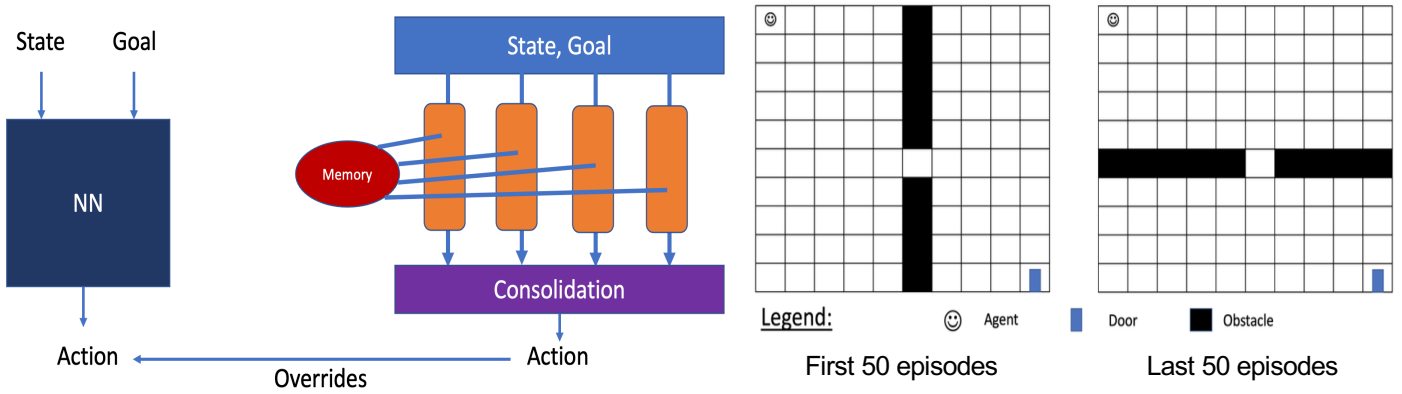
Learning, Fast and Slow (10 Years Plan)

John Tan Chong Min

Base Model (15 min recap)

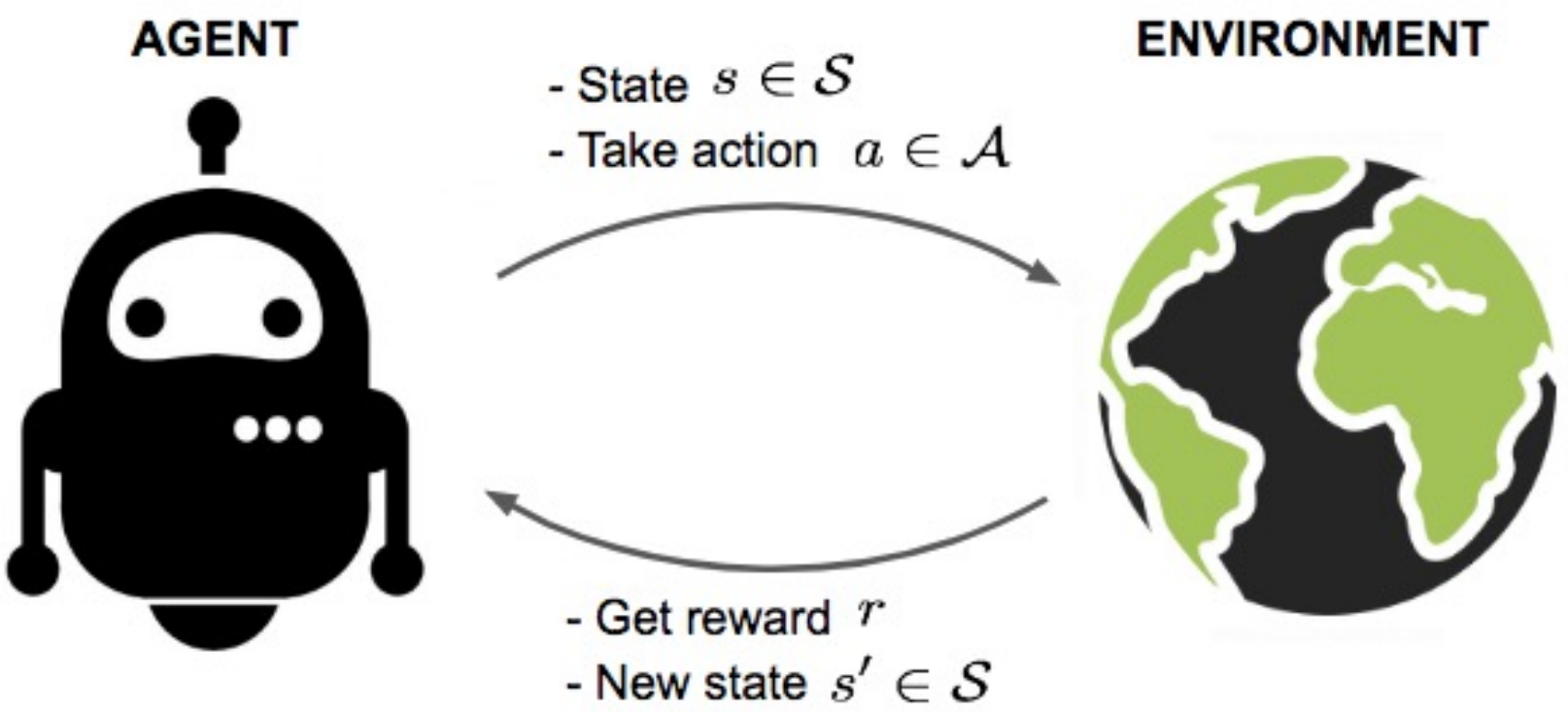
John Tan Chong Min and Mehul Motani
Dept of Electrical and Computer Engineering, National University of Singapore



- Uses a **Fast Neural-Network Based Goal-Directed Network** to select next action without lookahead (learned using self-supervised learning without rewards)
- Uses a **Slow Memory-Based Retrieval Network** to identify trajectories to goal state in parallel and repeat next successful action
- Achieves 91.9% solve rate in a dynamically changing grid world (better than PPO (61.2%), TRPO (26.1%), A2C (23.9%), DQN (4.9%))
- Code: https://github.com/tanchongmin/learning-fast-and-slow

Best Paper Finalist at:
IEEE International Conference on Development and Learning (ICDL) 2023

Traditional Reinforcement Learning



Insight: Value-based reinforcement learning is slow

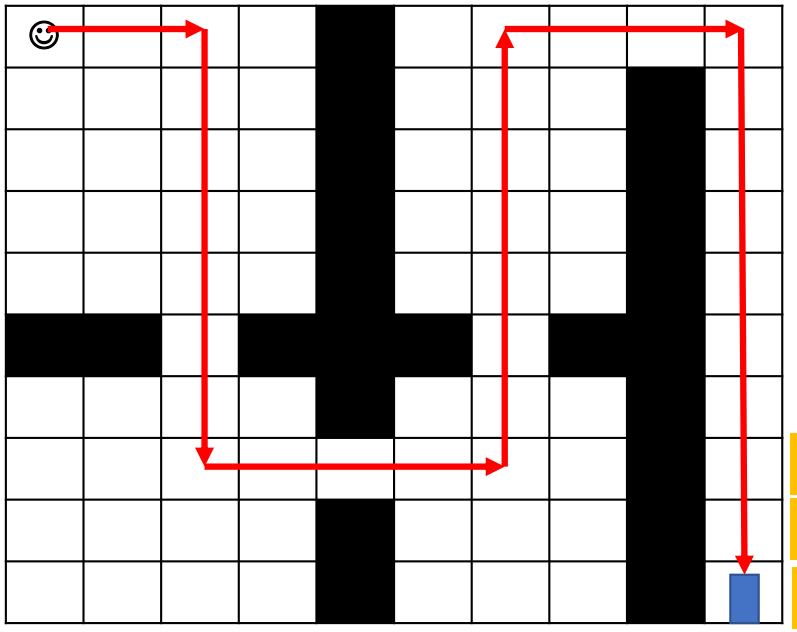
- Typically updated by one-step Bellman update (Temporal Difference Error)
- Takes **multiple updates to update the entire path** with the correct value
- Need to **learn a different value function** each time the goal changes

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

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

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

Walled Maze



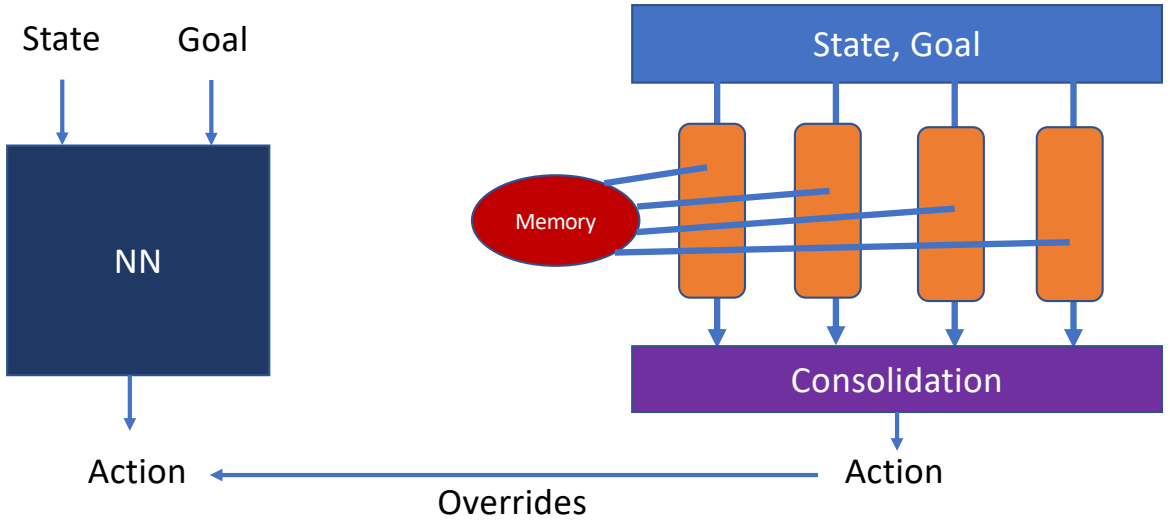
Legend:

Agent

Door

Two Networks – Fast and Slow

- Memory is important for fast adaptation before neural networks learn



Neural Networks: Fast retrieval, slow learning

Predicts best initial action given start state and goal

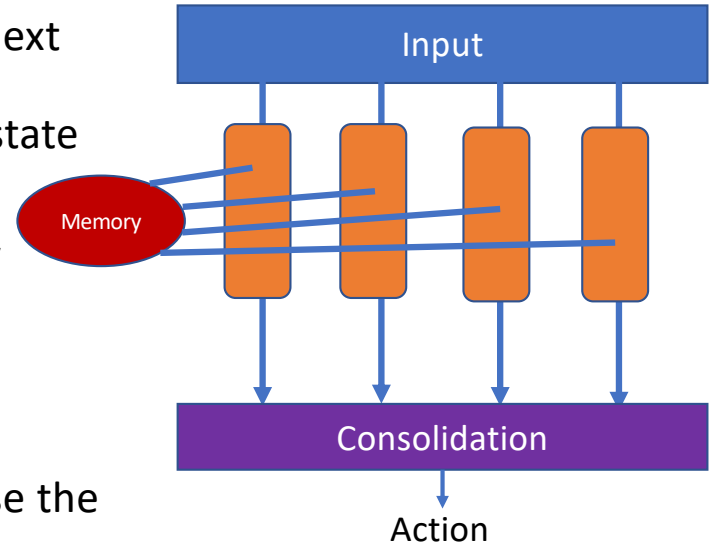
Memory: Slow retrieval, fast learning
(World Model planning as Memory Retrieval)

Lookahead multiple trajectories to goal state and choose best one

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



Goal-Directed Neural Network

- At each time step, learns from:
  - **Previous states replay**
  - **Future states replay (only if lookahead trajectory found)**

- Intuition:
  - If we have a trajectory A -> B -> C
  - We know A -> B, A -> C, B -> C
  - Maximise learning from experience/lookahead

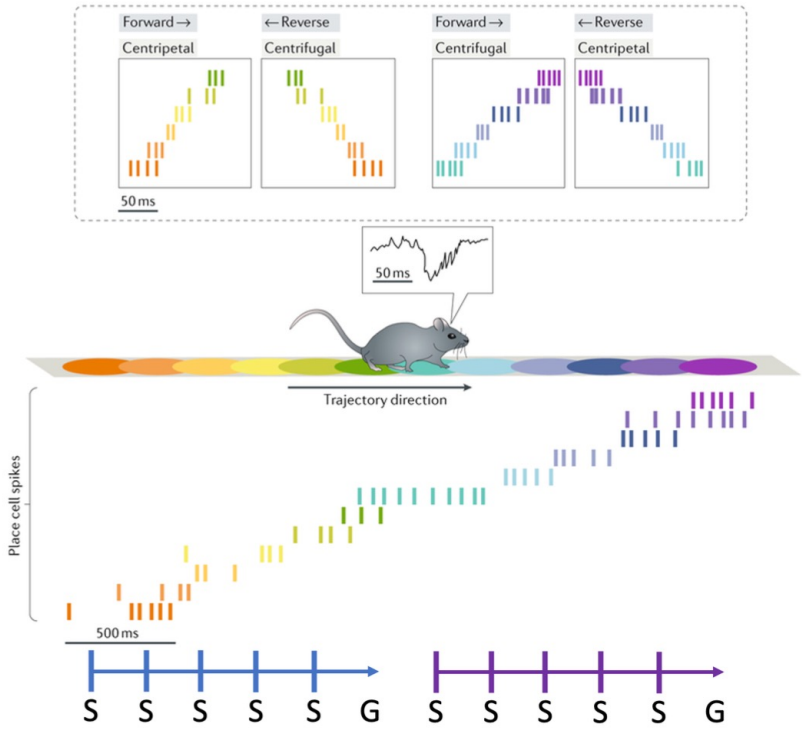


Figure extracted from Joo, H. R., & Frank, L. M. (2018). The hippocampal sharp wave-ripple in memory retrieval for immediate use and consolidation. Nature reviews. Neuroscience, 19(12), 744–757. https://doi.org/10.1038/s41583-018-0077-1

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, choose the shortest path and 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

Conclusion

- Traditional value-based systems (e.g. Actor-Critic) are **slow to converge** and are **slow to adapt** in a changing environment
  - Use goal-directed action-prediction learning (self-supervised learning)

- Memory mechanism is **very robust** and can enable agent to learn fast even when environment changes
  - Memory adapts faster than neural networks

- World model need **not use probabilities**, which can be intractable
  - Memory sampling and retrieval is sufficient
  - No need to model the whole world, only the locally observed parts that are relevant

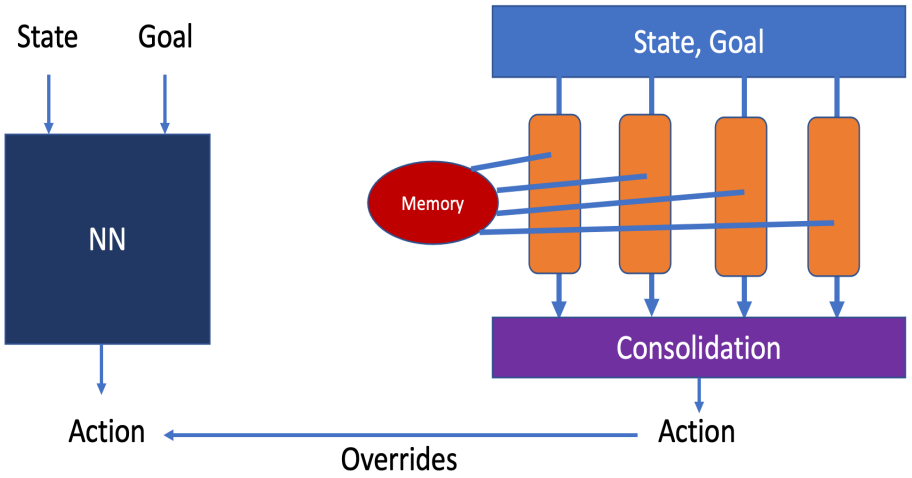
10 Year Plan

Multiple F&S Agents:

Memory Transfer between Agents
Multimodal Memory Search

Multiple F&S Modules:

Hierarchical / Parallel Prediction
Memory Soup for various Abstraction Spaces
Abstracted Goals by Proximity



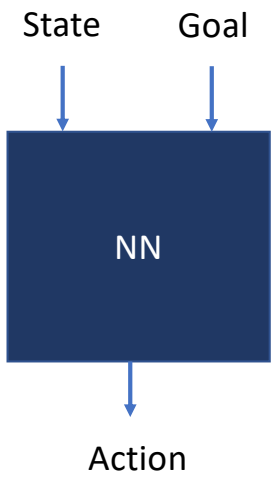
Goal-Directed Decision Transformer

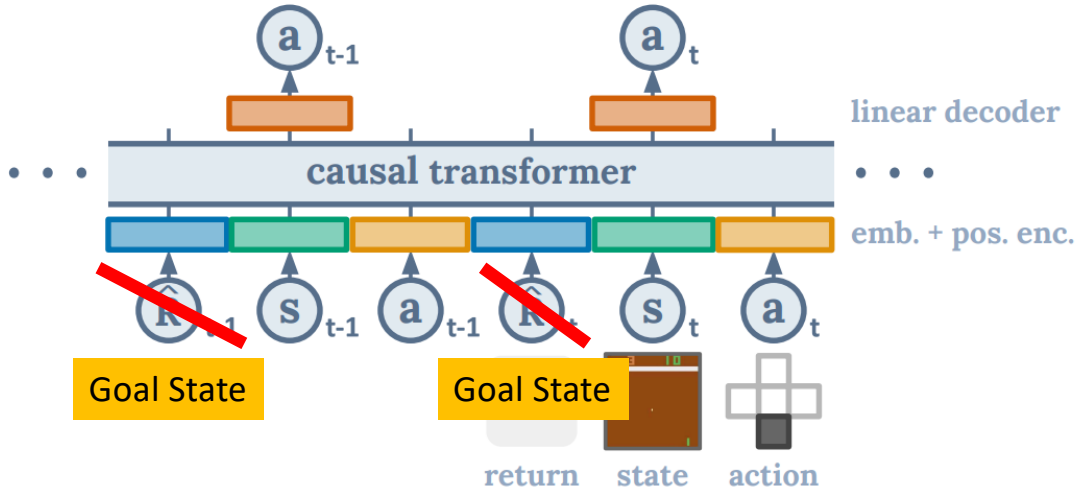
Emotions for Preference

Other memory types:
e.g. Semantic Memory
e.g. Knowledge Graphs

Goal-Directed Decision Transformer

- Enhance the Neural Network with an architecture that can take in sequences of past states and actions

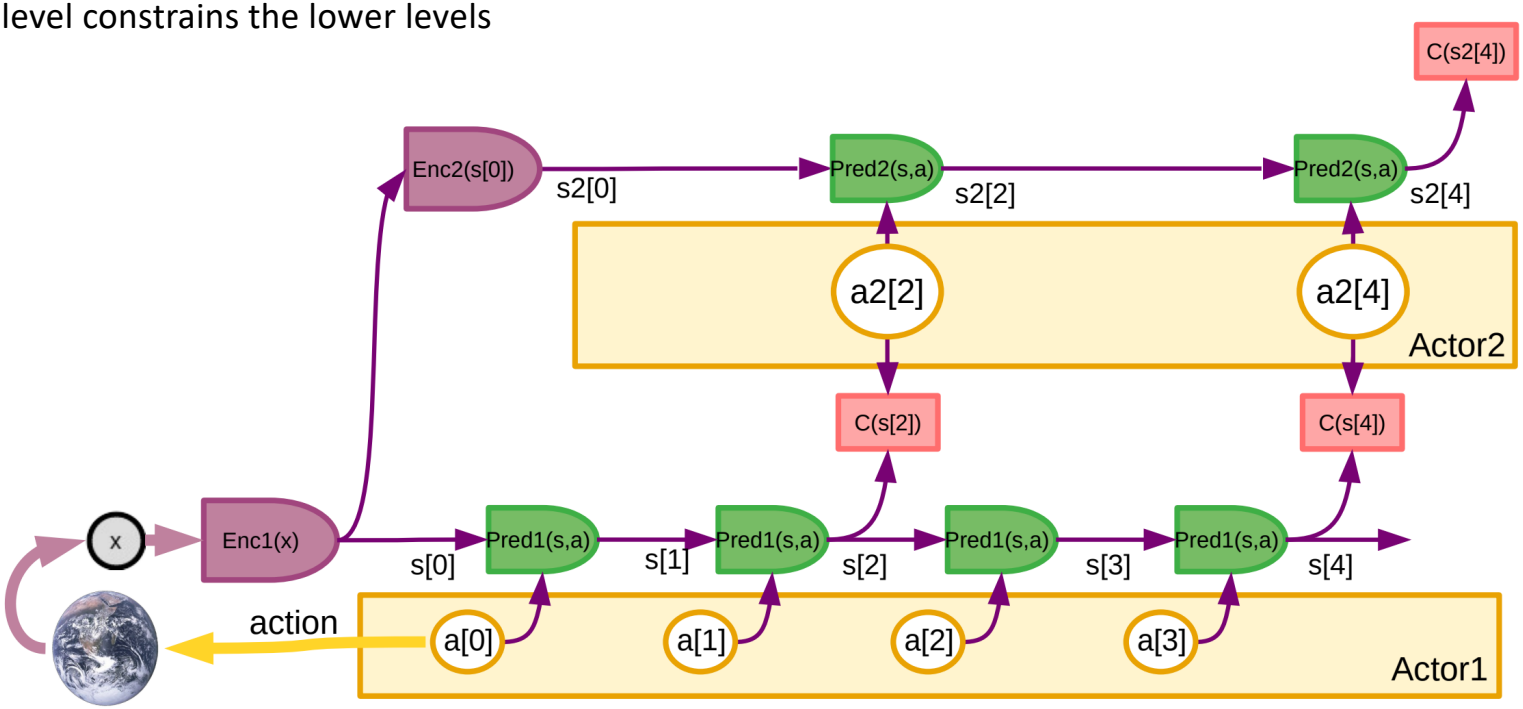




Decision Transformer: Reinforcement Learning via Sequence Modeling. Chen et al. 2021.

Hierarchical Prediction (Hierarchical JEPA)

- Sequential and hierarchical
- Top level constrains the lower levels

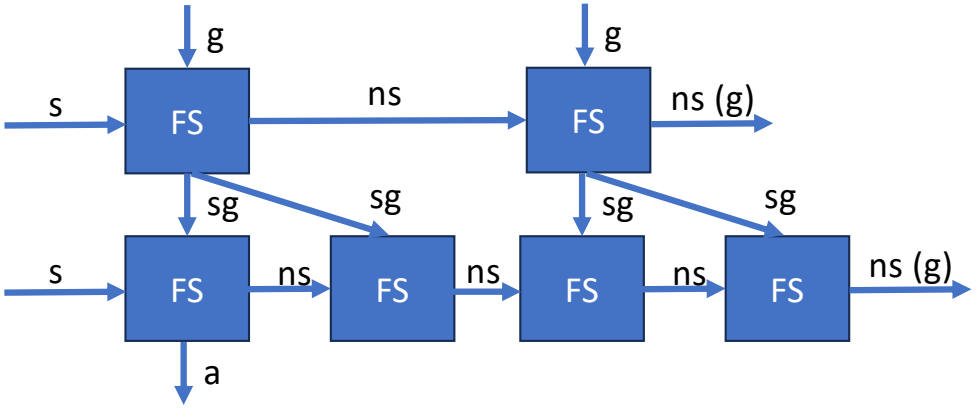


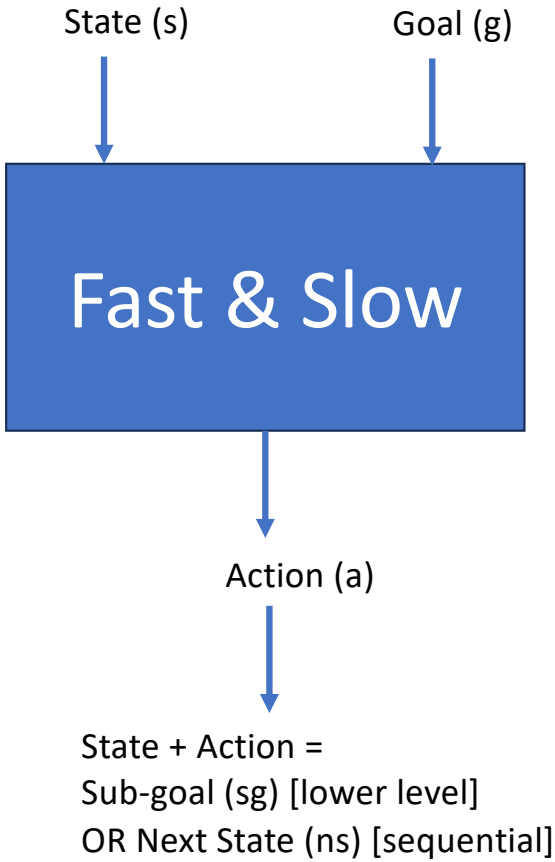
A Path Towards Autonomous Machine Intelligence. Yann LeCun. 2022.

Hierarchical Prediction

- Sequential and hierarchical
- Top level constrains the lower levels
- State + Action forms Goal for lower hierarchy

- Questions to ponder:
  - How to form the abstracted goals/states?
  - How to get abstracted memory?





Building Abstracted Goals/States by Proximity

- Goals/states at higher levels can be formed by removing some transitions
  - Multiple actions become aggregated at higher levels
- As we go down the hierarchy, distances between states are smaller
- Can goals be formed via Natural Language?

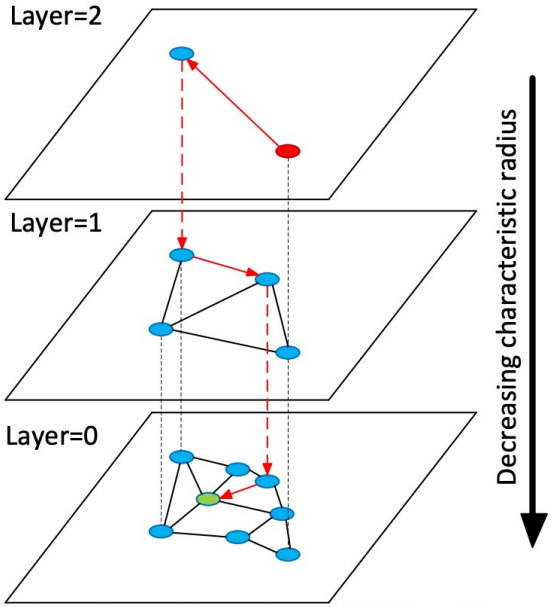
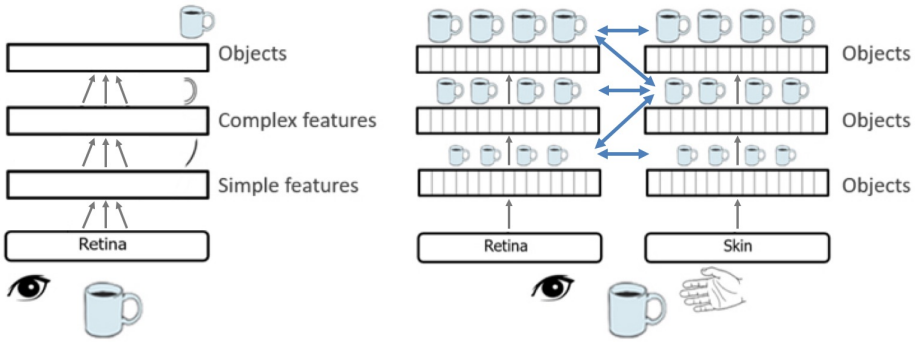


Fig. 1. Illustration of the Hierarchical NSW idea. The search starts from an element from the top layer (shown red). Red arrows show direction of the greedy algorithm from the entry point to the query (shown green).

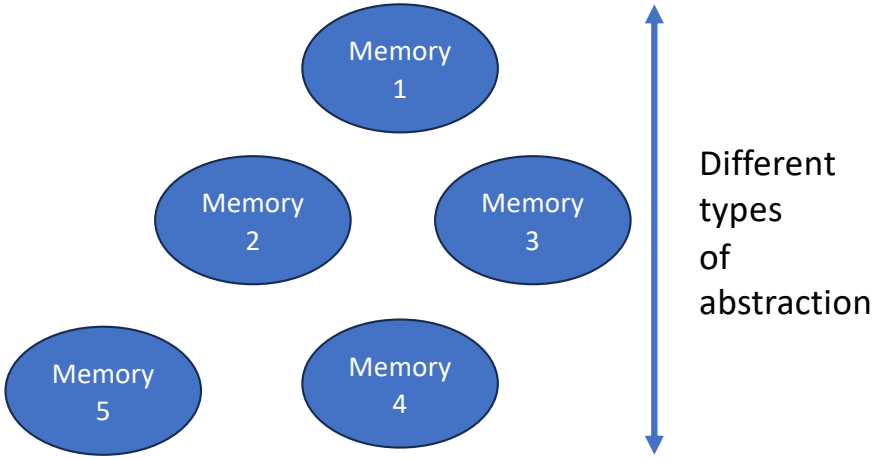
Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs
Malkov and Yashunin. 2016

Memory Soup

- All kinds of representations at various levels can be all put into one common **memory soup**
- Planning is done by pattern matching to the right level of abstraction in the **memory soup**



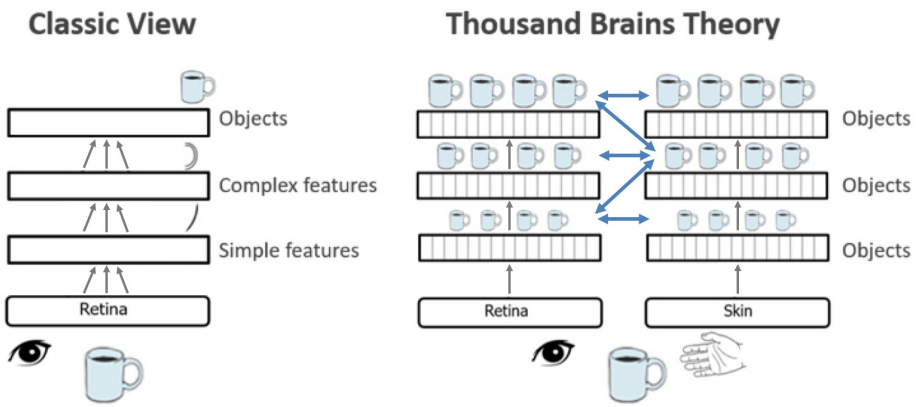
Thousand Brains Theory of Intelligence (Jeff Hawkins). Numenta.



"Memory Soup" Theory
(from John's AI Discord group)

Multiple Abstraction Spaces

- Multiple abstraction spaces based on varying inputs / varying structure
  - Self-supervised learning
- Semantic memory also possible
  - One such space could be knowledge graph

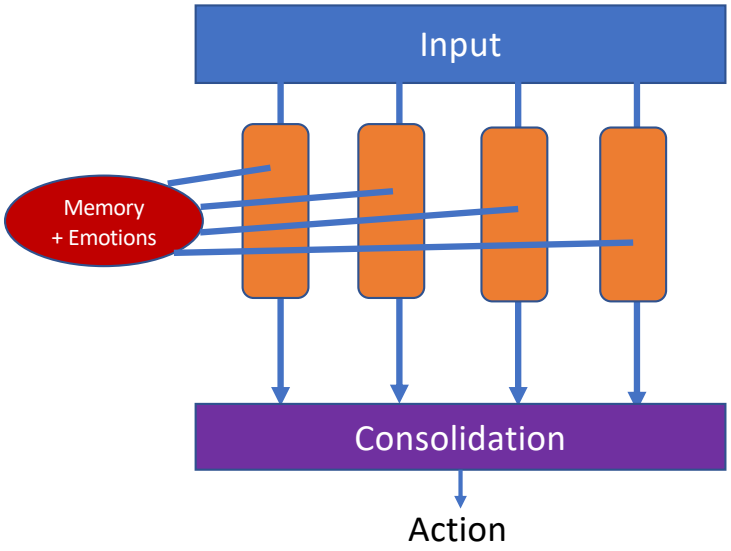


Thousand Brains Theory of Intelligence (Jeff Hawkins). Numenta.

Infuse memory with Emotions

- Emotions help to tell us which memories are important
- We don’t have to forget old memories, we simply need to imbue new memories with **higher preference score based on emotions**
- Retrieve memories by similarity first, then by preference score
- Emotions like fear, happiness might be highly preferred, while sadness highly avoided

- **Separate Fear and Desire Circuit:**
  - There is likely a ”desire” goal-directed network, and an “avoidance” network (fear) that constrains actions based on undesirability



Memory Transfer Between Agents

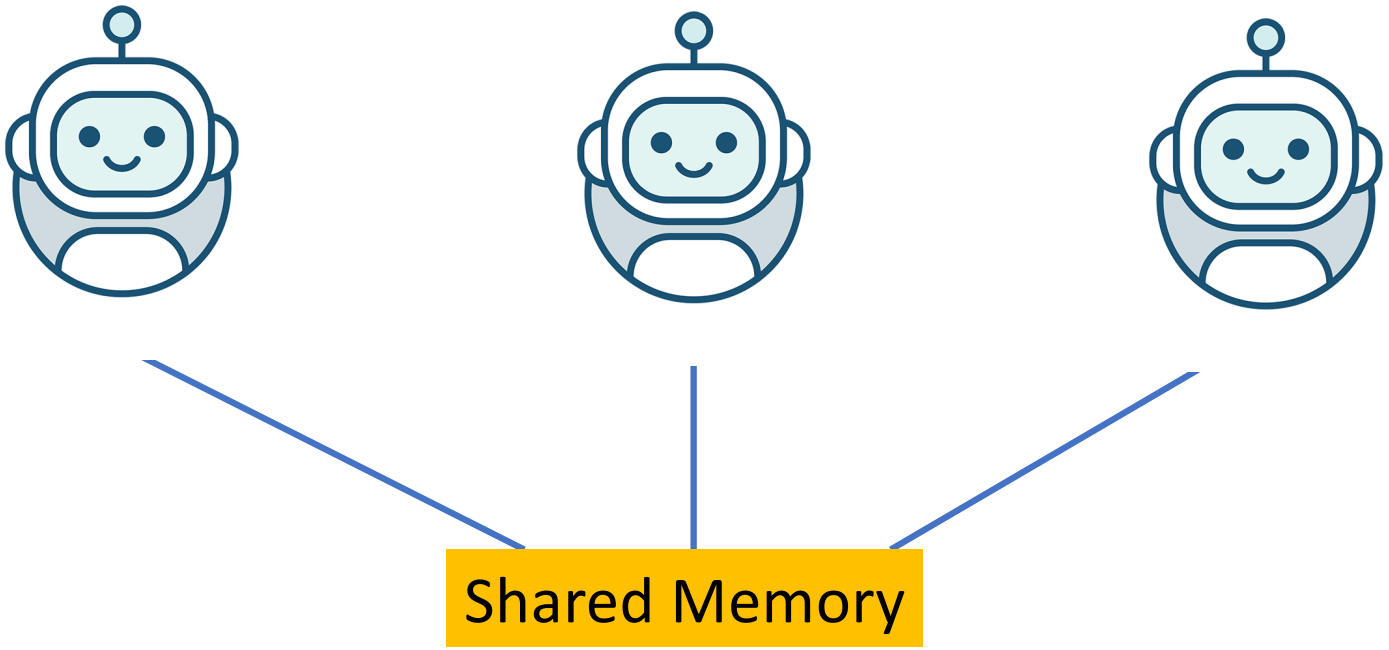
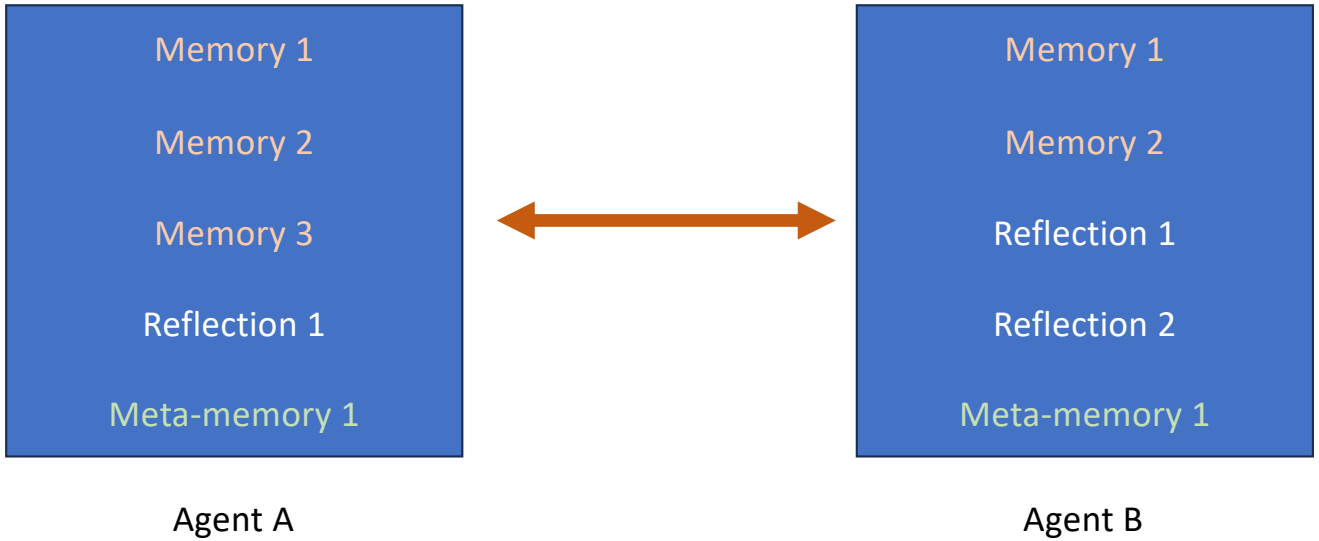


Image from: https://dev.to/akhilarjun/css-art-let-s-create-a-cute-robot-part-1-3ng5

Memory Transfer between Agents

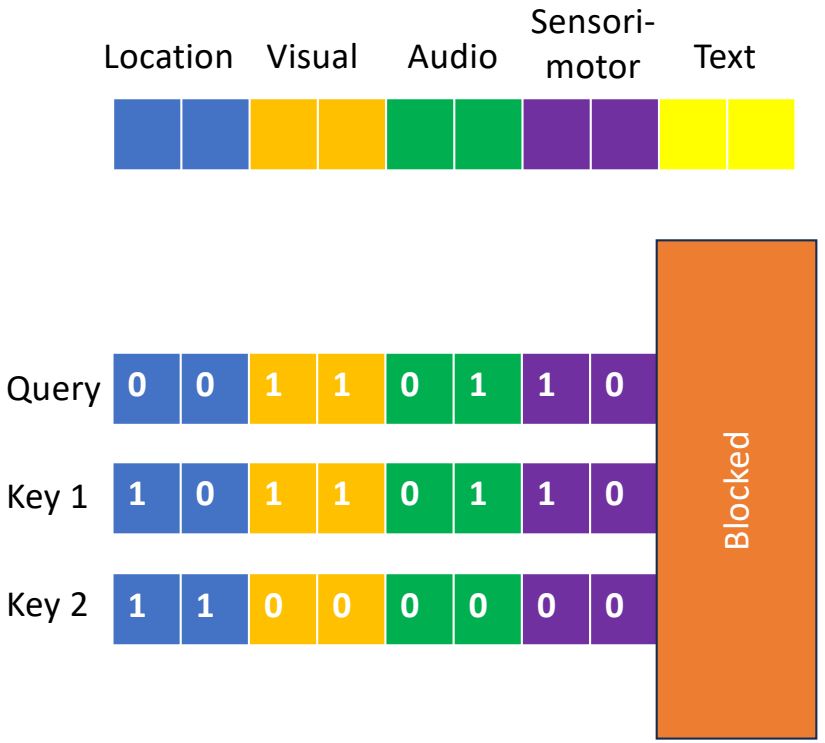
- How much memory to share?
  - Perhaps only the best agent should transfer memory to the rest

- Would sharing of too much information be bad for agents?
  - Perhaps we should only have memory transfer amongst groups of agents in same kind of environment
  - Local proximity memory transfer



Multimodal Memory Search – Incomplete Hash Search

- Search memory with the parts of the memory that are present in stimuli
- No need to search the entire embedding vector
- Can also search **subparts of vectors** mapping to one or more modality to do a “random forest”-like search



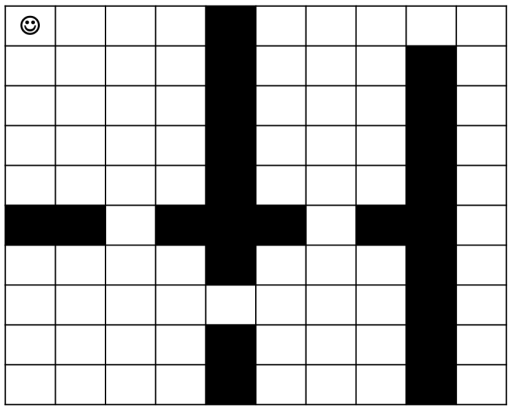
- Values can be continuous too, like vector embeddings

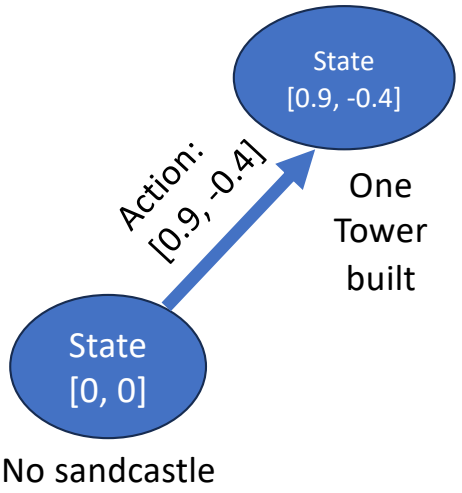
Can everything be modelled as a Travelling Salesman Problem?

- Intelligent tasks can be mapped into a search space with states in embedding space and distances based on a distance metric
- Action spaces can also be mapped into an embedding space
- Planning can be done by memory over new abstracted graph space

Arbitrary Task from **Start State** to **Goal State**

e.g. Build a Sandcastle





Discussion