Towards Fast and Adaptable Agents via Goal-Directed, Memory-Based Learning

PhD Thesis Overview

John Tan Chong Min

Supervisor: Mehul Motani

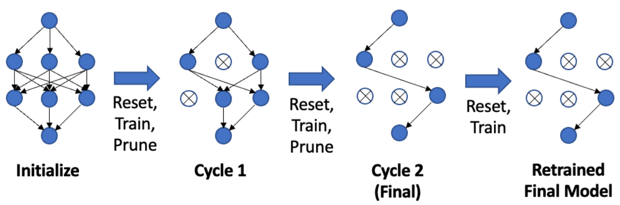
National University of Singapore

AI Singapore

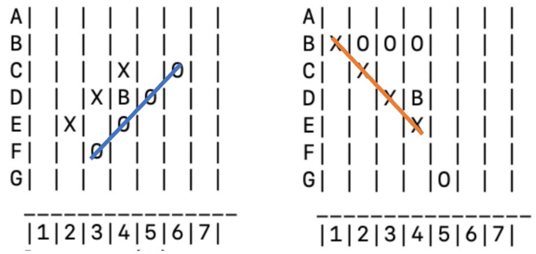
Overview

- Humans learn quickly and adapt fast
- Deep learning systems need **many examples** to learn, and **do not adapt fast** to changing environments
- How can we use **insights from human cognition** to build better AI models?

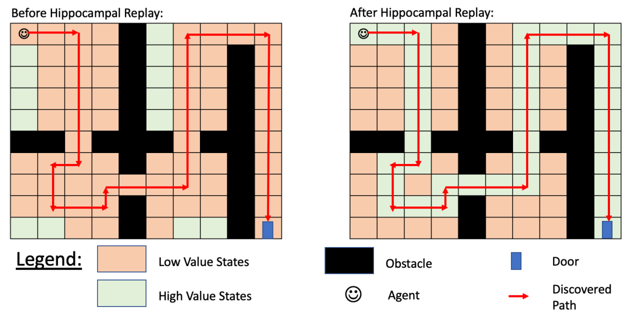
Overview: Moving from reward to goal-directed, memory-based learning



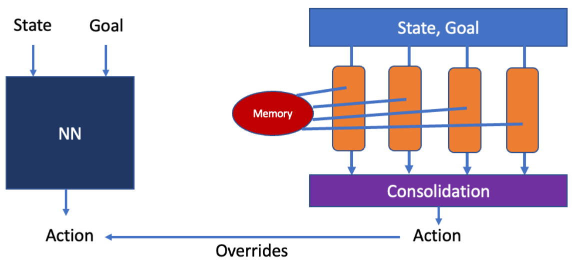
DropNet: Learning by pruning



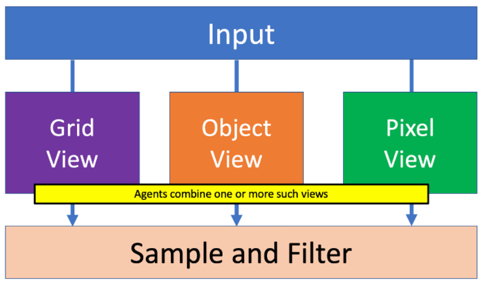
Brick Tic Tac Toe:
Reward is not enough



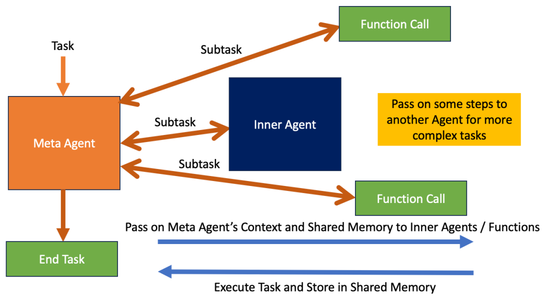
Hippocampal Replay:
Memory-based Learning



Learning, Fast & Slow:
Goal-Directed, Memory-Based Learning

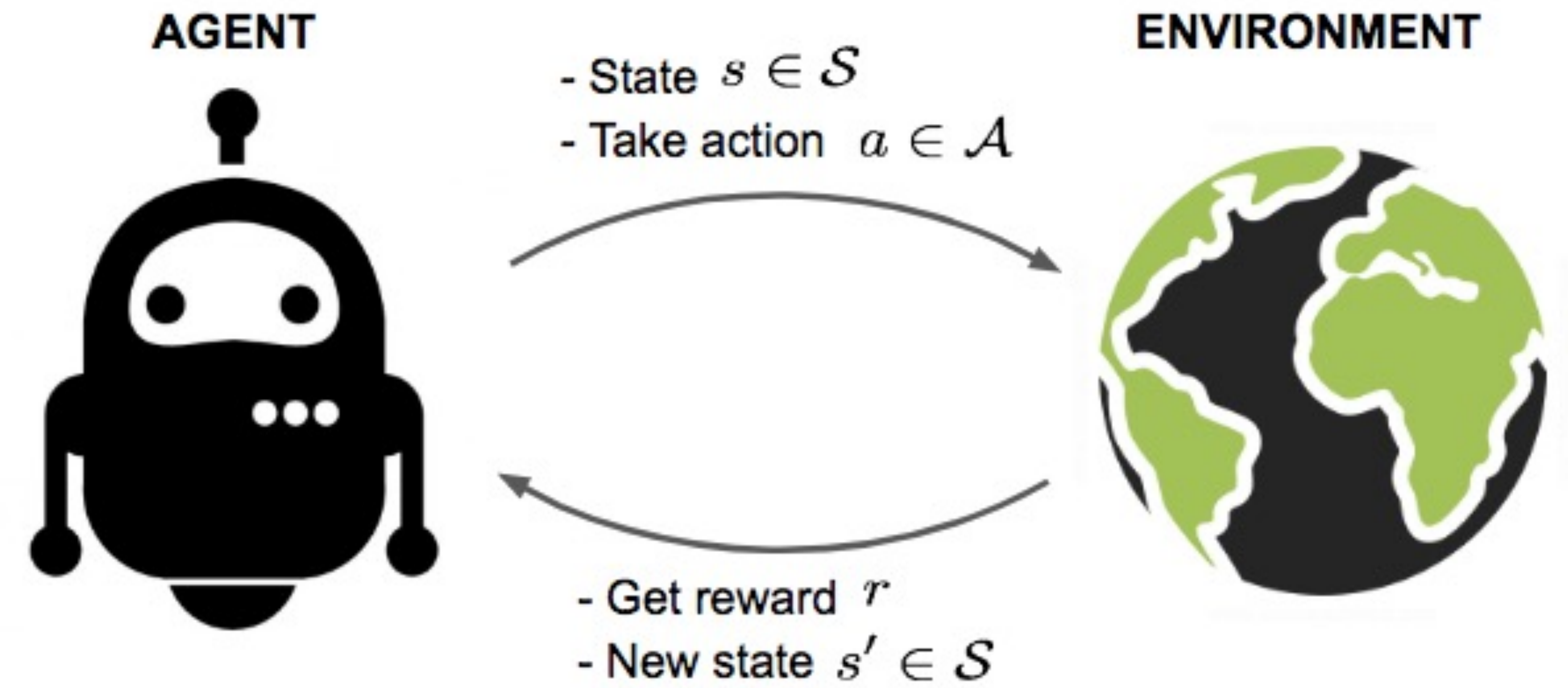


ARC Challenge:
LLMs + Multiple Abstraction Spaces



TaskGen / AgentJo:
LLMs + Goal-Directed,
Memory-Based Learning

Traditional Reinforcement Learning



Section 1: Learning by Pruning

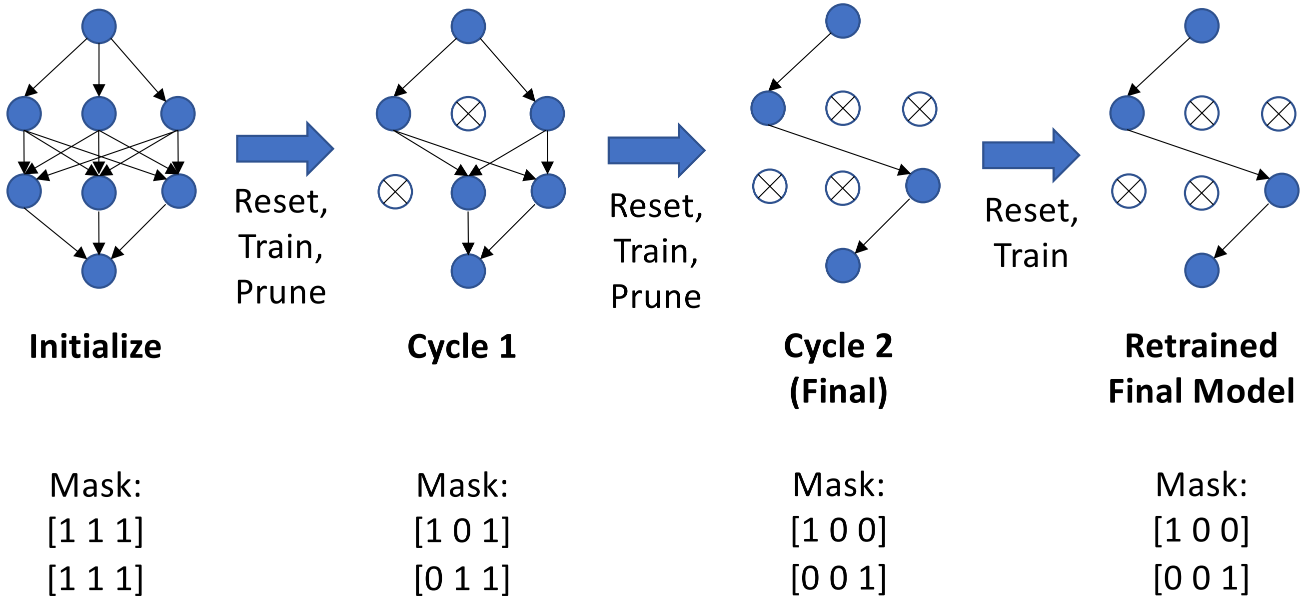
DropNet: Reducing Neural Network Complexity via Iterative Pruning (ICML 2020)

https://proceedings.mlr.press/v119/tan20a.html

DropNet – Algorithm
Applying iterative dropping to nodes/filters

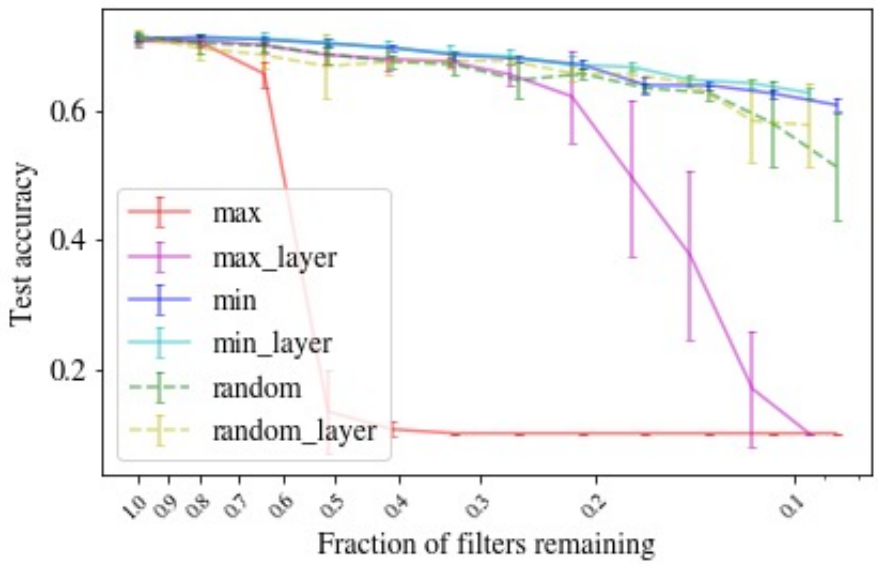
- Randomly initialize starting state of network and set mask *m* to all 1s

- Steps (Iterative):
  1. Reset network to starting state
  2. Apply mask *m* to nodes/filters
  3. Train network for at most *j* iterations until early stopping
  4. Apply pruning metric to choose a fraction *p* of nodes/filters to drop and update mask *m*
  5. Repeat steps 1 to 4 as necessary to get final mask
  6. Run steps 1 to 3 to retrain final network

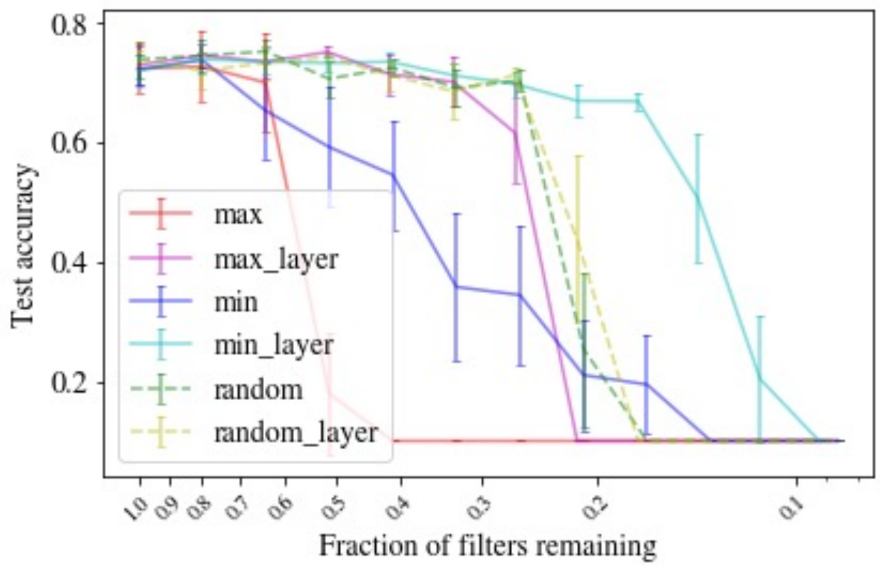


Results – CNN: CIFAR-10 (ResNet18, VGG19)

- min/min_layer has the best performance



ResNet18

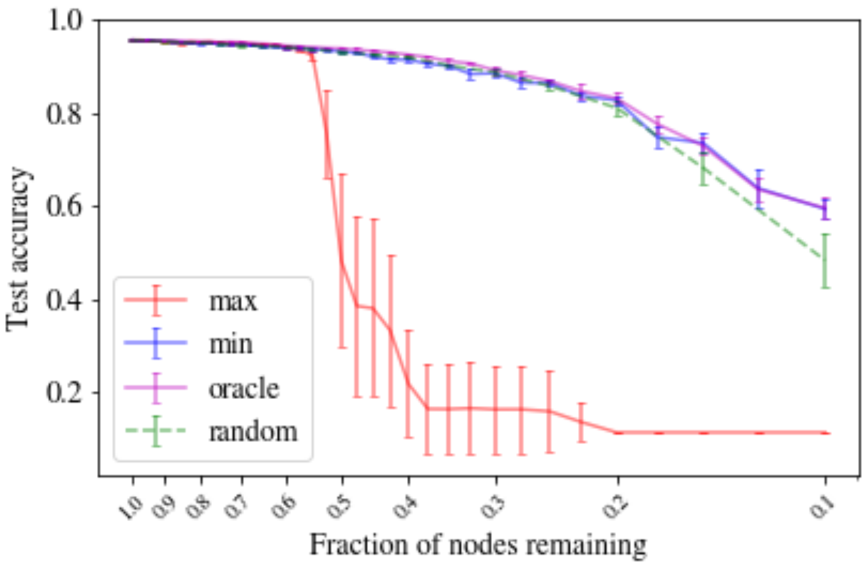


VGG19

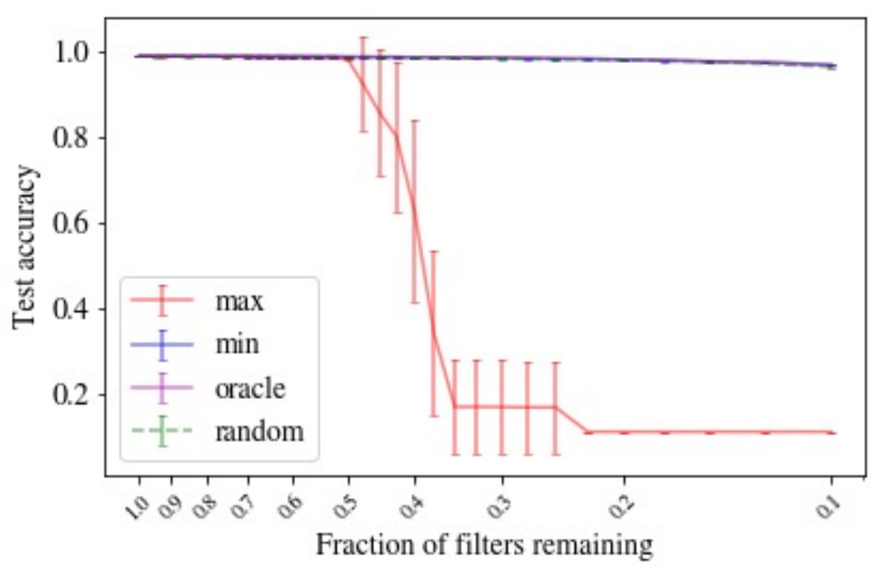
Error bars show the 95% confidence interval (CI) across 15 trials

Oracle Comparison (MNIST)

- Oracle **greedily** drops a node/filter at every training cycle in order to minimize overall training loss
- DropNet ($\min$) has comparable performance to oracle



Model A: FC20-FC20



Model B: Conv20-Conv20

Error bars show the 95% confidence interval (CI) across 15 trials

Section 2: Reward is not enough

Brick Tic Tac Toe - Exploring the Generalisability of AlphaZero to Novel Test Environments (arXiv 2022)

https://arxiv.org/pdf/2207.05991

Brick Tic-Tac-Toe (BTTT) Environment

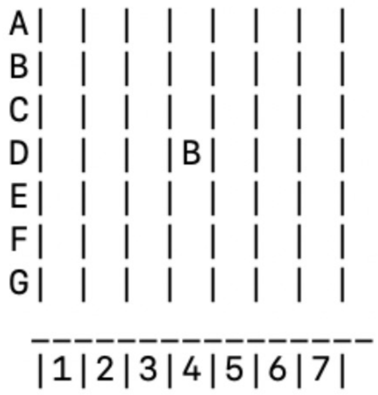


Figure 1. BTTT environment for Training (Variant 1)

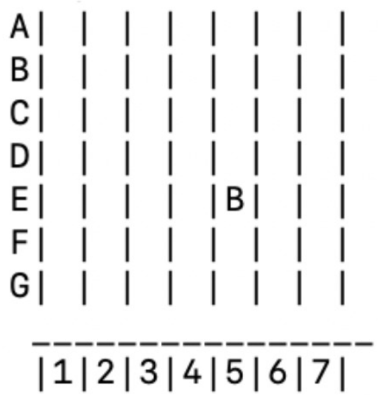


Figure 2. BTTT environment for Testing (Variant 2)

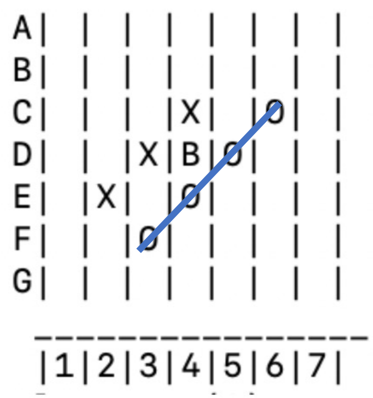


Figure 3. O winning in BTTT

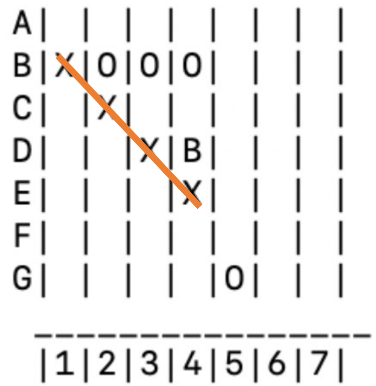


Figure 4. X winning in BTTT

- **Key idea: Train environment (Variant 1) different from Test environment (Variant 2)**
- Use simple game Tic-Tac-Toe with a twist:
  - Brick B is fixed at the start of the game, both players cannot place there
  - Player 1 (O) starts first, followed by Player 2 (X), first player to form 4-in-a-row wins
- Game designed to be always winnable as Player 1

AlphaZero may suffer from overfitting to train environment

Player 1	Player 2	Result (Var 1)	Result (Var 2)
MCTS 1000	MCTS 1000	70 - 30	70 - 30
MCTS 1000	MCTS 10000	15 - 85	15 - 85
MCTS 10000	MCTS 1000	100 - 0	100 - 0
MCTS 10000	MCTS 10000	70 - 30	70 - 30
Minimax	MCTS 1000	100 - 0	100 - 0
Minimax	MCTS 10000	100 - 0	100 - 0
Minimax	Minimax	100 - 0	100 - 0
MCTS 1000	Minimax	9 - 91	9 - 91
MCTS 10000	Minimax	51 - 49	51 - 49
AlphaZero NS	Minimax	100 - 0	0 - 100
AlphaZero 100	Minimax	100 - 0	0 - 100
AlphaZero 1000	Minimax	100 - 0	0 - 100
Minimax	AlphaZero 1000	100 - 0	100 - 0

- AlphaZero generalises worse than traditional search methods like Minimax and Monte Carlo Tree Search (MCTS)

Increasing training distribution can help mitigate overfitting

Player 1 (P1)	**Player 2**	**Result**	**P1 Win Rate(%)**
AlphaZero 100 (trained under D4 only) **[Baseline]**	Minimax	139 - 351	28.4
AlphaZero 100R2 (trained randomly under D3 and D4)	Minimax	306 - 184	62.4
AlphaZero 100R3 (trained randomly under C3, D3 and D4)	Minimax	443 - 47	90.4
AlphaZero 1000 (trained under D4 only)	Minimax	364 - 126	74.3
AlphaZero 1000R2 (trained randomly under D3 and D4)	Minimax	415 - 75	84.7
AlphaZero 1000R3 (trained randomly under C3, D3 and D4)	Minimax	483 - 7	98.6
Minimax **[Benchmark for Generalizable Perfect Play]**	Minimax	490 - 0	100

- AlphaZero generalises better when given more starting configurations
- 10 games per possible starting brick block position (7 x 7 = 49), total 490 games

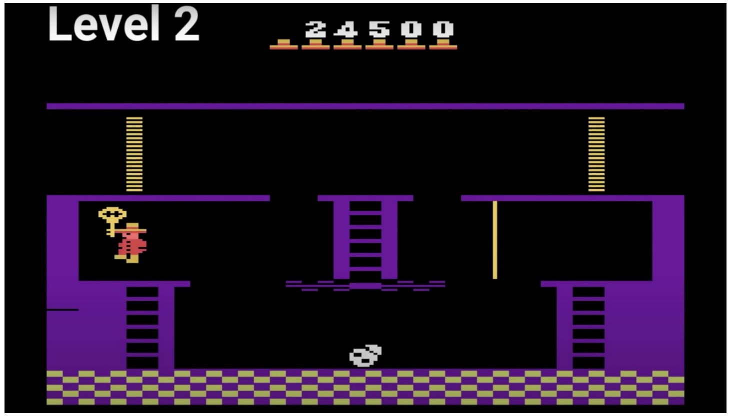
Section 3: Memory-based Learning

Using Hippocampal Replay to Consolidate Experiences
in Memory-Augmented Reinforcement Learning (NeurIPS memARI workshop 2022)

https://memari-workshop.github.io/papers/paper_38.pdf

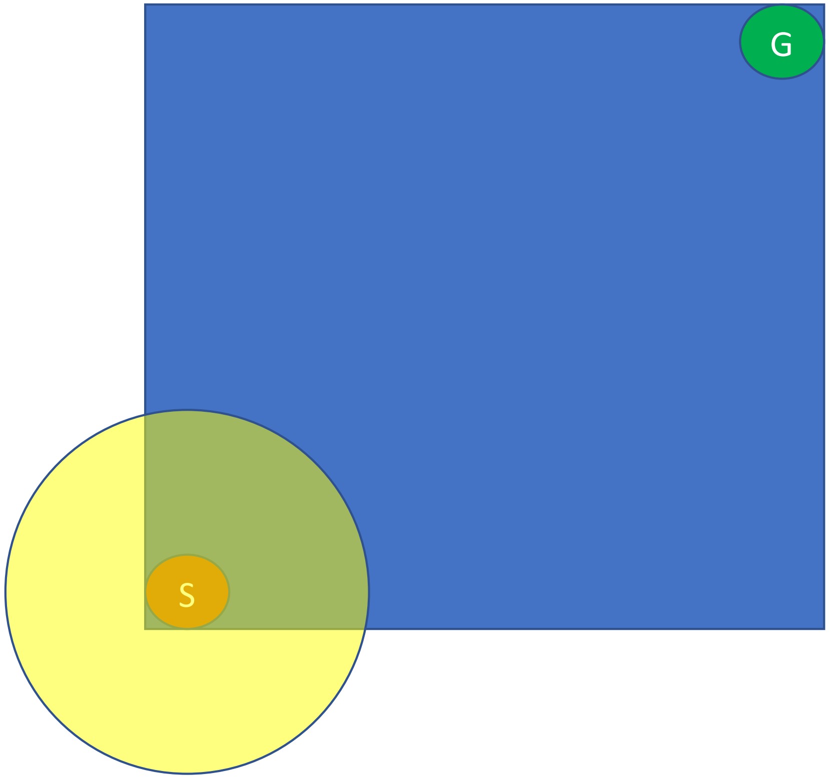
Go-Explore

- Using reward alone may be insufficient for sparse reward settings
- **Go-Explore** (Ecofett, 2019) uses external memory to update states
- In order to explore more states:
  - **Go:** Jump probabilistically to a state
  - **Explore:** Explore randomly from the state
- Update state's memory if current trajectory to that state is better

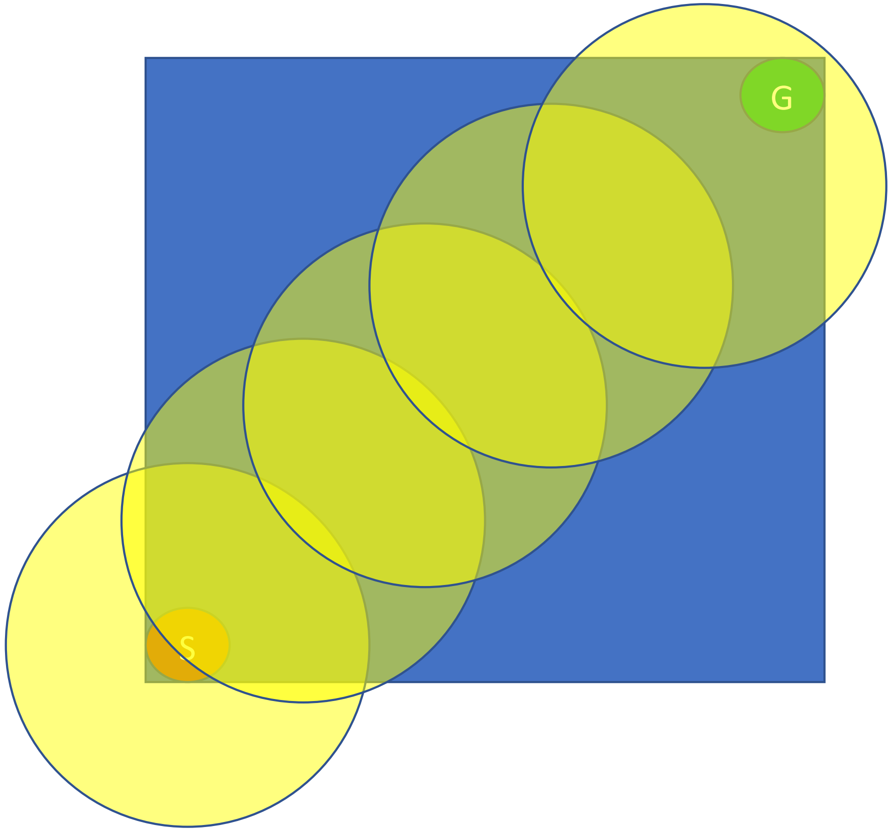


Montezuma's Revenge,
a game with sparse rewards

The torchlight analogy



Exploration from starting state



Exploration from frontier

How to “Go” and “Explore”

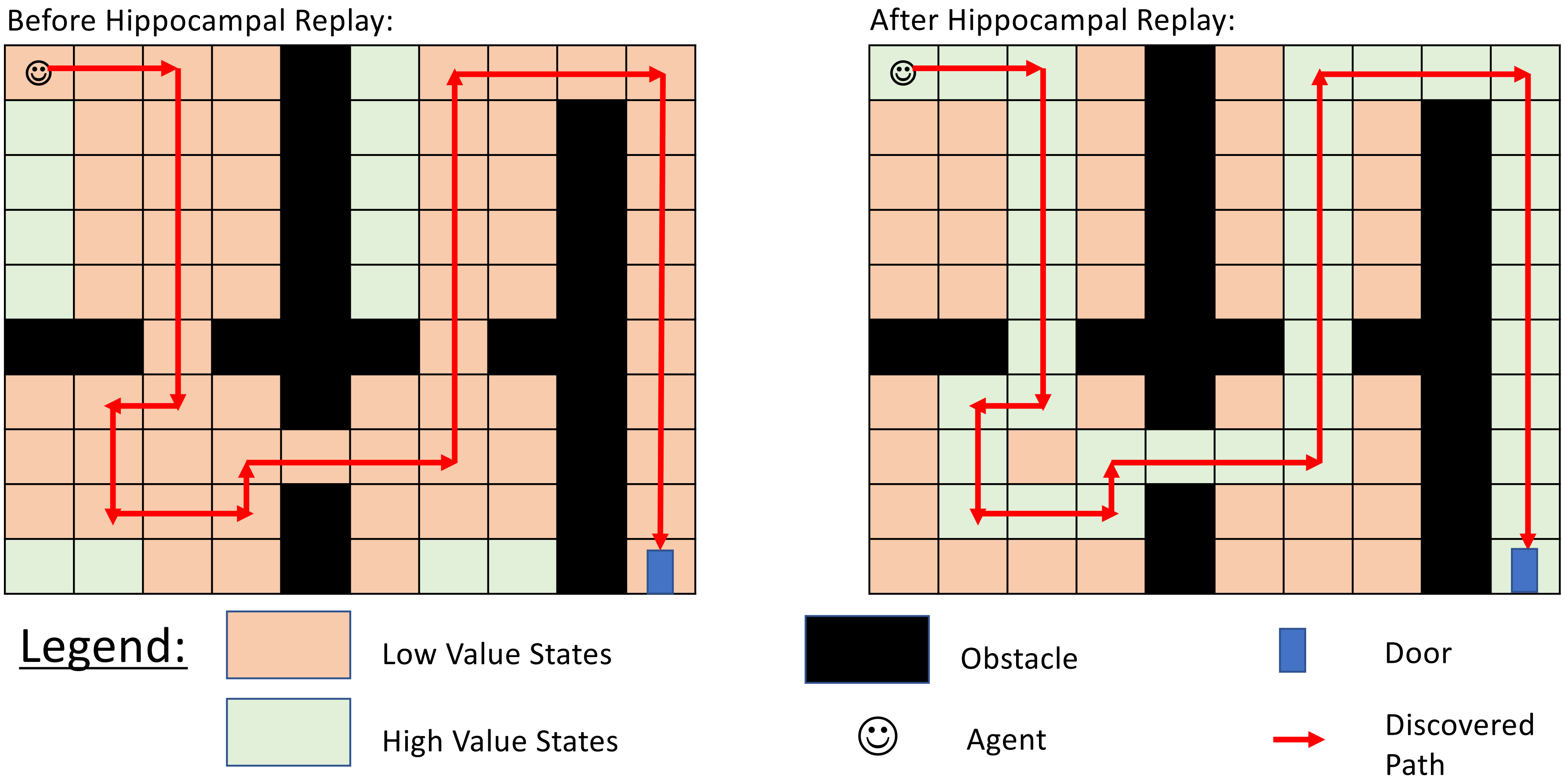
- Random exploration can be inefficient
- **Solution –** Use Deterministic Selection to balance explore and exploit

$$\alpha \ \cdot reward + \kappa\sqrt{moves} \ - \ \gamma\sqrt{numselected + numvisited}$$

  - ***reward***: environment reward
  - ***moves***: number of moves to reach state
  - ***numselected***: number of times state is selected in “Go” phase
  - ***numvisited***: number of times state is visited in “Explore” phase

- Similar to Upper Confidence Bounds (UCB) equation and encourages greedy action selection in the long run

Hippocampal Replay creates Exploration Highway



Results for Walled Maze

Overall		First Solve		Steps to Solve		
Agent	**Solve Rate**	**Run**	**Memory size**	**Avg**	**Min**	**Max**
Random	0/100	-	-	-	-	-
Go-Explore	0/100	-	-	-	-	-
Go-Explore-HR	0/100	-	-	-	-	-
Go-Explore-Count	**100/100**	**1**	**7552**	4918.2	**4718.0**	6362.0
Go-Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**
Explore-Count	52/100	**1**	**7552**	7039.0	**3094.0**	9758.0
Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**

- Our count-based approaches (Go-Explore-Count, Explore-Count) perform better than vanilla Go-Explore

- Hippocampal Replay leads to more consistent performance (higher solve rate) and less exploration (higher minimum number of steps to solve)

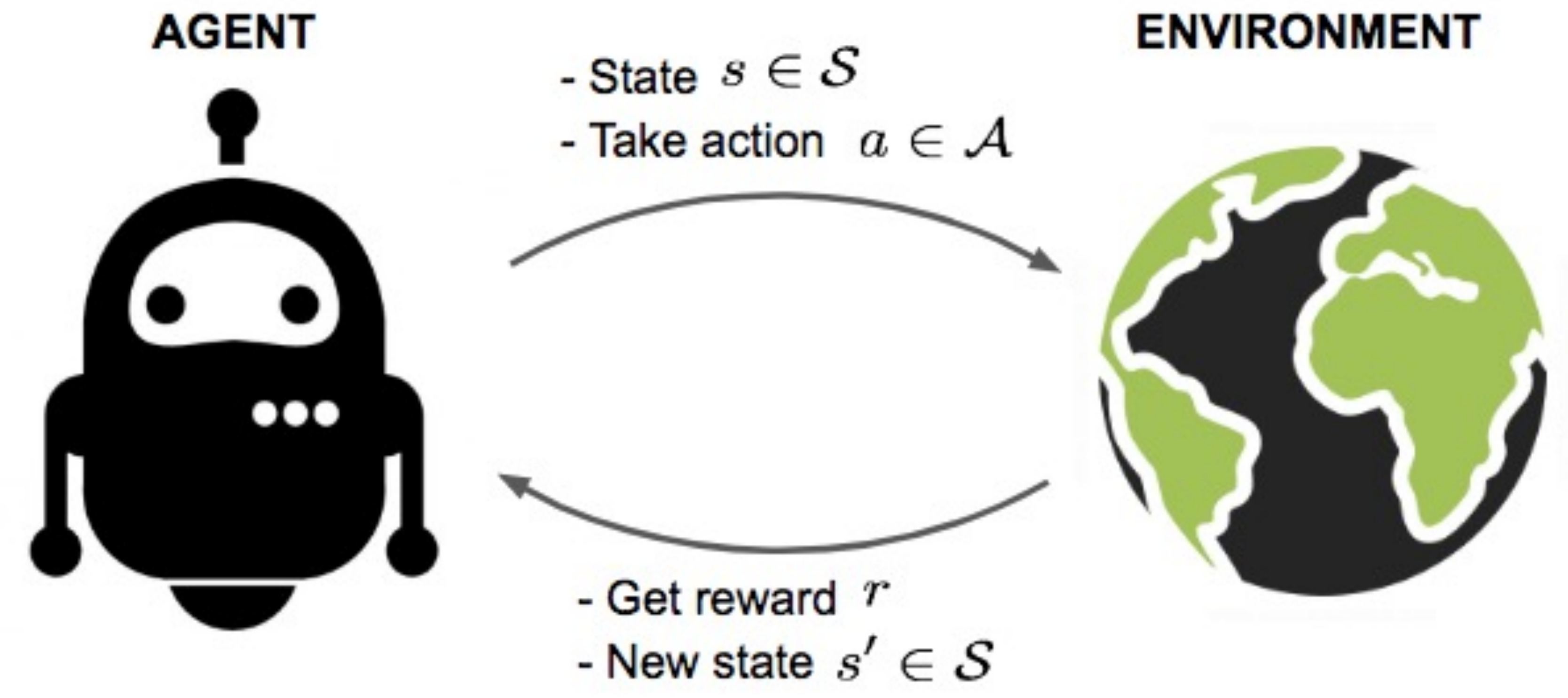
Section 4: Goal-Directed, Memory-based Learning

Learning, Fast and Slow: A Goal-Directed, Memory-Based Approach for Dynamic Environments (IEEE ICDL 2023)

https://ieeexplore.ieee.org/document/10364540/

Best Paper Finalist

Traditional Reinforcement Learning



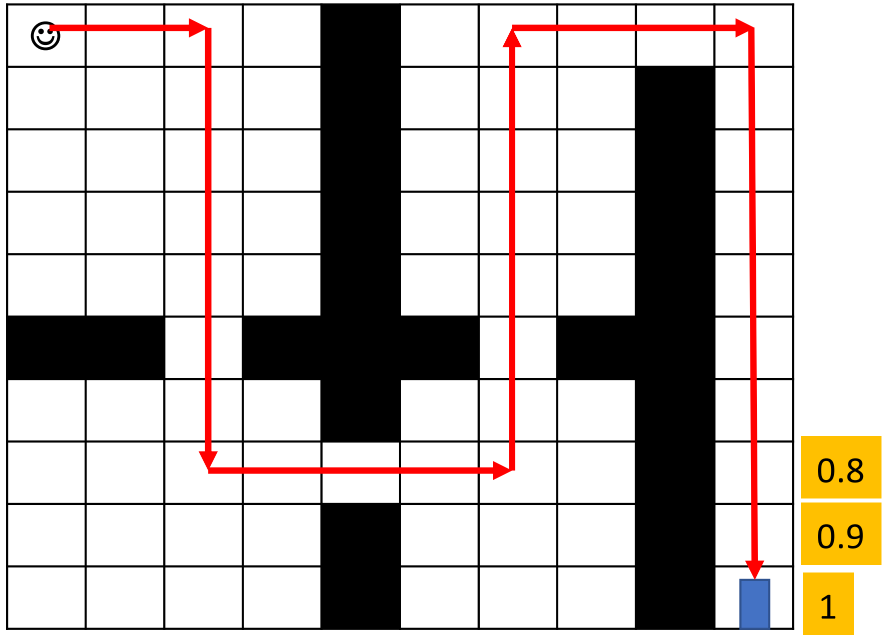
Insight: Value-based reinforcement learning is slow

- 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

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

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

Walled Maze



Legend:

☺ Agent

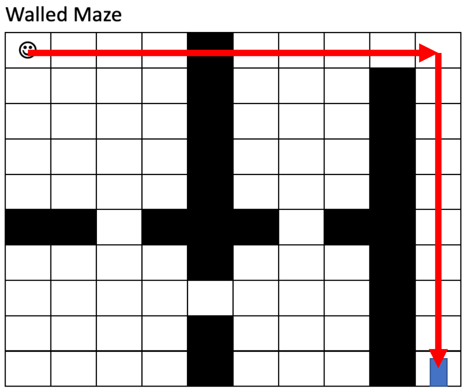
Door

Goal-Directed Action Prediction

- Instead of using value functions, use a **goal-directed action prediction given start state and goal state**

- Initial inference can be approximate, just needs to **head towards general direction of goal**
  - Prevent going in cycles by using count-based methods

- Learn goal-directed network **via self-supervised learning** (similar to Transformers pre-training)
  - Our own trajectories is the source of truth

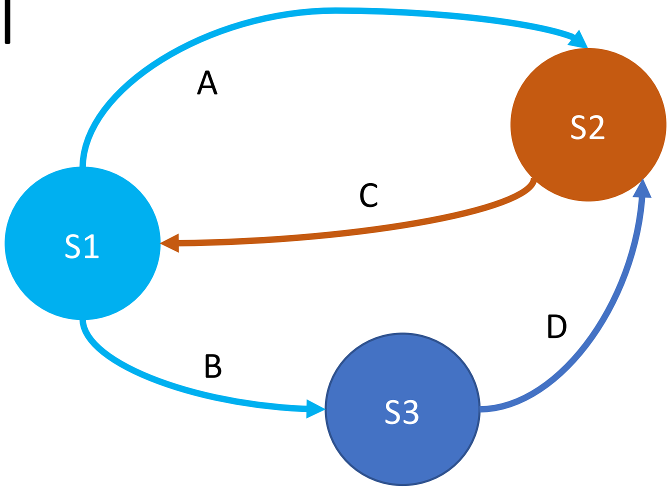


Using memory as world model

- **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

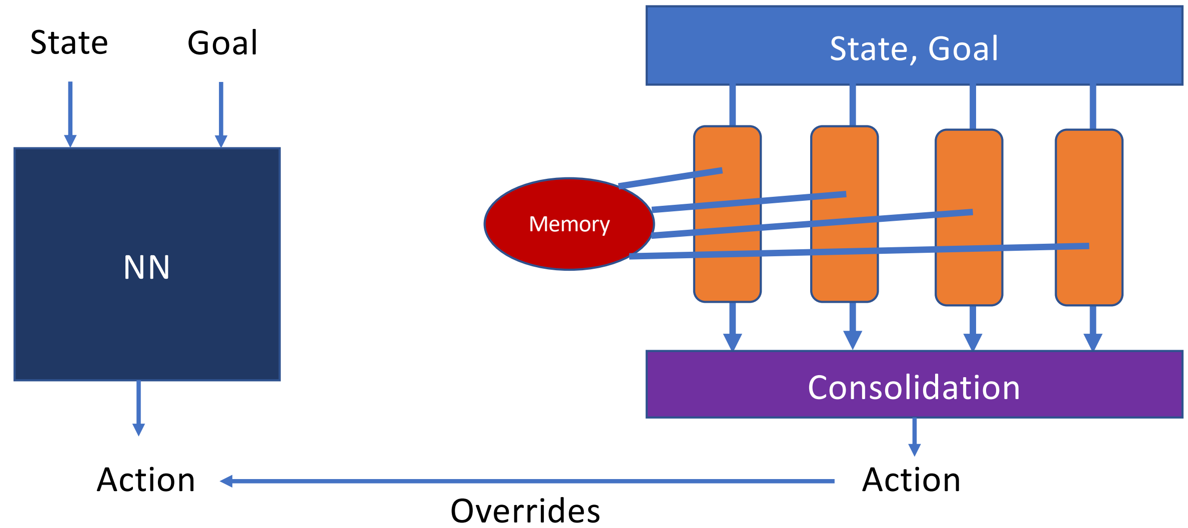
- Just need to **remember experienced transitions** and then **retrieve it** the next time we encounter a similar state



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

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

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

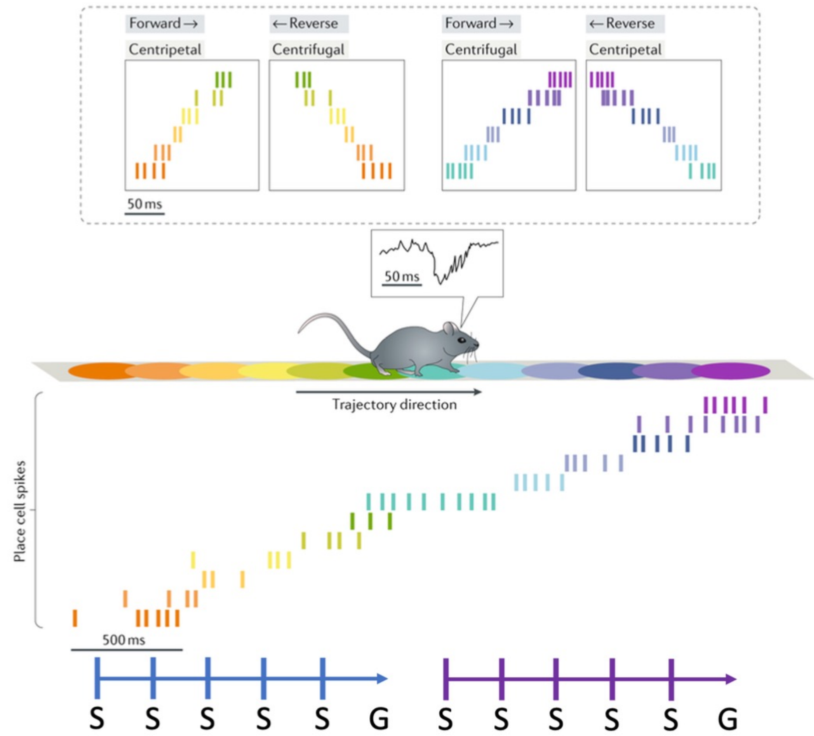
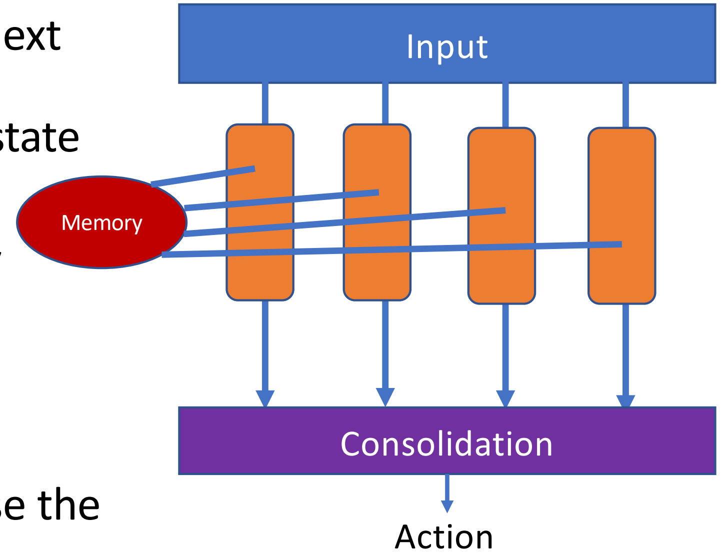


Figure extracted from Joo, H. R., & Frank, L. M. (2018). The hippocampal sharp wave-ripple in memory retrieval for immediate use and consolidation. Nature reviews. Neuroscience, 19(12), 744–757. https://doi.org/10.1038/s41583-018-0077-1

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 Fast and Slow Networks

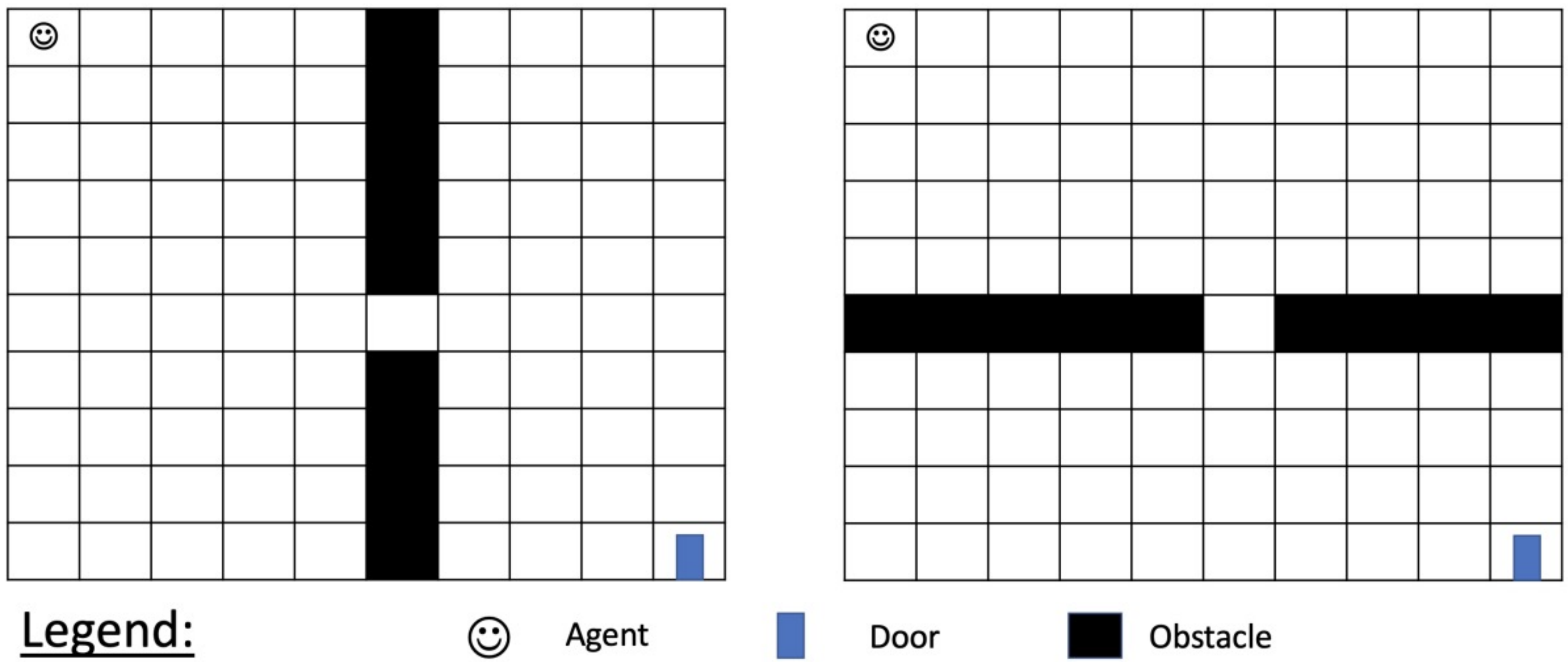
- State and Action Prediction
  - Agent has a goal state in mind, and knows its current state
  - **System 1:** Agent queries the fast neural network to get action probabilities for the goal (exploit)
  - Get state-action visit counts via retrieval from episodic memory and choose action in explore-exploit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

Dynamic Navigation with changing obstacles after 50 episodes (Left -> Right)



Main Results (10x10)

- F&S performs the best (91.9% solve rate), followed by PPO (61.2%)
- TRPO, A2C, DQN perform worse than random
- Selecting shortest path for memory retrieval makes it locally optimal

TABLE IV

ADAPTABILITY OF METHODS EVALUATED BY NUMBER OF SOLVES ON A DYNAMIC 10X10 NAVIGATION TASK. HIGHER IS BETTER (IN BOLD).

Agent	Number of Solves		
	First 50 episodes	Last 50 episodes	Total
F&S	**44.0 ± 1.7**	**47.9 ± 2.6**	**91.9 ± 2.7**
PPO	29.4± 6.2	31.8± 4.3	61.2± 7.5
TRPO	14.6± 6.0	11.5± 5.7	26.1± 8.5
A2C	11.9± 2.5	12.0± 5.2	23.9± 6.3
DQN	2.4± 1.6	2.5± 1.9	4.9± 2.0
Random	15.6± 3.4	14.1± 2.0	29.7± 3.3

TABLE V

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
F&S	**1029.5 ± 145.4**	**675.4 ± 223.3**	**1704.9 ± 280.6**
PPO	2516.0± 416.7	2154.2± 307.2	4670.2± 527.3
TRPO	3634.0± 446.5	3821.5± 364.5	7455.5± 606.7
A2C	3884.7± 105.8	3908.0± 287.5	7792.7± 308.2
DQN	4424.2± 142.8	4408.3± 173.6	8832.5± 184.8
Random	3736.3± 187.2	3795.5± 165.8	7531.8± 236.1

Ablation Study

- Baseline uses 20 depth and 100 threads
- Fast and slow are both essential components
- Increasing depth and threads are both beneficial for performance

TABLE VIII

ABLATION STUDY ON ADAPTABILITY OF F&S AGENT ON A DYNAMIC 10X10 NAVIGATION TASK. HIGHER IS BETTER (IN BOLD).

Agent	Number of Solves		
	First 50 episodes	Last 50 episodes	Total
Baseline	44.0± 1.7	47.9± 2.6	91.9± 2.7
No Slow	31.2± 2.6	32.2± 4.7	63.4± 5.1
No Fast	23.3± 2.5	26.3± 5.3	49.6± 6.1
No Fast,Slow	13.0± 2.8	13.1± 3.9	26.1± 4.4
10 depth	43.0± 2.2	46.4± 3.1	89.4± 4.0
50 depth	**44.7 ± 1.7**	**48.9 ± 1.2**	**93.6 ± 2.2**
50 threads	43.1± 2.2	47.3± 1.3	90.4± 2.7
200 threads	44.6± 1.7	48.6± 1.1	93.2± 2.4

TABLE IX

ABLATION STUDY ON EFFICIENCY OF F&S AGENT ON A DYNAMIC 10X10 NAVIGATION TASK. LOWER IS BETTER (IN BOLD).

Agent	Steps Above Minimum		
	First 50 eps	Last 50 eps	Total
Baseline	1029.5± 145.4	675.4± 223.3	1704.9± 280.6
No Slow	2625± 234.4	2517.0± 316.0	5142.7± 389.7
No Fast	2694.7± 216.8	2386.6± 445.0	5081.3± 496.2
No Fast,Slow	3890.6± 222.6	3853.0± 207.5	7743.6± 317.3
10 depth	1225.5± 225.1	821.3± 292.3	2046.8± 455.8
50 depth	941.2± 168.0	617.6± 115.4	1558.8± 250.6
50 threads	1112.6± 216.0	761.2± 156.9	1873.8± 231.0
200 threads	**870.2 ± 152.9**	**521.2 ± 132.7**	**1391.4 ± 224.9**

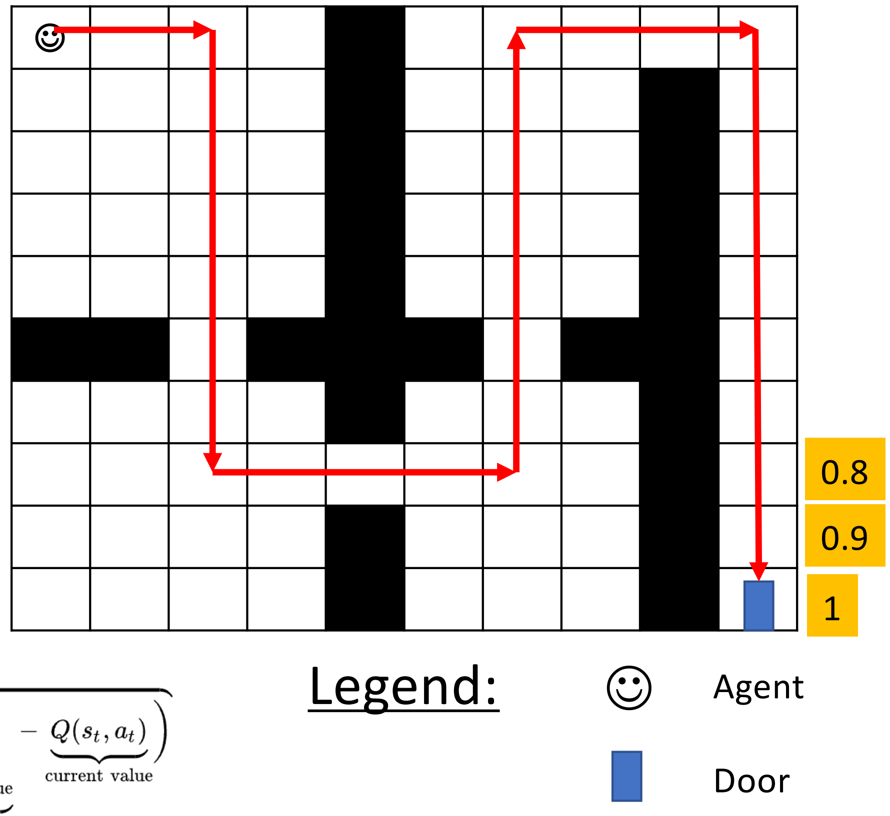
Insight: Value-based reinforcement learning is slow

- 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

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

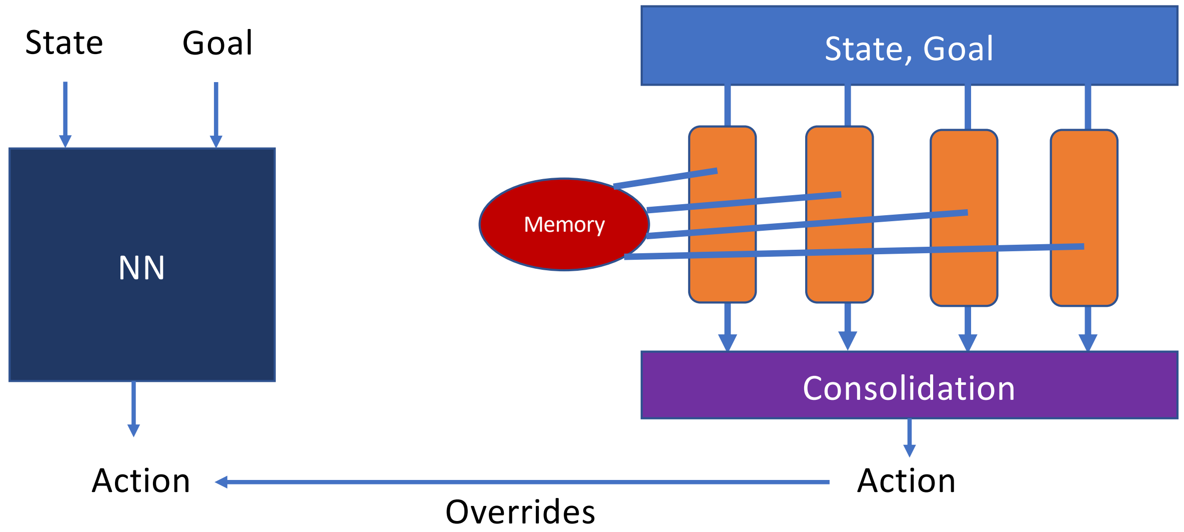
$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal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

Walled Maze



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

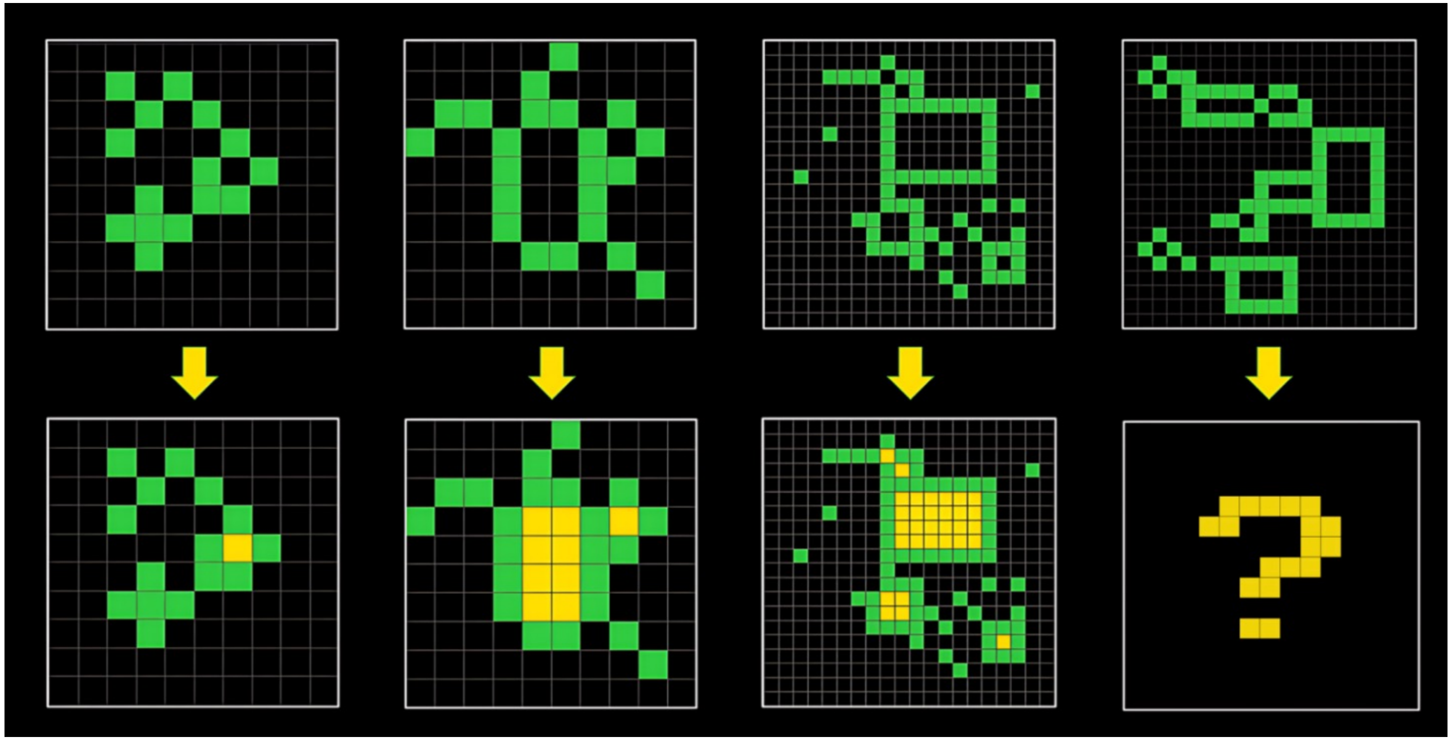
Section 5: Multiple Abstraction Spaces for Learning

Large Language Model as a System of Multiple Expert Agents: An Approach to solve the Abstraction and Reasoning Corpus Challenge (IEEE CAI 2024)

https://arxiv.org/pdf/2310.05146

https://ieeecai.org/2024/wp-content/pdfs/540900a793/540900a793.pdf

ARC-AGI



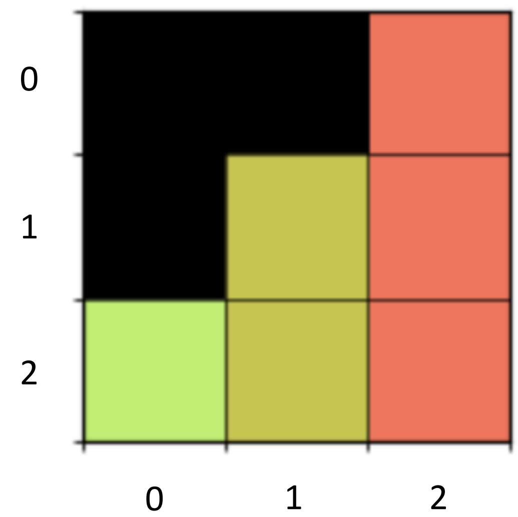
https://arcprize.org/arc

View Representations

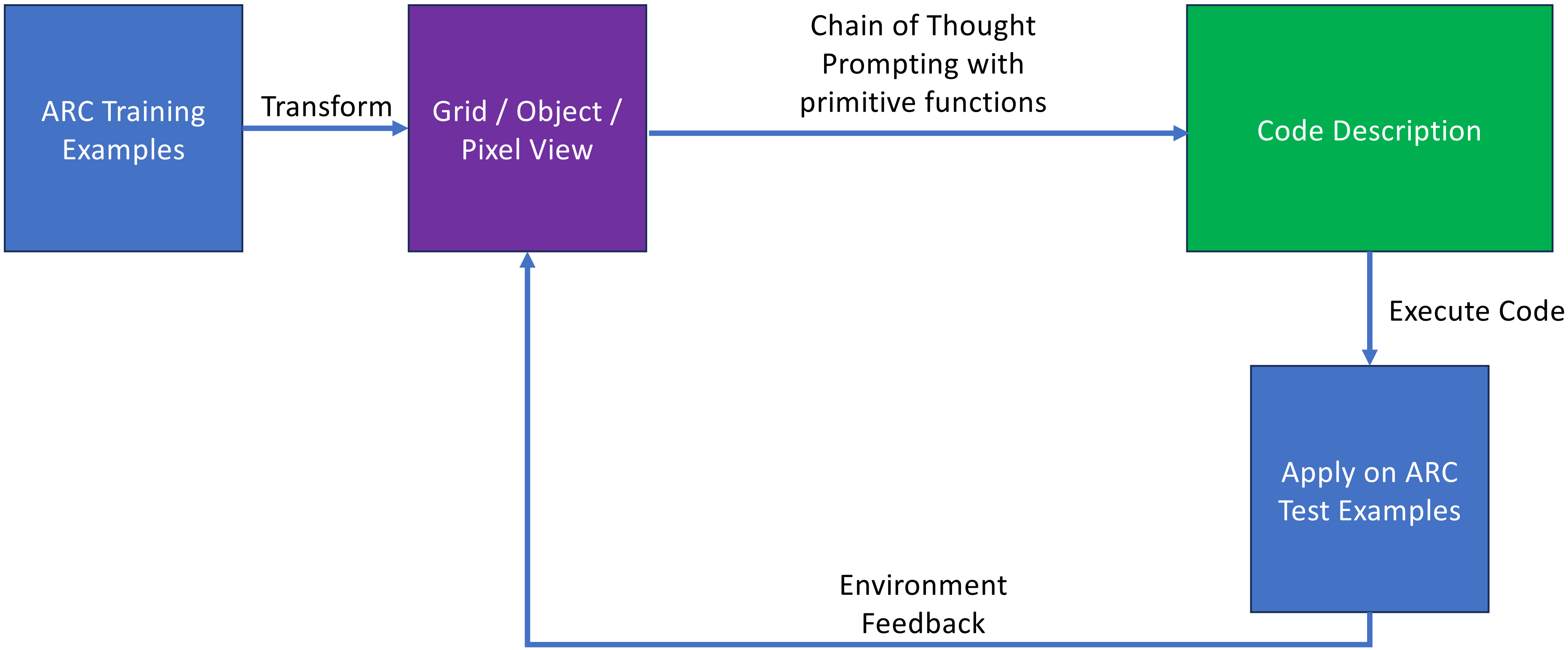
Grid View: [
['.', '.', 'f'],
['.', 'd', 'f'],
['c', 'd' ,'f']]

Object View (Mono-Color):
[{'tl':(0,2), 'grid':[['f'],['f'],['f']], 'size':(3,1), 'cell_count':3, 'shape':[['x'],['x'],['x']]},
{'tl':(1,1), 'grid':[['d'],['d']], 'size':(2,1), 'cell_count':2, 'shape':[['x'],['x']]},
{'tl':(2,0), 'grid':[['c']], 'size':(1,1), 'cell_count':1, 'shape':[['x']]}]

Pixel View: {
'f':[(0,2),(1,2),(2,2)],
'd':[(1,1),(2,1)],
'c':[(2,0)]}



Overall Process: Using Code as Grounding



Chain of Thought (CoT) prompting via JSON

You are to output the following in json format:

{**’reflection’:** ’reflect on the answer’,

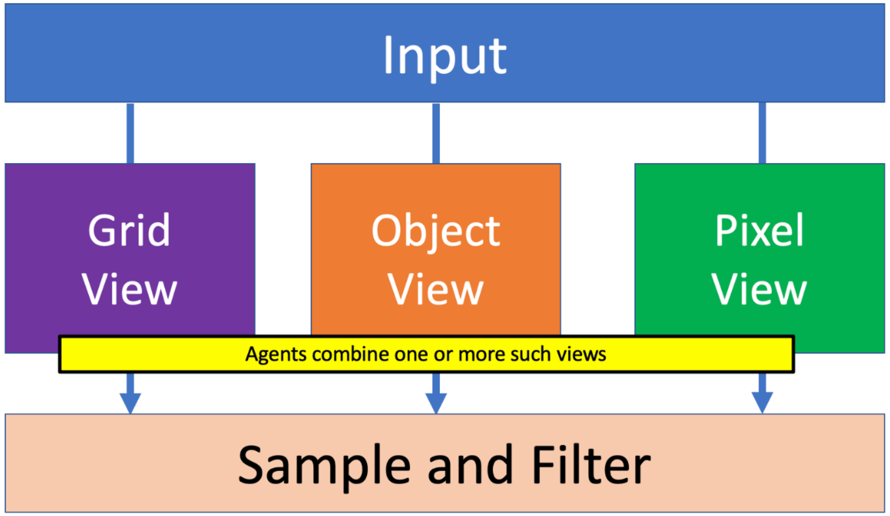
’pixel_changes’: ’describe the changes between the input and output pixels, focusing on movement or pattern changes’,

’object_changes’: ’describe the changes between the input and output objects, focusing on movement, object number, size, shape, position, value, cell count’,

’helper_functions’: ’list any relevant helper_functions for this task’,
’overall_pattern’: ’describe the simplest input-output relationship for all input-output pairs’,
’program_instructions’: ’Plan how to write the python function and what helper functions and conditions to use’,
’python_program’: "Python function named ’transform_grid’ that takes in a 2D grid and generates a 2D grid. Output as a string in a single line with \n and \t."}.

Do not use quotation marks ’ or " within the fields unless it is required for the python code

Results



View Type	Number of Tasks Solved
Total	50
Object View	23
Pixel View	19
Object & Pixel View	1
No Object & Pixel View (only Grid View)	7

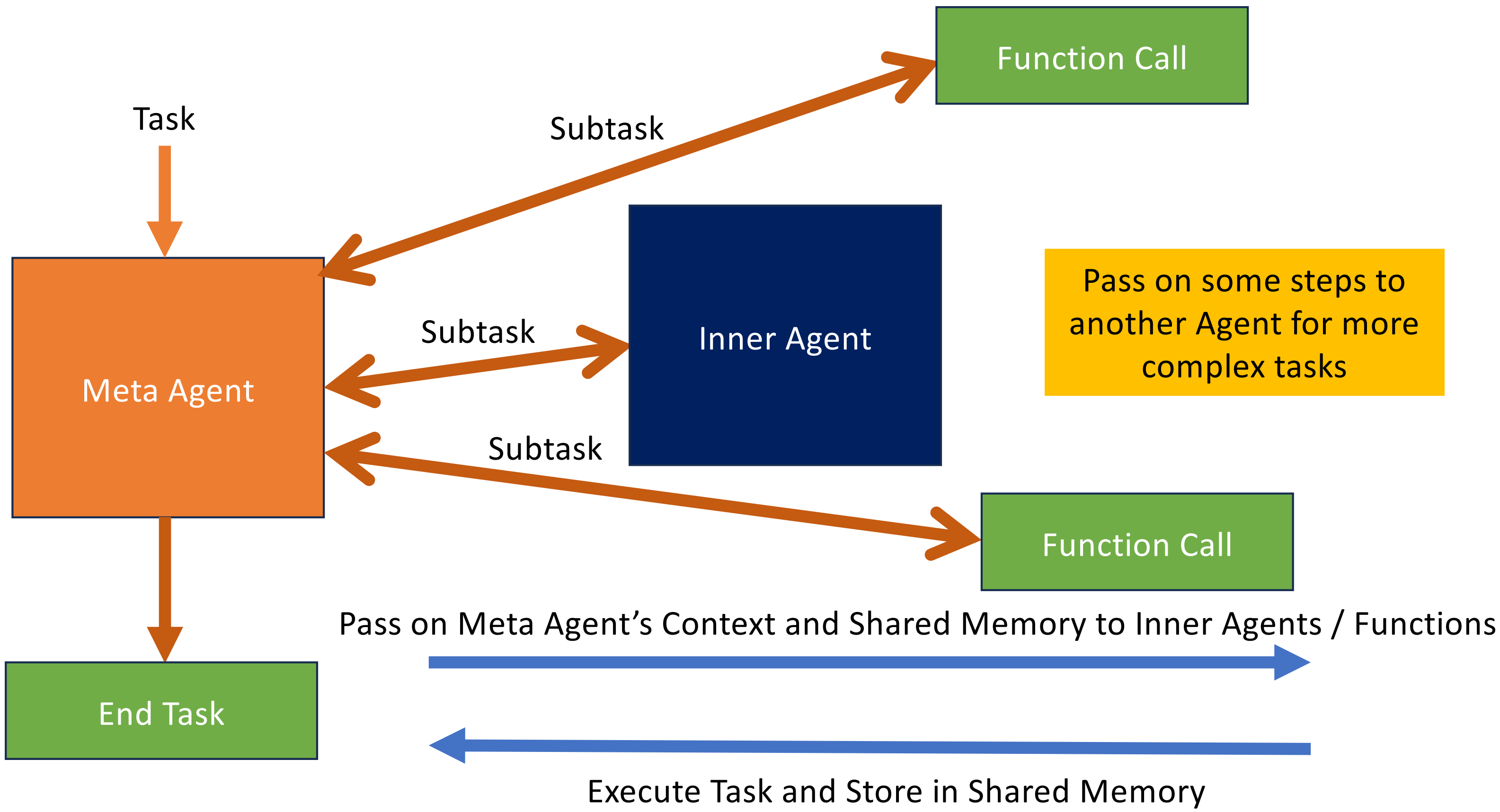
- **50 out of 111 tasks solved (45%)**
- Using more abstraction spaces / bias increases the solve rate
- Increased sampling will likely increase the solve rate

Section 6: LLMs + Goal-Directed, Memory-Based Learning

TaskGen: A Task-Based, Memory-Infused Agentic Framework using StrictJSON
https://arxiv.org/pdf/2407.15734

Acknowledgements
Collaborators: Prince Saroj, Brian Lim Yi Sheng, Richard Cottrill, Hardik Maheshwari, Bharat Runwal
Funders: Simbian AI (Ambuj Kumar and Alankrit Chona)

Overall Framework



Conciseness: Reduce tokens by using StrictJSON

JSON Schema for Parameters – 110 tokens

```
{
    "parameters": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA",
            },
            "format": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The temperature unit to use. Infer this from the users location.",
            },
        },
        "required": ["location", "format"],
    }
}
```

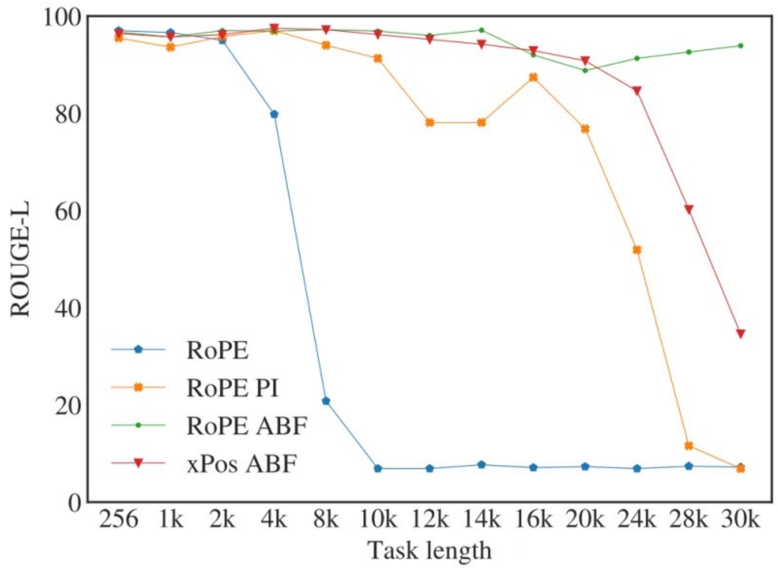
StrictJSON Schema for Parameters – 58 tokens

```
{
    "###Location###": "The city and state, e.g. San Francisco, CA, type: str",
    "###Format###": 'The temperature unit to use. Infer this from the users location, type: Enum["celsius", "fahrenheit"]'
}
```

https://github.com/tanchongmin/strictjson

Tokens impact not just cost, but performance

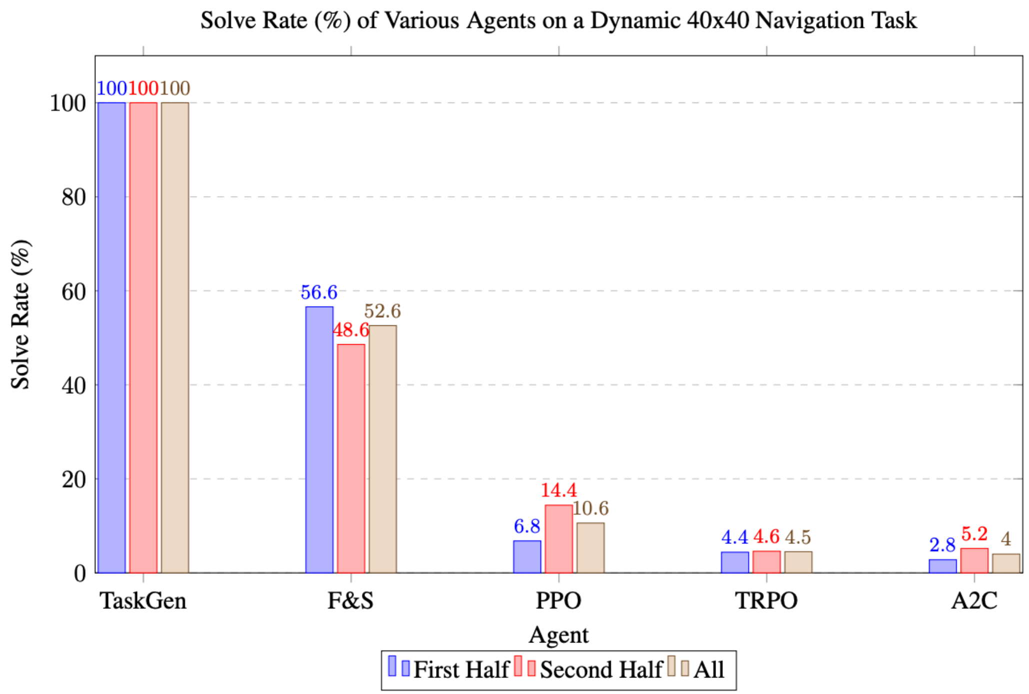
- Performance sharply degrades after 2-3k tokens
  - For Rotary Positional Embeddings (RoPE) in Llama 2



(b) Performance on FIRST-SENTENCE-RETRIEVAL task.

Effective Long-Context Scaling of Foundation Models. 2023. Xiong et. al.

Results: TaskGen Agent is close to optimal in Dynamic 40x40 Grid Maze



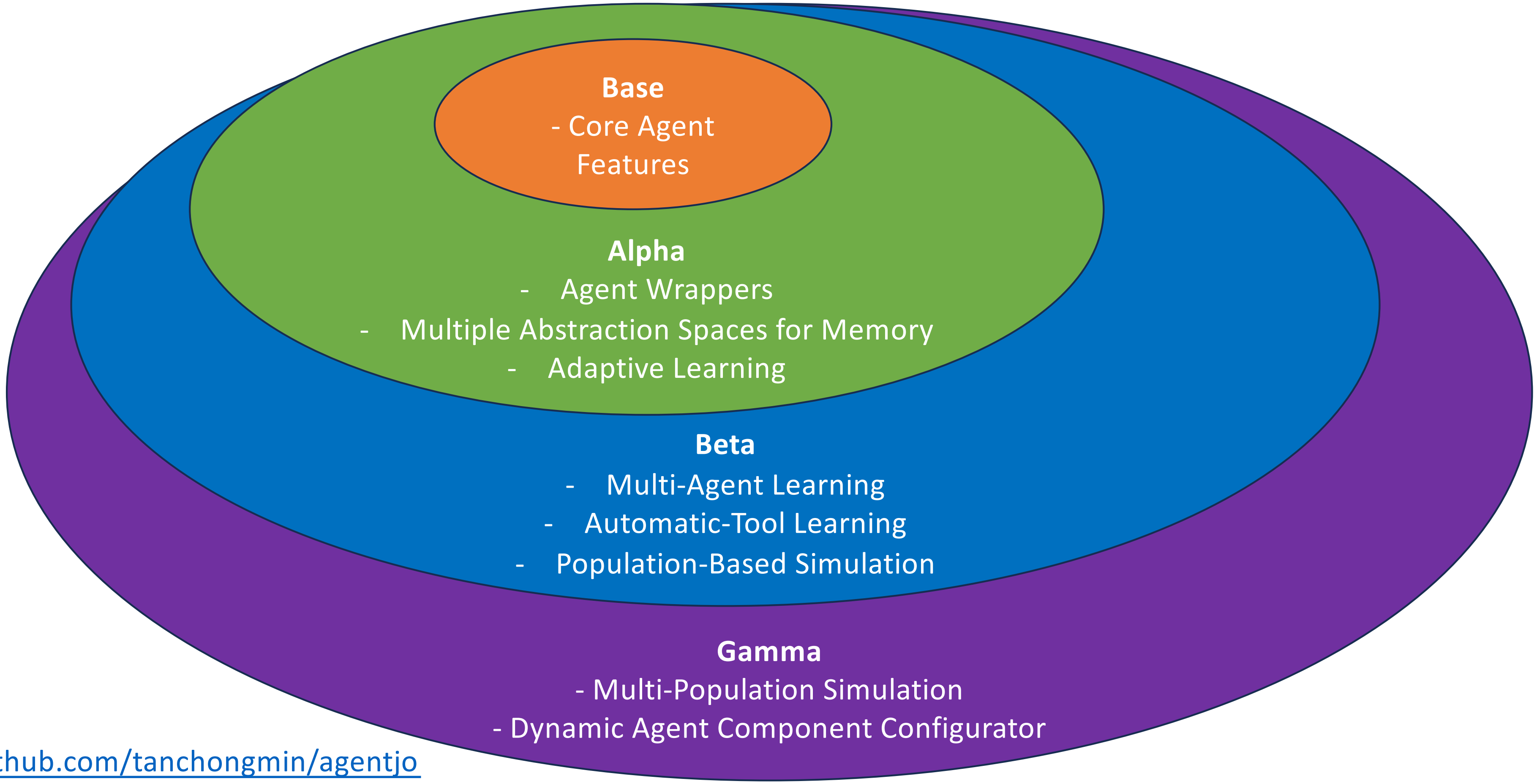
- With few-shot prompting used in planner, and feeding in next step into TaskGen agent, it outperforms Fast&Slow and other actor-critic methods

- Insight: If we can encode knowledge in text and such knowledge is within training distribution, LLMs will likely fare better than native neural networks

AgentJo

What’s next for the next 5 - 10 years

AgentJo – **Human-Friendly**, Fast Learning and Adaptable Agent Communities



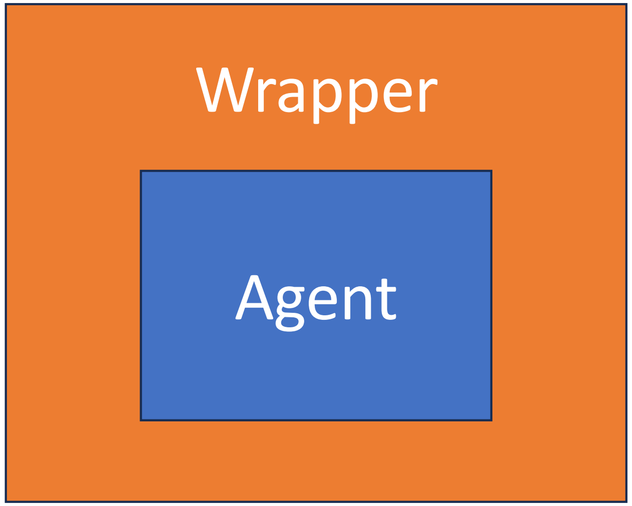
https://github.com/tanchongmin/agentjo

Agent Wrappers

- Base Agent functionality is kept simple to minimise overhead

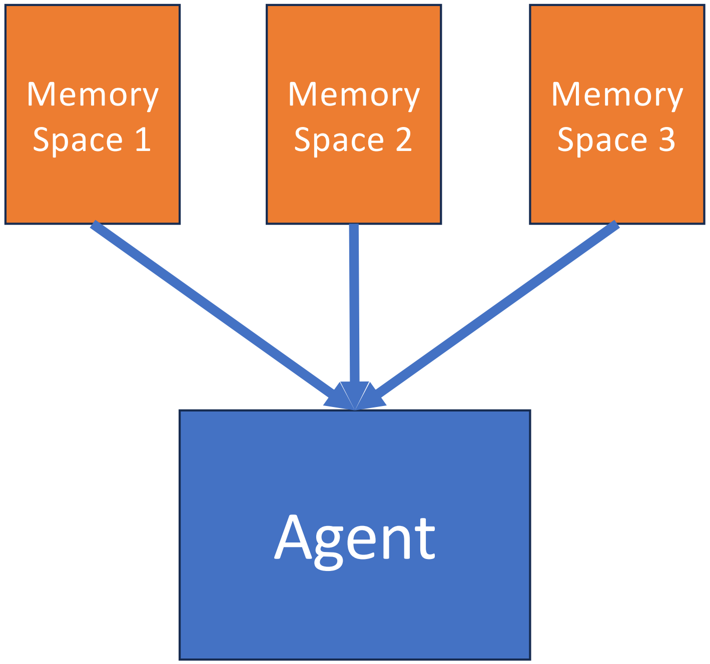
- Agents are meant to be modular and many can be "spawned" for usage in various pipelines

- For different tasks, can augment with extra functionalities via wrappers:
  - **PlanningWrappers:** How to plan and execute the plan
  - **ReflectionWrappers:** How to reflect and learn
  - **VerifierWrappers:** How to verify agent's outputs
  - **ConversationWrappers:** How to converse with the agent
  - **MultiAgentWrappers:** How multiple agents can converse



Memory Abstraction Spaces

- Memory is important for learning
- Memory is stored in different abstraction spaces, different modality
- Retrieve what is needed at each space to solve the task

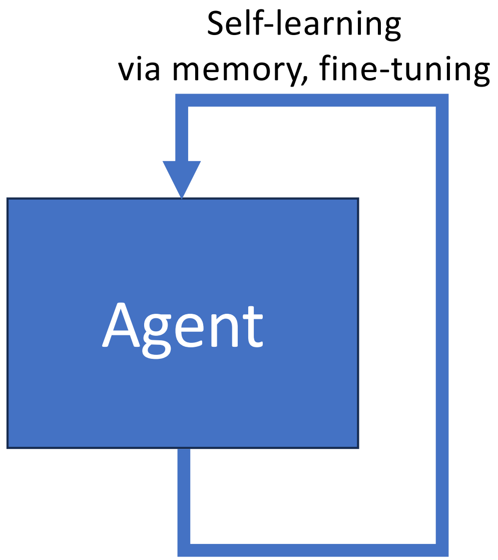


Adaptive Learning

- Agent is able to consolidate and store reflections in memory/fine-tuning, and use it for future tasks

- Agent is able to configure its own functions, context according to need

- Agent is able to learn **within a task**, and **through tasks**



Multi-Agent Learning

- Each agent interacts with others and **shares knowledge**
- **Not all knowledge is shared** with everyone, only some of knowledge shared with neighbours if agent is performant
- Agents intentionally **kept different** and not homogeneous so that there is adaptability should environment change

Questions to Ponder

- Is reward still needed for learning? If so, how can we combine reward and goal-directed learning?
- Is memory always fixed, or is it changeable? What are the pros / cons of changing memory during retrieval?
- How do we do adaptive learning by reflection when we do not have the ground truth? How can we ensure reflection / reasoning is grounded?
- How should multiple agents interact with one another? How much should each agent share? How should memory be inherited from one generation to the next?

Thank you

Special mention to my supervisor, Prof. Mehul,

for allowing me to explore my interests ☺