ANYTIME PLANNING WITH CONTINUOUS THOUGHT MACHINES

Eduard Anton
McGill University
eduard.anton@mail.mcgill.ca

ABSTRACT

Robotic agents must operate under partial observability, limited data, and real-time constraints, which pose challenges for many existing planning and control models. Continuous Thought Machines (CTMs) offer an alternative by modelling internal neural dynamics through iterative latent computation, enabling adaptive reasoning and emergent planning capabilities. In this work, we investigate whether CTMs can support rollout-like reasoning under partial observability by leveraging their internal dynamics. We compare CTMs and LSTM-based recurrent agents trained with reinforcement learning on navigation tasks, evaluating performance as observations are withheld for increasing rollout lengths. Our results show that internal recurrence benefits both architectures, but CTMs exhibit distinct behavior: they achieve higher performance and lower policy entropy for short rollout horizons, reflecting structured internal reasoning, while degrading more rapidly for longer rollouts due to accumulated internal drift. These findings highlight CTMs as promising models for adaptive computation and uncertainty-aware planning in real-time robotic systems.

Keywords Continuous Thought Machines · Reinforcement Learning · Partial Observability · Planning · Robotics

1 Introduction

Robotic systems operating in real-world environments must reason under partial observability, limited data, and strict real-time constraints. Recently, large pretrained foundation models, such as large language models (LLMs), vision–language models (VLMs), and vision–navigation models (VNMs), have been increasingly adopted in robotics due to their strong generalization capabilities and great performance across a wide range of domains. These models have shown particular strengths for high-level reasoning, task specification, and planning [8]. Despite these advantages, the deployment of these models in robotics presents several challenges. Foundation models often require large amounts of training data, exhibit slow and computationally expensive inference, and in some cases rely on continuous internet connectivity, making them ill-suited for real-time, low-level embedded, and safety-critical robotic applications. Moreover, they typically lack reliable uncertainty quantification and are known to hallucinate, producing confident but incorrect outputs [4]. These limitations motivate the exploration of alternative models that can retain some of the flexibility and generalization benefits of large models while being better aligned with the constraints of low-level systems. In particular, models that can reason internally and maintain robust behavior under uncertainty are of significant interest.

1.1 Continuous Thought Machines

Continuous Thought Machines (CTMs) from Figure 1 are a recent class of models motivated by the observation that biological intelligence arises from time-dependent neural dynamics. In biological systems, mechanisms such as spike-timing-dependent plasticity [1] shape how neural activity evolves over time, enabling flexible computation and planning intelligence. By contrast, most conventional neural networks rely on memoryless nonlinearities such as ReLU, and even recurrent neural networks primarily propagate state rather than modeling continuous internal neural dynamics. CTMs introduce three key ingredients to address this limitation. First, CTMs incorporate an explicit internal time dimension in the latent space. While recurrence and iterative computation have been explored previously, for example in chain-of-thought reasoning [7] and other recurrent latent-space methods [5], CTMs perform multiple internal computation steps, referred to as ticks, within a single environment step, during which it uses the learned continuous time-based dynamics model. This enables iterative refinement and adaptive computation, where additional internal processing can improve decision quality. Second, CTMs employ neuron-level models (NLMs), where neurons learn their own continuous, time-dependent activation functions through individual temporal update rules. Third, CTMs use neural synchronization to produce coherent representations from distributed neural activity [3]. Synchronization provides a structured output of the model's internal dynamics and enables the expression of confidence in its outputs, for example through entropy. This makes uncertainty an explicit, measurable property of the model rather than an implicit artifact. Together, these mechanisms provide CTMs with structured internal dynamics that can be leveraged for cheaper planning and decision-making compared to foundation models.

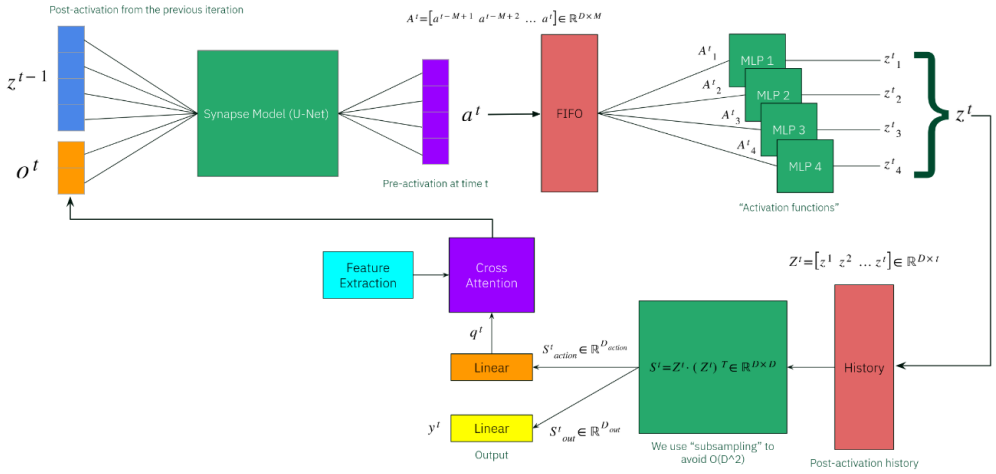


Figure 1: CTM Architecture [3]. The synapse model produces neuron-wise pre-activations from the observation and previous latent state. Neuron-level models transform pre-activation histories into post-activations using learned continuous activation dynamics. A synchronization module aggregates post-activations to produce the output and the modifies the latent state, forming a recurrent loop.

1.2 Internal Rollouts Without Explicit World Models

A key motivation of this work is to investigate whether the internal dynamics of CTMs can be leveraged in a manner analogous to model-based reinforcement learning approaches such as Dreamer. In Dreamer-style algorithms, agents learn an explicit world model and perform imagined rollouts to plan future actions [6]. While powerful, such approaches rely on learned latent dynamics models that can suffer from compounding errors and hallucinations. In contrast, CTMs do not explicitly learn a world model. Rather than predicting future observations, they evolve a recurrent latent state over multiple internal computation steps. This raises the question of whether CTMs can perform rollout-like reasoning using partial observations, without

requiring an explicit predictive model or the complexity and opacity associated with learned latent world models, which is particularly relevant for real-time, low-level control systems. The goal of this paper is to explore this hypothesis by systematically evaluating the robustness of CTMs under increasing observation gaps. Specifically, we study how CTMs and LSTM-based recurrent agents perform when observations are frozen for k consecutive steps, forcing the model to rely on its internal dynamics rather than on a reactionary policy. By increasing k, we directly probe each model's ability to maintain coherent internal representations and action policies in the absence of external feedback.

2 Methods

We evaluate CTMs and LSTM-based agents trained using reinforcement learning with Proximal Policy Optimization (PPO) in the MiniGrid FourRooms environment [2] seen in Figure 2. This task consists of a maze composed of four interconnected rooms, where both the agent and a goal square are randomly placed at the start of each episode. The agent must navigate to the goal within a maximum of 300 steps. Observations are limited to a grid-based view of nearby tiles, where no explicit position, velocity, or global state information is provided.

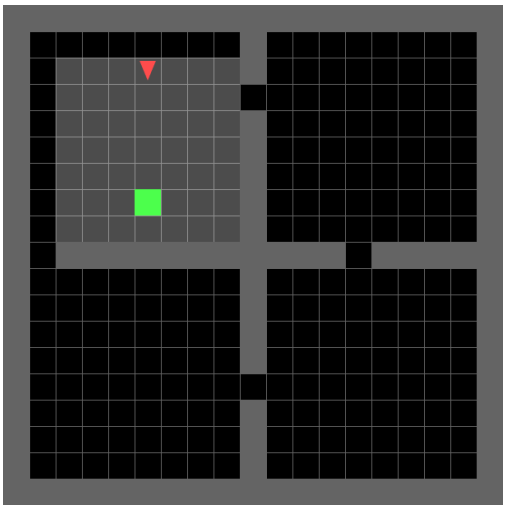


Figure 2: Four Rooms MiniGird [2].

To measure the performance of the agents under partial observability, we introduce a structured observation-free rollout algorithm. Each episode alternates between a short warmup phase, during which observations are provided normally, and a rollout phase in which observations are frozen for k consecutive steps. During rollout, the agent must act solely based on its internal recurrent state. After the rollout phase, observations are restored and the cycle repeats. In the measured results, evaluation uses a fixed warmup of three steps, while the rollout length is varied to systematically increase reliance on internal dynamics. We evaluate agents using episode return, success rate, and policy entropy. Episode return reflects how efficiently the agent reaches the goal, while success rate measures the frequency of successful episodes. Policy entropy is computed during rollout steps to capture the uncertainty in the absence of sensory input.

2.1 Pretrained Baselines

For the pretrained setting, we use the baseline CTM and LSTM models provided by the authors of the original CTM paper [3]. Both models have comparable sizes, with approximately 7 million parameters, a model input dimension of 128, and a hidden width of 512, ensuring a fair comparison. Both agents are trained using PPO with two internal thinking iterations and are optimized under fully observable conditions (k=0), where observations are available at every

environment step. These models serve as reference points for evaluating rollout robustness without additional adaptation.

2.2 Finetuned

To adapt both architectures to partial observability and rollout conditions, we further finetune the pretrained CTM and LSTM agents on a NVIDIA RTX 4070 SUPER. Finetuning is performed for 10 million environment steps, during which each episode samples a random rollout length $k \in [0, 8]$ and a random warmup length of $w \in [1, 4]$. To stabilize learning, rollout difficulty is progressively increased over training, with larger values of k becoming more likely as training progresses. This encourages the models to gradually rely more heavily on their internal dynamics. During finetuning, the PPO entropy coefficient is reduced from the original 0.1 to 0.01 and the learning rate is set to 2×10^{-5}. This change encourages more confident and consistent action selection, which is particularly important when the agent must act without immediate observational feedback. All other training hyperparameters are kept identical across architectures.

3 Results

We evaluate the performance of CTM and LSTM agents under increasing observation gaps, implemented by freezing observations for k consecutive steps in Figure 3. As expected, increasing k forces the agents to rely more heavily on their internal state, and performance degradation occurs in proportion to k. Moreover, across all configurations, both recurrent architectures benefit from internal recurrence, even when not explicitly trained for rollout conditions. However, we observe diminishing returns from using more than two internal computation ticks.

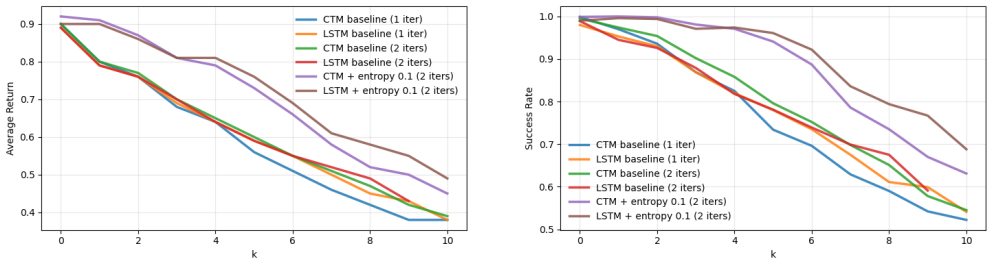
For pretrained models, CTMs and LSTMs achieve comparable average returns and success rates under full observability. With two internal iterations, CTMs maintain a higher success rate than LSTMs up to k=6. In contrast, CTMs with a single internal iteration perform similarity to LSTMs until they begin to degrade more rapidly for k>4. Across all settings, CTMs generally maintain or slightly reduce their policy entropy during rollout steps, suggesting increasing confidence as internal computation unfolds. In contrast, LSTM baselines exhibit a monotonic increase in entropy as rollout lengthens, indicating growing uncertainty in the absence of new observations.

When models are fine-tuned under partial observability, overall performance improves for small rollout gaps. In particular, the fine-tuned CTM outperforms the fine-tuned LSTM and the pretrained CTMs for short rollouts until k=4, achieving higher returns and success rates while maintaining substantially lower entropy. Beyond this threshold, however, CTM performance degrades more rapidly than that of the LSTM as k increases. Despite this decline, the absolute entropy values of the CTM remain consistently lower than those of the LSTM across all rollout lengths, indicating a less stochastic policy. While both architectures exhibit increasing entropy during rollouts, because of the accumulating uncertainty, the CTM maintains a more concentrated action distribution throughout.

4 Discussion

4.1 Analysis

The results demonstrate that internal recurrence in latent space is beneficial for both CTM and LSTM agents when operating under observation gaps. As observations are reduced, both architectures demonstrate reliance on their recurrent state, and performance degrades more gracefully than with no ticks. This confirms that internal state propagation is a key mechanism for improving coherent behavior and planning under partial observability. However, the two models exhibit fundamentally different behaviors in how they leverage this recurrence. The LSTM behaves primarily as a reactive, stochastic policy: as the observation gap grows, the



(a) Reward return over k

(b) Success rate over k

Figure 3: Results for both CTM and LSTM as the rollout window k is increased.

policy entropy increases steadily, reflecting rising uncertainty in action selection due to missing external feedback. In contrast, the CTM operates as an internally dynamics-driven policy. For short rollout horizons, its evolving latent dynamics enable iterative refinement of action selection, effectively planning, allowing the model to compensate for missing observations and maintain coherent behavior, which explains its superior performance at small values of k. As the rollout horizon increases, however, the CTM's internal state progressively drifts from the true environment state, since no new observations are available to correct its latent dynamics. This accumulated internal mismatch leads to a rapid degradation in performance at larger k, producing a sharper collapse than that observed for the LSTM. This behavior is similar to classical state estimation methods such as Kalman filtering, where uncertainty grows as predictions drift from the true state in the absence of measurements, rather than to model-based approaches such as Dreamer [6] that explicitly learn and roll out a predictive world model.

4.2 Limitations

A primary limitation of this study is the choice of evaluation environment. While the MiniGrid Four Rooms task is a standard benchmark for partial observability and navigation [2], it may not fully expose the strengths of CTMs relative to LSTMs. The environment is relatively simple, and both baseline models achieve near-perfect performance under full observability, leaving limited potential performance gain to distinguish architectural advantages. As a result, performance differences under increasing rollout gaps are subtle and primarily reflected in secondary metrics such as entropy rather than large return gaps. More importantly, this environment provides limited global context and requires only modest long-term planning once the goal is discovered. CTMs are designed to exploit rich internal dynamics and are particularly well-suited for tasks that require long-horizon reasoning when sufficient contextual information is available. In the original CTM work [3], substantial gains were demonstrated on tasks such as maze solving and parity problems under full context, where iterative internal reasoning plays a central role. In contrast, the partial observability constraint in this study limits the amount of information available to the model, which in turn restricts the long-term planning capabilities of CTMs.

5 Conclusion

This work examined whether the internal dynamics of Continuous Thought Machines can support rollout-like reasoning under partial observability. By comparing CTMs with LSTM-based recurrent agents in the Four Rooms environment [2], we showed that internal recurrence in latent space benefits both architectures, but is exploited in different ways. CTMs are particularly effective for short rollout horizons, where their internal time dimension and neuron-level dynamics enable adaptive computation and iterative refinement of actions [3]. This results in higher performance and lower policy entropy compared to LSTMs under smaller observation gaps. However, as rollouts extend, internal dynamics increasingly drift from the lack of corrective observations, leading to faster performance degradation. These findings highlight CTMs as a promising architecture for adaptive computation and planning in dynamic systems, rather than purely reactive control. Their ability to model internal dynamics and explicitly

represent uncertainty makes them especially relevant for real-time robotic applications. Future work should evaluate CTMs in more complex environments with richer context and longer planning horizons, where their advantages may become more relevant.

References

[1] N. Caporale and Y. Dan. Spike timing–dependent plasticity: A hebbian learning rule. *Annual review of neuroscience*, 31:25–46, 02 2008.

[2] M. Chevalier-Boisvert, B. Dai, M. Towers, R. Perez-Vicente, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

[3] L. Darlow, C. Regan, S. Risi, J. Seely, and L. Jones. Continuous thought machines, 2025.

[4] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, and M. Schwager. Foundation models in robotics: Applications, challenges, and the future, 2023.

[5] J. Geiping, S. McLeish, N. Jain, J. Kirchenbauer, S. Singh, B. R. Bartoldson, B. Kailkhura, A. Bhatele, and T. Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025.

[6] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models, 2025.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[8] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans. Foundation models for decision making: Problems, methods, and opportunities, 2023.

Table 1: Performance metrics for baseline models pretrained for 1 iteration.

Model	k	Return ($\mu \pm \sigma$)	Entropy	Success
CTM Baseline (1 iter)	0	0.90 ± 0.10	–	0.999
	1	0.80 ± 0.19	1.539	0.970
	2	0.76 ± 0.24	1.565	0.936
	3	0.68 ± 0.31	1.614	0.869
	4	0.64 ± 0.33	1.632	0.825
	5	0.56 ± 0.37	1.667	0.734
	6	0.51 ± 0.38	1.679	0.696
	7	0.46 ± 0.39	1.682	0.629
	8	0.42 ± 0.39	1.690	0.590
	9	0.38 ± 0.39	1.688	0.542
	10	0.38 ± 0.39	1.697	0.522
LSTM Baseline (1 iter)	0	0.89 ± 0.17	–	0.980
	1	0.79 ± 0.22	1.558	0.953
	2	0.76 ± 0.25	1.587	0.930
	3	0.69 ± 0.31	1.639	0.869
	4	0.64 ± 0.34	1.661	0.820
	5	0.59 ± 0.36	1.681	0.780
	6	0.55 ± 0.37	1.688	0.735
	7	0.50 ± 0.39	1.704	0.675
	8	0.45 ± 0.40	1.721	0.611
	9	0.43 ± 0.39	1.719	0.599
	10	0.38 ± 0.39	1.728	0.541

Table 2: Performance metrics for baseline models pretrained for 2 iterations.

Model	k	Return ($\mu \pm \sigma$)	Entropy	Success
CTM Baseline (2 iters)	0	0.90 ± 0.11	–	0.995
	1	0.80 ± 0.19	1.562	0.974
	2	0.77 ± 0.22	1.573	0.954
	3	0.70 ± 0.28	1.606	0.902
	4	0.65 ± 0.31	1.626	0.858
	5	0.60 ± 0.35	1.655	0.796
	6	0.55 ± 0.36	1.666	0.752
	7	0.51 ± 0.38	1.678	0.698
	8	0.47 ± 0.39	1.684	0.651
	9	0.42 ± 0.39	1.696	0.578
	10	0.39 ± 0.39	1.705	0.545
LSTM Baseline (2 iters)	0	0.89 ± 0.15	–	0.989
	1	0.79 ± 0.23	1.572	0.945
	2	0.76 ± 0.26	1.595	0.926
	3	0.70 ± 0.30	1.630	0.879
	4	0.64 ± 0.34	1.664	0.818
	5	0.59 ± 0.35	1.672	0.781
	6	0.55 ± 0.37	1.689	0.739
	7	0.52 ± 0.38	1.694	0.699
	8	0.49 ± 0.38	1.699	0.675
	9	0.43 ± 0.40	1.711	0.591
	10	–	–	–

Table 3: Performance metrics for models finetuned with entropy coefficient 0.1 (2 iterations).

Model	k	Return ($\mu \pm \sigma$)	Entropy	Success
CTM + Entropy 0.1	0	0.92 ± 0.07	–	0.999
	1	0.91 ± 0.06	0.127	1.000
	2	0.87 ± 0.12	0.147	0.998
	3	0.81 ± 0.20	0.155	0.981
	4	0.79 ± 0.22	0.182	0.971
	5	0.73 ± 0.27	0.206	0.941
	6	0.66 ± 0.31	0.214	0.887
	7	0.58 ± 0.37	0.231	0.786
	8	0.52 ± 0.38	0.256	0.735
	9	0.50 ± 0.40	0.282	0.670
	10	0.45 ± 0.40	0.292	0.631
LSTM + Entropy 0.1	0	0.90 ± 0.13	–	0.990
	1	0.90 ± 0.09	0.330	0.996
	2	0.86 ± 0.15	0.376	0.994
	3	0.81 ± 0.21	0.401	0.971
	4	0.81 ± 0.21	0.395	0.974
	5	0.76 ± 0.23	0.404	0.961
	6	0.69 ± 0.29	0.443	0.922
	7	0.61 ± 0.34	0.460	0.836
	8	0.58 ± 0.36	0.465	0.794
	9	0.55 ± 0.37	0.481	0.767
	10	0.49 ± 0.39	0.511	0.688