Generative Recursive Reasoning

Junyeob Baek[1*] Mingyu Jo[1*] Minsu Kim[1,2] Mengye Ren[3] Yoshua Bengio[2,4] Sungjin Ahn[1,3]

[1] KAIST [2] Mila [3] [4] Université de Montréal

* Equal contribution

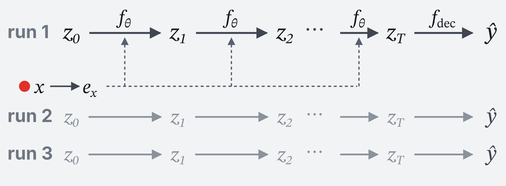
PROJECT PAGE CODE

Q. Can latent thought recursion support probabilistic, multi-hypothesis reasoning?

Deterministic recursion HRM, TRM **vs** **GRAM** (ours)

One path, one answer.

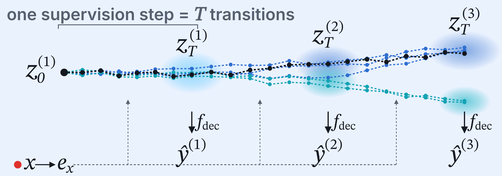
- Recursive Reasoning Models **refine a latent state**, not a sequence, so depth is free of model size and output length.
- But updates are **deterministic**: the same input always gives a **single latent trajectory** and a single prediction.



All plausible reasoning paths collapse into **one trajectory**.

Many paths, many answers.

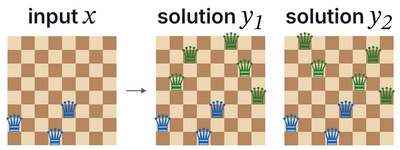
GRAM turns recursion into **probabilistic multi-trajectory computation**. The input is embedded once, $e_x = f_{\text{enc}}(x)$, and every **latent transition** is a **sample**: $z_t \sim p_\theta(z_t \mid z_{t-1}, e_x)$.



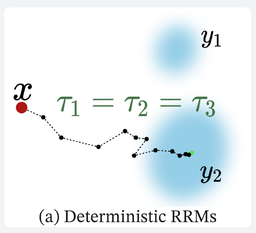
Stochastic latent trajectory. Trajectories $\tau = (z_0 \to \cdots \to z_{T_{total}})$ form a **distribution**; the answer marginalizes over them:

$$p_\theta(y \mid x) = \int \underbrace{p_\theta(y \mid \tau, x)}_{\text{answer given a trajectory}} \underbrace{p_\theta(\tau \mid x)}_{\text{distribution over trajectories}} d\tau$$

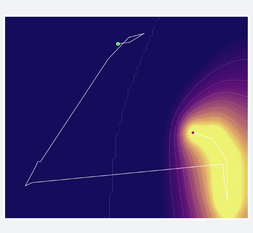
Same N-Queens input x, two valid solutions



Latent landscapes on one Sudoku: high-level state h per supervision step, PCA to 2D over the loss landscape; dark = low loss, ● start, ● end.

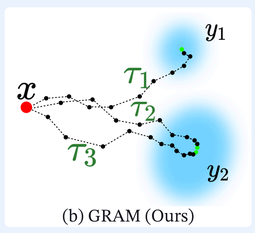


(a) Deterministic RRMs

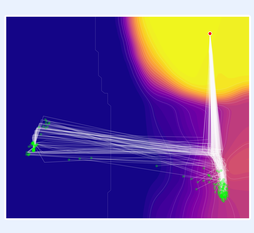


TRM: **one path**, no way to escape a suboptimal region.

All runs identical, **$\tau_1 = \tau_2 = \tau_3$**: one solution.



(b) GRAM (Ours)



Diverse paths reach both y_1 and y_2.

GRAM, 50 paths: **some reach the global optimum**.

Why one trajectory is not enough

- **Stuck on one path.** If refinement gets trapped in a suboptimal state, the model cannot escape it.
- **Collapse on multiple answers.** With several valid solutions, it must collapse them into one.
- **Depth-only scaling is slow.** More reasoning means more iterations, run one after another.

Our approach: what sampling brings

→ **(i) Diverse latent trajectories** explore alternative paths and can escape a suboptimal state.

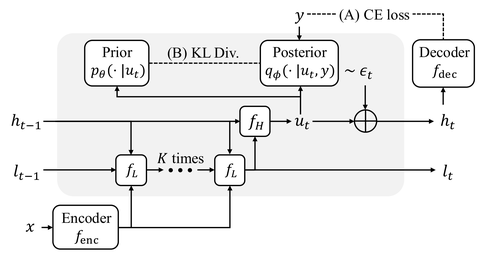
→ **(ii) Multiple valid solutions** from diverse trajectories on the same input.

→ **(iii) Sampling in parallel** adds width scaling.

Method

Architecture — Steering latent trajectories via variational guidance

Each transition acts on $z = (h, l)$: a **deterministic update** u_t, plus **stochastic guidance** ε_t on the abstract level h that **steers the overall trajectory** (μ steers, σ^2 explores).



Low level, K times
$l_{t,k} = f_{\text{L}}(h_{t-1}, l_{t,k-1}, e_x)$

High-level update
$u_t = f_{\text{H}}(h_{t-1}, l_t)$

Stochastic guidance
$\varepsilon_t \sim \mathcal{N}(\mu_\theta(u_t), \sigma^2_\theta(u_t)\, I)$

Guided update
$h_t = u_t + \varepsilon_t$

Training — A posterior that sees the answer teaches the prior

Maximizing $\log p_\theta(y \mid x)$ is intractable, so GRAM uses a **variational posterior** and the **ELBO**; prior and posterior **differ only in the noise**. **Prior** $p_\theta(\varepsilon_t \mid u_t)$ sees only x (test time); **posterior** $q_\varphi(\varepsilon_t \mid u_t, y)$ also sees y (training).

$$\mathscr{L}_{\text{ELBO}} = \underbrace{\mathbb{E}_{q_\varphi}[\, \log p_\theta(y \mid z_{T_{total}}, x)\,]}_{\text{reach the answer}} - \underbrace{\Sigma_t\, \text{KL}(\, q_\varphi(\varepsilon_t \mid u_t, y) \,\|\, p_\theta(\varepsilon_t \mid u_t)\,)}_{\text{at every step, the prior learns the posterior}}$$

Practical objective at each supervision step n (each $\hat{y}^{(n)}$ in the rollout), last transition only:

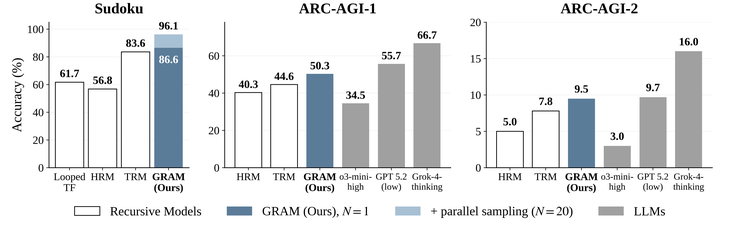
$$\mathscr{L}^{(n)}{}_{\text{GRAM}} = \mathbb{E}_{q_\varphi}[\, \log p_\theta(y \mid z_T{}^{(n)}, x)\,] - \text{KL}(\, q_\varphi(\varepsilon_T{}^{(n)} \mid u_T{}^{(n)}, y) \,\|\, p_\theta(\varepsilon_T{}^{(n)} \mid u_T{}^{(n)})\,)$$

Inference — Sample the prior in parallel

- No answer at test time: drop the posterior; every ε_t comes from the **learned prior** $p_\theta(\varepsilon_t \mid u_t)$.
- **Width**: decode N parallel trajectories into $\hat{y}^{(i)}$ and take the **majority vote**, $\text{argmax}_y\, \Sigma_i\, 1[\hat{y}^{(i)} = y]$.
- **Depth**: more transitions also help, as in HRM and TRM, but run in sequence; **width** runs at once.

Highlighted Results

(1) Structured Reasoning – a **single sample** already beats every recursive baseline, and **parallel sampling amplifies the gain**.



(2) Multiple Solutions – deterministic recursion collapses onto few solutions, while GRAM reaches the **highest coverage** and, surprisingly, **sharper constraint satisfaction than AR and MDLM**.

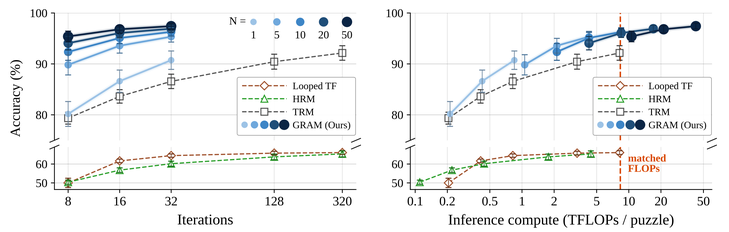
Method	# Params	N	N-Queens 8×8 Acc.	N-Queens 8×8 Cov@20	N-Queens 10×10 Acc.	N-Queens 10×10 Cov@20	Graph Coloring 8-vertex Conflict↓	Graph Coloring 8-vertex Cov@20	Graph Coloring 10-vertex Conflict↓	Graph Coloring 10-vertex Cov@20
Looped TF	7M	–	68.1±7.4	23.6±1.9	50.0±7.6	6.2±3.2	136.0±16.1	20.5±1.5	157.3±9.0	7.2±0.7
HRM	27M	–	78.7±2.9	26.7±1.3	37.8±0.7	4.7±0.1	109.7±1.5	21.8±0.3	164.3±21.6	8.9±1.7
TRM	7M	–	66.8±5.6	22.6±0.8	17.5±11.2	2.0±1.3	109.3±3.1	22.3±0.6	170.7±17.9	6.8±0.3
AR	10.6M	1	96.3±1.0	–	**90.0**±2.2	–	37.3±6.1	–	101.7±4.0	–
MDLM	12.6M	1	96.1±1.5	–	74.3±6.6	–	16.3±2.5	–	57.3±4.7	–
		20	**100.0**±0.0	87.2±0.6	98.6±0.8	47.4±2.2	**2.7**±0.6	84.5±4.0	12.0±7.0	48.2±1.4
GRAM (Ours)	10M	1	**99.5**±0.3	–	89.7±2.7	–	**8.3**±2.5	–	**31.3**±15.0	–
		20	**100.0**±0.0	**89.7**±1.7	**100.0**±0.0	**57.5**±3.4	**2.7**±2.1	**85.8**±0.5	**3.3**±1.5	**51.3**±2.8

(3) Width Scaling – parallel sampling adds a **new scaling axis** and beats deeper recursion at **matched compute**.

15.0× faster than TRM at equal FLOPs

+3.9 pp higher accuracy at equal FLOPs

3.7× less compute to reach 90%



(4) Unconditional Generation – GRAM shows **generative potential**, refining digits from no input and **improving beyond its training depth** while TRM collapses.

