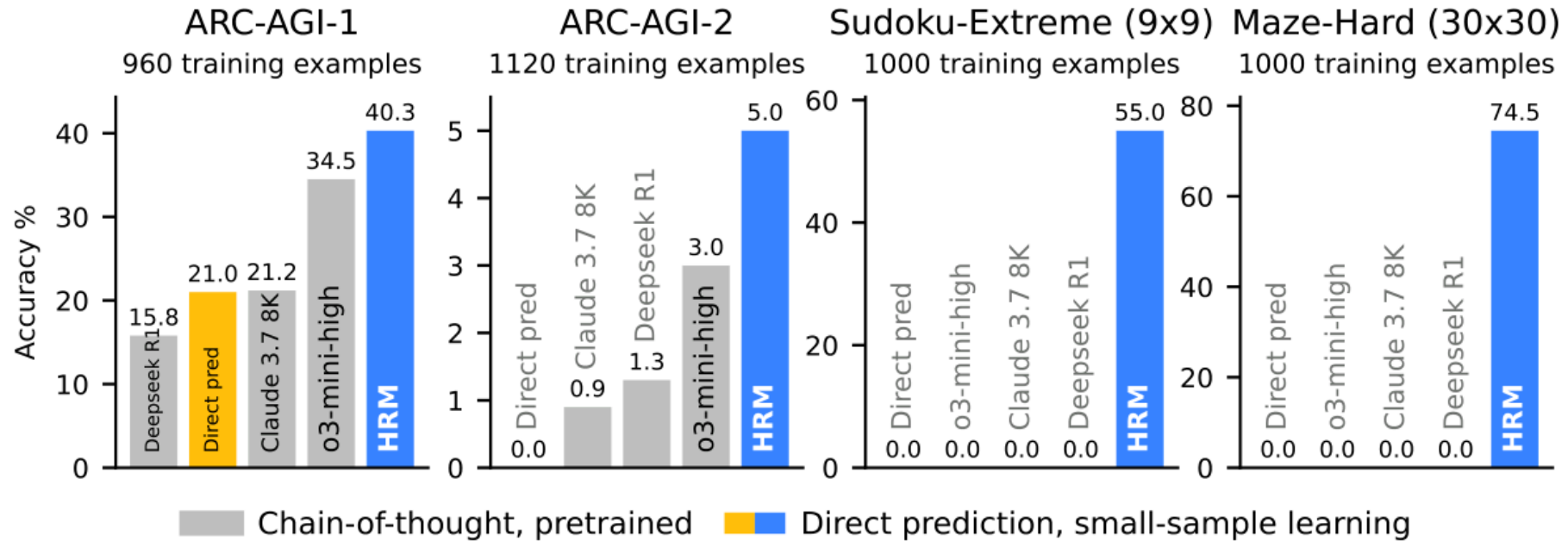
Two recurrent components operate on different timescales through feedback connections. The implemented transitions define the model. Biological inspiration supplies motivation rather than an additional performance guarantee.

Hierarchical Reasoning Model: mechanism detail



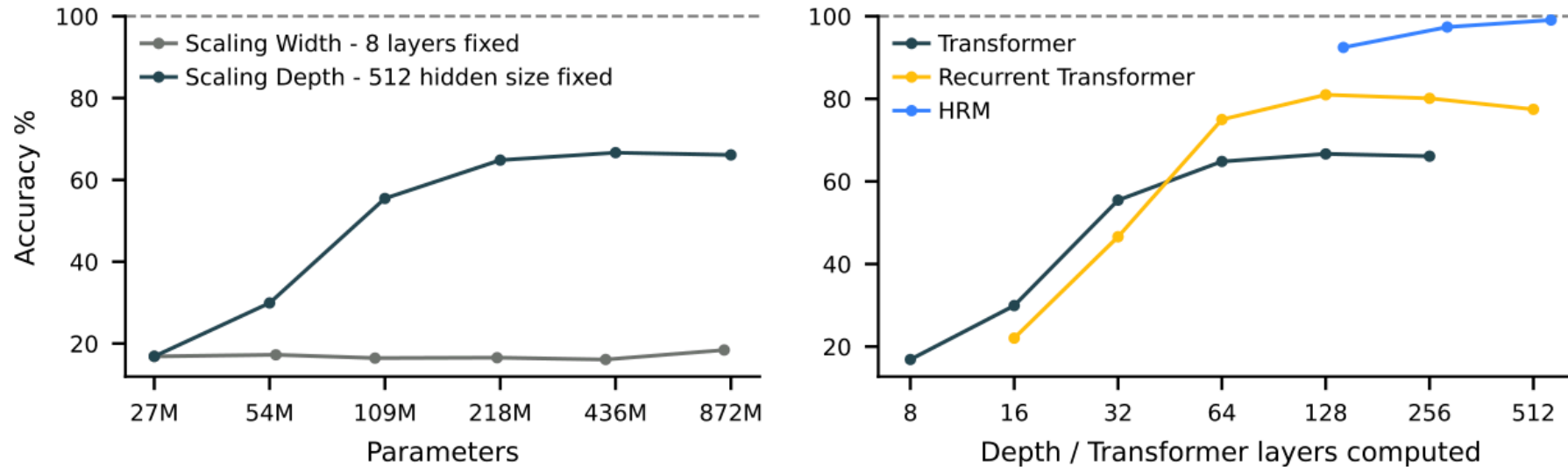
The two recurrent states receive different information and update on different schedules. This documented coupling describes the mechanism; its interpretation as scientific levels is a separate, untested claim.

Evidence: the matched architectural control



HRM reports 55.0% on Sudoku-Extreme and 74.5% on Maze-Hard, versus 0.0% for the direct-prediction control. The paper describes that control as sharing HRM's training setup. The pretrained CoT models have different training histories. ARC additionally uses 1,000 augmented test variants and aggregation; its inference protocol differs from these two tasks.

Hierarchical Reasoning Model: interpretation and limits



Small structured datasets, augmentation, supervision choices, and test-time computation affect comparisons. Puzzle success and a two-timescale architecture do not establish a general scientific hierarchy or a biological explanation of reasoning.

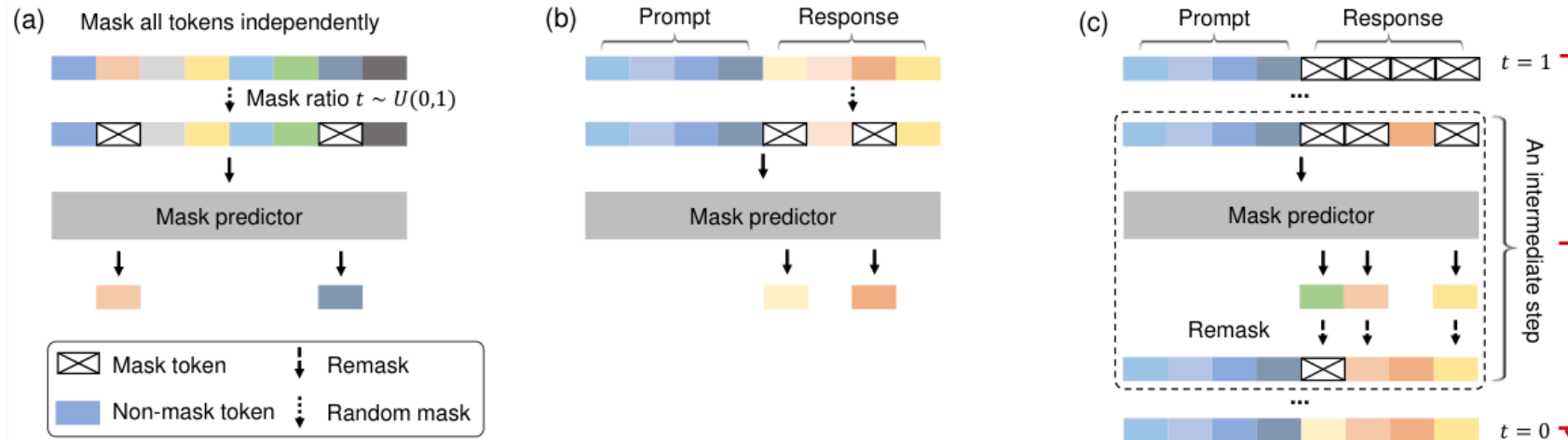
LLaDA: the method

LLaDA trains a masked diffusion language model to recover clean tokens from corrupted sequences and generates text through iterative denoising.

Pretraining, supervised fine-tuning, and inference use related but distinct masking and conditioning procedures.

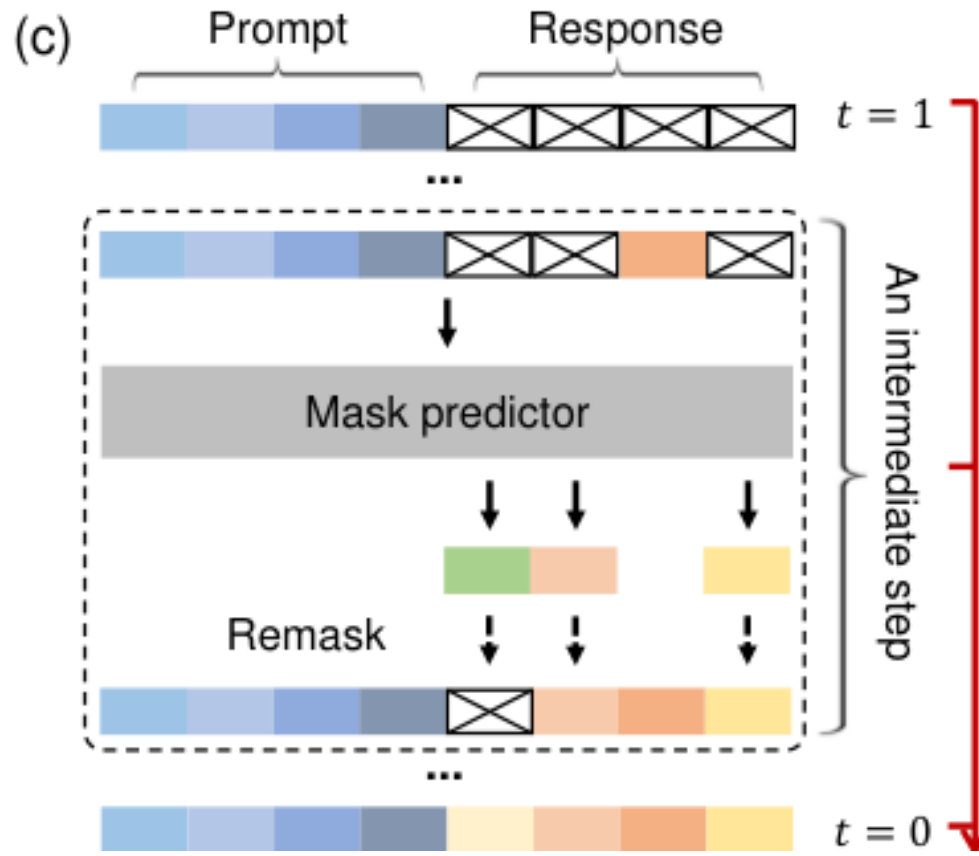
The architecture provides an alternative to strictly autoregressive output generation. Its value depends on the training recipe, denoising schedule, task constraints, and total computational cost.

LLaDA: training and sampling procedures



Pretraining, supervised fine-tuning and sampling use observed and masked positions differently. During generation, the policy for selecting and revisiting positions determines the sequence of denoising decisions.

LLaDA: mechanism detail



The sampling policy moves positions from masked states to predictions and possible revisions. Its retention rule controls which proposed tokens remain fixed, affecting both consistency and computation.

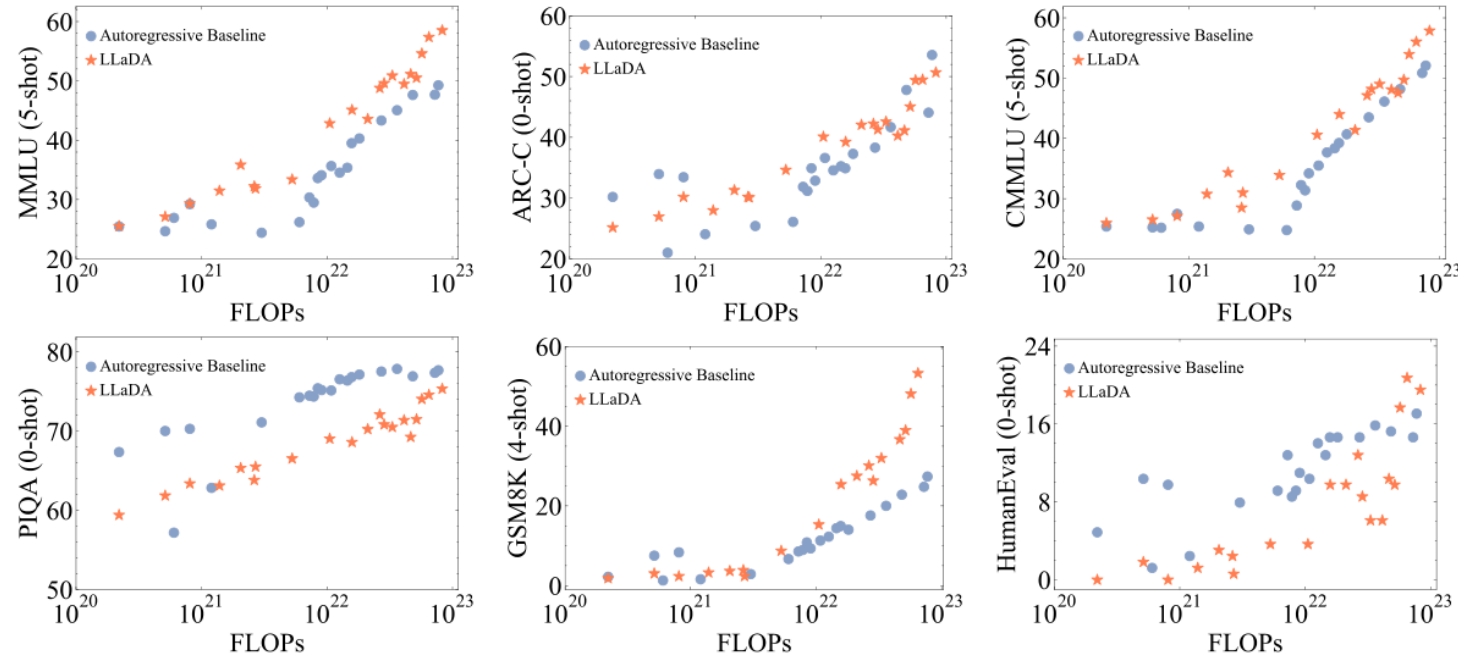
Evaluation scope

The source compares benchmark performance as training computation and model scale increase. The original plots contain both diffusion and autoregressive reference systems.

Match tasks and resource definitions before interpreting a point as a better frontier.

Broad language-model competence demonstrates the feasibility of the model family, while reasoning efficiency and scientific usefulness require targeted evaluations beyond standard benchmark scores.

LLaDA: interpretation and limits



Parallel prediction does not imply proportional speedup, and generation-order flexibility may introduce its own errors. Training data, optimisation, and sampling schedules can confound architecture comparisons.

Why intervene before denoising finishes?

L06 defined a target favoring high-scoring objects. Terminal ranking spends a full trajectory on every candidate, even those with poor prospects.

A partial denoised state might predict final value. That suggests reallocating the remaining computation while generation is still in progress.

Feynman–Kac (FK) steering constructs a population of trajectories. Intermediate potentials determine which branches survive; poor early predictions can discard valuable branches.

From intermediate scores to a particle update

Step	Toy two-particle population	Consequence
Propagate	Advance A and B by one denoising transition.	Both have consumed the same step budget.
Weight	Positive potentials: $G(A)=1$; $G(B)=3$.	Normalize: A has weight $\frac{1}{4}$; B has weight $\frac{3}{4}$.
Resample two paths	Draw two ancestors from these normalized weights.	Expected copies: $A=\frac{1}{2}$; $B=1\frac{1}{2}$. Next transitions add diversity.

Resampling reallocates work stochastically. Two particles do not guarantee that the best branch survives.

The potentials must agree with the final target

For base-transition proposals, choose potentials whose product is $\exp(\beta r(x_0))$, up to a constant. Here x_0 is the finished object, r its score and β the steering strength.

An intermediate score can guide search, but terminal corrections must reconcile it with the intended reward. Arbitrary repeated rewards can change the target.

FK steering is a finite particle approximation. Compare against terminal ranking at matched compute, and measure the real target separately from the steering score.

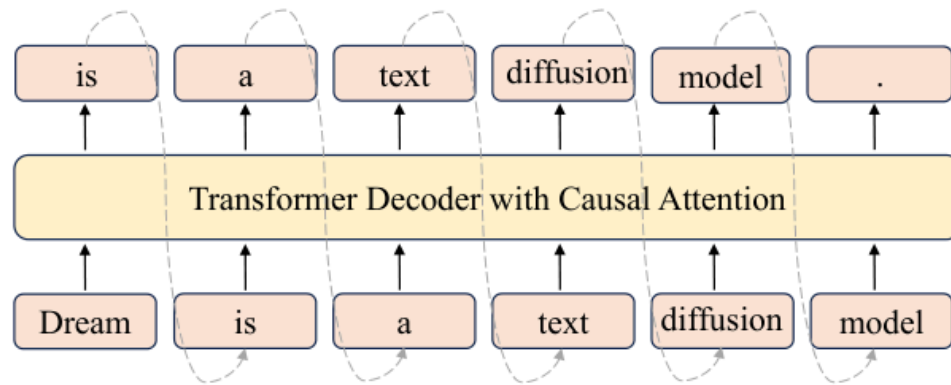
Dream diffusion language models: the method

Dream develops a diffusion language-model training and inference recipe using full-context denoising rather than exclusively causal next-token generation.

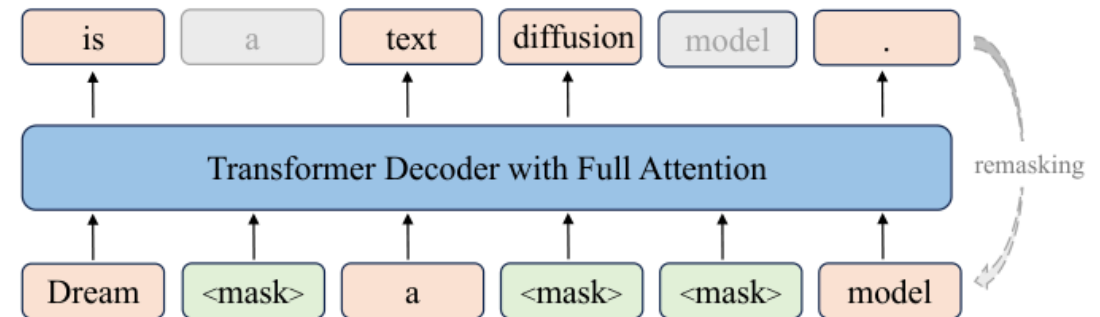
The source studies how to initialise and train the model and compares its capabilities with diffusion and autoregressive references.

The relevant scientific questions include how iterative global updates affect constraint-heavy tasks and whether the resulting quality justifies the measured inference resources.

Dream: attention and denoising structure



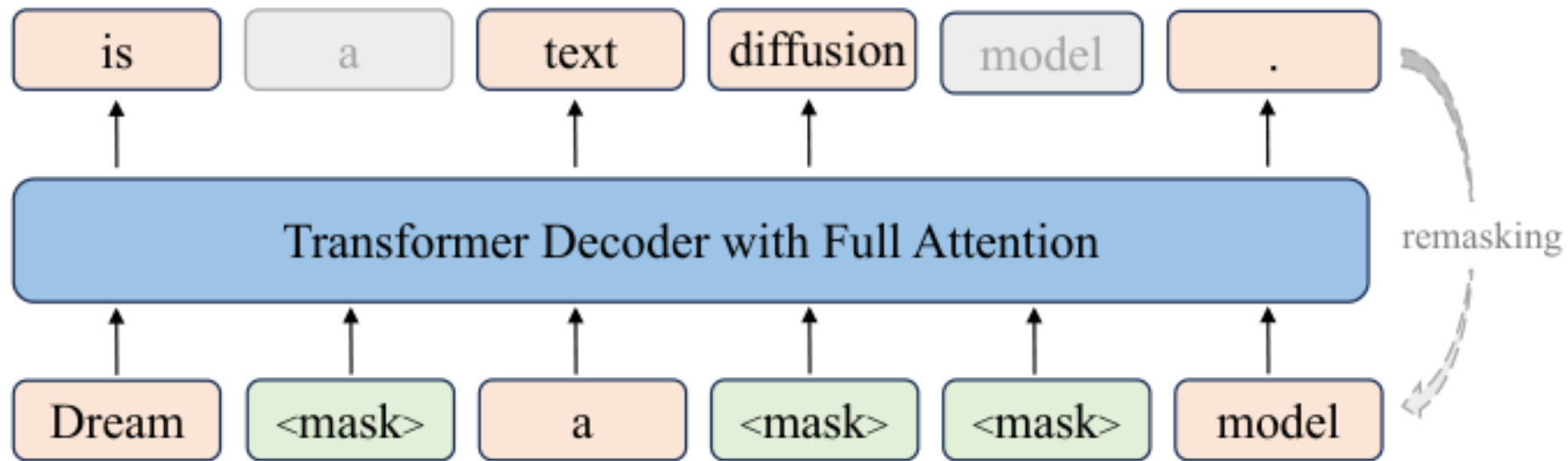
(a) Autoregressive Modeling



(b) Diffusion Modeling in Dream

Autoregressive causal attention and full-attention denoising expose different information to each prediction. The conditioning structure is a computational distinction whose task-level consequences require empirical evidence.

Dream diffusion language models: mechanism detail



(b) Diffusion Modeling in Dream

The diffusion attention and masking pattern permits dependencies on proposed outputs that a causal generation step cannot access in the same way. The update policy determines which of those dependencies affect the next prediction.

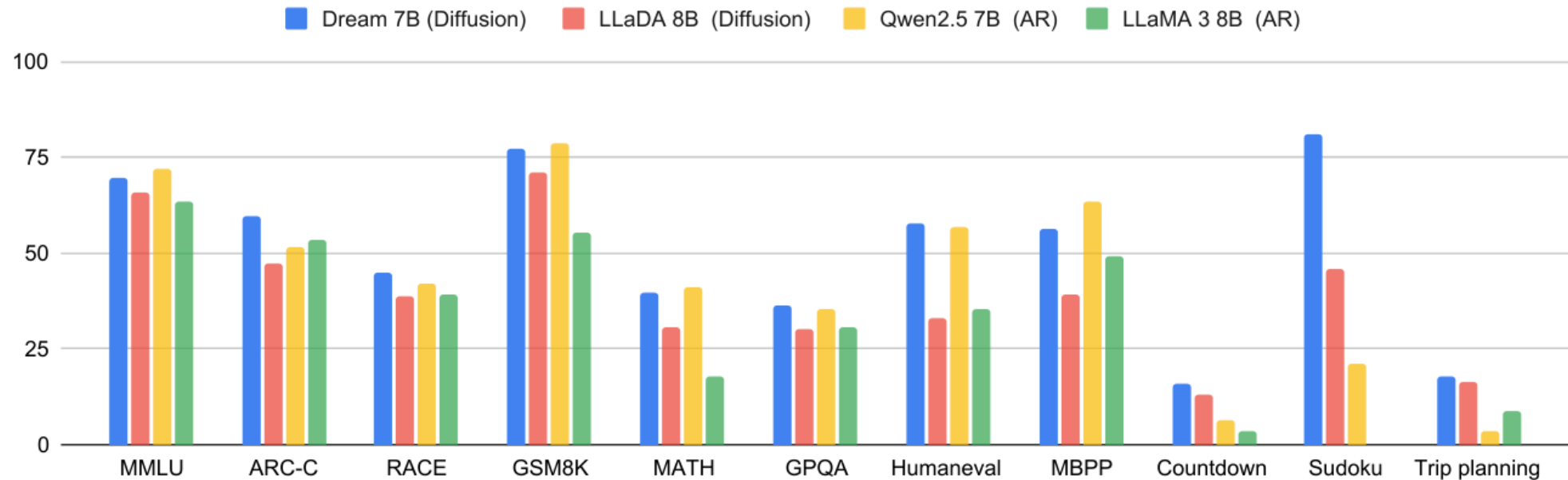
Evaluation scope

The original benchmark comparison spans language, mathematics, code, and constraint-oriented tasks.

Different tasks can favour different generation procedures. Read model sizes and benchmark definitions before comparing bars.

Strong performance on a puzzle-like task motivates investigating iterative revision, but it does not establish a general advantage for all reasoning or a resource-matched improvement over every autoregressive alternative.

Dream diffusion language models: interpretation and limits



The training recipe and data differ across reference models, and a benchmark bar does not include every latency or compute cost. Results should be interpreted as a system comparison unless a controlled ablation isolates the architecture.

Parallel updates still perform work

Masked revision can update many token positions in one pass. Repeating that pass performs additional computation over the sequence.

Separate total work from sequential depth. A reduction in dependent steps can improve latency on suitable hardware while increasing total operations.

A deliberately simplified cost model

Method	Passes	Work per pass	Total work / depth
Autoregressive	12	1	12 / 12
Masked revision	4	12	48 / 4
Masked revision	8	12	96 / 8

Synthetic units omit caching, attention, batching and hardware effects. The table isolates why fewer passes need not mean less computation.

Revision must preserve global constraints

Locally plausible token updates can jointly violate a constraint. The schedule, conditioning context and acceptance rule determine whether a valid candidate survives revision.

Evaluate feasibility after the complete update. If a repair step restores validity, include its computation and measure whether it changes solution quality.

Latency depends on the implementation

Method	Pass latency	Passes	Assumed total latency
Autoregressive	1 unit	12	12 units
Masked revision	1.8 units	4	7.2 units
Masked revision	3.5 units	4	14 units

Synthetic latencies show that the same four-pass algorithm can be faster or slower under different execution conditions.

Parallel best responses can cycle

Operation	Calculation or state	Consequence
Constraint	$x+y=1$; x,y are binary	Each variable wants 1–other
Parallel, step one	$(0,0)$ becomes $(1,1)$	Both read the old state
Parallel, step two	$(1,1)$ becomes $(0,0)$	The invalid cycle repeats
Sequential sweep	$x \leftarrow 1-y$; then $y \leftarrow 1-x$	$(0,0)$ becomes $(1,0)$

Synthetic constraint problem. The update schedule changes the trajectory even though both policies use the same local rule.

LLaDA: prediction is not yet commitment

Sampling state	Predictions at masked positions	Visible state after selection
Start	Response is fully masked	[M, M, M, M]
First denoising step	[2, +, 3, 6]	[2, +, M, M]
Second step	Predict 3 and 5 in remaining positions	[2, +, 3, M]
Final step	Predict 5 for the last mask	[2, +, 3, 5]

Synthetic four-position tuple (a, operator, b, result). M means masked. This is a schematic trace, not observed LLaDA output.

Confidence schedules a sampler; it does not verify

Control	Mechanism	Limitation
Remasking schedule	At times $t \rightarrow s$, remask fraction s/t	Controls how many new predictions persist
Low-confidence selection	Remask less confident new predictions	Token confidence can favor a wrong value
Response length and step count	Set the state size and iteration budget	More steps do not ensure global consistency
External arithmetic check	Test whether $a + b$ equals the result	An added verifier, not intrinsic to LLaDA

Masked denoising provides a generation procedure. A scientific consistency check is a separate source of feedback.

Account for sequential depth and total work

Suppose an autoregressive system processes one new output position over twelve decoding steps. A denoising system uses four passes over twelve positions.

The second system has fewer sequential passes but performs up to 48 position updates, compared with twelve output updates in the simplified accounting. Actual transformer costs also depend on caching and attention structure.

Measure wall-clock time and FLOPs rather than equating four passes with a threefold speedup.

An informative ablation matrix

Holding the backbone, training examples and output constraints fixed separates the contributions of update family, schedule and halting rule.

Final correctness, constraint violations, latency, total computation and performance on harder inputs describe different trade-offs. An explicit search baseline distinguishes iterative refinement from ordinary search.

This design tests whether recurrence or denoising provides an advantage beyond simply repeating computation, and whether that advantage transfers beyond the tasks used for development.

LLaDA's SFT loss supervises masked response tokens

Training operation	Source-specific object
Sample corruption	t is uniform in $(0, 1)$; mask each response token with probability t
Predict a token	$p_{\theta}(r_i \mid \text{prompt}, r_{-t})$, where r_{-t} is the corrupted response
Loss for one example	$\ell = -(1/t) \sum (\text{masked } i) \log p_{\theta}(r_i \mid \text{prompt}, r_{-t})$
Optimize the model	Minimize expected ℓ over examples, times and sampled masks

LLaDA supervised fine-tuning (SFT), Eq. 5: preserve the prompt; score only masked response positions. The $1/t$ weight belongs to this formulation.

HRM and TRM do not use the same hierarchy

Design choice	Hierarchical Reasoning Model	Tiny Recursive Model
Learned modules	Separate high- and low-level modules	One small shared network
Recurrent state	Two interacting time scales	Latent reasoning state and candidate answer
Training signal	Supervision across reasoning cycles	Deep supervision across recursive updates
Interpretation	Hierarchy is architectural	Recursion need not imply separate planners

HRM and TRM show different ways to reuse computation. Neither establishes that a state corresponds to a human-readable subgoal.

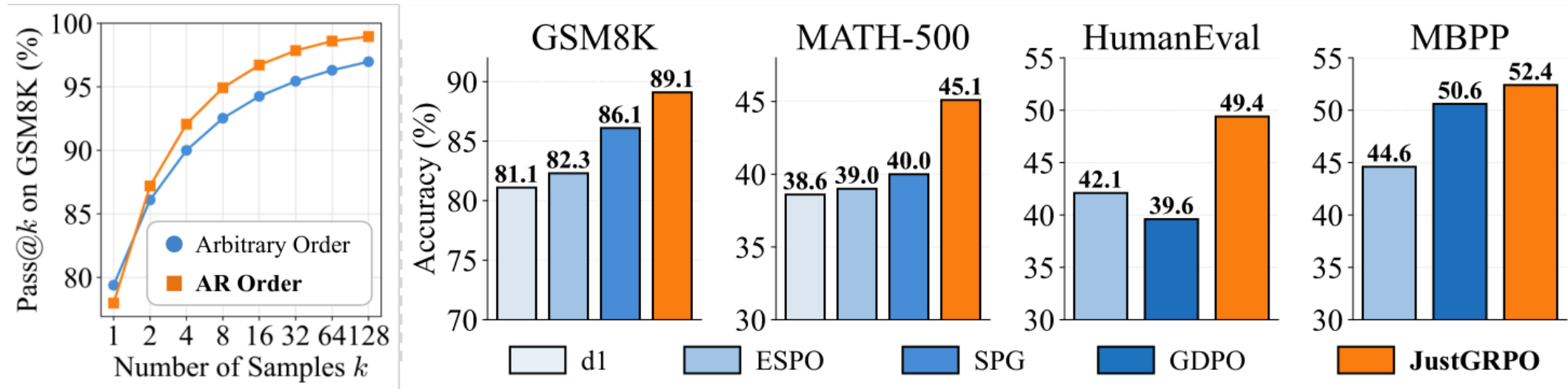
Recent research: The Flexibility Trap

The study investigates how arbitrary-order generation in diffusion language models affects reasoning coverage.

It reports that some generation policies avoid uncertain early decisions, and studies a simpler order constraint with group-relative reinforcement learning.

The literature snapshot is dated 11 September 2026. Findings remain conditional on the source version and evaluation conditions. Recent preprints contribute reported results with distinct validation requirements.

Diffusion reasoning: generation-policy comparisons



The labelled generation and training conditions determine whether differences in the curves concern decoding order, reinforcement learning or their combination.

Recent methods in context

Discrete Diffusion Forcing (ICLR 2026). Combines blockwise autoregressive structure and parallel diffusion decoding to make caching and pipelining possible. Throughput depends on the model, hardware, output length, batching and resulting quality.

Latent Thinking Optimization (ICLR 2026). Uses a correctness predictor over latent states to guide latent reasoning updates. A decodable reward signal is not a faithful explanation of the underlying reasoning.

Recursive Language Models (preprint; revised 2026). Treats long context as an external environment that can be inspected programmatically and processed through recursive model calls. Nested calls can increase cost unpredictably; long-context benchmark gains do not establish universal task efficiency.

References (1/2)

[Guan Wang, Jin Li, Yuhao Sun et al. \(2025\). Hierarchical Reasoning Model](#)

[Shen Nie, Fengqi Zhu, Zebin You et al. \(2025\). Large Language Diffusion Models](#)

[Jiacheng Ye, Zhihui Xie, Lin Zheng et al. \(2025\). Dream 7B: Diffusion Large Language Models](#)

[Alexia Jolicoeur-Martineau \(2025\). Less is More: Recursive Reasoning with Tiny Networks](#)

[Raghav Singhal, Zachary Horvitz, Ryan Teehan et al. \(2025\). A General Framework for Inference-time Scaling and Steering of Diffusion Models](#)

References (2/2)

Zanlin Ni, Shenzhi Wang, Yang Yue et al. (2026). The Flexibility Trap: Rethinking the Value of Arbitrary Order in Diffusion Language Models

Xu Wang, Chenkai Xu, Yijie Jin et al. (2025). Diffusion LLMs Can Do Faster-Than-AR Inference via Discrete Diffusion Forcing

Hanwen Du, Yuxin Dong, Xia Ning (2025). Latent Thinking Optimization: Your Latent Reasoning Language Model Secretly Encodes Reward Signals in Its Latent Thoughts

Alex L. Zhang, Tim Kraska, Omar Khattab (2025). Recursive Language Models