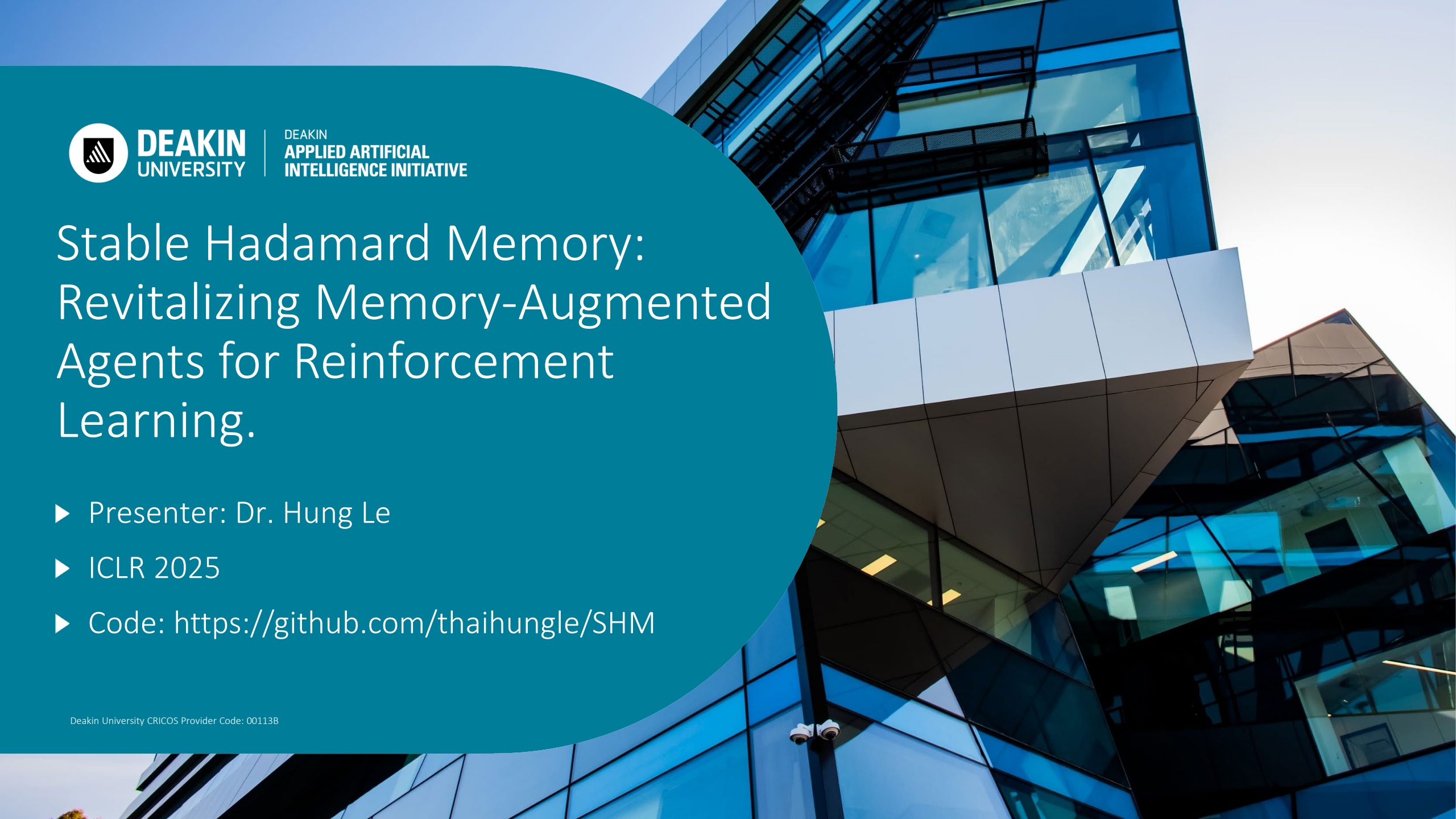


DEAKIN UNIVERSITY | DEAKIN APPLIED ARTIFICIAL INTELLIGENCE INITIATIVE

Stable Hadamard Memory: Revitalizing Memory-Augmented Agents for Reinforcement Learning.

- Presenter: Dr. Hung Le
- ICLR 2025
- Code: https://github.com/thaihungle/SHM

About Us and the Topic

- Our lab: Applied AI Initiative, Deakin University
- Hung Le is a DECRA fellow, senior lecturer at Deakin University, leading research on deep sequential models and reinforcement learning
- Why memory in RL?
  - Deep RL Agents struggles in long-term tasks that require remembering past events in the trajectory
  - In real-world setting, with robots as life-long agents, the ability to maintain memory of past events is critical for assisting human in daily activities

A2I2 Lab Foyer, Waurn Ponds

DEAKIN UNIVERSITY | DEAKIN APPLIED ARTIFICIAL INTELLIGENCE INITIATIVE

Background

Reinforcement Learning Basics

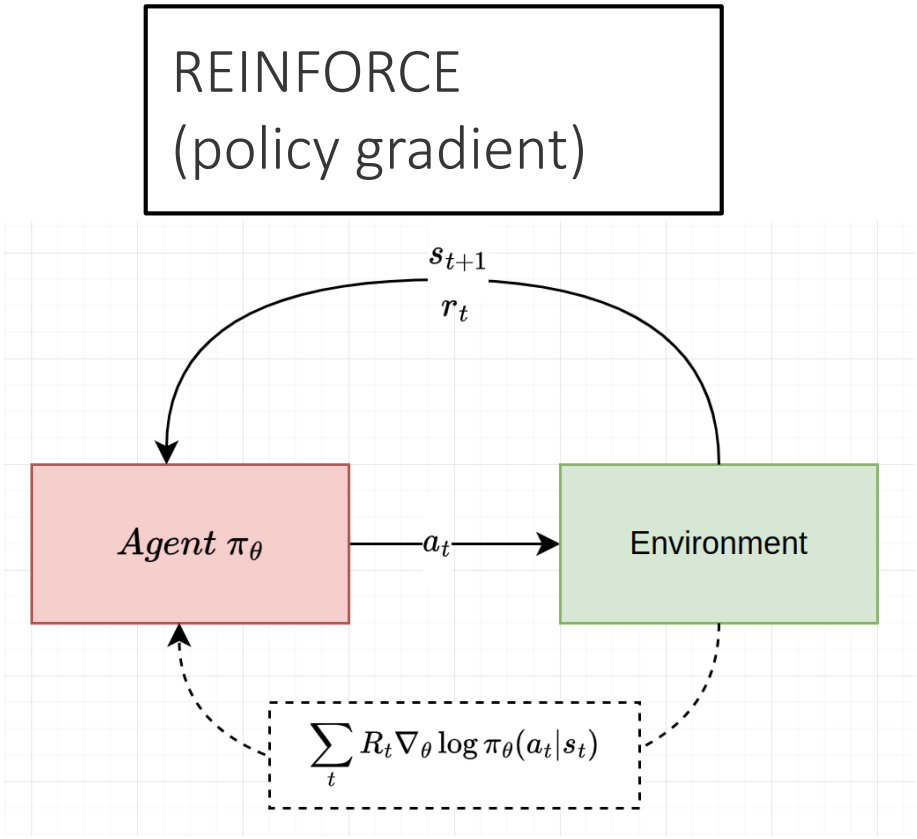
- In reinforcement learning (RL), an agent interacts with the environment, taking actions *a*, receiving a reward *r*, and moving to a new state *s*
- The agent is tasked with maximizing the accumulated rewards or returns *R* over time by finding optimal actions (policy)

Example: Mouse Robot

- The state space is discrete. We have 6 states corresponding to 6 locations in the map
- The action space is discrete. We have 4 actions corresponding to 4 movements
- The reward can be “nothing”, “poison”, “1 cheese” or “3 cheese”. We can convert to scalars: 0, -1,1,3

RL algorithms: Policy Gradient

- Basic idea: directly optimise the policy as a function of states
- Need to estimate the gradient of the objective function E(∑R) w.r.t the parameters of the policy
- Some algorithms: REINFORCE, PPO, GRPO …

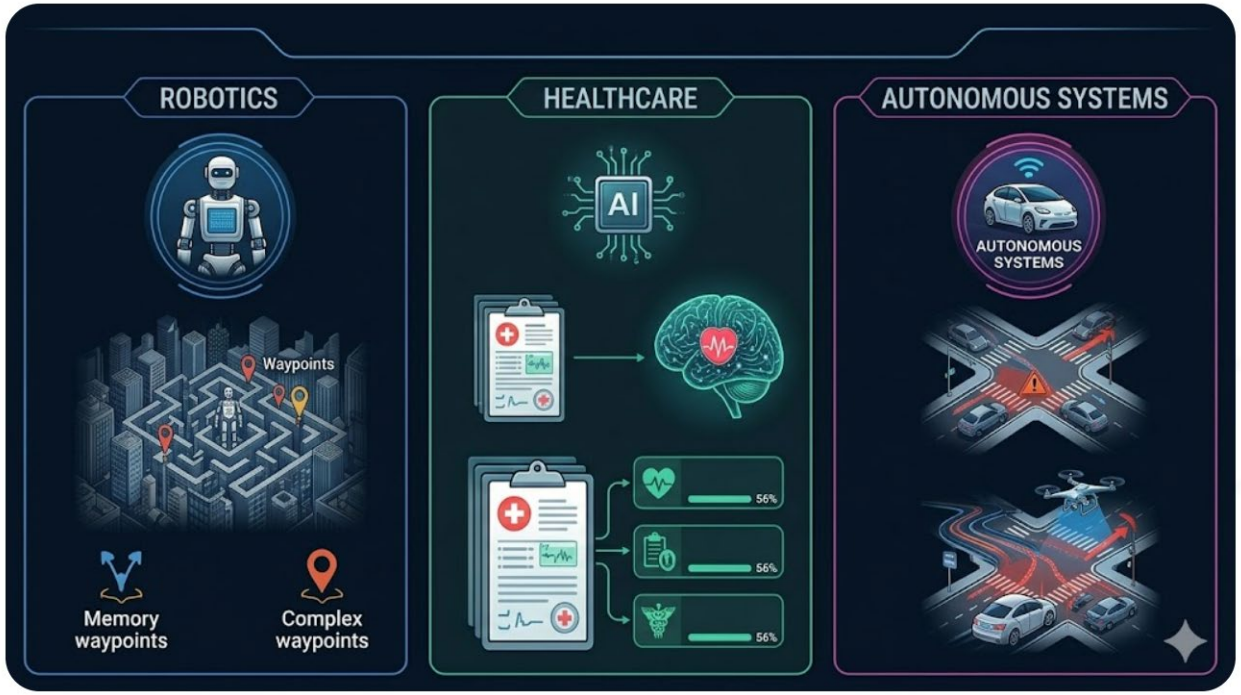


DEAKIN UNIVERSITY | DEAKIN APPLIED ARTIFICIAL INTELLIGENCE INITIATIVE

The Problem

Real-world Agents Need Memory

- Real agents must look past immediate observations to solve complex tasks
- Partial Observability: Critical information is often hidden from the current view
- Without memory, complex sequential decision-making becomes intractable



Source: AI (Nano Banana 2)

POMDP: State is Not Enough

- Partially Observable Environments:
  - States do not contain all required information for optimal action
  - E.g. state=position, does not contain velocity
- Ways to improve:
  - Build richer state representations
  - Memory of all past observations/actions
  - Example: RNN Memory

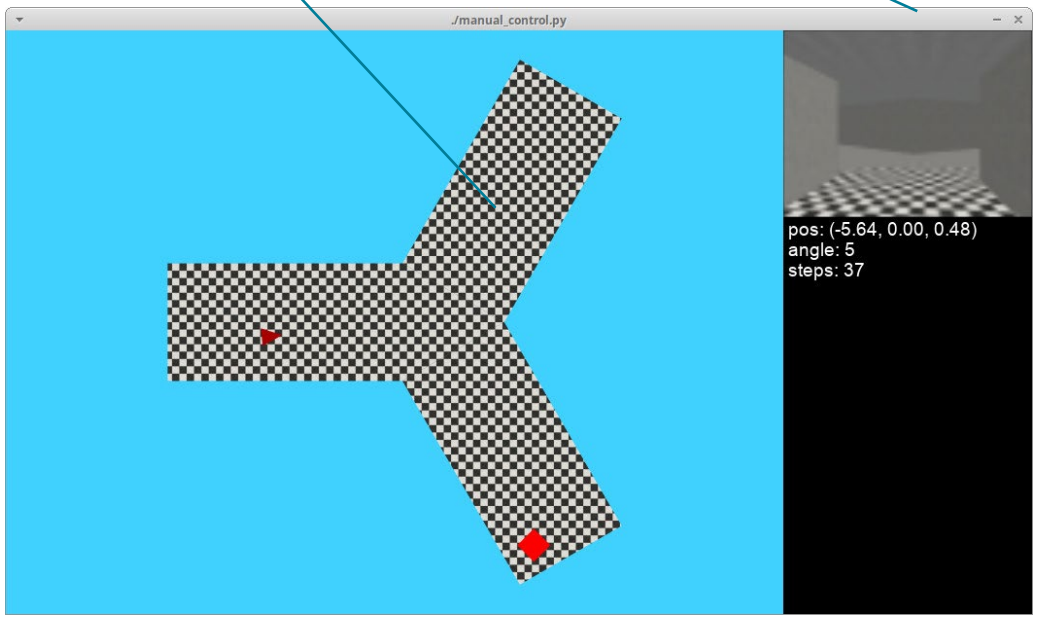
$$h_t = \langle o_0, a_0, o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t \rangle$$

- Policy gradient

$$\nabla_\theta J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi(a_t | h_t^n) R_t^n$$

Full map

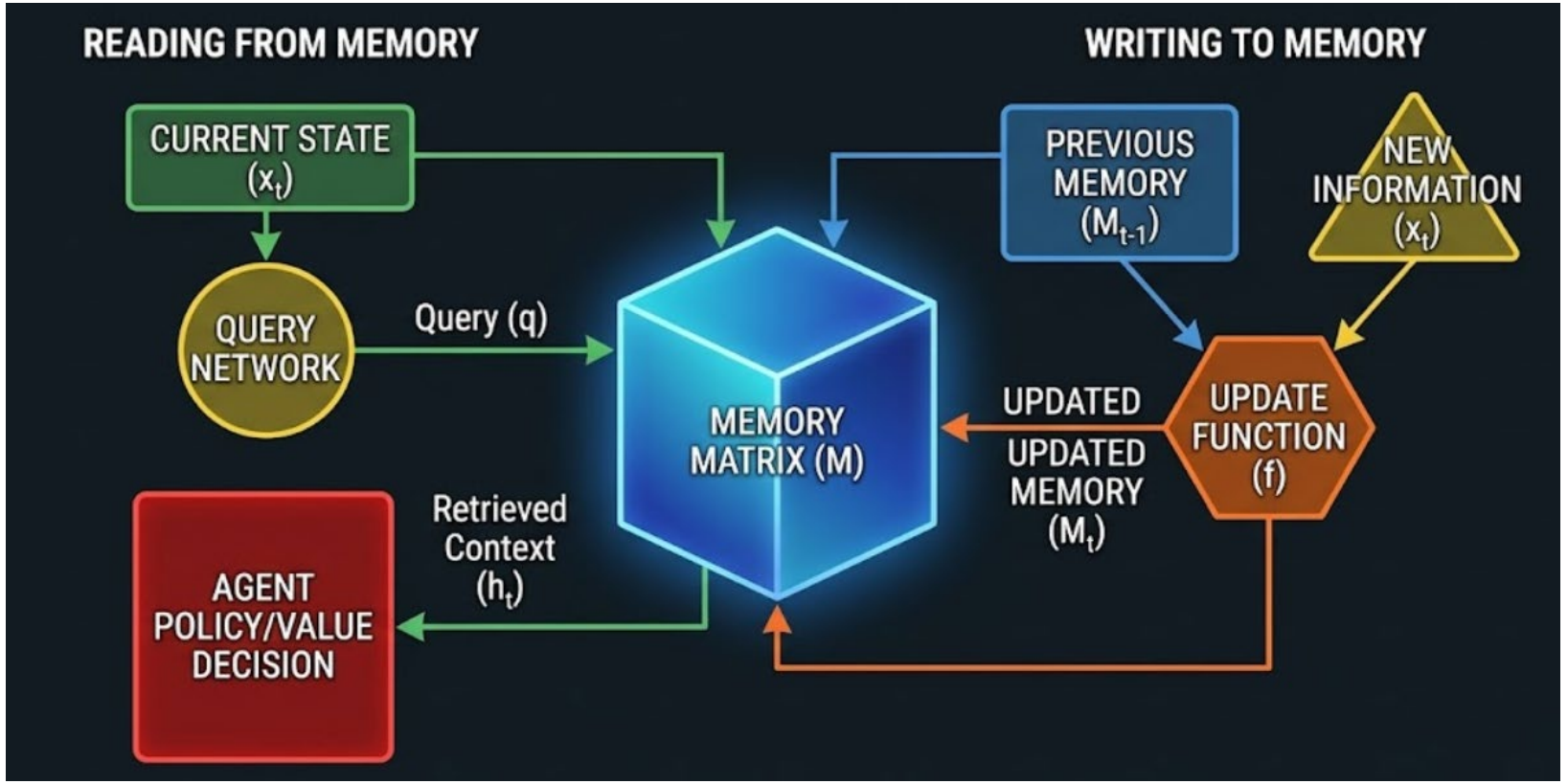
Observed state



RNN hidden state

RNN as policy model

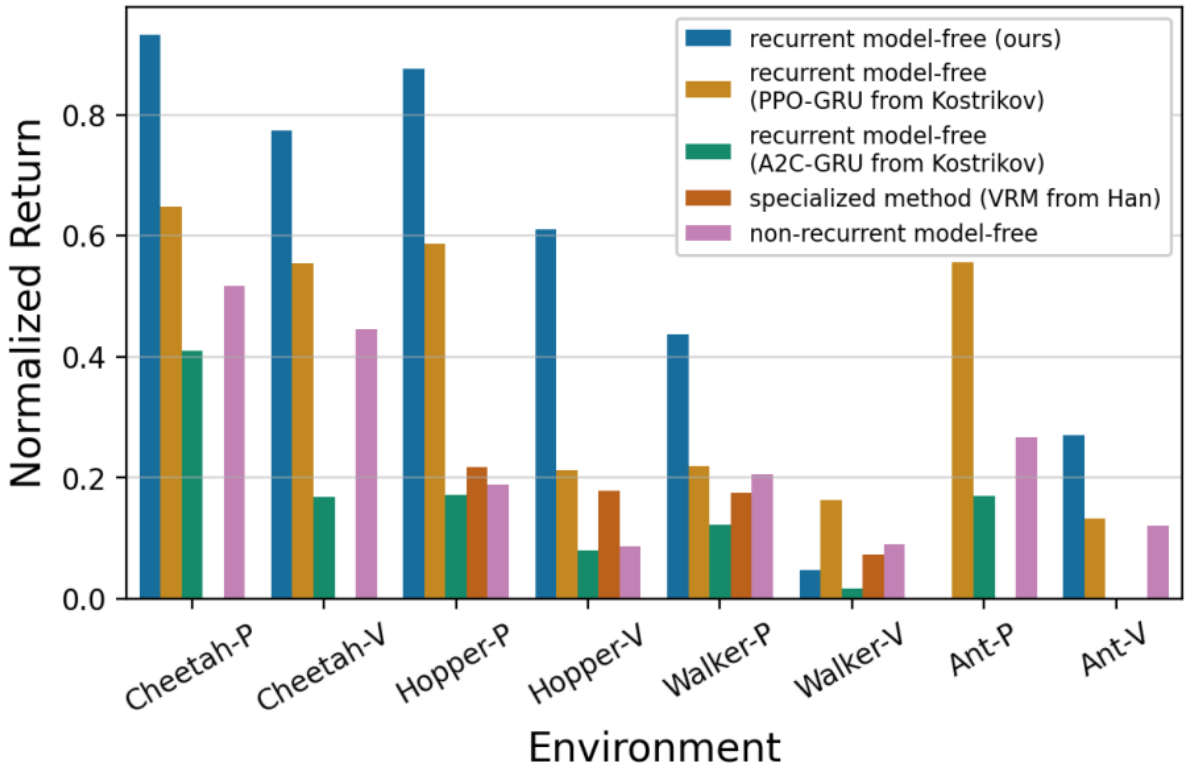
Generic Memory Operations



Source: AI (Nano Banana 2)

- RNN, LSTM, GRU
- Transformer
- Mamba, XLSTM
- Linear Attention
- NTM, DNC
- ...

Memory: Recurrence or Attention?



Performance

Speed

RNN

Transformer

Ni, Tianwei, Benjamin Eysenbach, and Ruslan Salakhutdinov. "Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs." In International Conference on Machine Learning, pp. 16691-16723. PMLR, 2022

Limitations of Current Memory

In dynamic, high-dimensional environments, agents struggle to filter out noise and prioritize truly relevant information

- The "Attention" Trade-off: While powerful, attention mechanisms are often computationally expensive, data-hungry, and slow to identify critical memories
- RNN Bottlenecks:
  - Short Horizon: Standard RNNs suffer from vanishing gradients, limiting their effective memory span to recent events
  - Sequential Dependency: Inherent step-by-step processing prevents parallelization, resulting in slower training and inference compared to Transformers

Limitations

DEAKIN UNIVERSITY | DEAKIN APPLIED ARTIFICIAL INTELLIGENCE INITIATIVE

The Solution

Stable Hadamard Memory (SHM)

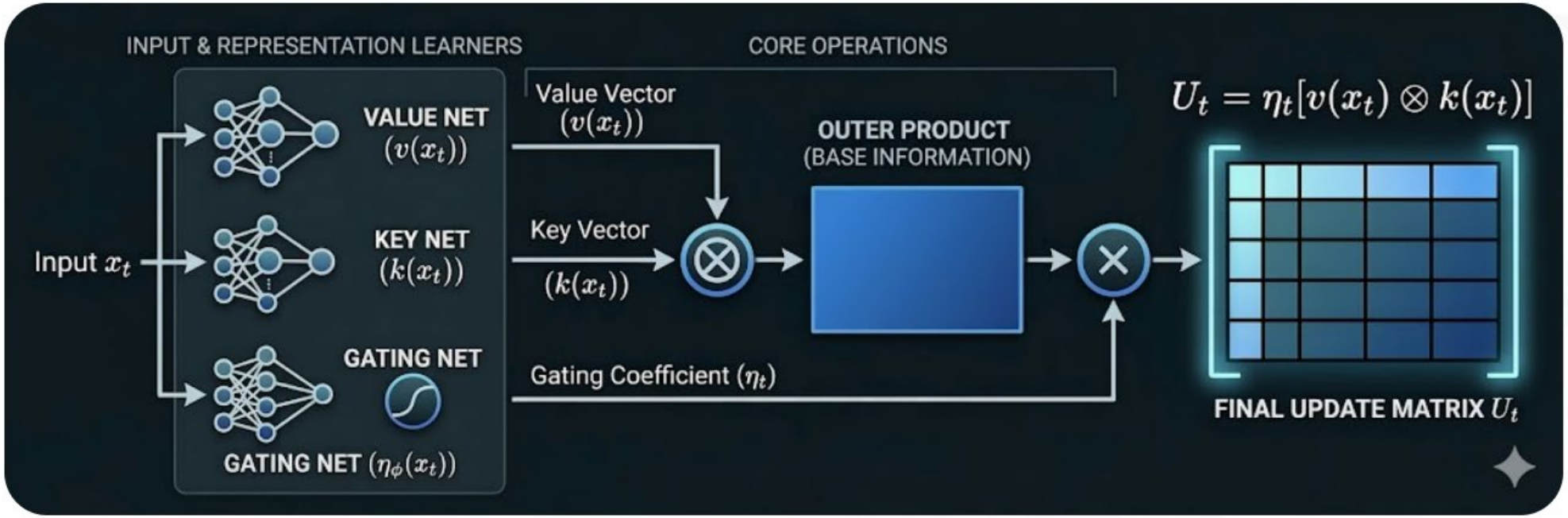
- SHM use large matrix memory M and formulate memory writing into two distinct, parallel operations: Calibration and Update
- Calibration (C): Dynamically strengthens or weakens existing memory cells based on the current input (x), effectively fine-tuning the importance of past experiences
- Update (U): Encodes and integrates fresh information into the memory matrix, functioning like adding new entries to a structured journal

$$M_t = M_{t-1} \odot \underbrace{C_\theta(x_t)}_{C_t} + \underbrace{U_\varphi(x_t)}_{U_t}$$

Hadamard Product avoids interference between unrelated memory cells, ensuring stable and context-aware updates.

Update Matrix Design

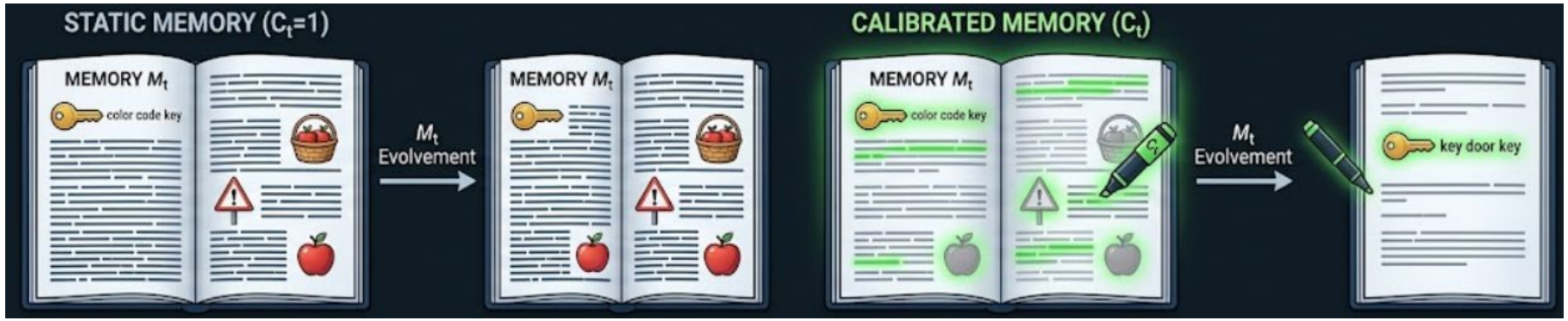
Source: AI (Nano Banana 2)



$$U_{\varphi}\left(x_t\right) = \eta_{\varphi}\left(x_t\right)\left[v\left(x_t\right) \otimes k\left(x_t\right)\right]$$

About The Calibration

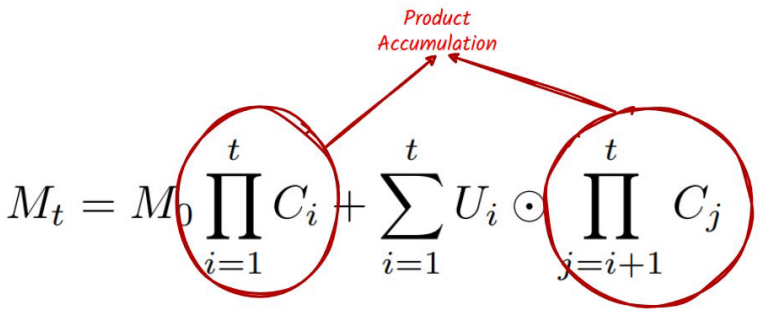
- Calibration acts as a learned "forgetting" mechanism (C→ 0) that prevents irrelevant data overload (e.g., ignoring apple-picking details while preserving a critical door code).
- By adjusting the magnitude of memory cells via the Hadamard product, calibration ensures that important signals persist over time while noise (C→ 1)
- Adaptive Contextualizing: calibrated memory can selectively "re-highlight" information if it becomes relevant again later in the sequence



Source: AI (Nano Banana 2)

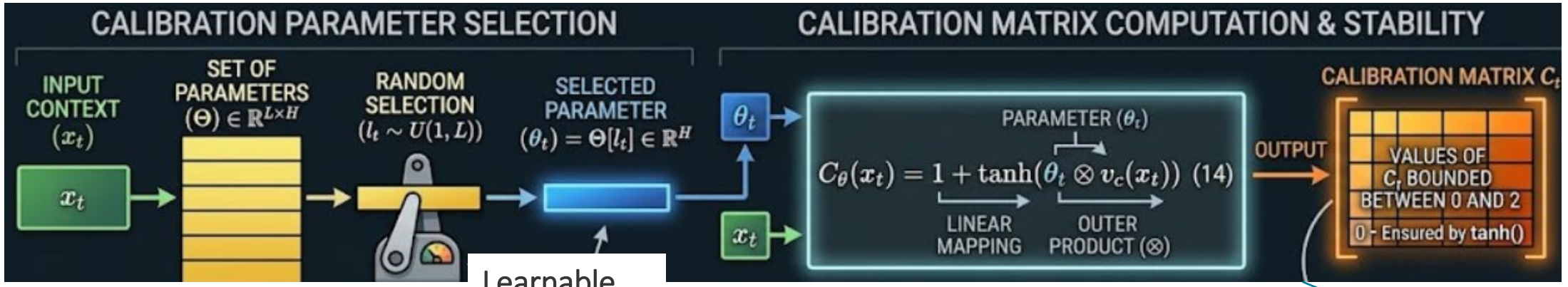
Designing Stable Calibration Matrix

- **Why Care About Stability?** Looking at the explicit form of the memory update, we can see the cumulation of calibration through the accumulated product of C
- Therefore, memory updates can face two major pitfalls:
  - Exploding or Vanishing Gradients: Without proper bounds, the cumulative effect of updates can grow too large (exploding) or shrink to zero (vanishing). This disrupts learning.
  - Overlapping Dependencies: If memory updates at one timestep are too closely tied to updates at another, the system loses flexibility and struggles with new contexts.



Calibration Matrix Design

Source: AI (Nano Banana 2)

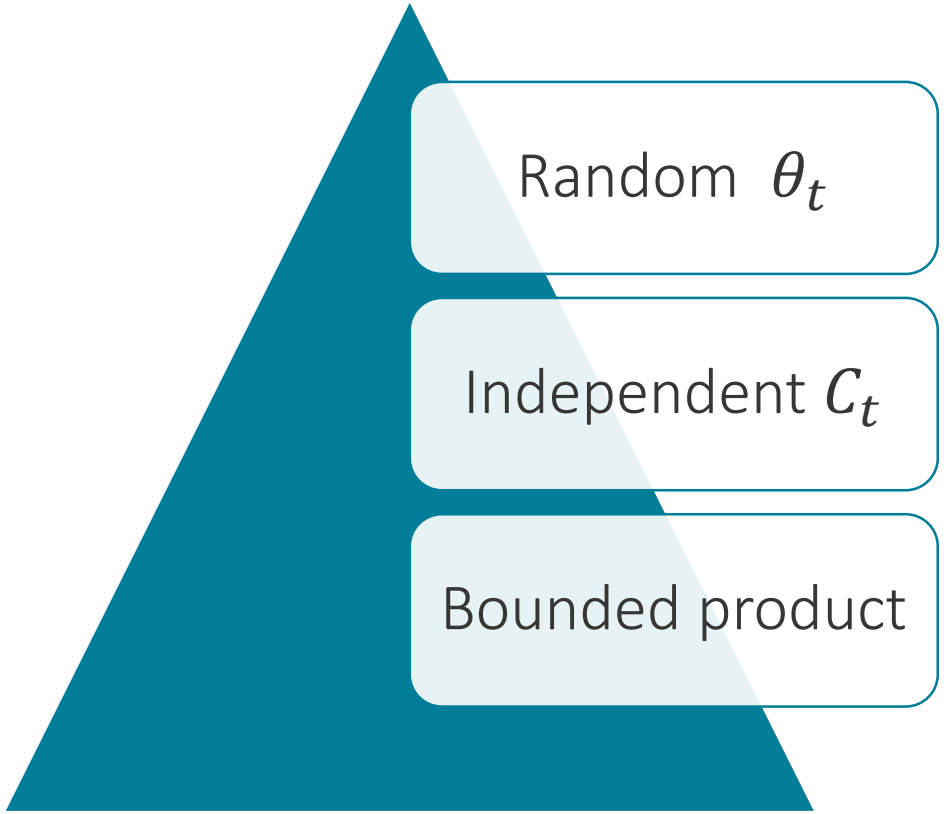


$$C_\theta(x_t) = 1 + \tanh(\theta_t \otimes v_c(x_t))$$

$$\mathbb{E}\left[\prod_{t=1}^{T} C_t\right] \approx 1$$

Why Randomly Select θ_t?

- Randomly sampling θt breaks dependencies between timesteps, reducing correlations and enabling the memory to adapt to new inputs
- Proposition 5 in the paper mathematically supports this idea by showing that random sampling minimizes the Pearson correlation between timesteps
- This ensures that the calibration matrix adjusts independently for each timestep, promoting that the cumulative product of calibration matrices is bounded



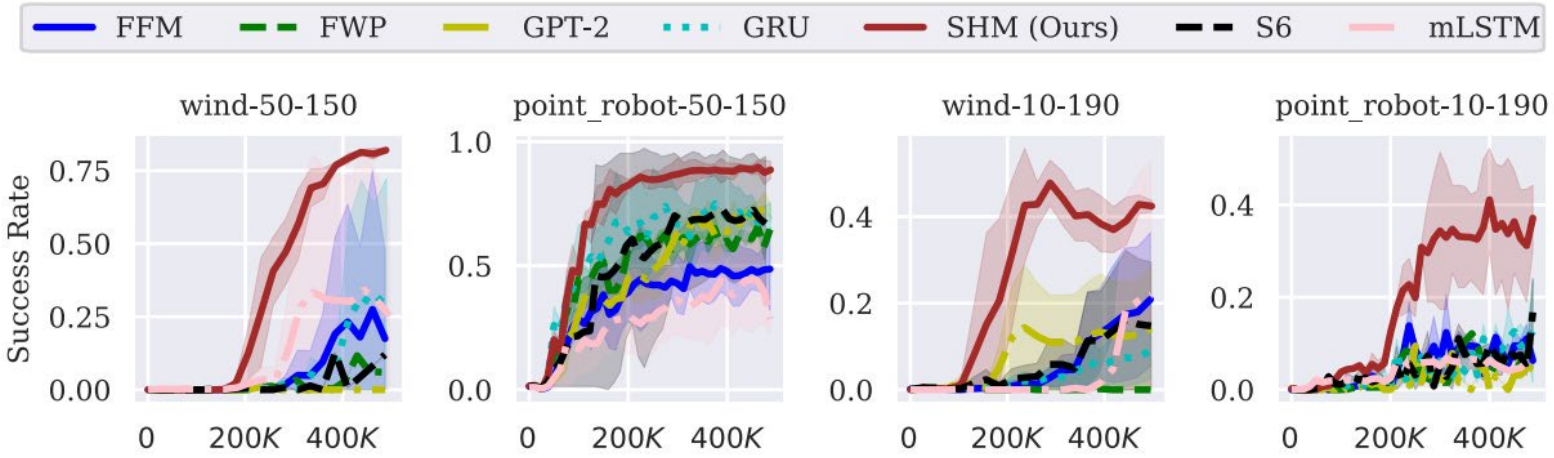
DEAKIN UNIVERSITY | DEAKIN APPLIED ARTIFICIAL INTELLIGENCE INITIATIVE

Experimental Results

Meta-Reinforcement Learning

- Meta-reinforcement learning (Meta-RL) challenges agents to adapt to ever-changing environments with varying tasks and goals
- **Wind:** The agent's goal is fixed but affected by random "wind" forces that shift its trajectory
- **Point Robot:** The goal changes between episodes, requiring the agent to adapt

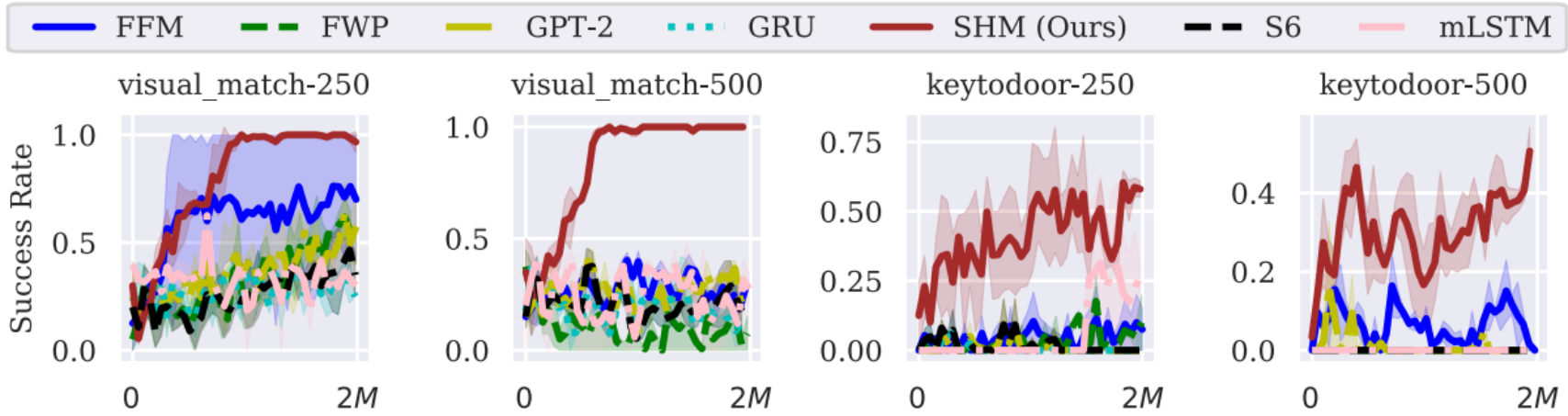
- Easy Mode (50 training tasks, 150 testing tasks): SHM achieved near-optimal success, outperforming other models by 20-50%.
- Hard Mode (10 training tasks, 190 testing tasks): SHM maintained a ~20% performance lead, learning faster and adapting earlier than baselines like GRU and FFM



Long-Term Credit Assignment

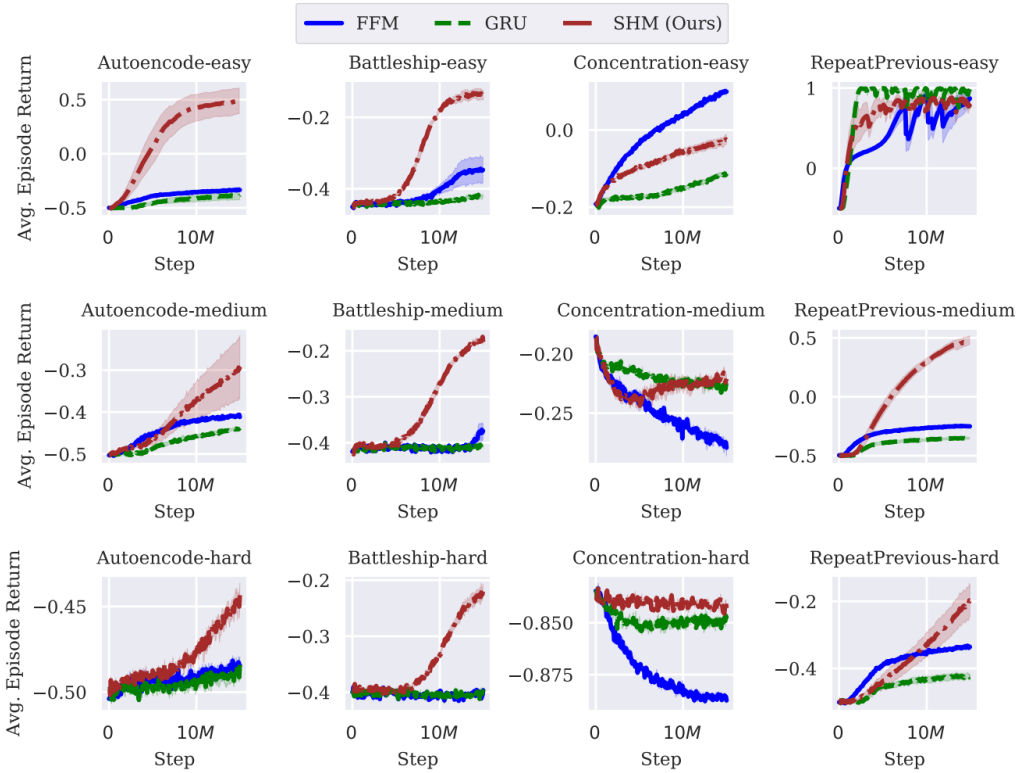
- Some tasks, like connecting distant events, require agents to retain and recall information over extended timeframes
- **Visual Match:** Agents observe a color code early on, pick apples for intermediate rewards, and match the code to a door for the final reward
- **Key-to-Door:** The agent collects a key in one phase, performs unrelated tasks, and later uses the key to open a door

- In Visual Match (250 and 500 steps), SHM was the only model to achieve perfect success rates, while FFM reached 77% (250 steps) and 25% (500 steps)
- In Key-to-Door, SHM demonstrated similarly high success rates, while other models, including GPT-2, showed little to no learning.



POPGym Benchmarks

- POPGym is the ultimate testbed for memory models, offering ultra-long tasks like Autoencode, Battleship, and RepeatPrevious that require agents to memorize and use information across 1,024 steps
- SHM outperformed state-of-the-art memory models (e.g., GRU, FFM) by 10-12% on average in the most challenging tasks:
  - RepeatPrevious: Only SHM consistently showed learning in the Medium/Hard modes.
  - Autoencode: SHM excelled across Easy/Medium tasks, proving its scalability in ultra-long scenarios.



Conclusion and QA

✓ Dynamic Memory Updates: C_t adjusts based on the current context x_t enabling adaptive forgetting and prioritization of critical information

✓ Numerical Stability: By bounding C_t, SHM prevents gradients from exploding or vanishing

✓ Efficient Training: The use of Hadamard products enables parallel training and inference