Learning, Fast and Slow

A new **goal-directed**, **memory-based** learning system

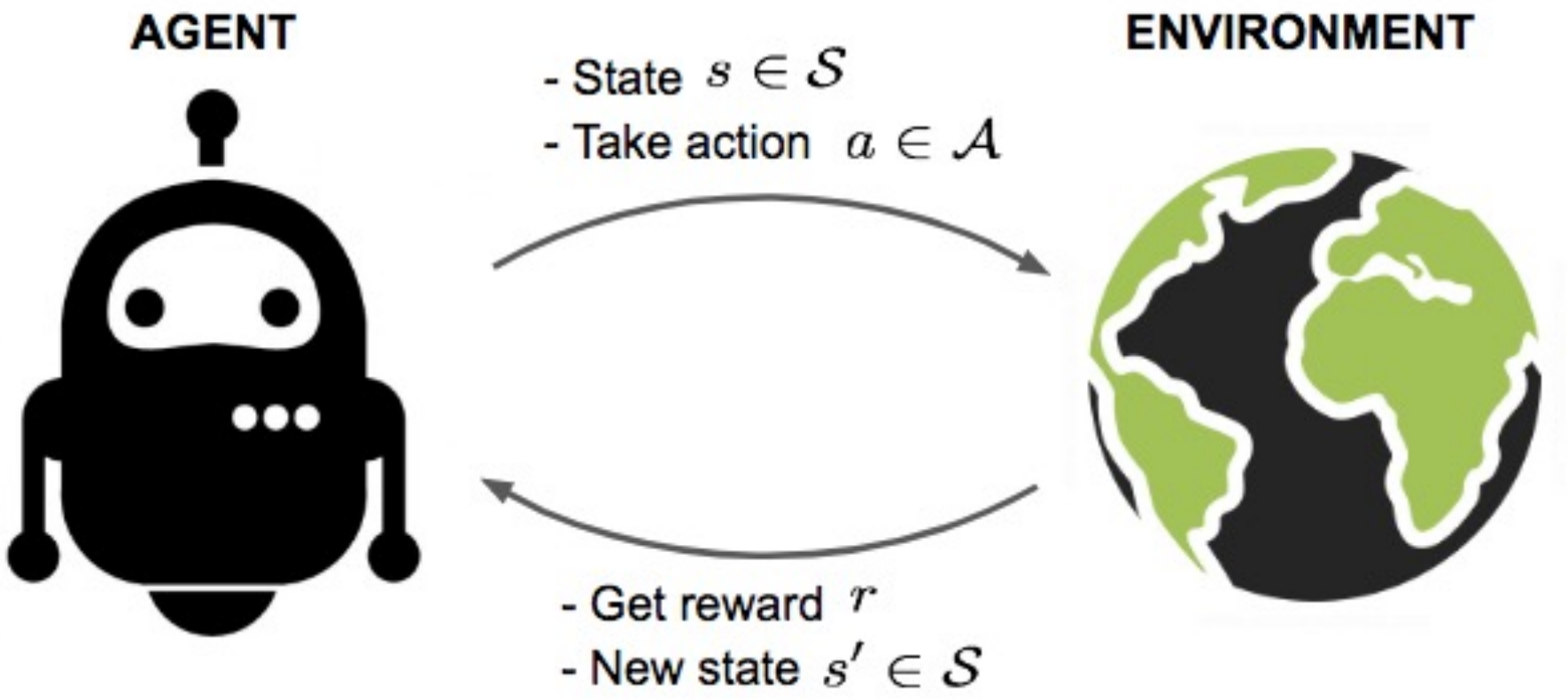
John Tan Chong Min

Mehul Motani

Aim

- To seek biological / neuroscience and psychology inspirations for how humans learn fast, and model it into AI
- Do not need to replicate faithfully the detailed mechanisms behind human thinking
- Just need to take broad concepts and apply it

Traditional Reinforcement Learning



How do humans think?

- Imagine if you were asked to choose one of two doors to walk to your office from your office lobby
- Which door would you choose?
- Do you calculate value functions for each door? Or based on memory?

Modelling the world

Insight: Next state prediction is slow

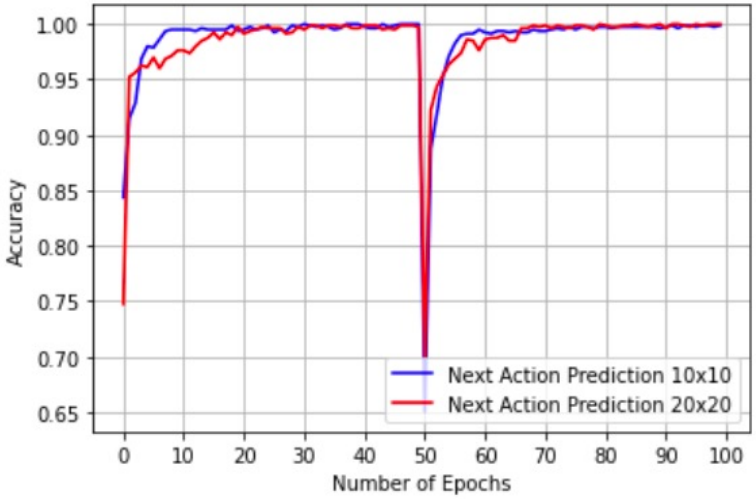


Figure 2: Accuracy of next action prediction for 10x10 grid (blue) and 20x20 grid (red) using 1000 samples

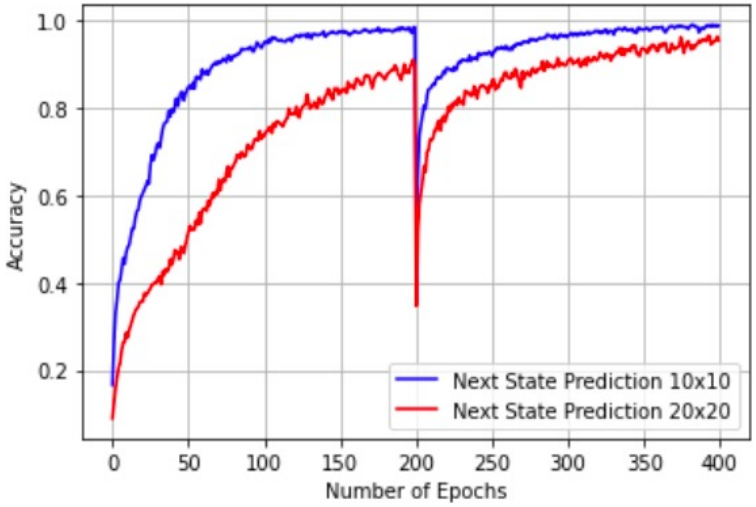
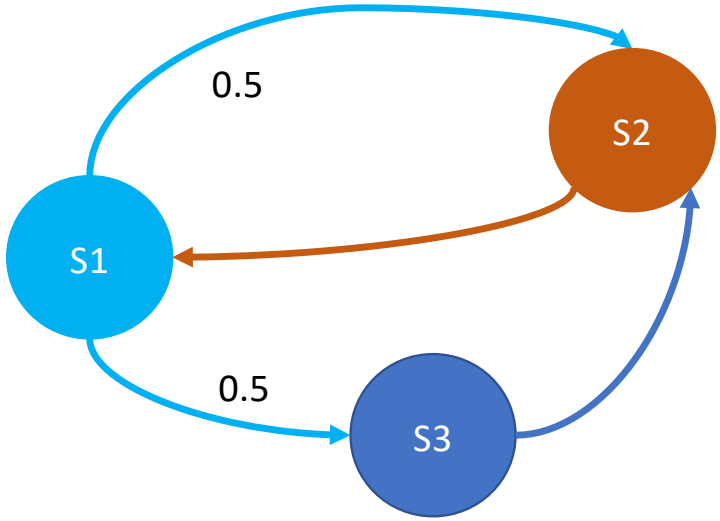


Figure 3: Accuracy of next state prediction for 10x10 grid (blue) and 20x20 grid (red) using 1000 samples

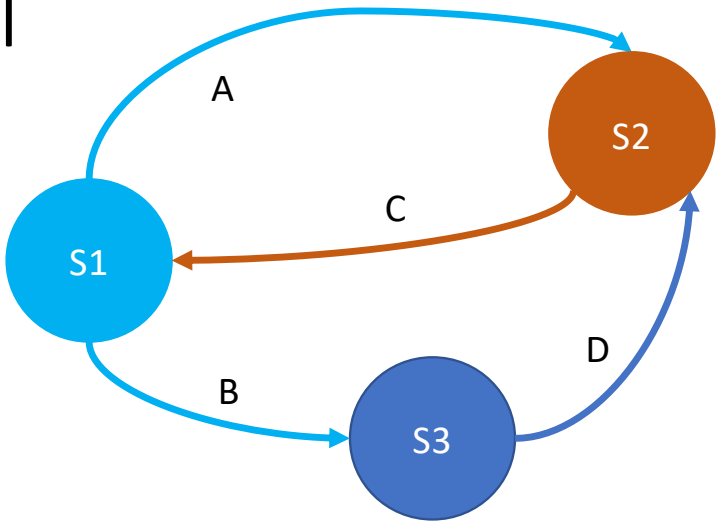
Should we model the Markov Decision Process of the world?

- Recent works have tried to utilize world models
  - MuZero
  - Dreamer v2/v3
- To get probability of transition, we would need many samples to learn it accurately
- Difficult to define state transition probability when number of possible states is unbounded



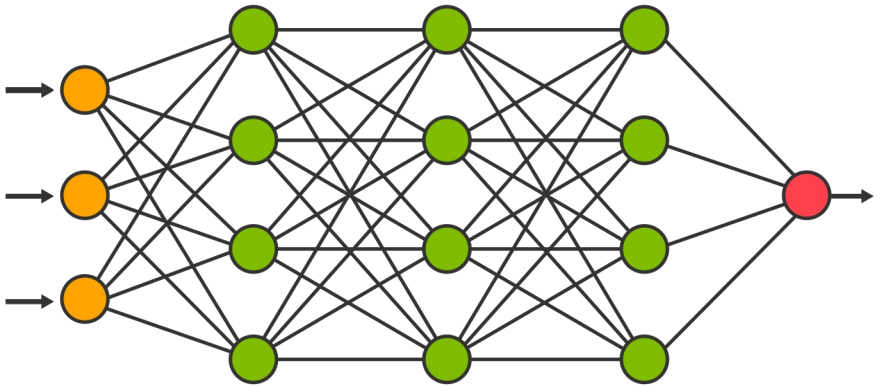
Using memory as world model

- **No need to model Markov Decision Process fully** – impractical to model environments with unbounded state and action spaces
- **No need to model probability of transition** – just need to see how often the next state is stored based on memory
- Just need to **remember experienced transitions** and then **retrieve it** the next time we encounter a similar state



Key (State)	Value 1 (Next State)	Value 2 (Action)
1	2	A
1	3	B
2	1	C
3	2	D

Neural Networks vs Memory



- Fast inference (one pass)
- Slow to learn (requires gradient descent)
- Can interpolate well
- Unreliable storage (weight update can change previously stored memory)

- Slow inference unless using approximate techniques
- Can be quick to learn from experience (instant memory update/deletion)
- Need the right abstraction space to generalize
- Reliable storage (previously stored memory will not be changed)

Exploring the World

Insight: Value-based reinforcement learning is slow

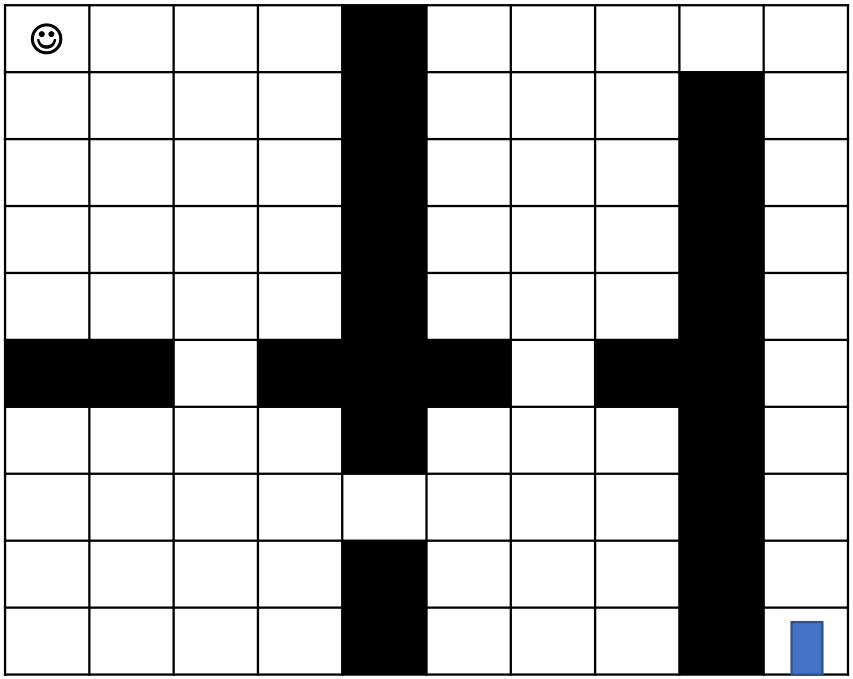
- Typically updated by one-step Bellman update (Temporal Difference Error)
- Takes as many updates as the path length to update the states with the final value
- Need to learn a different value function each time the goal changes

$$V(s) \leftarrow V(s) + \alpha(\overbrace{r + \gamma V(s')}^{\text{The TD target}} - V(s))$$

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{current value}}\Big)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Walled Maze



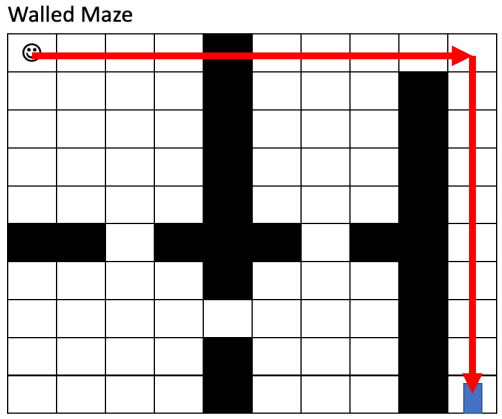
Legend:

☺ Agent

Door

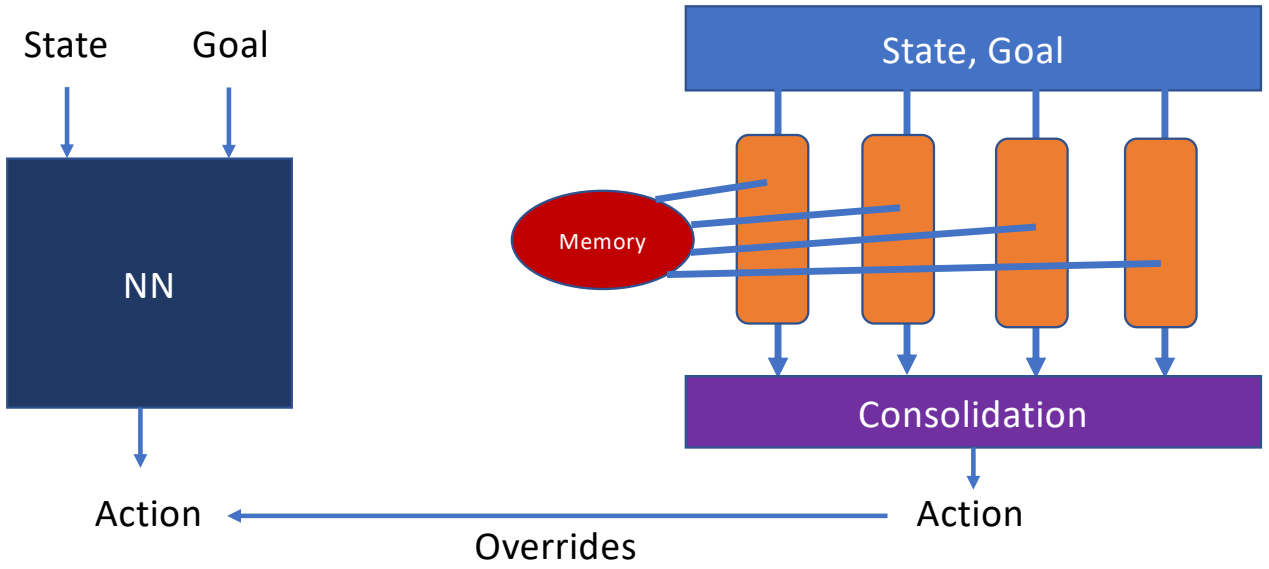
Goal-Directed Exploration

- Instead of value-based exploration, we use a goal-directed exploration
  - Humans also explore for needs like food/shelter
- Initial inference can be approximate, just needs to head towards general direction of goal
  - Prevent going in cycles by using count-based methods
- Hard to generate heuristic for inference for various scenarios
  - With hippocampal replay to train the goal-directed network, **no need for heuristic**!!



Fast & Slow

Two Networks – Fast and Slow



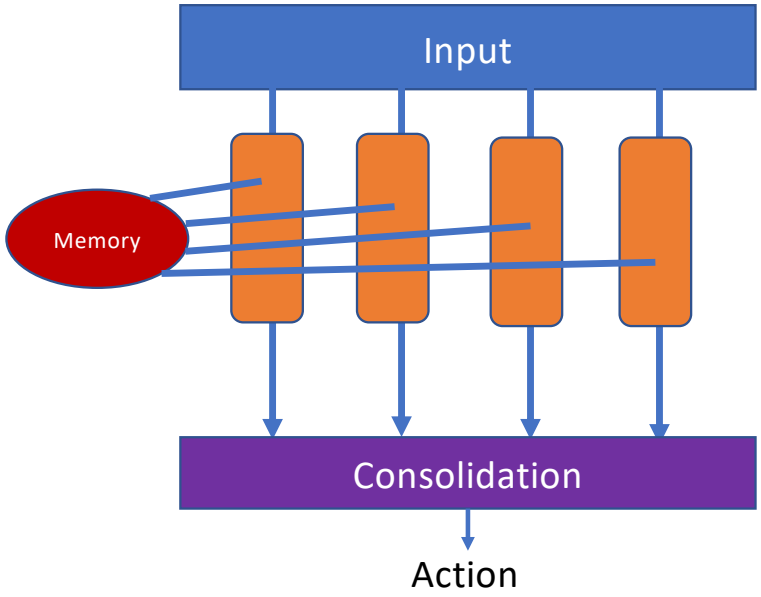
Neural Networks: Fast retrieval, slow learning

Memory: Slow retrieval, fast learning

Two Networks – Fast and Slow

- **Fast (System 1):** Neural network predicts best initial action given a goal

- **Slow (System 2):** Memory retrieval network to form possible trajectories to goal state



Fast Goal-Directed Neural Network

- Fast Neural Network Update

(at each time step)

  - **Previous states replay**
  - **Future states replay (only if trajectory found**

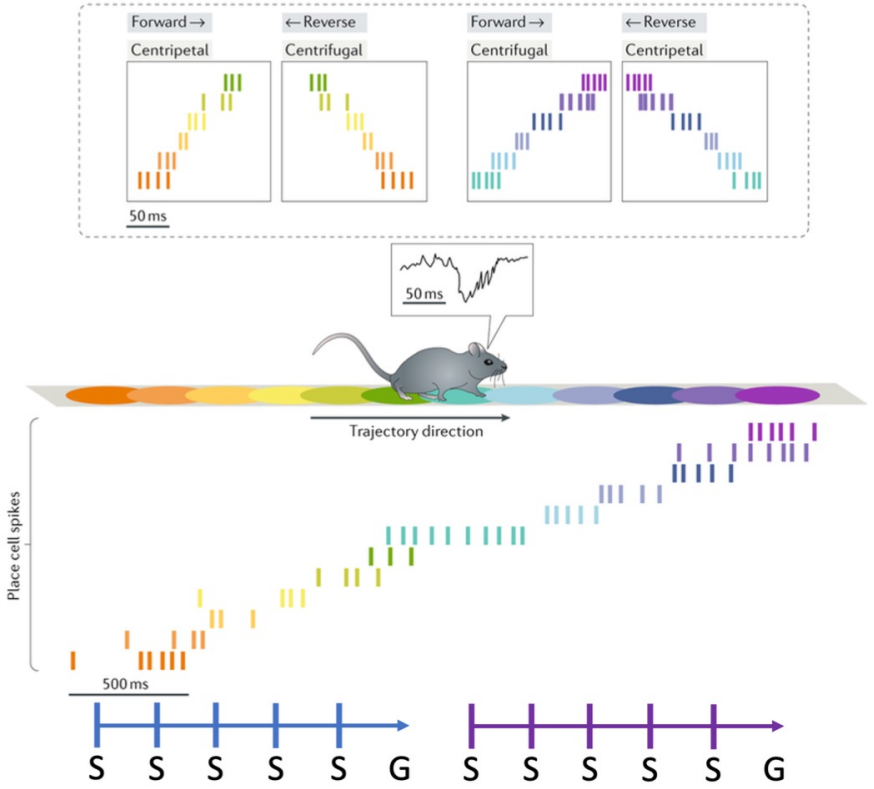
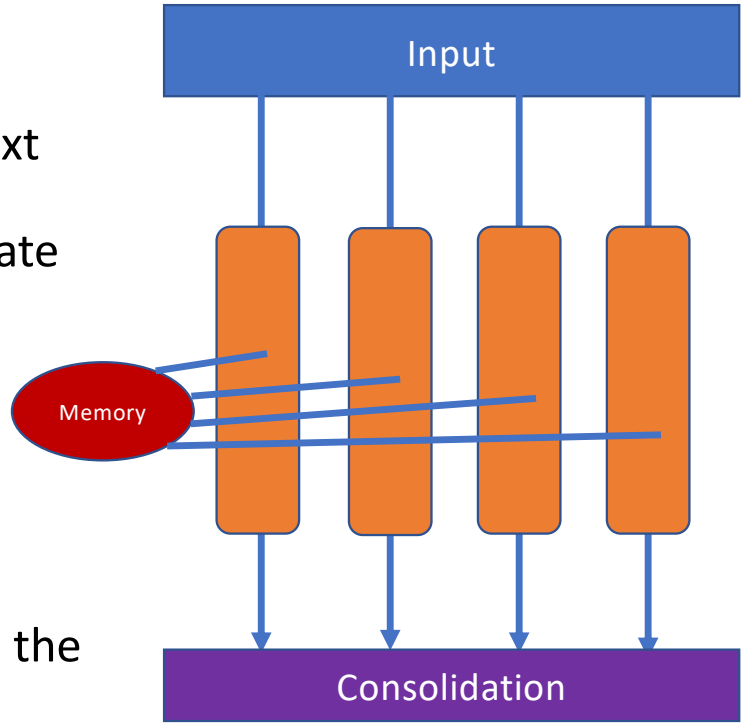


Figure extracted from Joo, H. R., & Frank, L. M. (2018). The hippocampal sharp wave-ripple in memory retrieval for immediate use and consolidation. Nature reviews. Neuroscience, 19(12), 744–757. https://doi.org/10.1038/s41583-018-0077-1

Slow Memory Retrieval Network

- Uses parallel processing with *B* branches
- Each parallel branch is like a minicolumn in the neocortex
- Takes the starting state and then reference memory for next state
- If more than one match, randomly pick one for the next state
- If next state is goal state, break
- Continue with next state as the key to reference memory
- Repeat until *D* lookahead timesteps

- All parallel branches will come back with a response
- See which branch has shortest trajectory to goal state, use the first action



Overall Procedure using Memory

- State and Action Prediction
  - Agent has a goal state in mind, and knows its current state
  - **System 1:** Agent queries the fast neural network to get action probabilities for the goal (exploit)
  - Get state-action visit counts via retrieval from episodic memory and choose action in explore-exploit way

    $$a^* = \arg\max_a (p(a) - \alpha\sqrt{numvisits(a)})$$

  - **System 2:** Agent uses the slow memory retrieval procedure to find out if there is any match in goal state in multiple lookahead simulations. If there is a match, overwrite the action from the explore-exploit mechanism

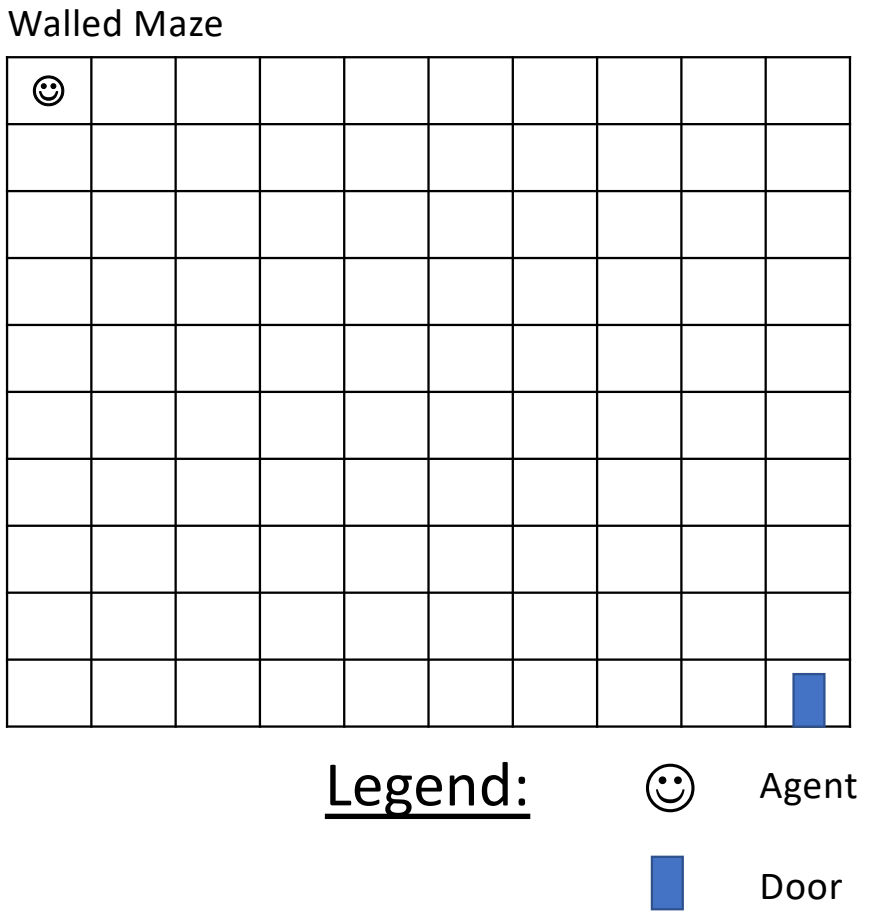
- Memory Update
  - Update the memory retrieval network with the new transition
  - Remove all memories that conflict with the current transition (if deterministic)
  - Perform hippocampal replay to update fast neural network

Experiments

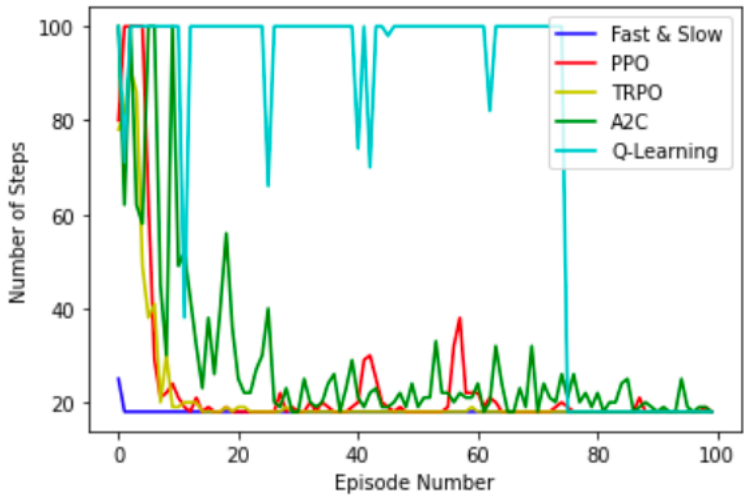
Navigating in a grid maze with obstacles

Static Environment

- **Setup:** Start State State to be (0, 0) and End State to be (*n-1*, *n-1*) in a *n* x *n* grid
- **Actions:** Up, Down, Left, Right
- **Reward:** 1 only if agent reaches the door within *n* x *n* steps, otherwise 0
- **Partial Observability:** Agent cannot see the entire grid, only knows its position and goal position



Results (Static)



Zoomed In

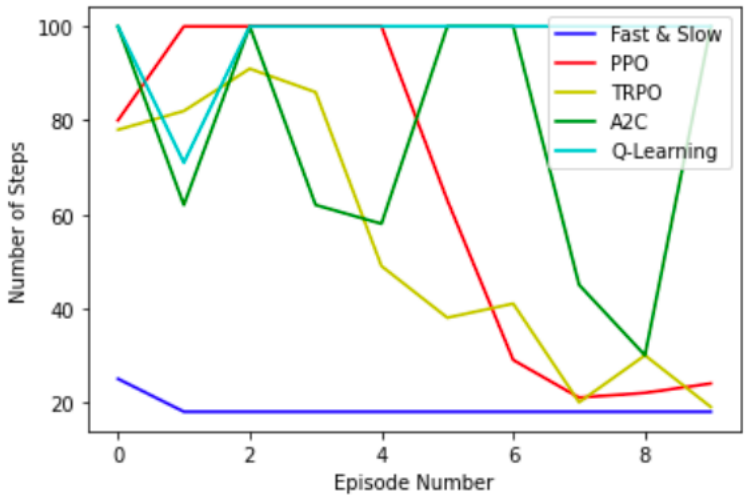


Figure 7: Steps per episode of the agents on a static 10x10 navigation task across 100 episodes

Figure 8: Steps per episode of the agents on a static 10x10 navigation task across first 10 episodes

Dynamic Environment

- Set Start and End State to be changing each episode
- Include obstacles which block certain squares
- Obstacle locations change every 50 episodes

Dynamic Environment

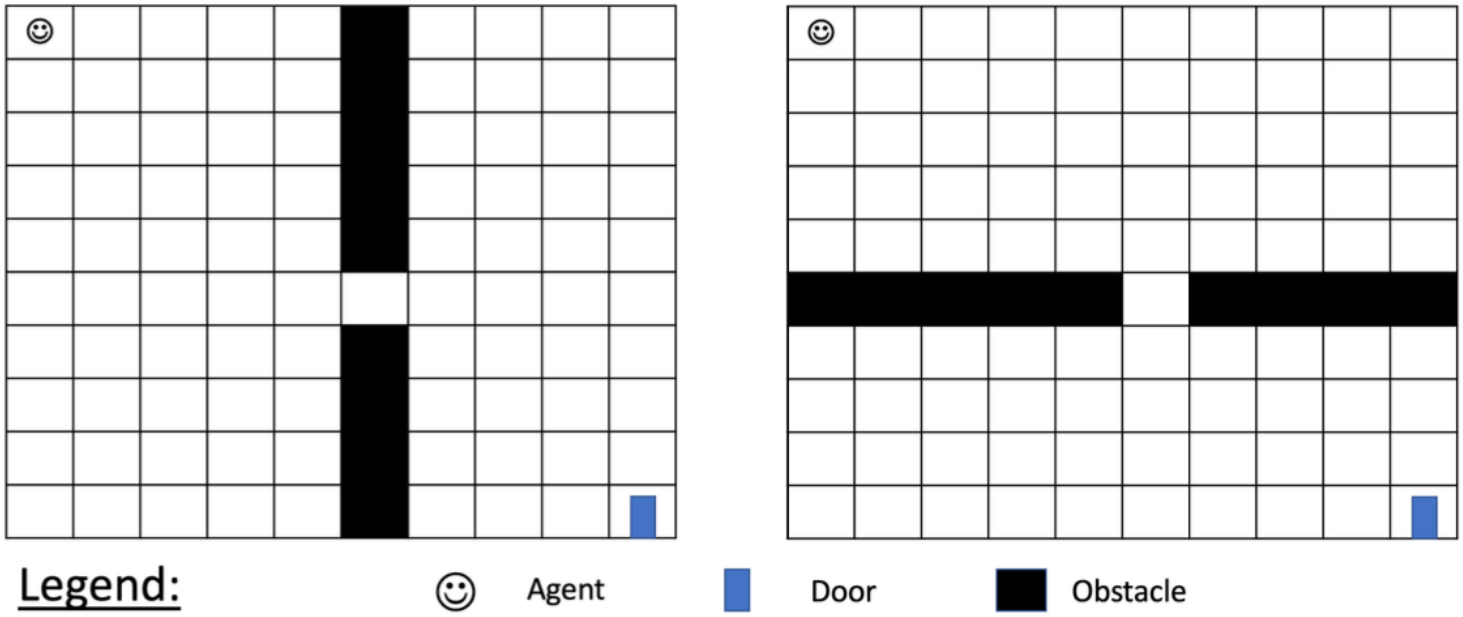


Figure 6: A sample maze environment of size 10x10. By default, the agent's start state is at the top left and the door is at the bottom right, but it can be varied. **(Left)** Obstacles before episode 50 form a vertical wall with a gap in the center across the mid-point. **(Right)** Obstacles after episode 50 from a horizontal wall with a gap in the center across the mid-point.

Main Results

Table 3: Adaptability of methods evaluated by solve rate on a dynamic 10x10 navigation task. Higher is better (in bold).

Agent	**Solve Rate(%)**		
	First 50 episodes	Last 50 episodes	Total
Fast & Slow	**88**	**96**	**92**
PPO	50	58	54
TRPO	56	44	50
A2C	20	28	24

Table 4: Efficiency of methods evaluated by steps above minimum on a dynamic 10x10 navigation task. Lower is better (in bold).

Agent	**Steps Above Minimum**		
	First 50 episodes	Last 50 episodes	Total
Fast & Slow	**923**	**555**	**1478**
PPO	2872	2336	5208
TRPO	2669	3001	5670
A2C	4032	3774	7806

Results (Ablation)

Table 5: Ablation study on adaptability of Fast & Slow agent evaluated by the solve rate of the agents on a dynamic 10x10 navigation task. Higher is better (in bold).

Agent	**Solve Rate(%)**		
	First 50 episodes	Last 50 episodes	Total
Baseline	88	96	92
No Slow	70	72	71
No Fast	52	50	51
No Fast, Slow	26	18	22
5 lookahead depth	82	96	89
10 lookahead depth	84	94	89
50 lookahead depth	88	**98**	**93**
10 parallel threads	88	96	92
50 parallel threads	84	**98**	91
200 parallel threads	**90**	96	**93**

Results (Ablation)

Table 6: Ablation study on efficiency of Fast & Slow agent evaluated by the steps above minimum of the agents on a dynamic 10x10 navigation task. Lower is better (in bold).

Agent	**Steps Above Minimum**		
	First 50 episodes	Last 50 episodes	Total
Baseline	923	555	1478
No Slow	2493	2677	5170
No Fast	2432	2678	5110
No Fast, Slow	3894	4146	8040
5 lookahead depth	1125	681	1806
10 lookahead depth	1100	905	2005
50 lookahead depth	898	488	1386
10 parallel threads	1297	930	2227
50 parallel threads	1080	528	1608
200 parallel threads	**791**	**445**	**1236**

Conclusion

- Memory mechanism is very robust and can enable agent to learn fast even when environment changes
- Increasing lookahead depth and parallel branches improve memory's performance
- Count-based mechanism alone is not sufficient to learn well; there is a benefit to using memory and goal-directed network for learning
- Value-based systems (e.g. Actor-Critic) are slow to converge and are slow to adapt in a changing environment

Extensions

Multi-Agent Learning
Arbitrary Goals and Subgoals
Forgetting as Learning
Hierarchical Planning

Extensions: Multi-Agent Learning

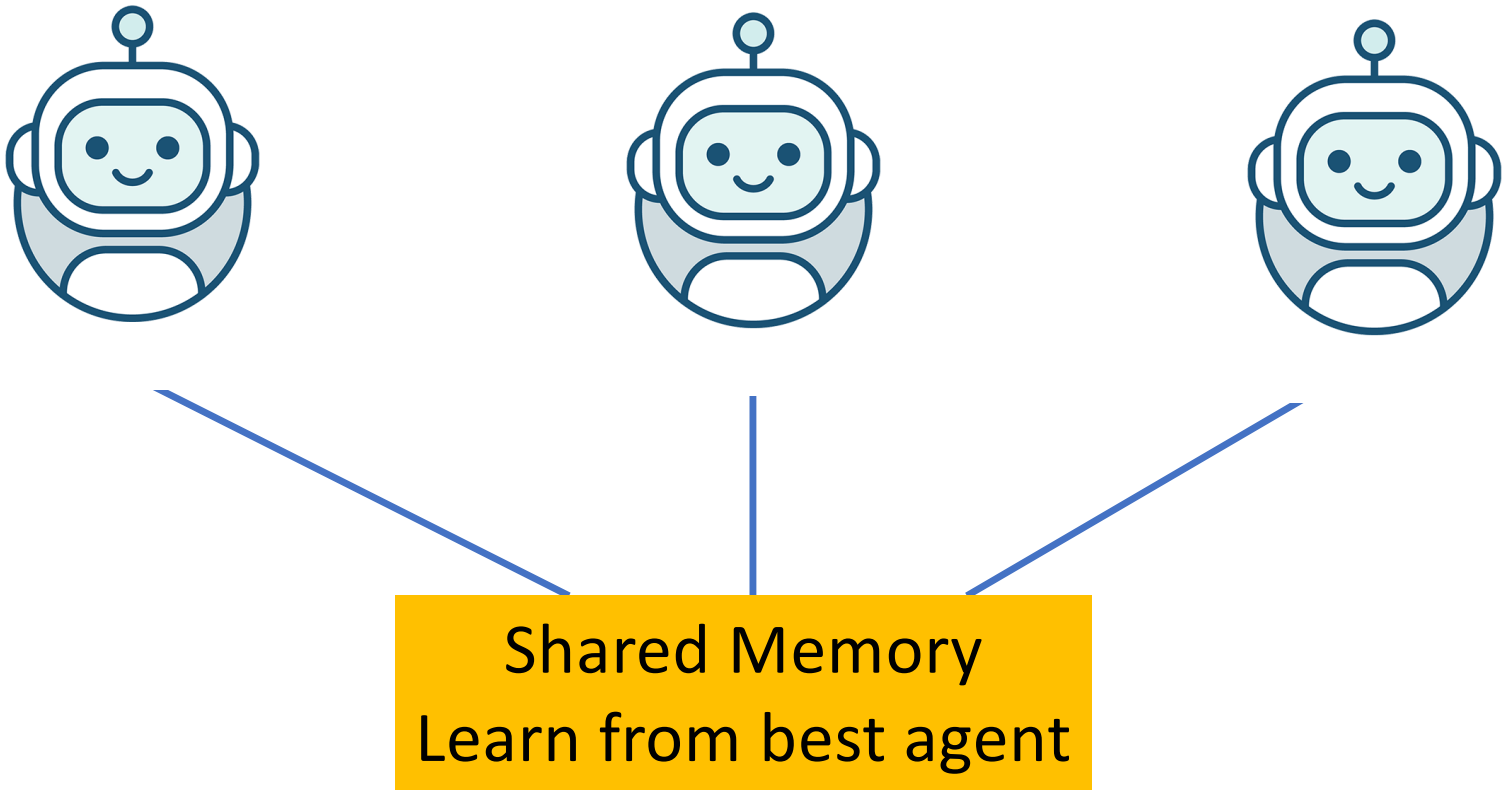
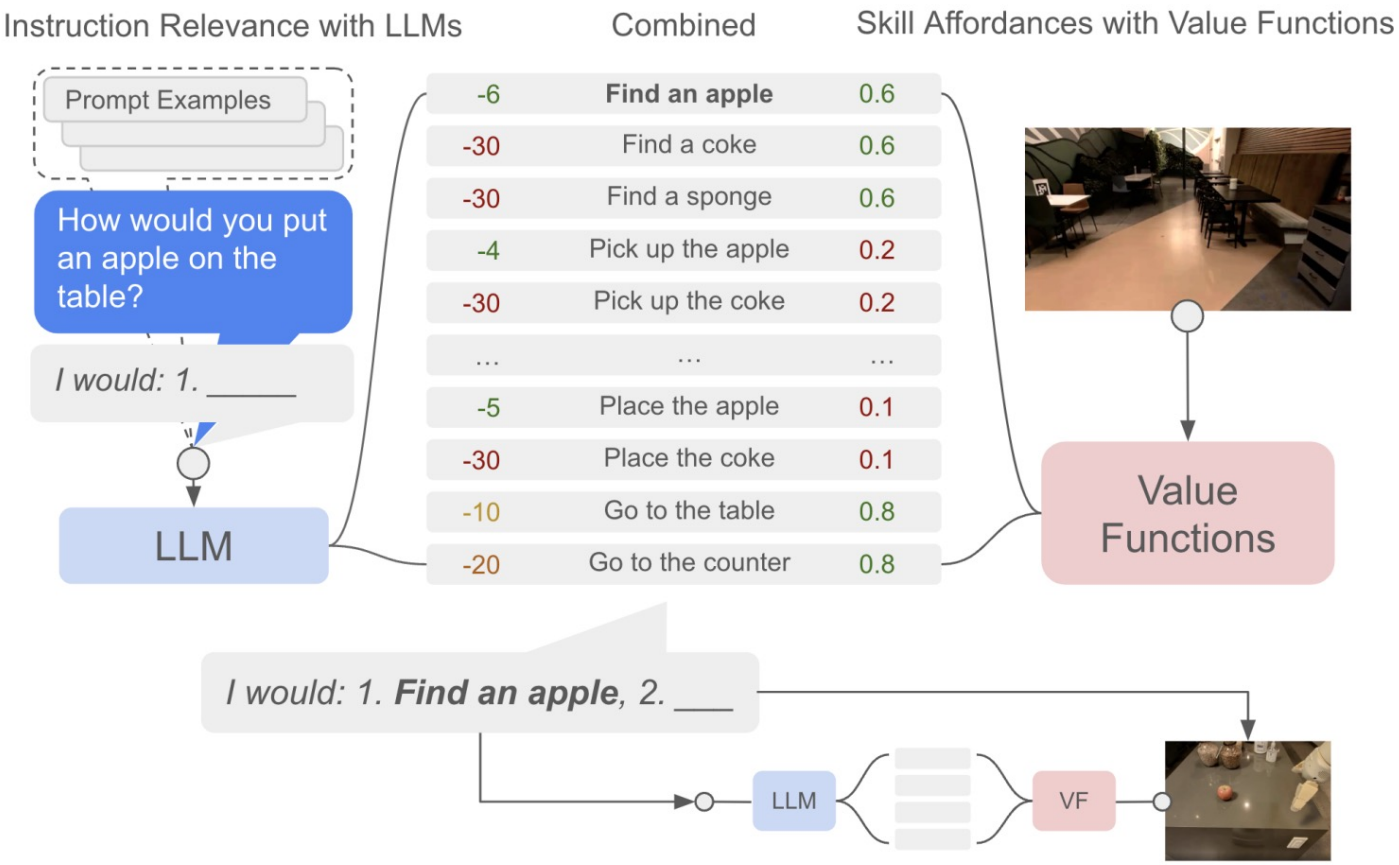


Image from: https://dev.to/akhilarjun/css-art-let-s-create-a-cute-robot-part-1-3ng5

Extensions: Arbitrary Goals and Subgoals



Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. Ahn et al. 2022

Extensions: Forgetting as Learning

- Memory retrieval helps to improve the strength of the memory
- Memories that are not accessed often are forgotten and make room for other memories to be stored
- Helps to make sure the memories we have are useful for the current environment
  - Less used memories are those not relevant and forgotten
  - Frequently used memories are those relevant and strengthened

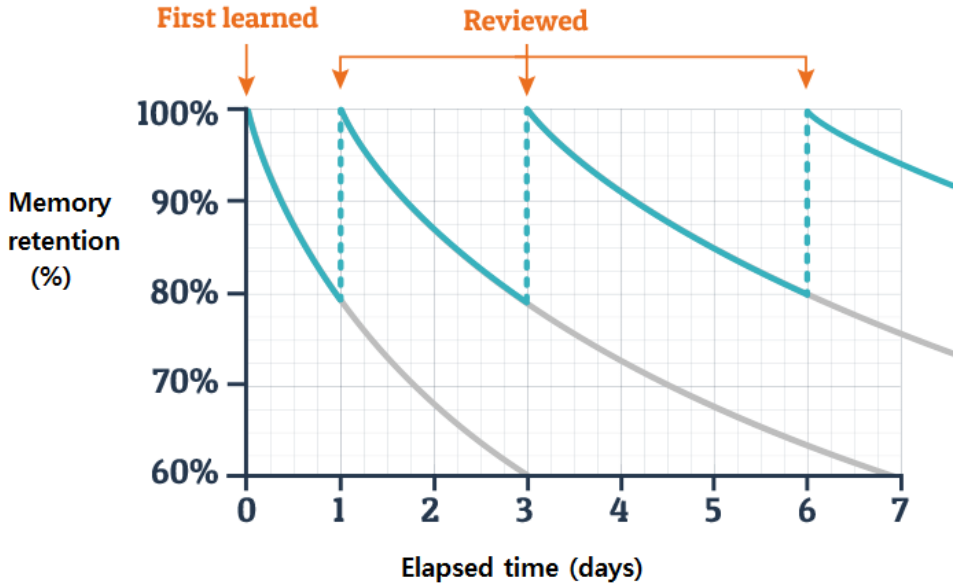
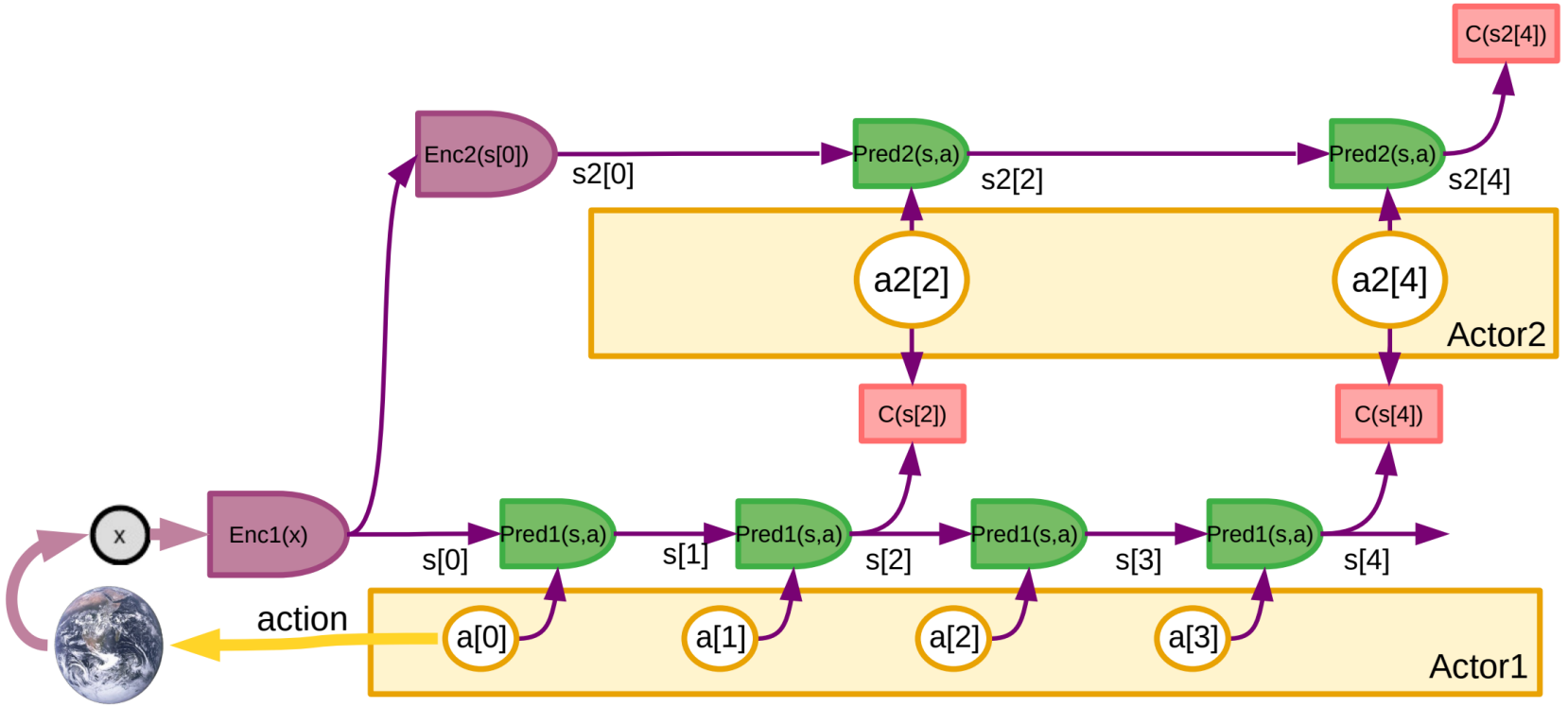


Chart from: Chun, Bo Ae & hae ja, Heo. (2018). The effect of flipped learning on academic performance as an innovative method for overcoming ebbinghaus' forgetting curve. 56-60. 10.1145/3178158.3178206.

Extensions: Hierarchical Planning



A Path Towards Autonomous Machine Intelligence. Yann LeCun. 2022.

Questions to Ponder

- How can we find sequence of trajectories with increasing state space?
  - Methods other than random search – goal distance function?
- How can we define arbitrary goals?
- How can we determine subgoals given the main goal we want to achieve?
- How can we abstract memory to generalize?