# Integrated Cognitive Architectures (Stanford CS379C)

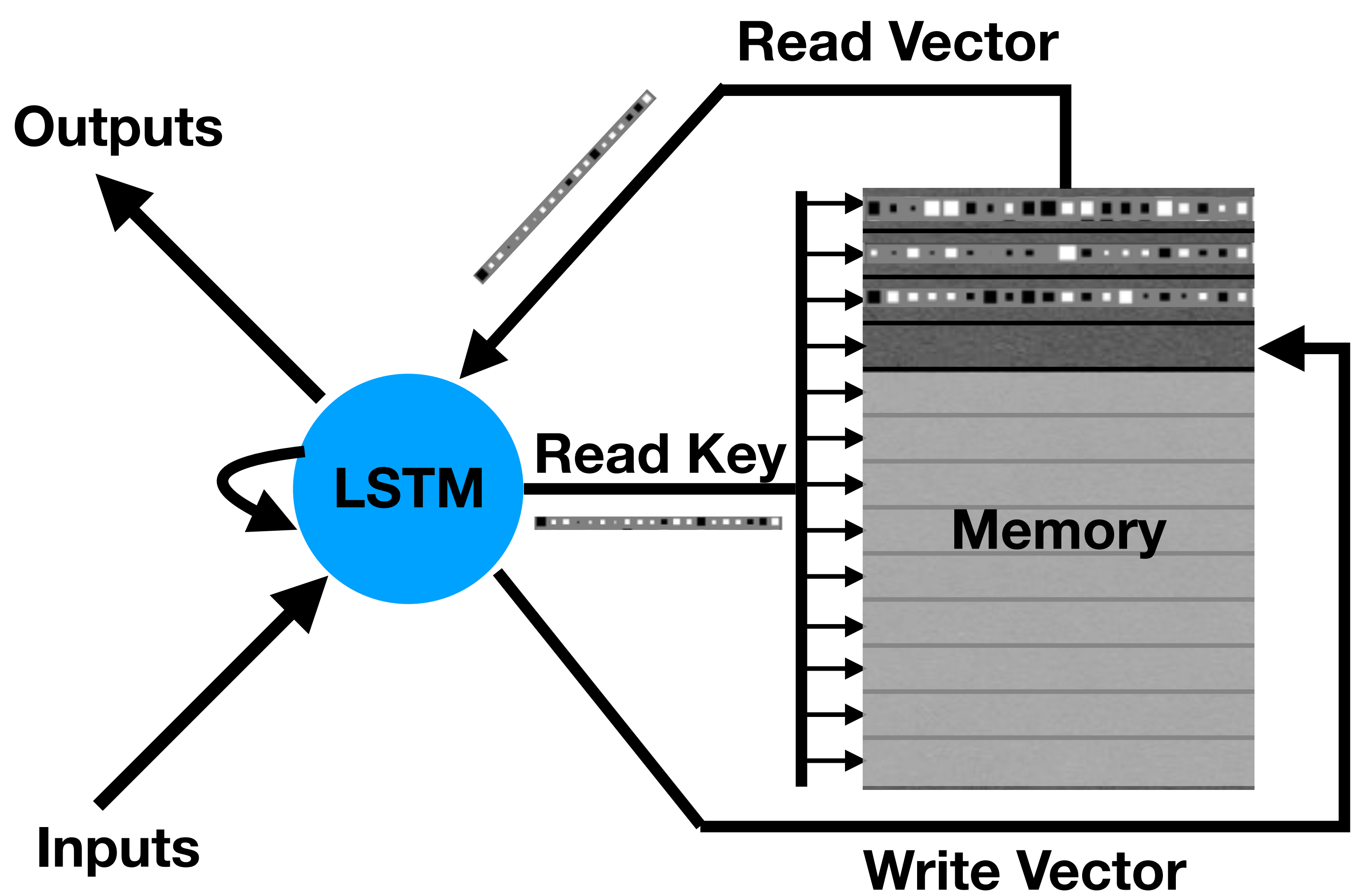
Greg Wayne
DeepMind Technologies Limited

# Talk Outline

- Neural Networks with External Memory
- Reinforcement Learning with Memory Systems
    - Some Limitations and Weaknesses Therein
- The MEmory, INference, and Reinforcement Learning Agent (MERLIN)
- Its Behavior on Interesting Tasks Characterized by Partial Observability
- Re The Programmer's Apprentice: Imitation-Learning for Complex Skills (Motor Skills)

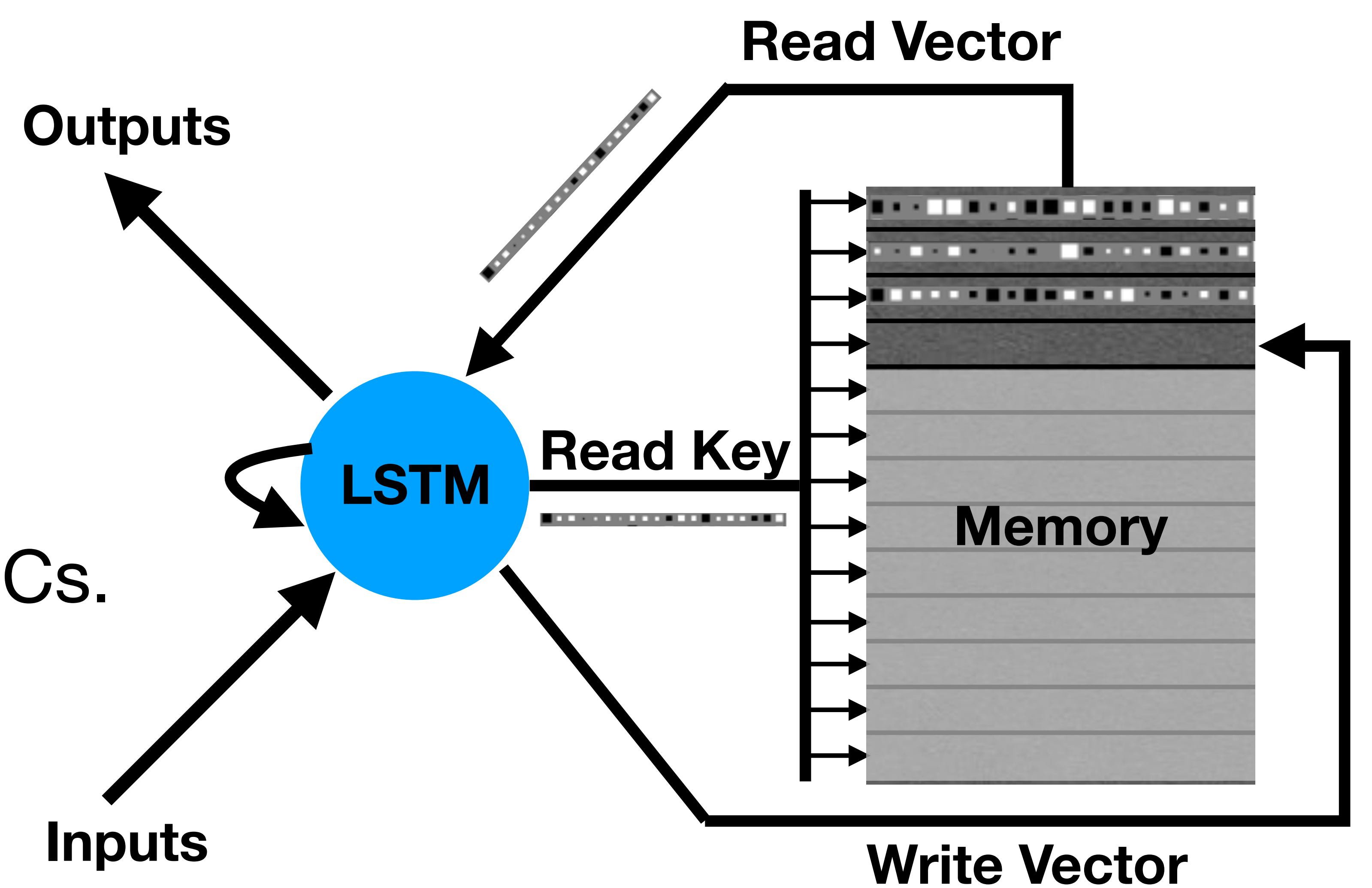
# Neural Turing Machines (NTMs) and Differentiable Neural Computers (DNCs)

- Trainable neural networks that can read and write to external memory.
- Can instantiate simple algorithms operating over simple data structures like lists and graphs.
- Have higher capacity memory than LSTMs (Hochreiter and Schmidhuber, 1997) alone.




(Graves, Wayne, Danihelka, arXiv 2014) (Graves, Wayne et al., Nature 2016)

# Neural Turing Machines (NTMs) and Differentiable Neural Computers (DNCs)




NTMs are a little more primitive / kludgy than DNCs.

(Graves, Wayne, Danihelka, arXiv 2014) (Graves, Wayne et al., Nature 2016)

# A Simple DNC

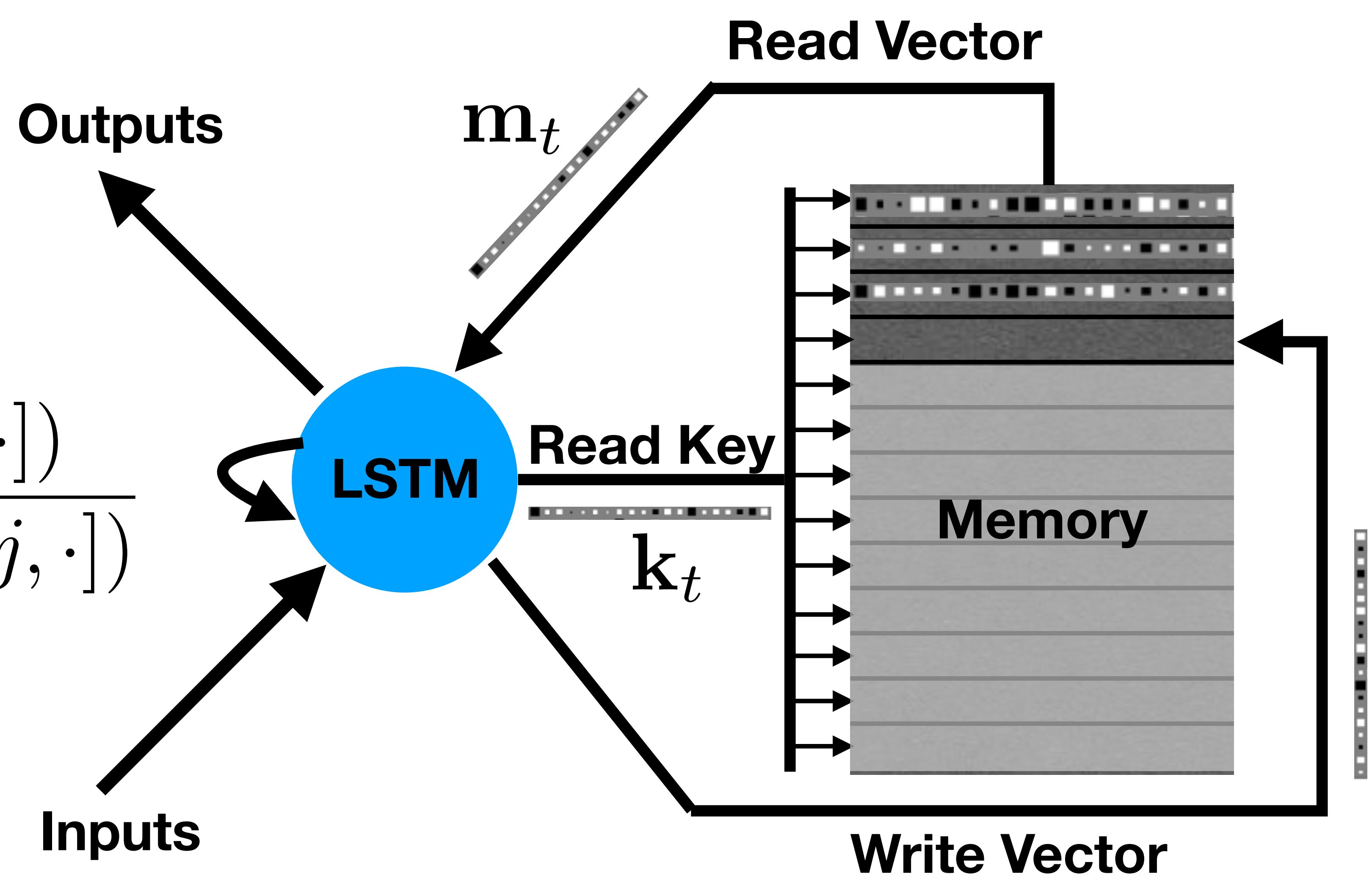
***Reading from Memory***

**Read "Attention Weighting"**

$$w_t[i] = \frac{\exp(\mathbf{k}_t \cdot M_t[i,\cdot])}{\sum_j \exp(\mathbf{k}_t \cdot M_t[j,\cdot])}$$

**Read Vector**

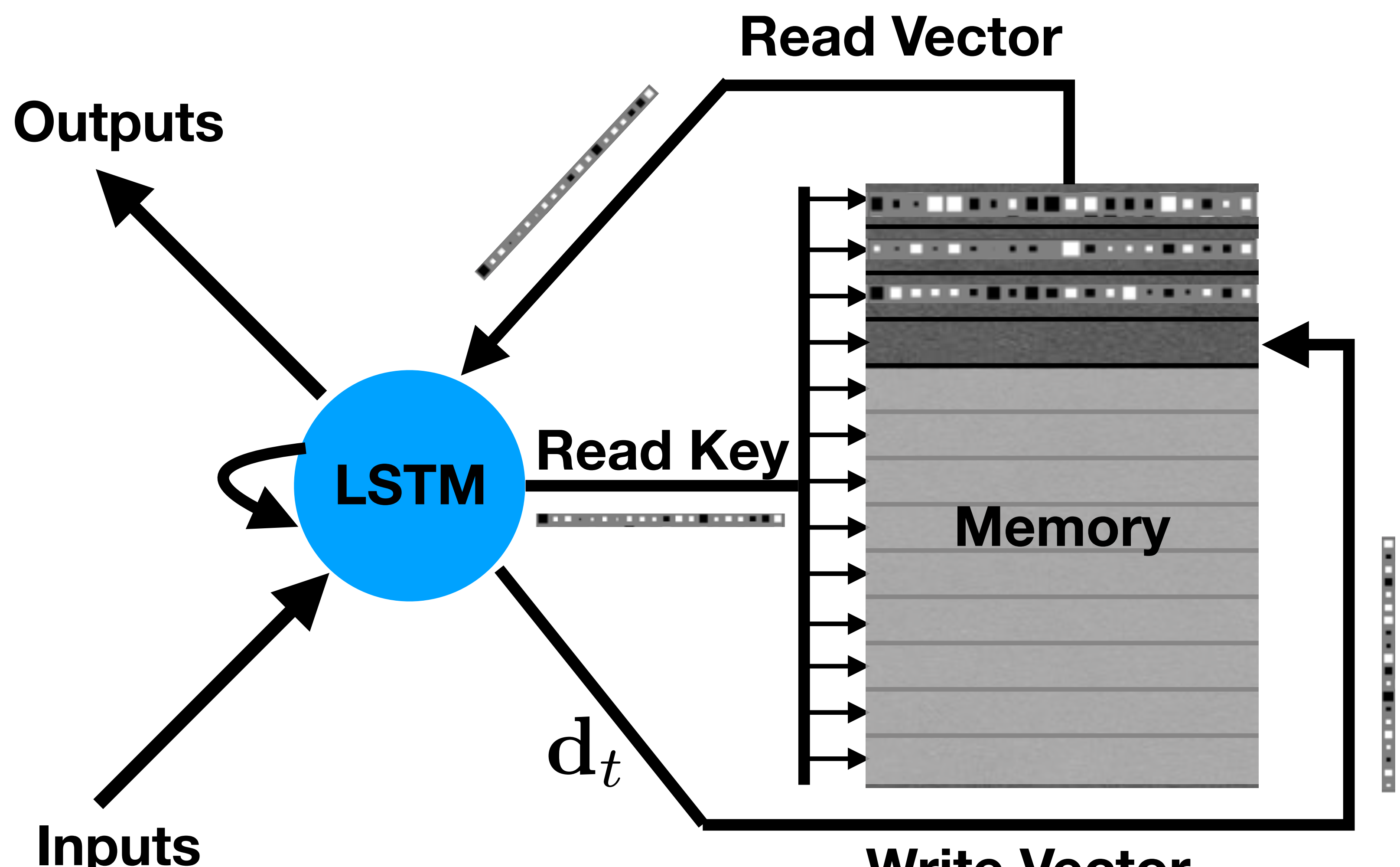
$$\mathbf{m}_t = \sum_i w_t[i] M_t[i,\cdot]$$



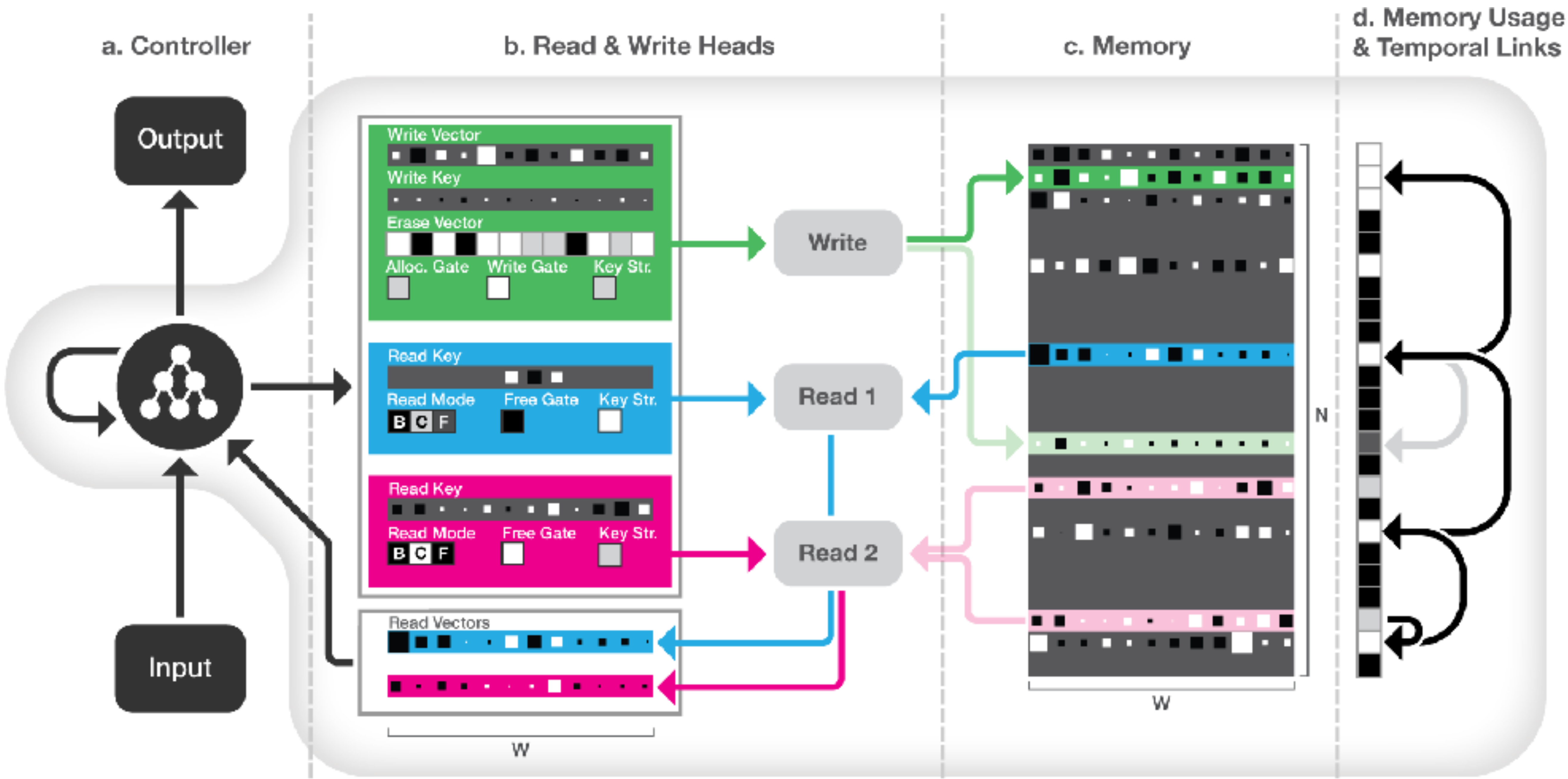
# A Simple DNC

***Writing to Memory***

$$M_{t+1}[t, \cdot] = \mathbf{d}_t$$

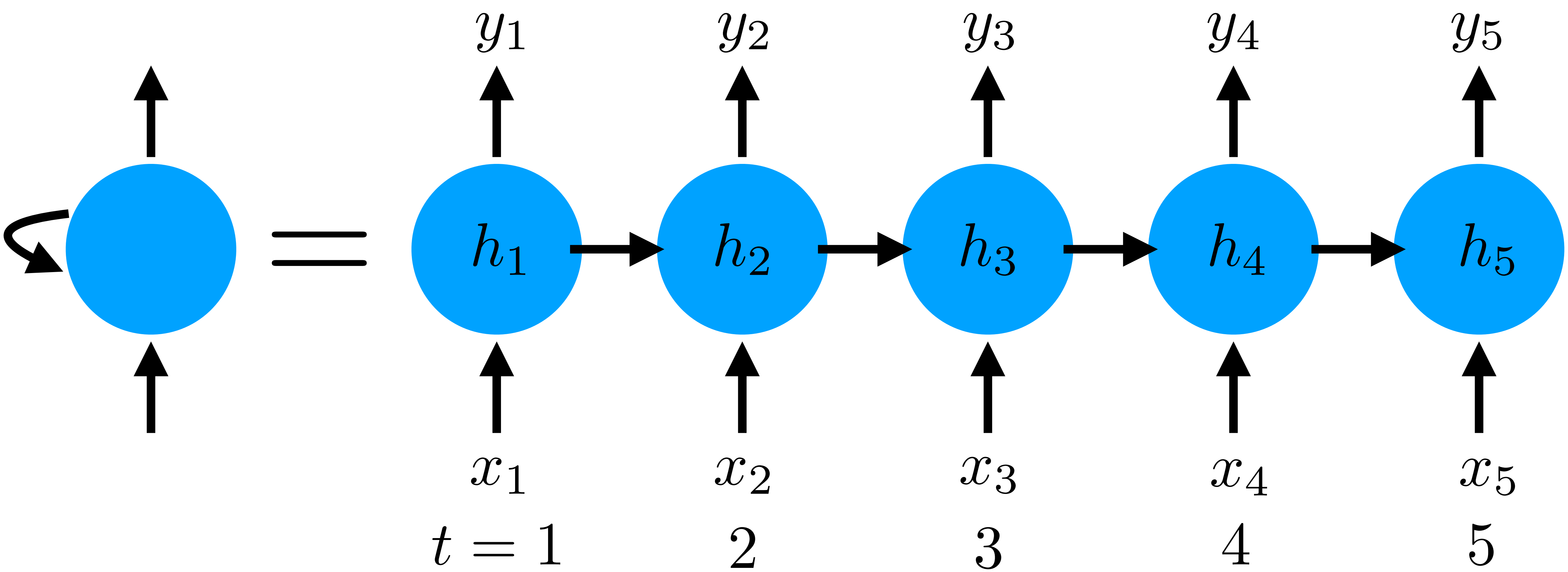


# A More Complicated DNC

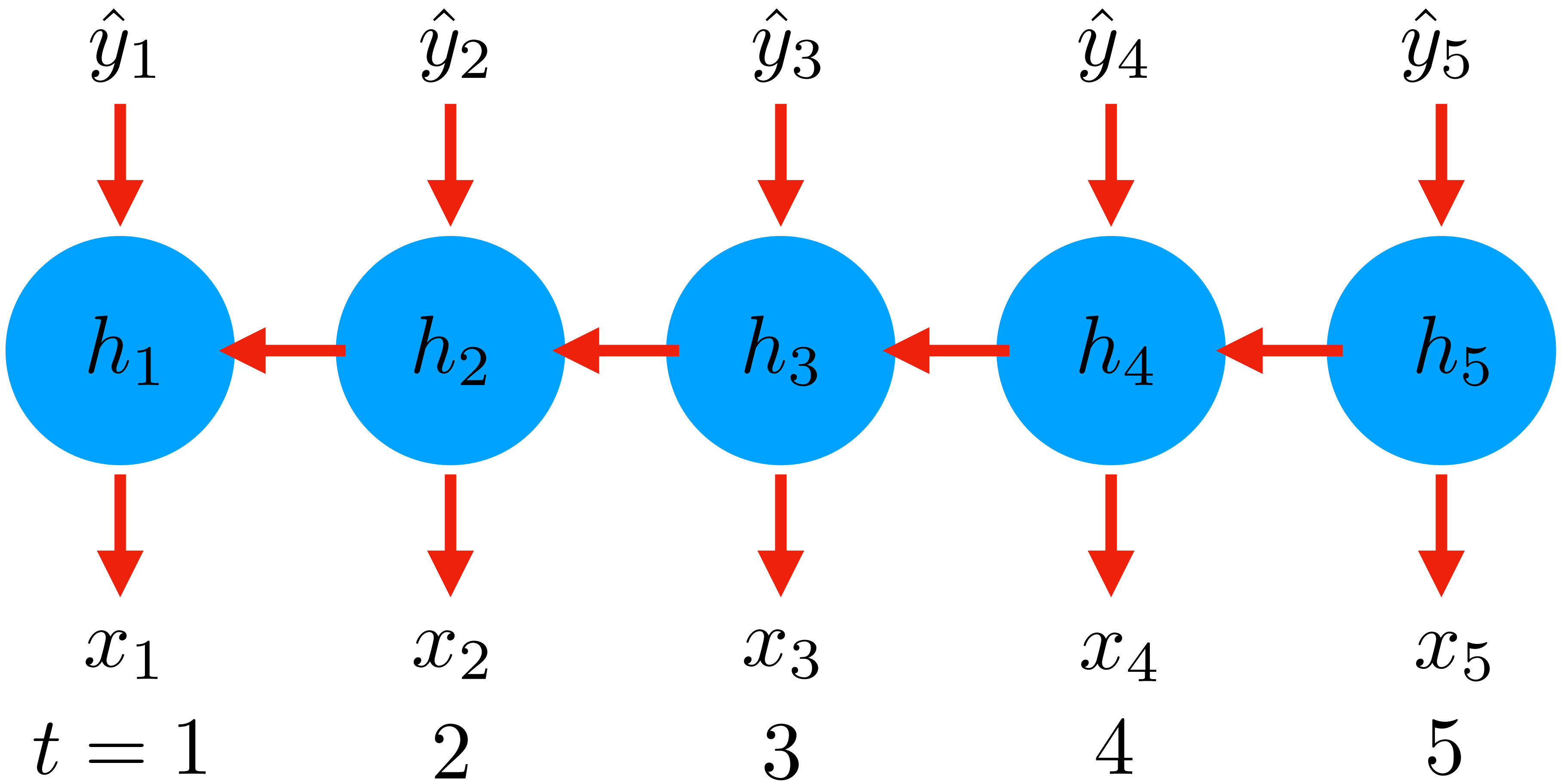



**Supports reading items based on the order they were written as well.**

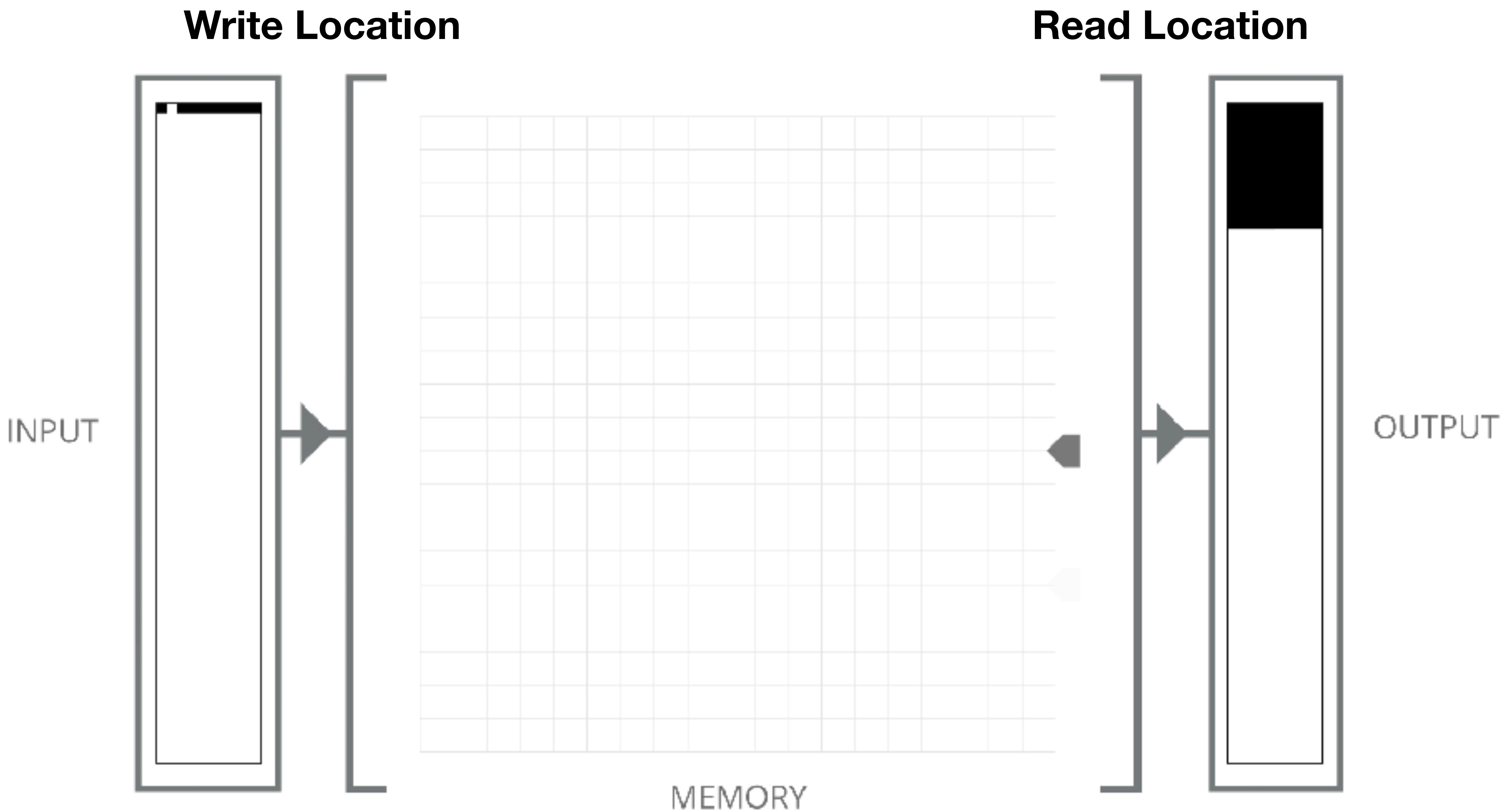
# Supervised Training of RNNs / DNCs



# Supervised Training of RNNs / DNCs



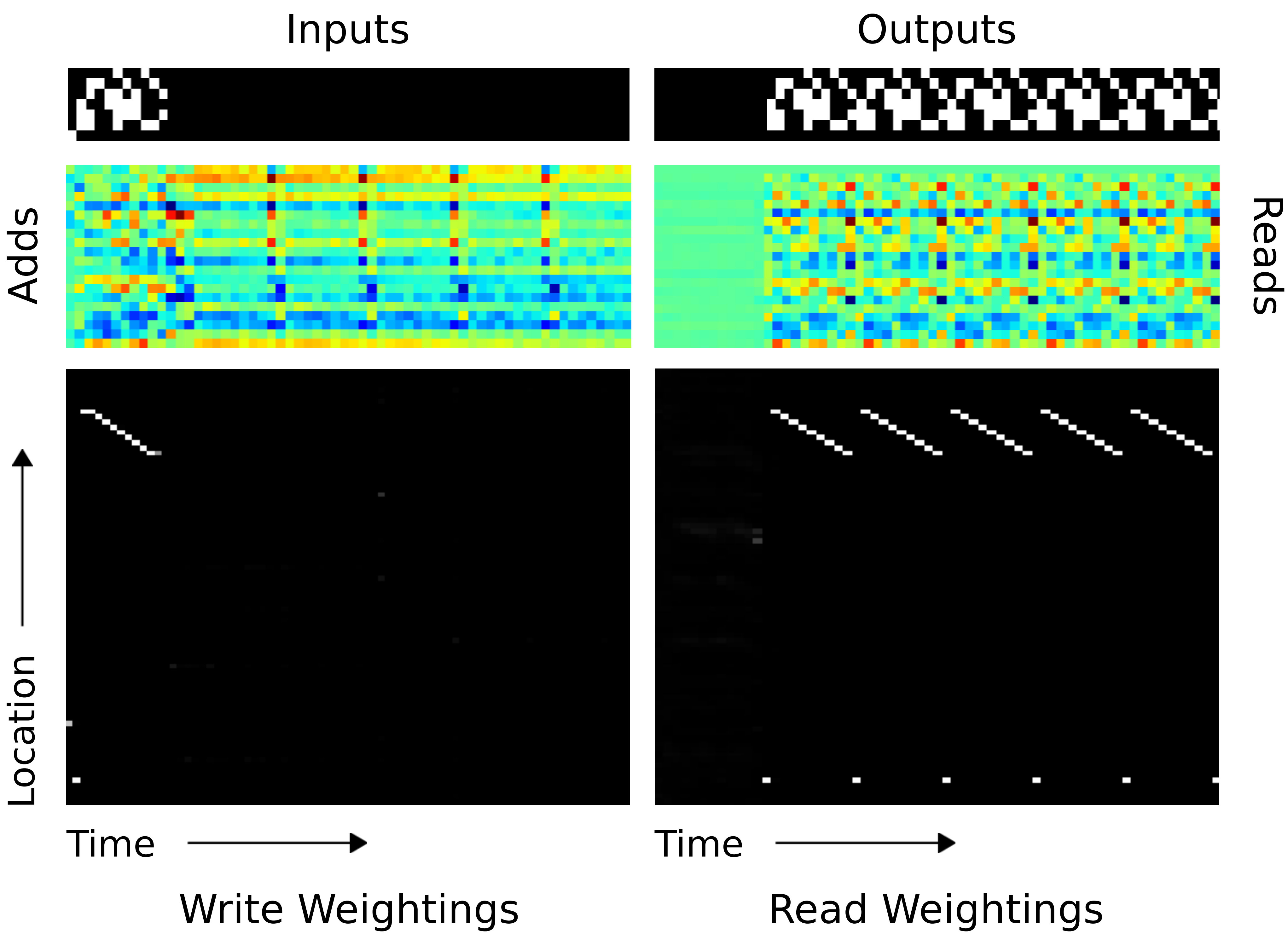

**Backpropagation through Time**




**Example of an NTM (circa 2014) on a looped copy problem.**

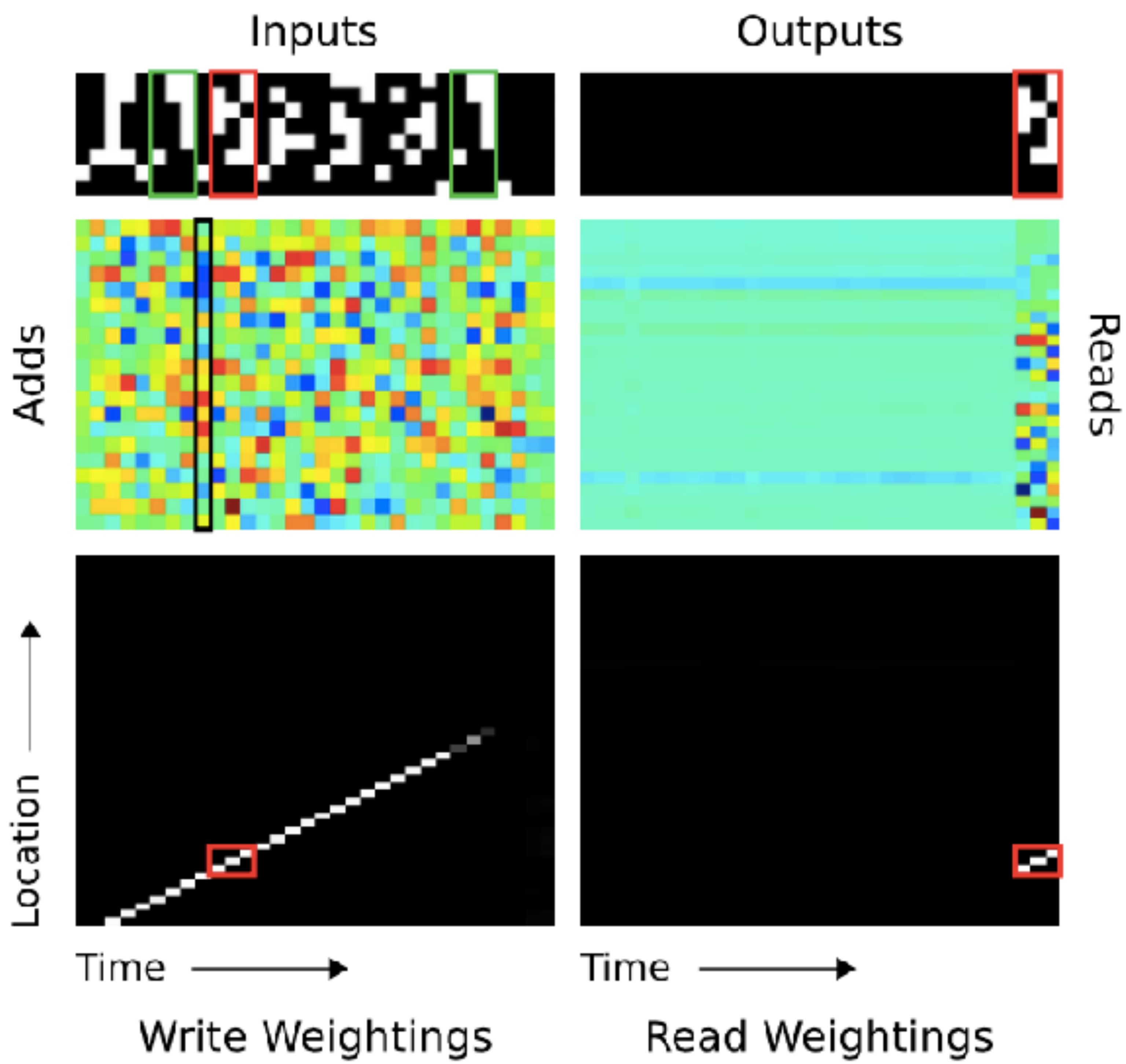
**(LSTMs struggle on this.)**

# Looped Copy



Inputs
Outputs
Adds
Reads
Location
Time
Time
Write Weightings
Read Weightings

# Associative List

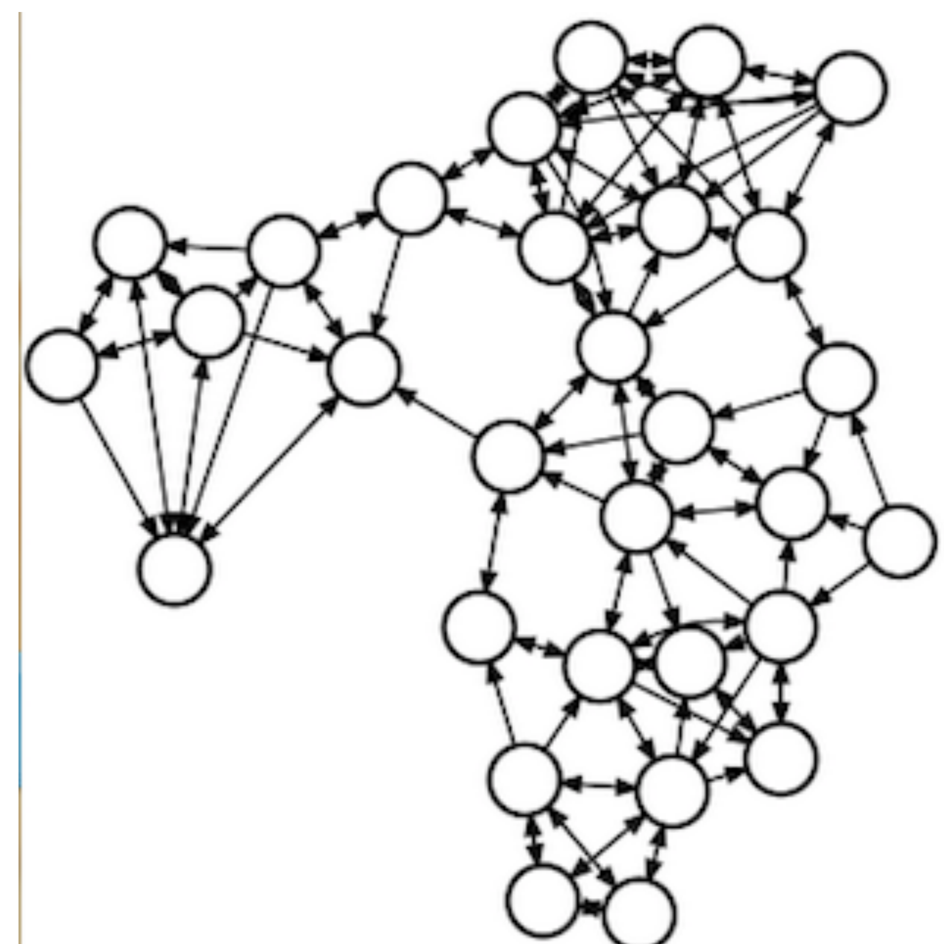


# Learning Complex Data Structures (Random Graphs)

Time

1. Source Node Label - Edge Label - Destination Node Label
2. Source Node Label - Edge Label - Destination Node Label
3. Source Node Label - Edge Label - Destination Node Label

Labels:
Digital representations of numbers between 0 and 999.



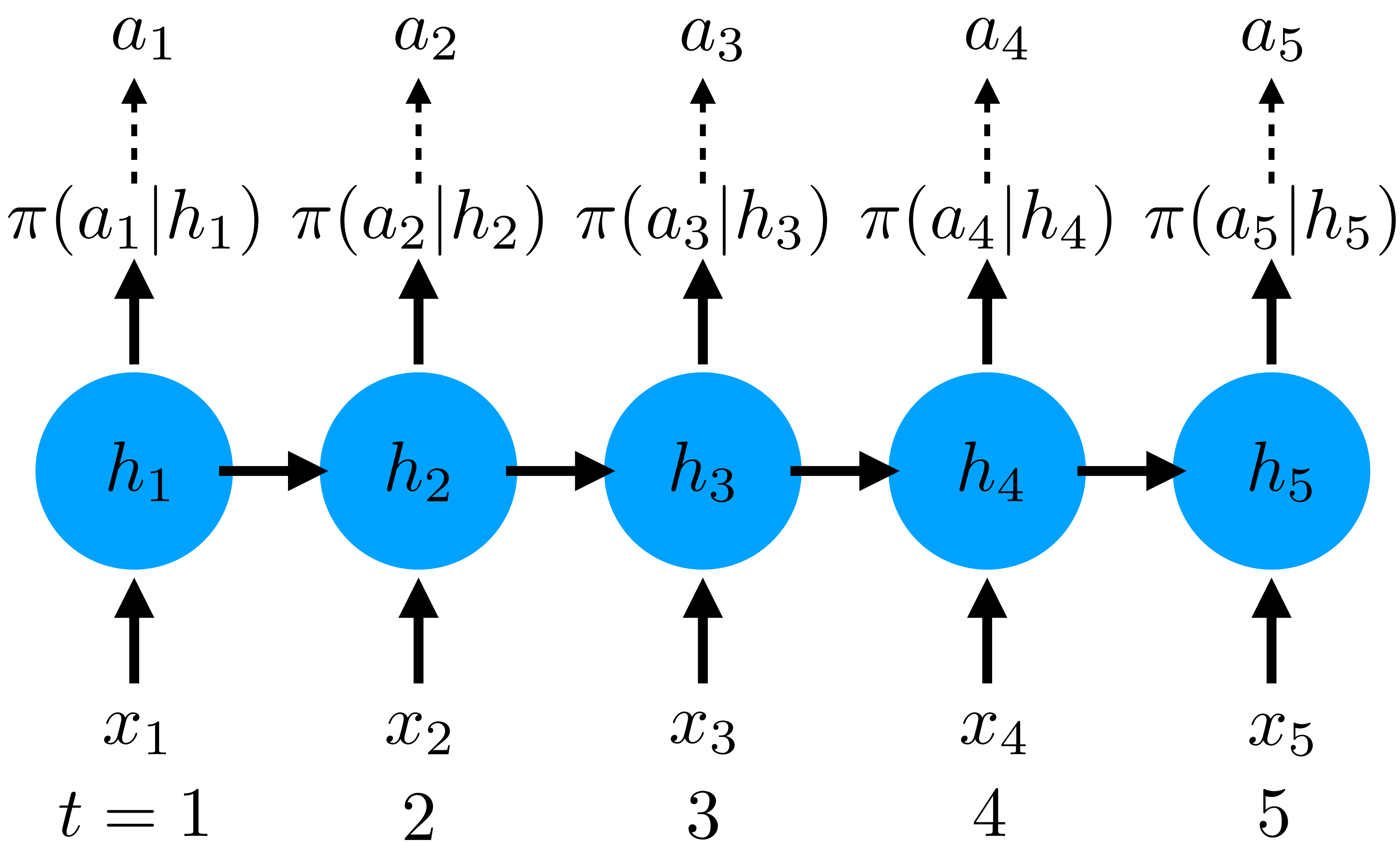
# Tube Traversal Problem


**Content-based lookup**

**Queries of form: From Victoria Station go North on Victoria Line, North on Victoria Line, East on Central Line, ….**

# Reinforcement Learning

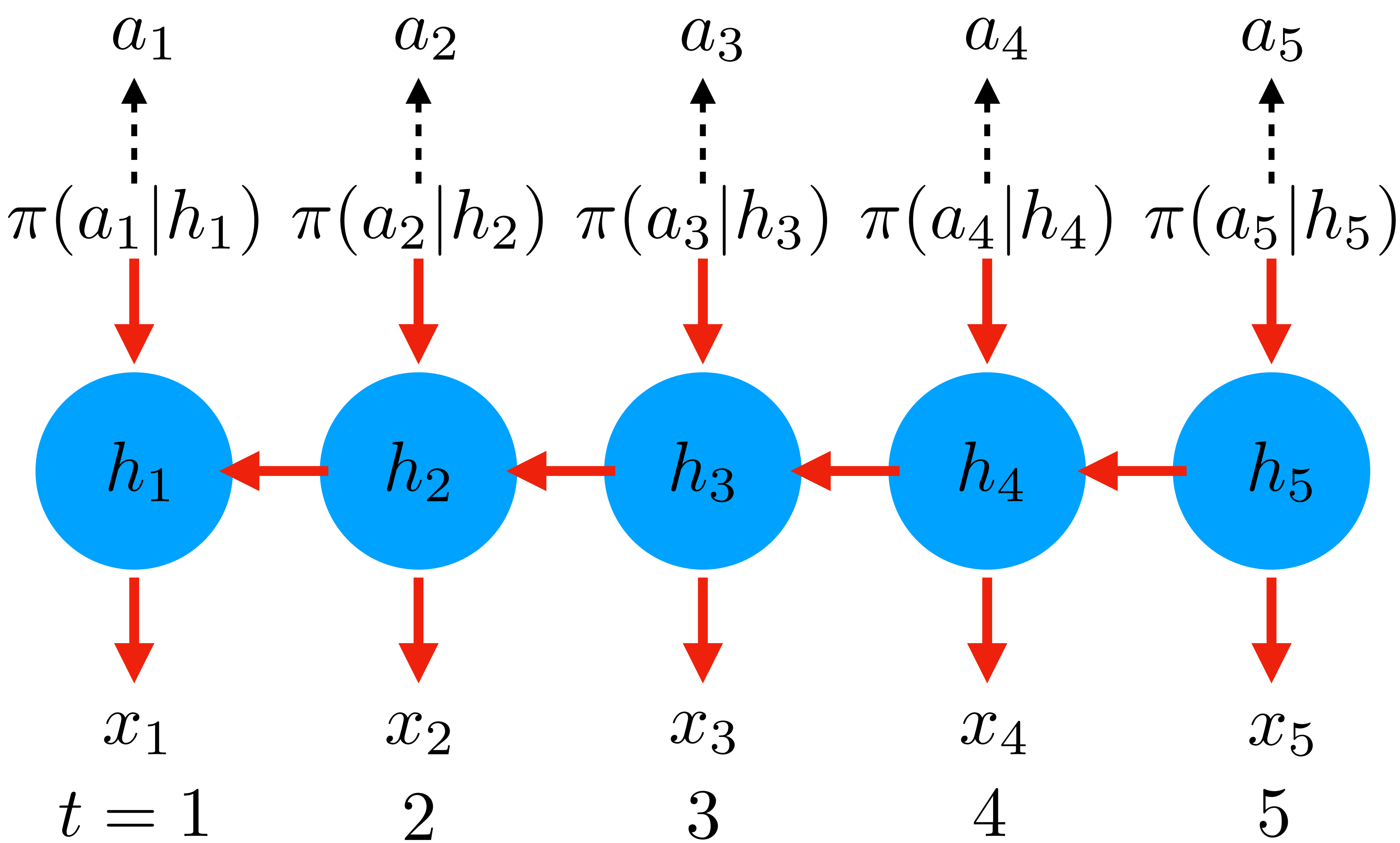
**Probability of each action**




**Maximize expected “return”** $R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_T$

# Reinforcement Learning

**Probability of each action**




**Policy Gradient**

$$\Delta\theta \propto \sum_{t=0}^{T} R_t \nabla_\theta \log \pi(a_t|h_t)$$

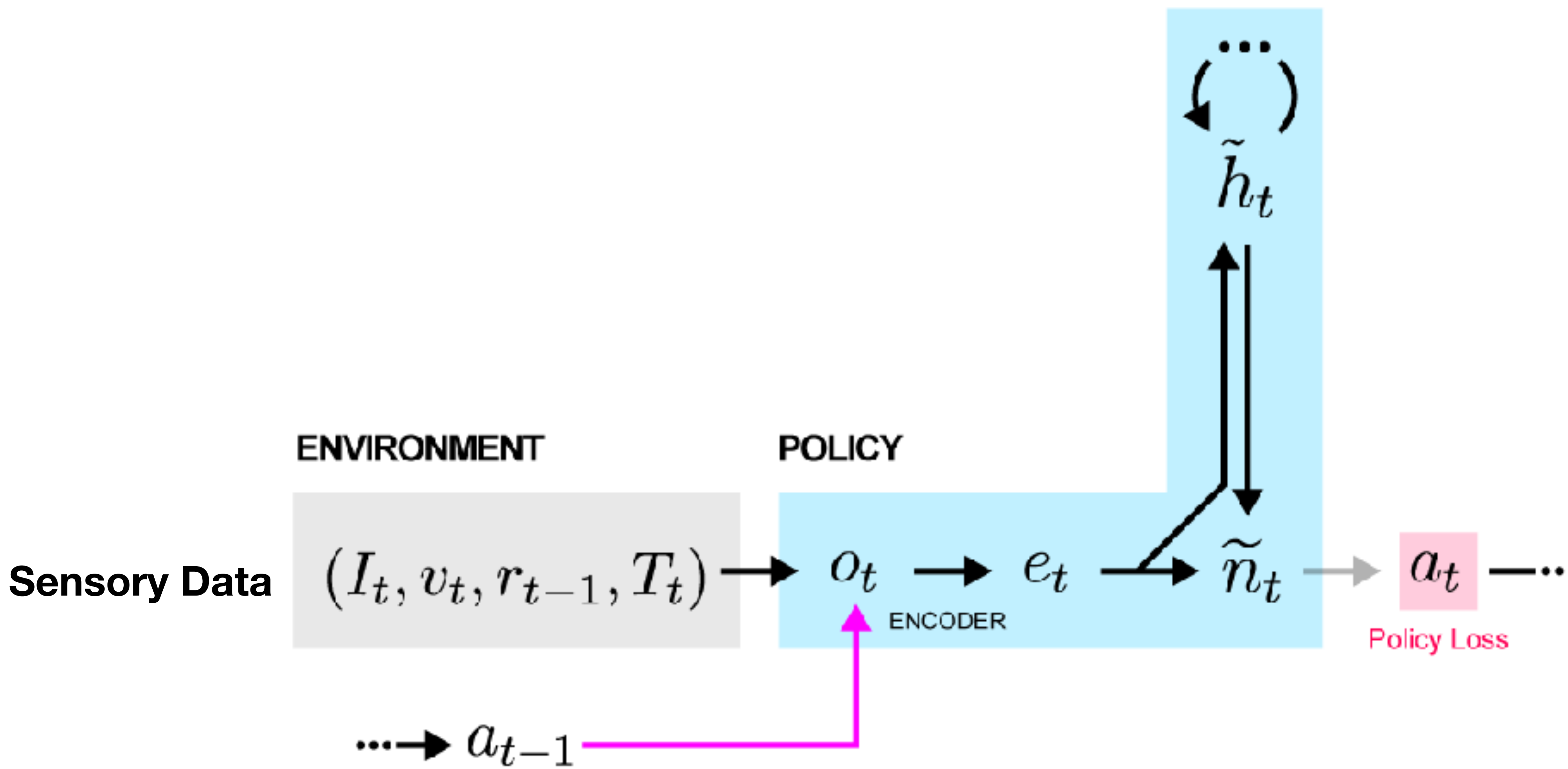
# A Simple Memory-Based RL Challenge (Memory Game)

**Partially observed,**
**need to know “what is where”**

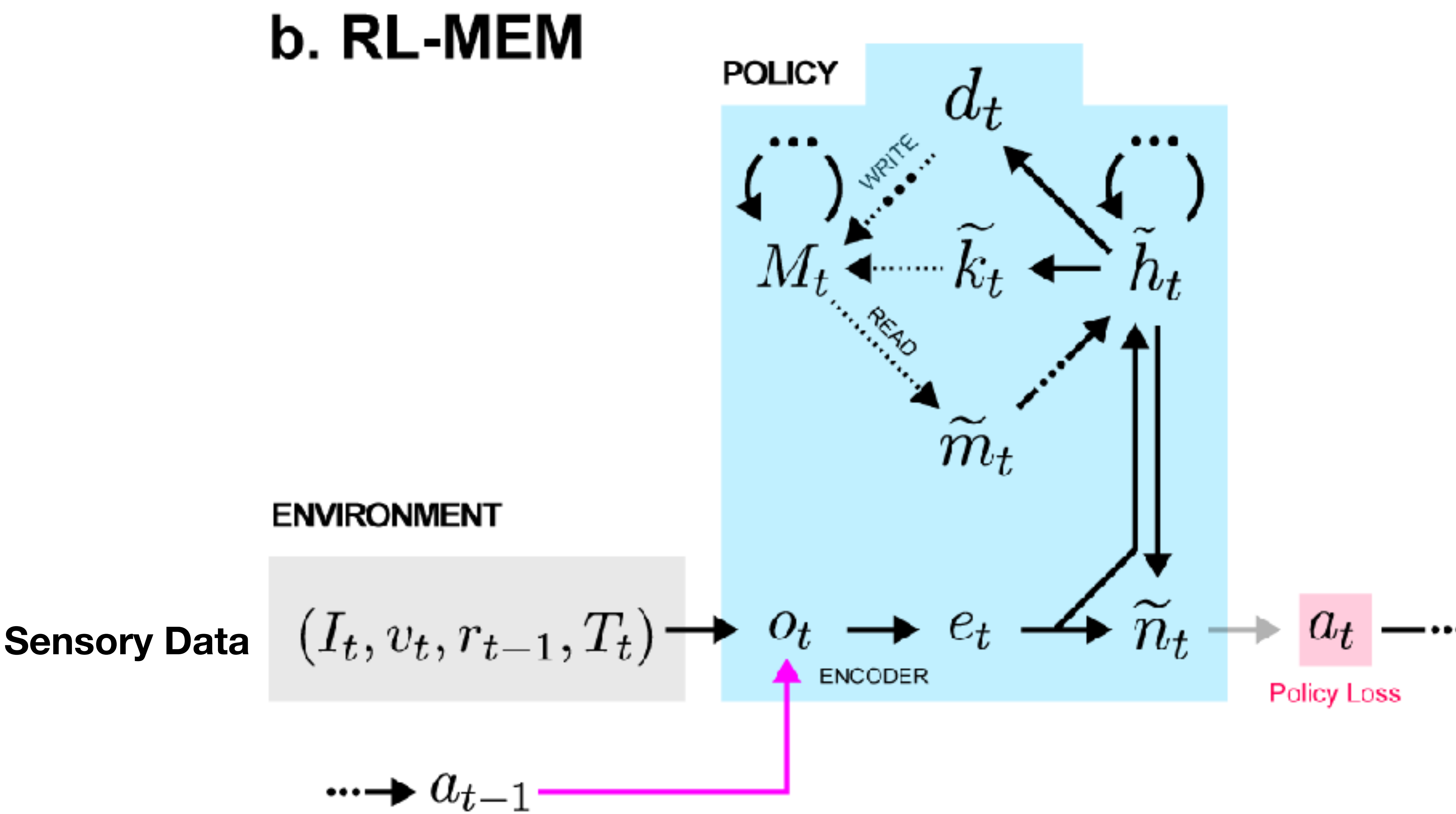
**(a kind of state estimation)**

# Control Architecture “a”

## a. RL-LSTM



# Control Architecture "b"

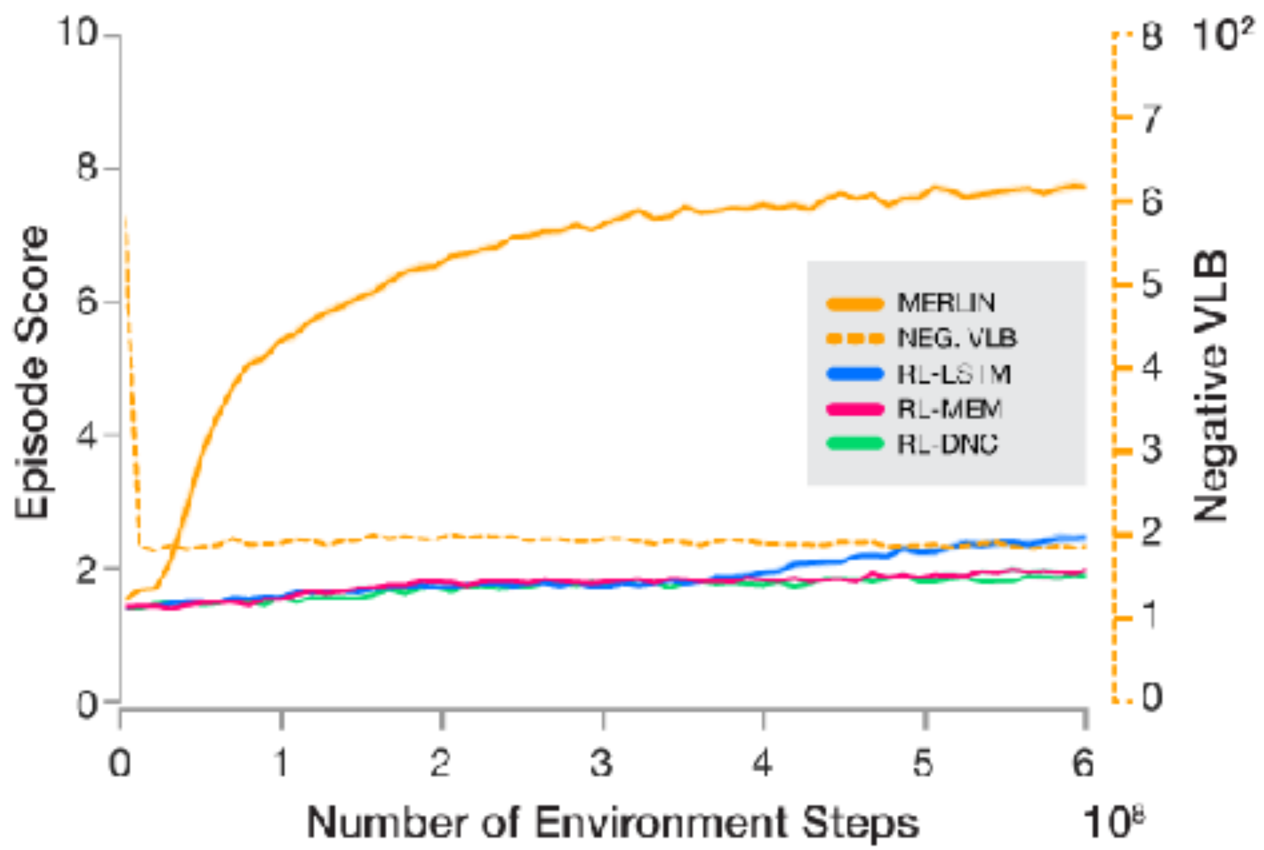


# Remarkably

- Both these models fail to solve this very simple task. Why?

- RL-LSTM: lacks **capacity for storing** and **search mechanism for recalling** positions and images.

- RL-MEM: has capacity and search mechanism, but (we argue) policy gradient RL does not provide the right **objective function** to determine what is stored in memory.



# Policy Gradient

- Policy gradient: relatively high variance (noisy) estimator of performance improvement direction.
    - trajectory dependent
- Intuitively, feels wrong: memory for, say, the memory game should not only develop through gradients from **success or failure** due to **trial and error** action choices (“motor twitches”).

# Prediction, Compression, and State Estimation

Classical Control: Separate State Estimation from Control Design

Our strategy: instead build perceptual representations and memories by unsupervised predictive modeling / state estimation

# Unsupervised Predictive Memory in a Goal-Directed Agent



Computer Science > Learning

## Unsupervised Predictive Memory in a Goal-Directed Agent

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Rezende, David Saxton, Adam Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matt Botvinick, Demis Hassabis, Timothy Lillicrap

*(Submitted on 28 Mar 2018)*

Animals execute goal-directed behaviours despite the limited range and scope of their sensors. To cope, they explore environments and store memories maintaining estimates of important information that is not presently available. Recently, progress has been made with artificial intelligence (AI) agents that learn to perform tasks from sensory input, even at a human level, by merging reinforcement learning (RL) algorithms with deep neural networks, and the excitement surrounding these results has led to the pursuit of related ideas as explanations of non-human animal learning. However, we demonstrate that contemporary RL algorithms struggle to solve simple tasks when enough information is concealed from the sensors of the agent, a property called "partial observability". An obvious requirement for handling partially observed tasks is access to extensive memory, but we show memory is not enough; it is critical that the right information be stored in the right format. We develop a model, the Memory, RL, and Inference Network (MERLIN), in which memory formation is guided by a process of predictive modeling. MERLIN facilitates the solution of tasks in 3D virtual reality environments for which partial observability is severe and memories must be maintained over long durations. Our model demonstrates a single learning agent architecture that can solve canonical behavioural tasks in psychology and neurobiology without strong simplifying assumptions about the dimensionality of sensory input or the duration of experiences.

Subjects: Learning (cs.LG); Machine Learning (stat.ML)

Cite as: arXiv:1803.10760 [cs.LG]
(or arXiv:1803.10760v1 [cs.LG] for this version)

**Download:**

- PDF
- Other formats

(license)

Current browse context:
cs.LG
< prev | next >
new | recent | 1803

Change to browse by:
cs
stat
stat.ML

References & Citations
- NASA ADS

Bookmark (what is this?)

# Unsupervised Predictive Memory in a Goal-Directed Agent



arXiv.org > cs > arXiv:1803.10760

Search or Article ID | All fields
(Help | Advanced search)

Computer Science > Learning

## Unsupervised Predictive Memory in a Goal-Directed Agent

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Rezende, David Saxton, Adam Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matt Botvinick, Demis Hassabis, Timothy Lillicrap

(Submitted on 28 Mar 2018)

Animals execute goal-directed behaviours despite the limited range and scope of their sensors. To cope, they explore environments and store memories maintaining estimates of important information that is not presently available. Recently, progress has been made with artificial intelligence (AI) agents that learn to perform tasks from sensory input, even at a human level, by merging reinforcement learning (RL) algorithms with deep neural networks, and the excitement surrounding these results has led to the pursuit of related ideas as explanations of non-human animal learning. However, we demonstrate that contemporary RL algorithms struggle to solve simple tasks when enough information is concealed from the sensors of the agent, a property called "partial observability". An obvious requirement for handling partially observed tasks is access to extensive memory, but we show memory is not enough; it is critical that the right information be stored in the right format. We develop a model, the Memory, RL, and Inference Network (MERLIN), in which memory formation is guided by a process of predictive modeling. MERLIN facilitates the solution of tasks in 3D virtual reality environments for which partial observability is severe and memories must be maintained over long durations. Our model demonstrates a sin… that can … tasks in psychology and neurobiology without strong simplifying assumptions about t… …ut or th…

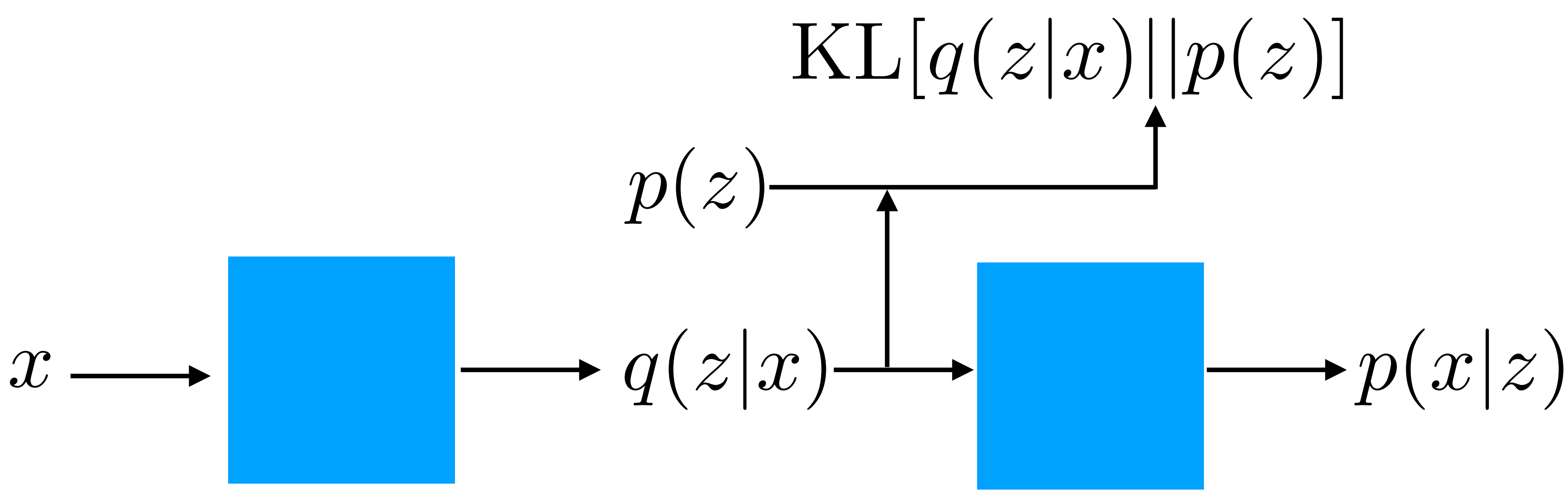
Subjects: Learning (cs.LG); Machine Learning (stat.ML)
Cite as: arXiv:1803.10760 [cs.LG]
(or arXiv:1803.10760v1 [cs.LG] for this version)




# Unsupervised Learning

Modern framework for unsupervised learning:

Variational Autoencoders (Kingma and Welling, 2013)
(Rezende and Mohamed, 2014)




$$q(z|x) = \mathcal{N}(\mu(x), \Sigma(x))$$

$$\log p(x) \geq \mathbb{E}_{q(z|x)}\Big[\log p(x|z) - \mathrm{KL}[q(z|x)||p(z)]\Big]$$

# Unsupervised Predictive Modeling

Sequential Variational Autoencoders or Variational RNNs (Gregor, 2015), (Chung, 2015).

**State variable (compressed representation)** $z_t$

**Prior** $p(z_t|z_1, a_1, z_2, a_2, \ldots, z_{t-1}, a_{t-1})$

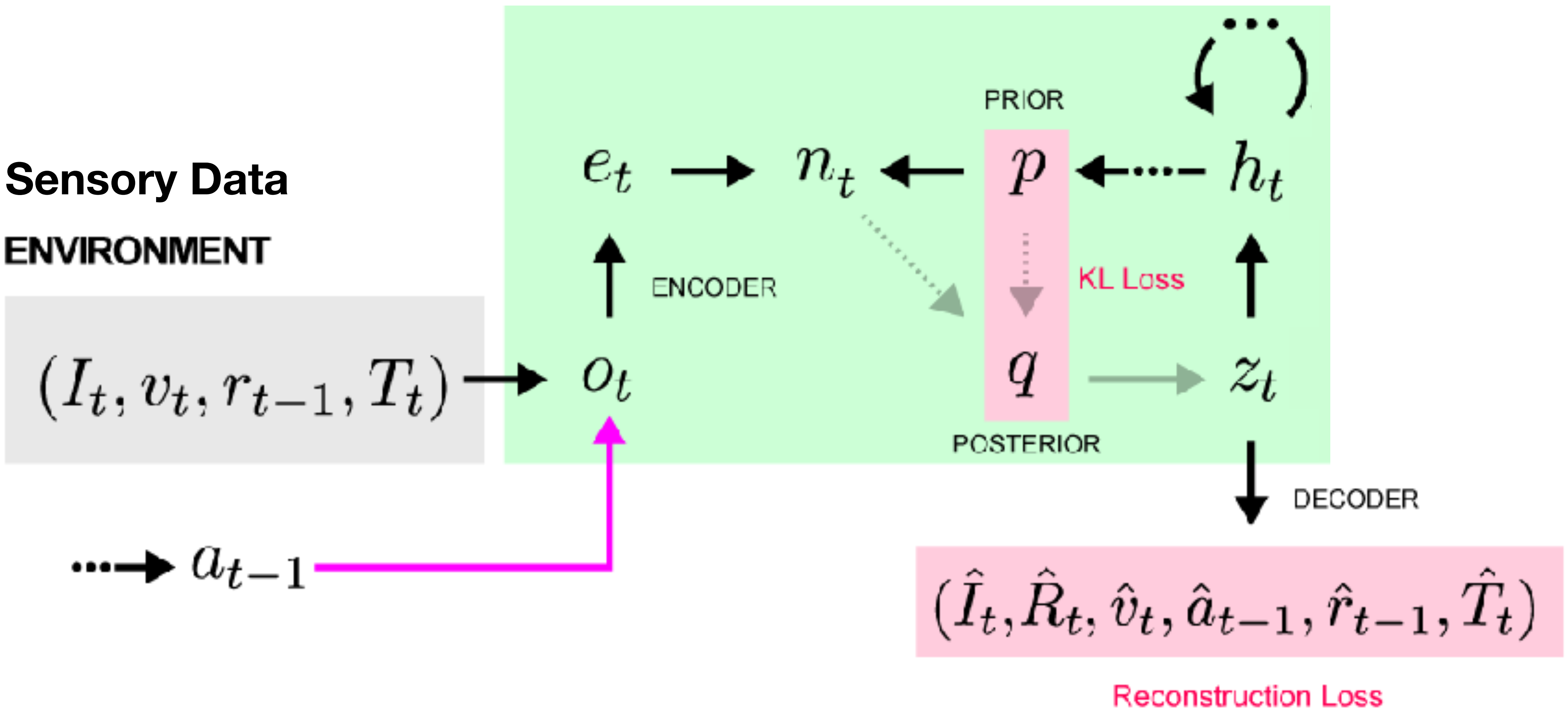
**Posterior** $q(z_t|z_1, a_1, z_2, a_2, \ldots, z_{t-1}, a_{t-1}, x_t)$

**Variational Lower Bound (VLB)** $\log p(x_t|z_t) - \mathrm{KL}[q||p]$

**Prior is trained to predict posterior.**

**Posterior is trained to make minimal deviation from prior whilst propagating information about observation.**

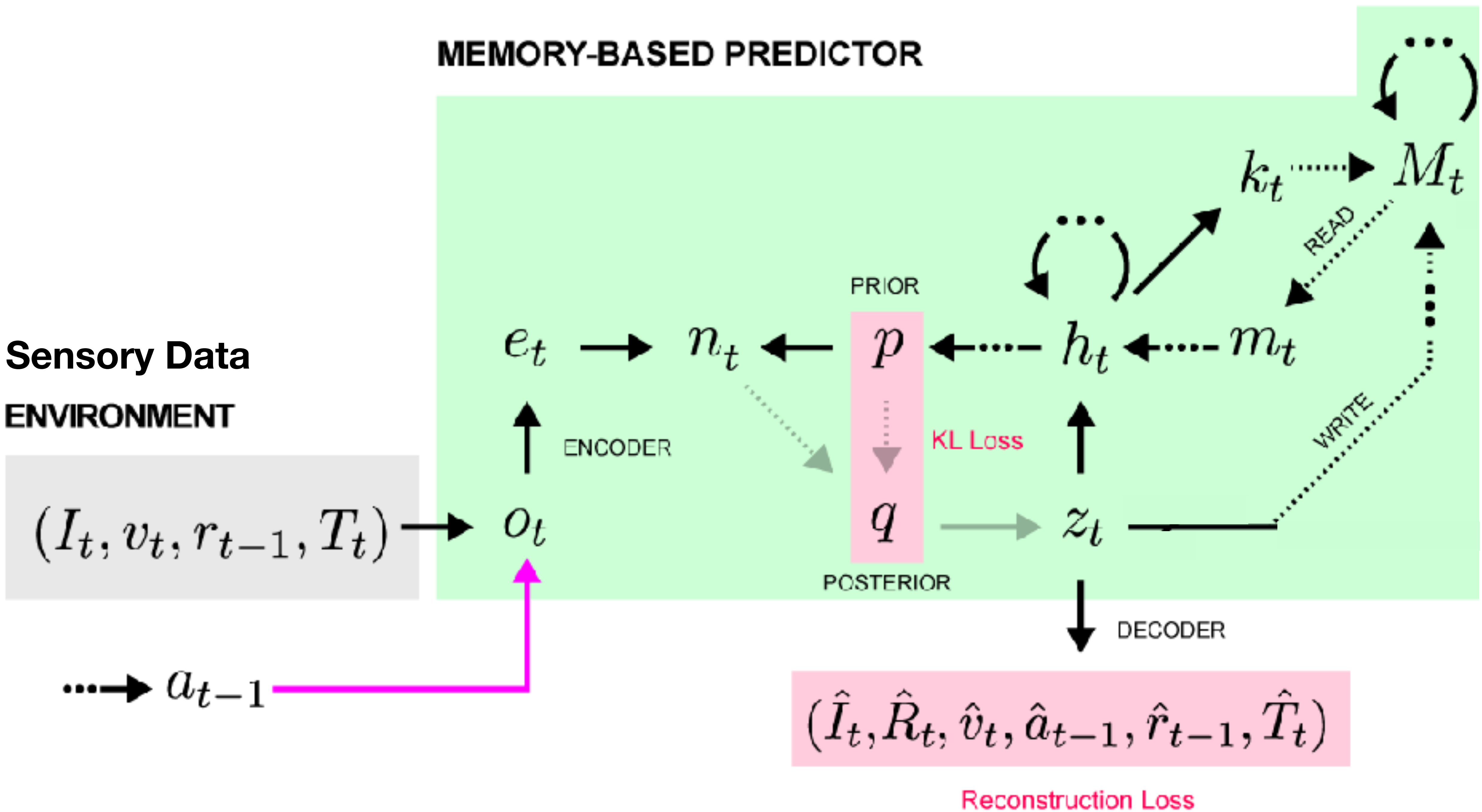
# What the Mechanism Looks Like




**Two Modes:**

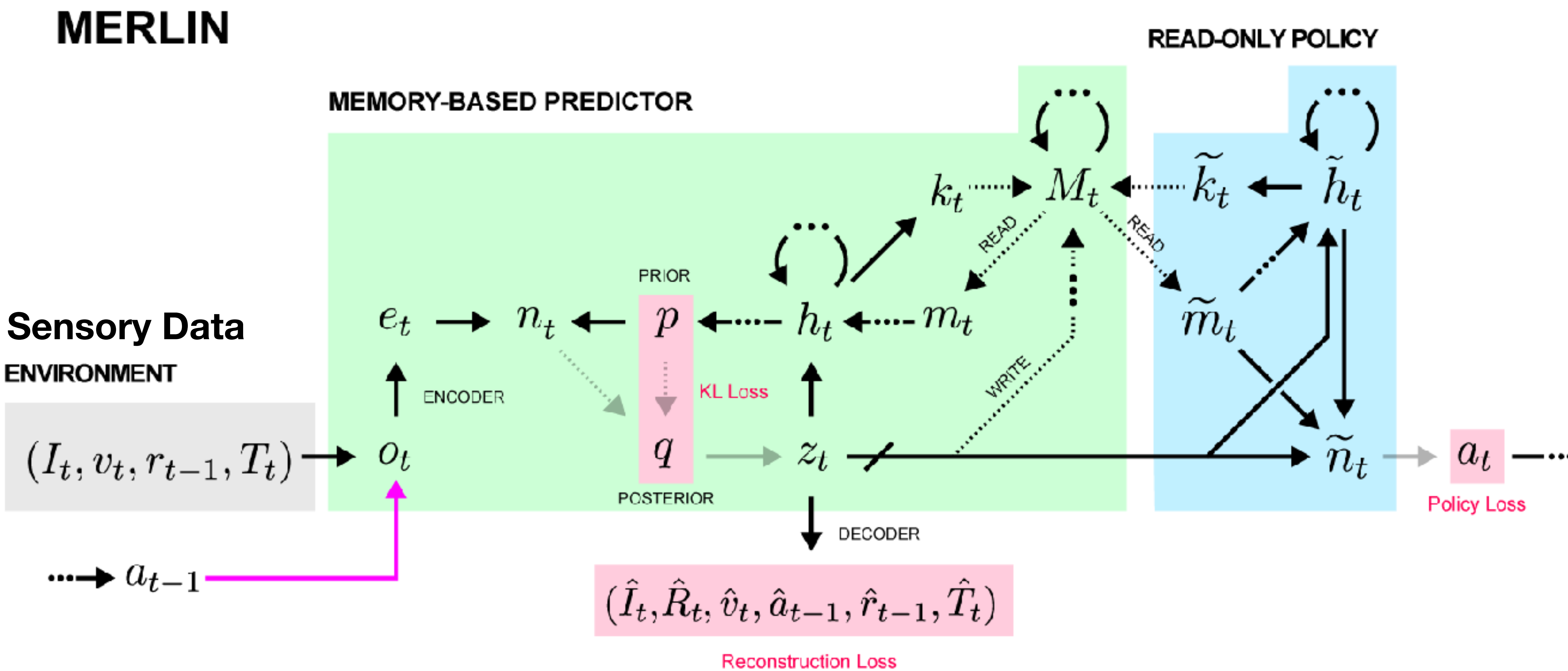
1) prediction
2) inference / state estimation

# Memory-Based Predictor

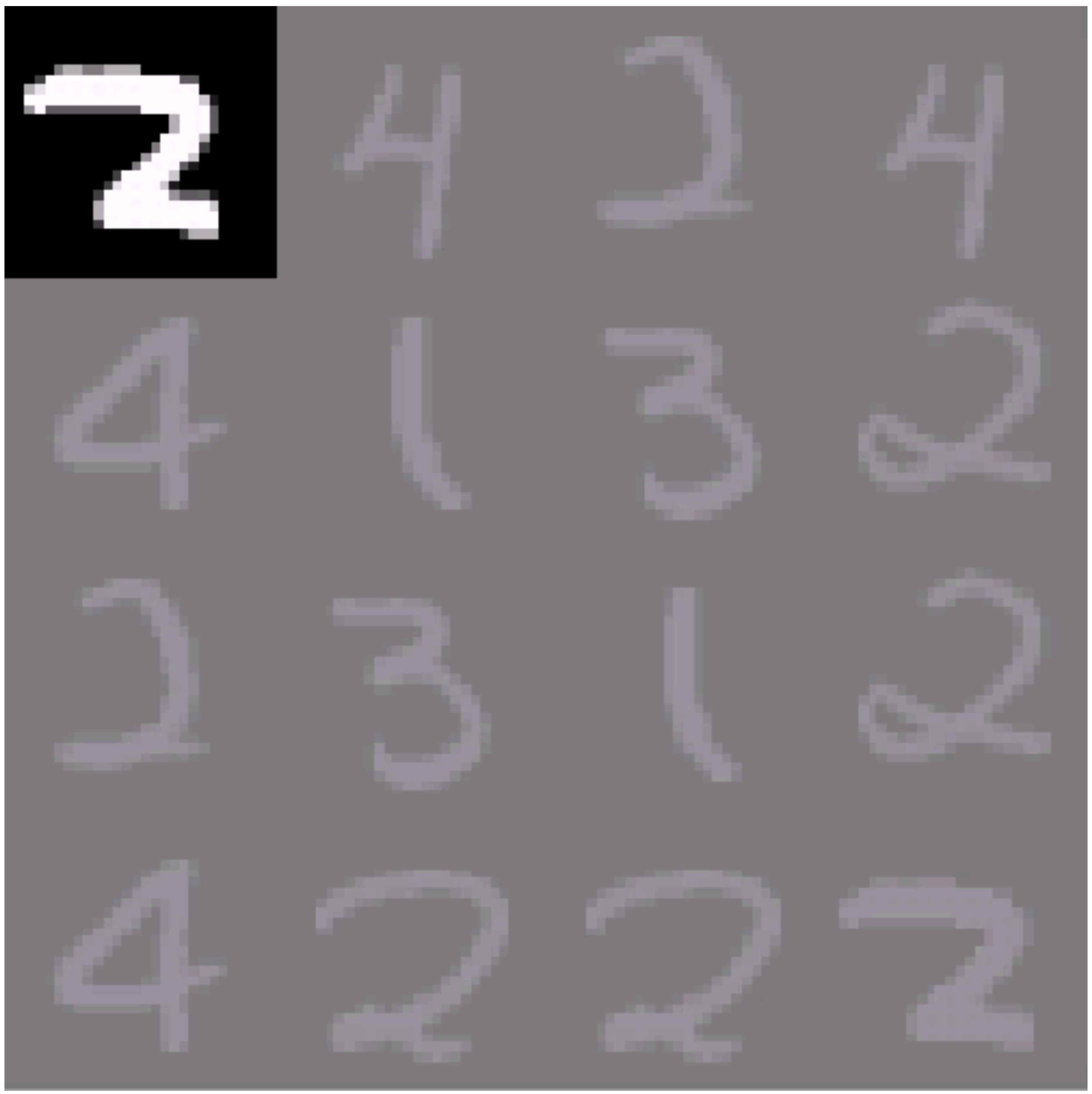


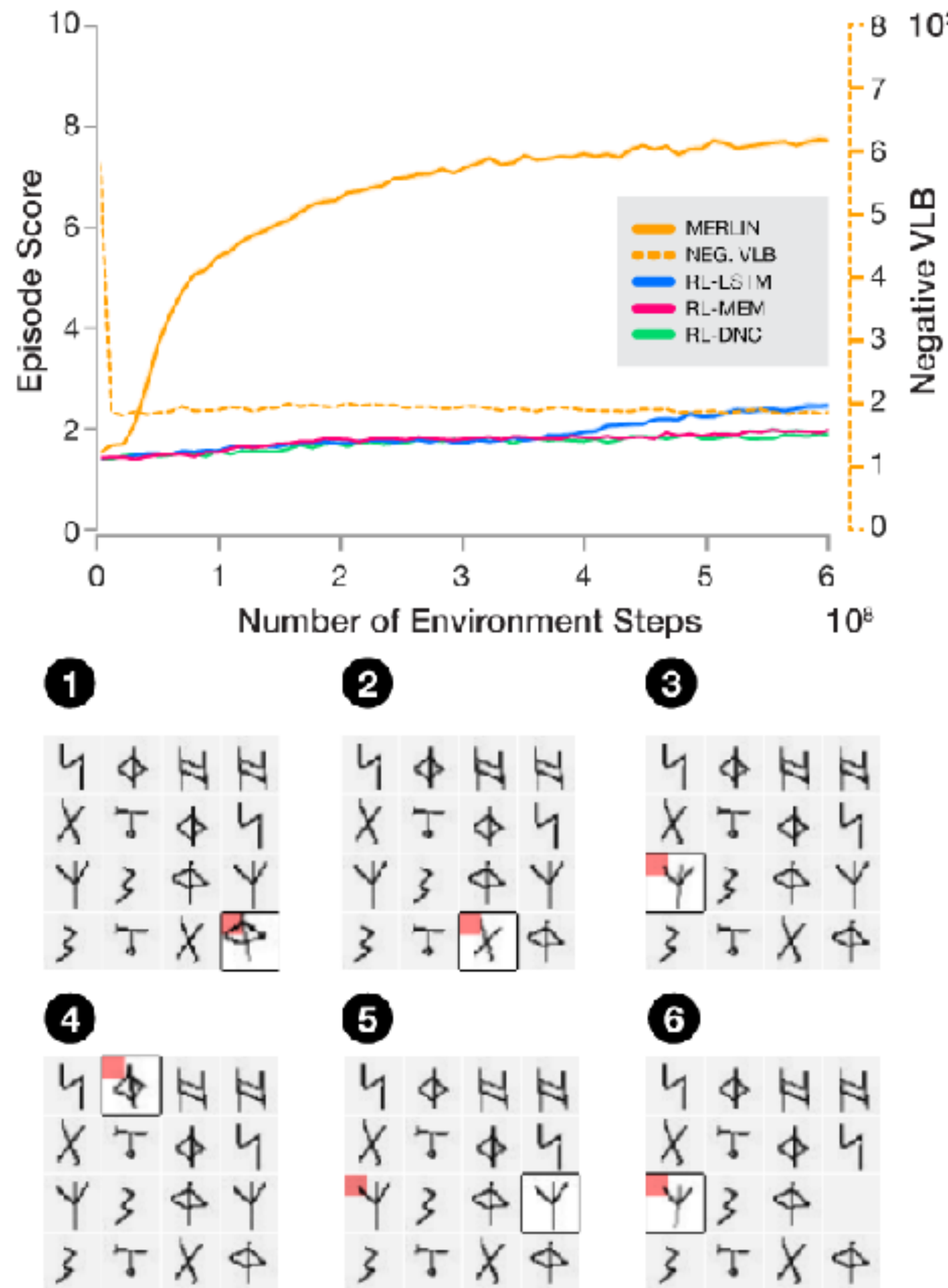

**Two-fold advantage: state variables are good representations; also, memory helps prediction.**

# MERLIN

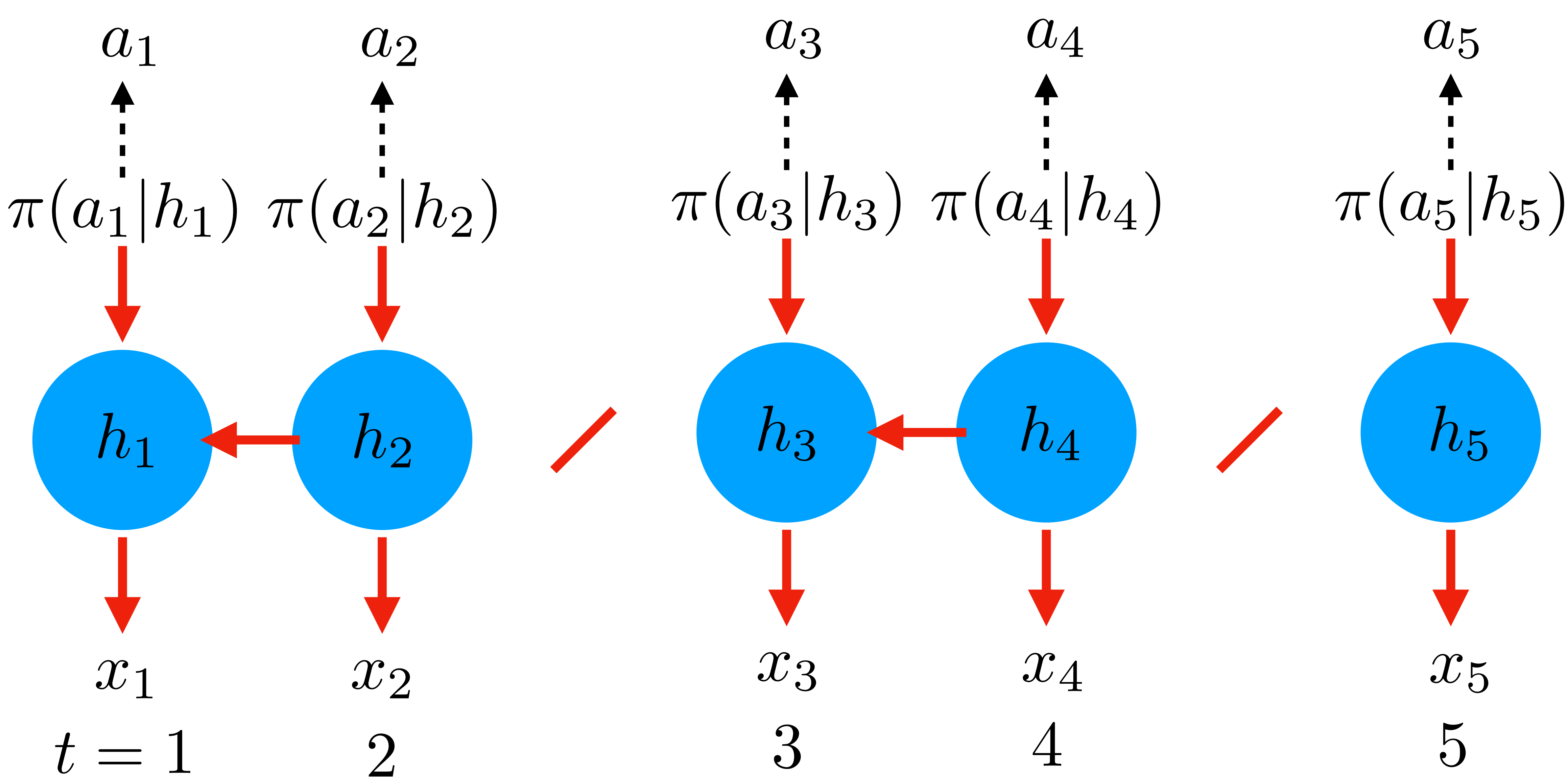

MERLIN
READ-ONLY POLICY
MEMORY-BASED PREDICTOR
Sensory Data
ENVIRONMENT
$(I_t, v_t, r_{t-1}, T_t)$
$o_t$
ENCODER
$e_t$
$n_t$
PRIOR
$p$
KL Loss
$q$
POSTERIOR
$h_t$
$z_t$
$m_t$
$k_t$
$M_t$
READ
READ
WRITE
$\tilde{k}_t$
$\tilde{h}_t$
$\tilde{m}_t$
$\tilde{n}_t$
$a_t$
Policy Loss
$a_{t-1}$
DECODER
$(\hat{I}_t, \hat{R}_t, \hat{v}_t, \hat{a}_{t-1}, \hat{r}_{t-1}, \hat{T}_t)$
Reconstruction Loss

# MERLIN on the Memory Game



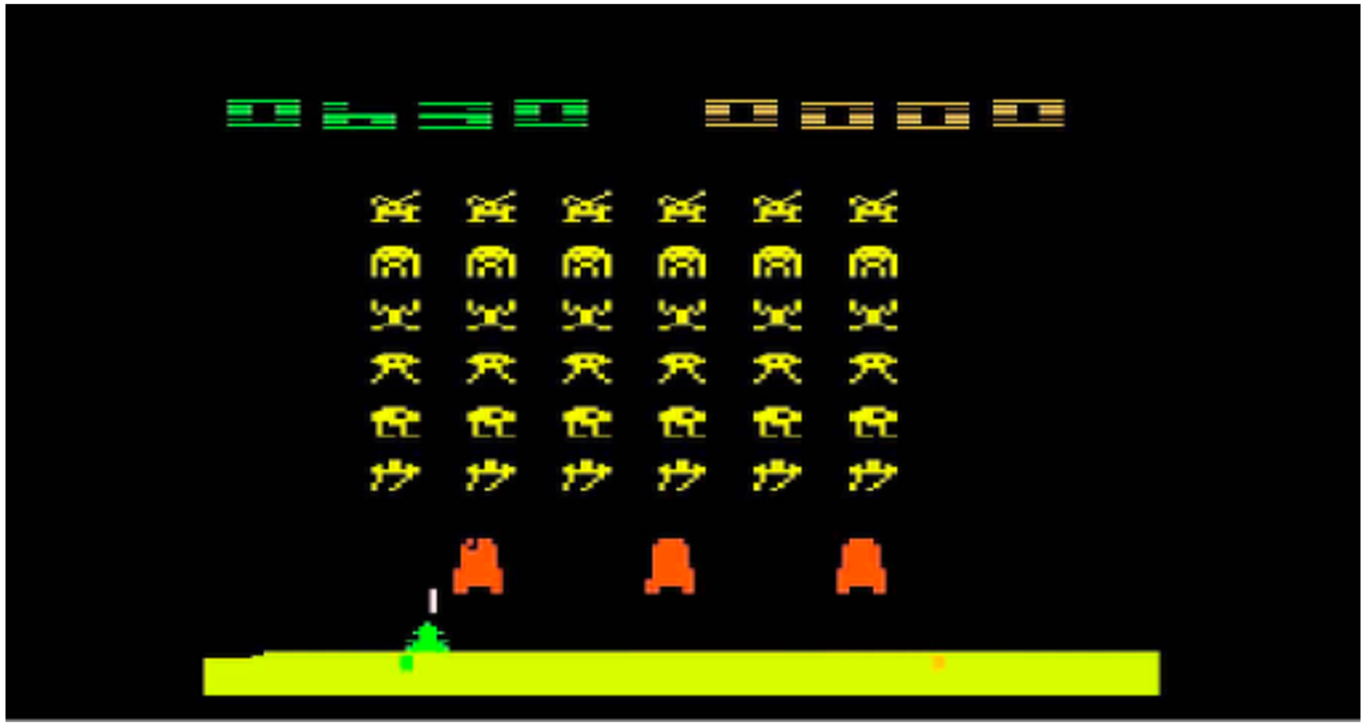


# Truncation for Very Long Tasks




**Here, unless otherwise specified, we truncate every 20 steps (15 steps per second).**

# The Bullet Problem



Saliency and
"The Bullet Problem"

496 *HIPPOCAMPUS* VOL. 3, NO. 4, OCTOBER 1993

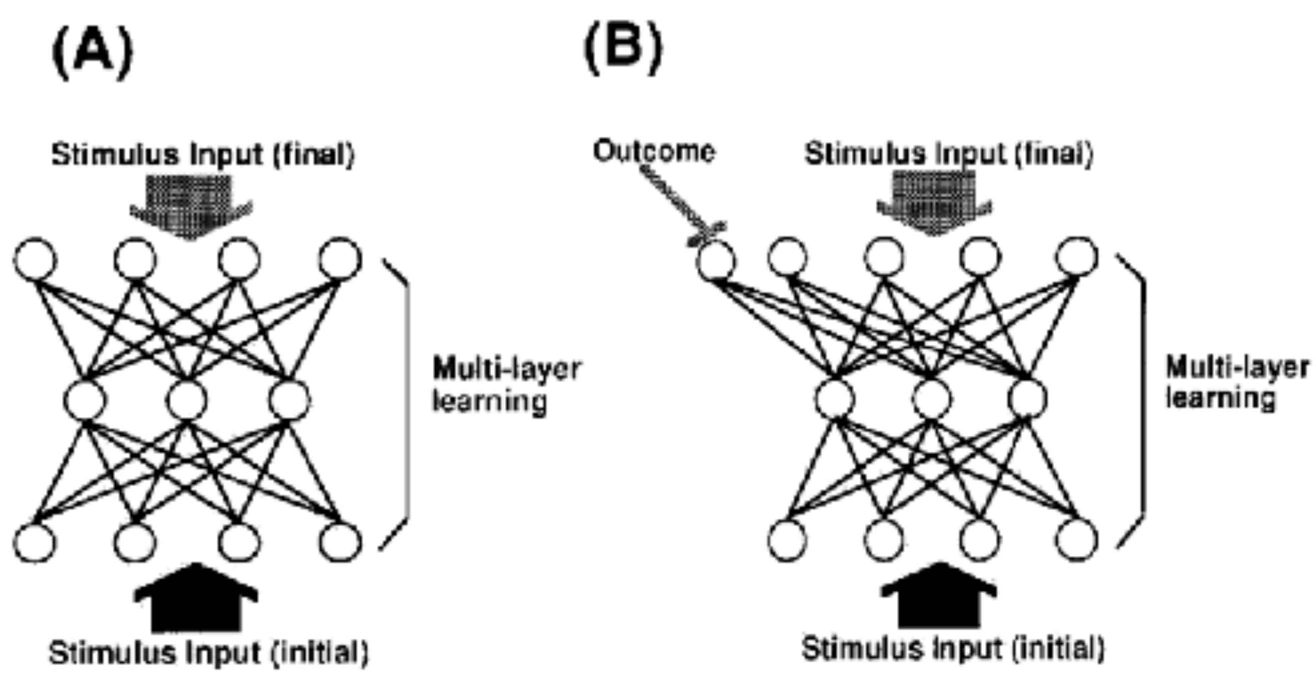
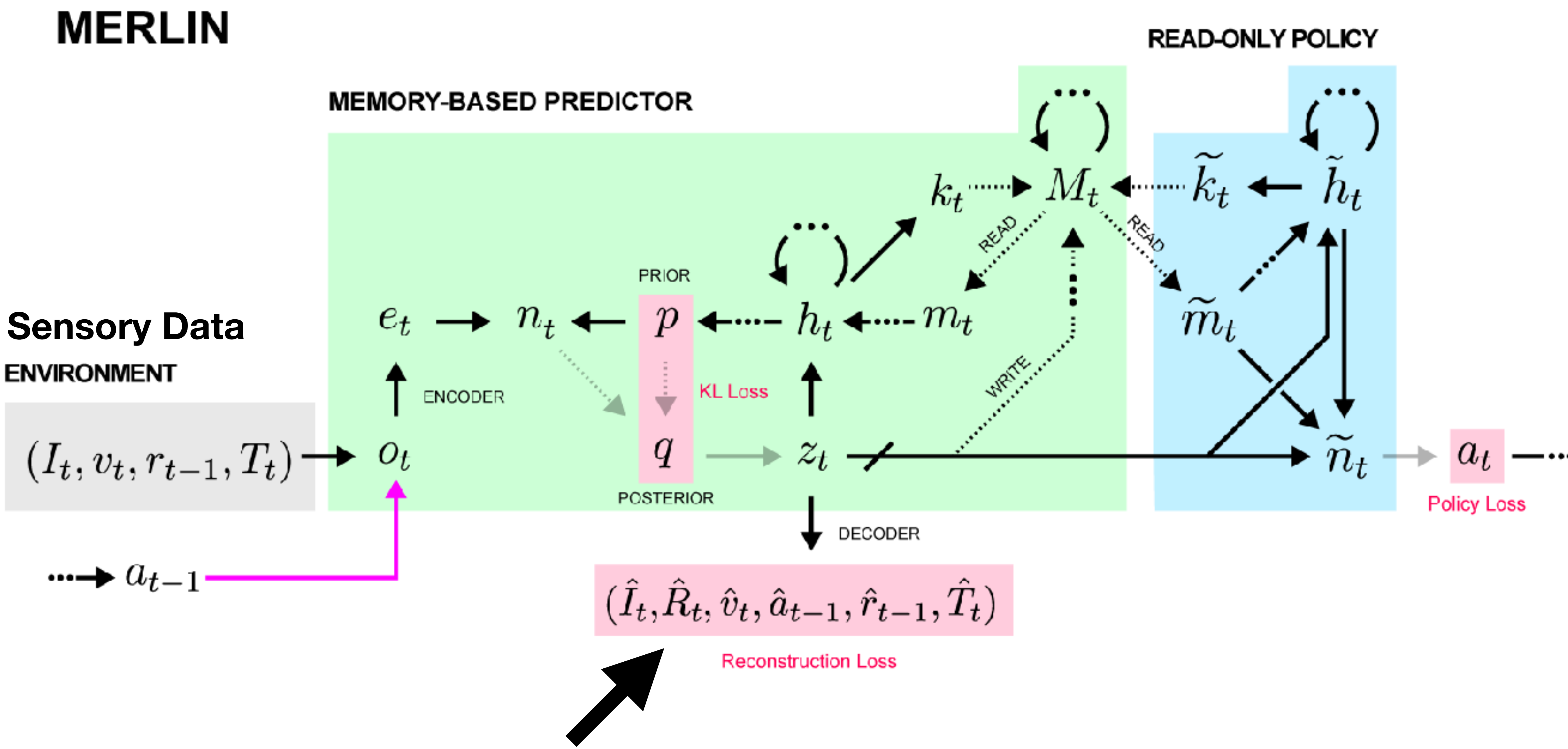



Fig. 4. (A) An autoencoder (Hinton, 1989). This network learns to reproduce its inputs at the output layer, using a multilayer learning algorithm. It contains a narrow internal layer, and so is forced to generate an internal representation that compresses redundancies in the input pattern. (B) A predictive autoencoder. This network is an autoencoder augmented with the constraint that it must also output a classification (or prediction of outcome) for the output pattern. The representation formed in the narrow internal layer must still compress redundancies; now, however, it must also differentiate representations of input features that are especially useful in predicting the outcome.

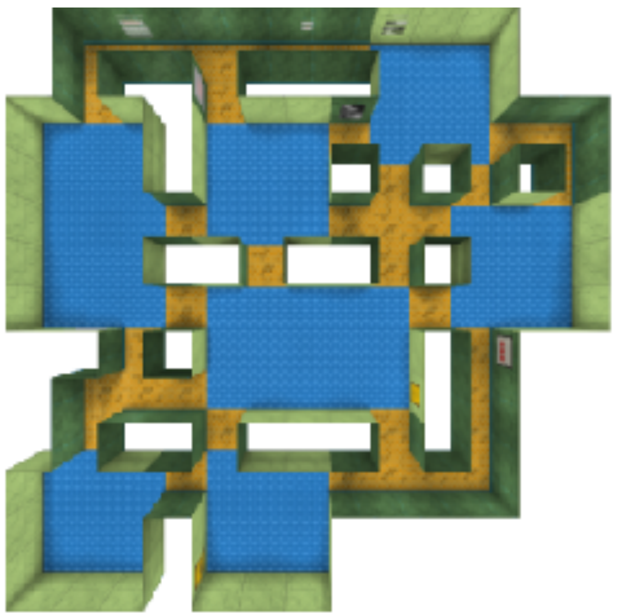
Gluck and Myers, 1993

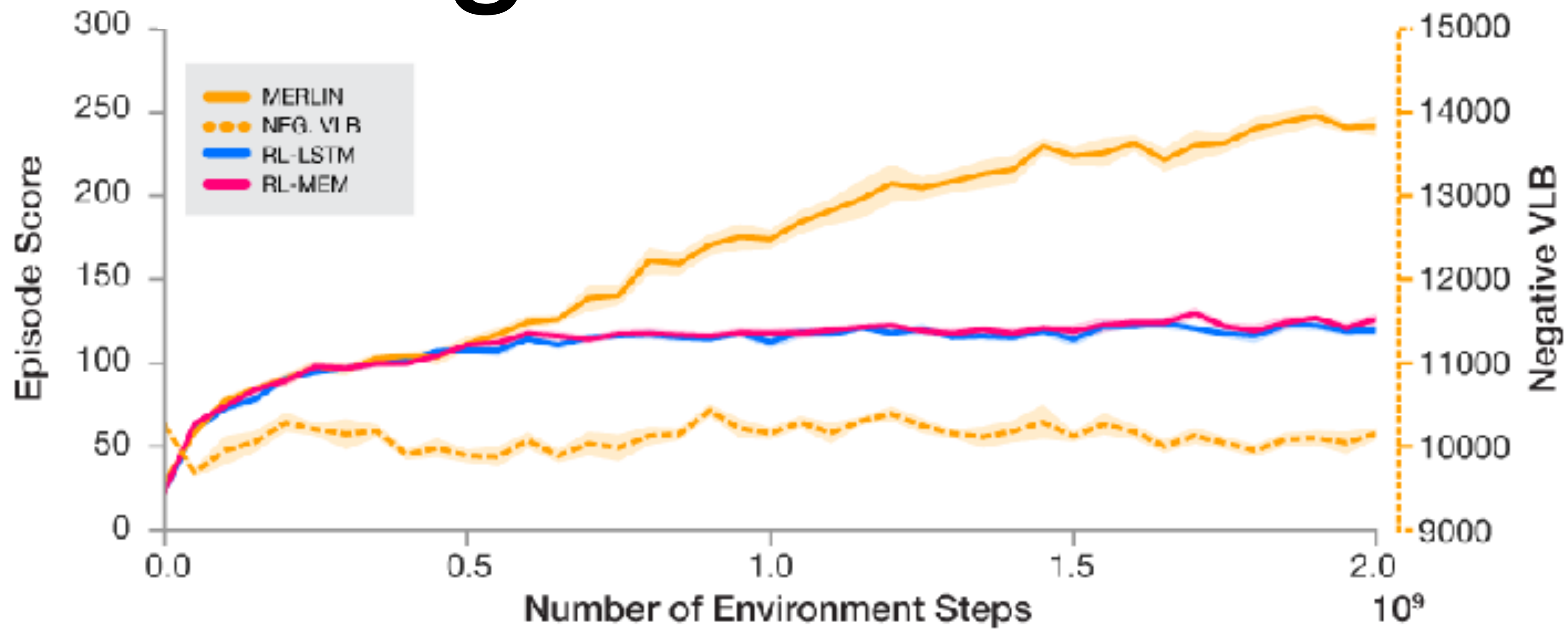
# MERLIN

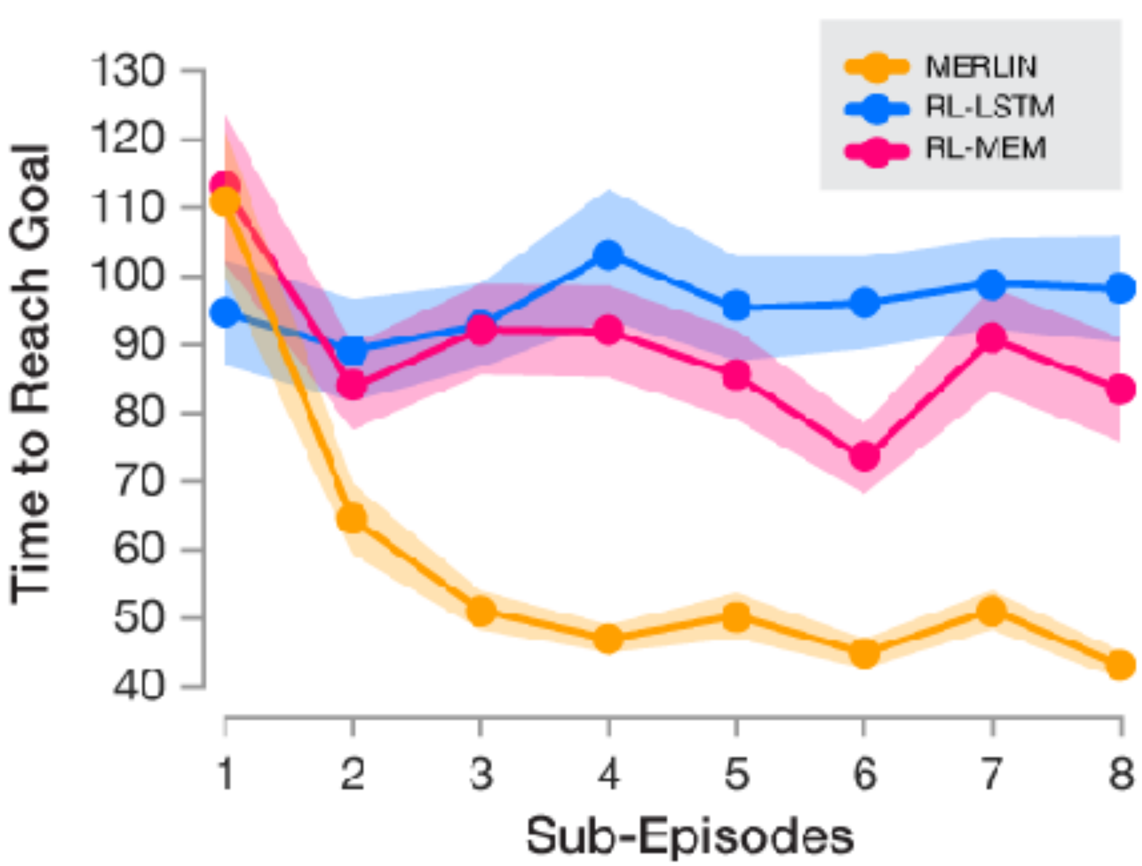


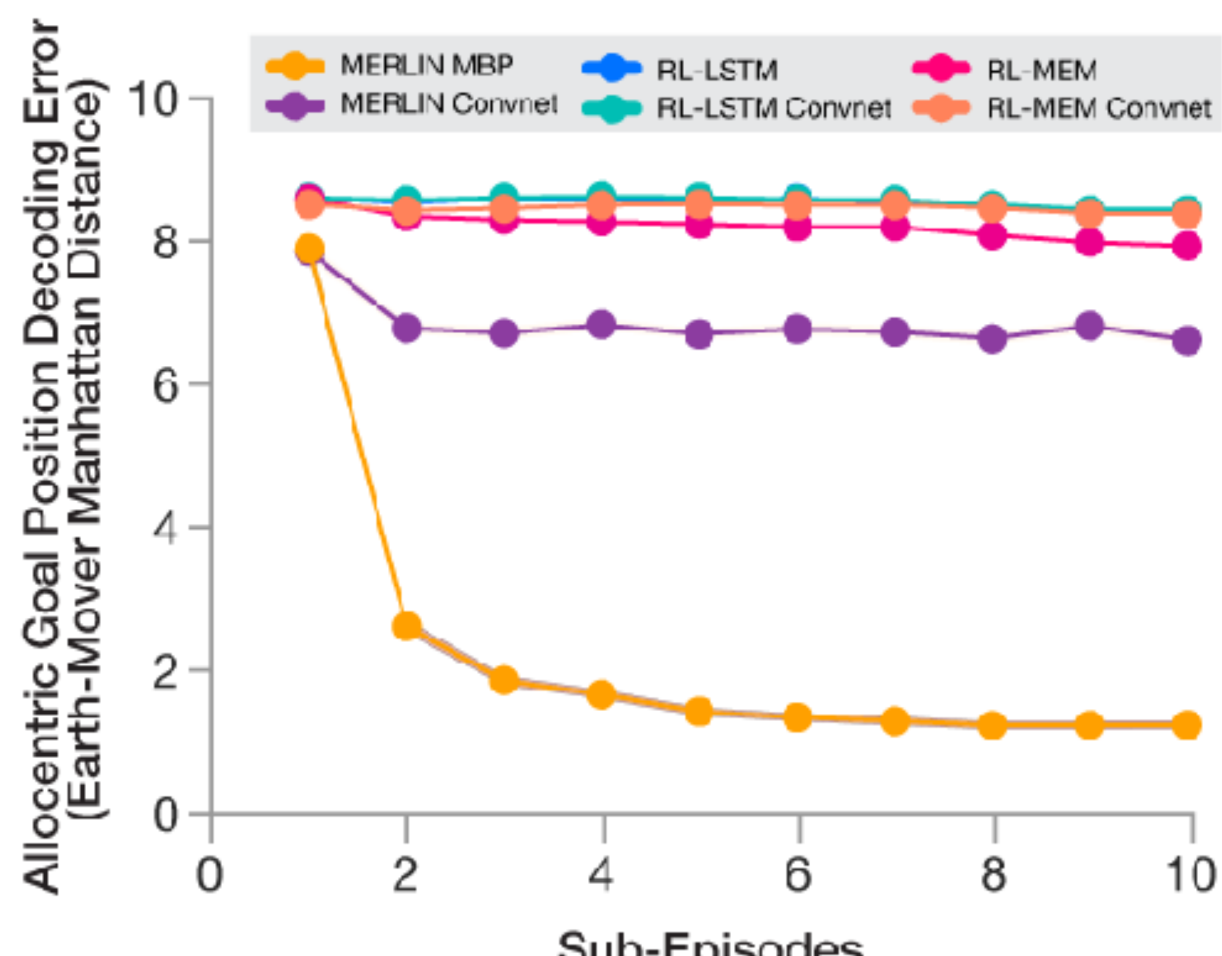

**Return Prediction** **State variable z is only 200 dimensional**

# Navigation



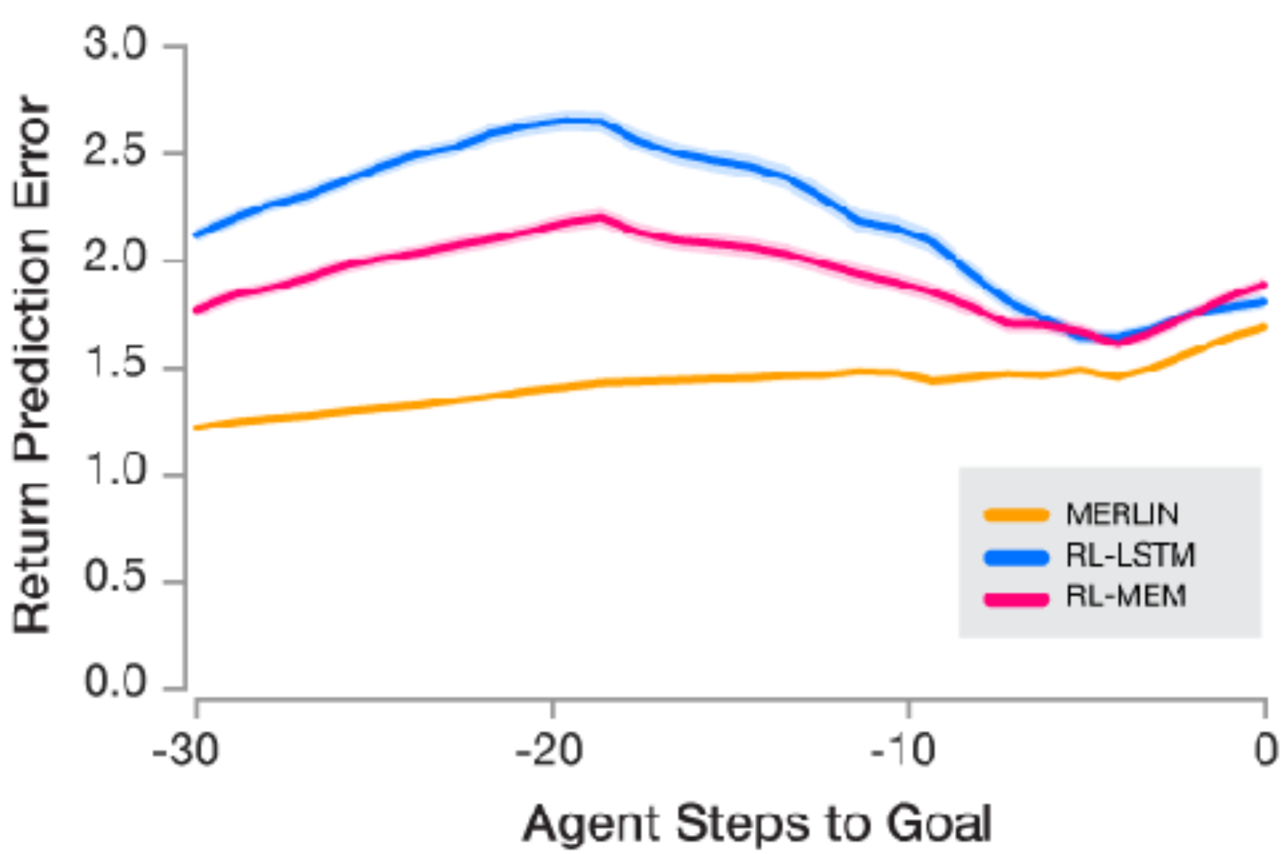

MERLIN
NEG. VLB
RL-LSTM
RL-MEM
Episode Score
Negative VLB
Number of Environment Steps
10^9

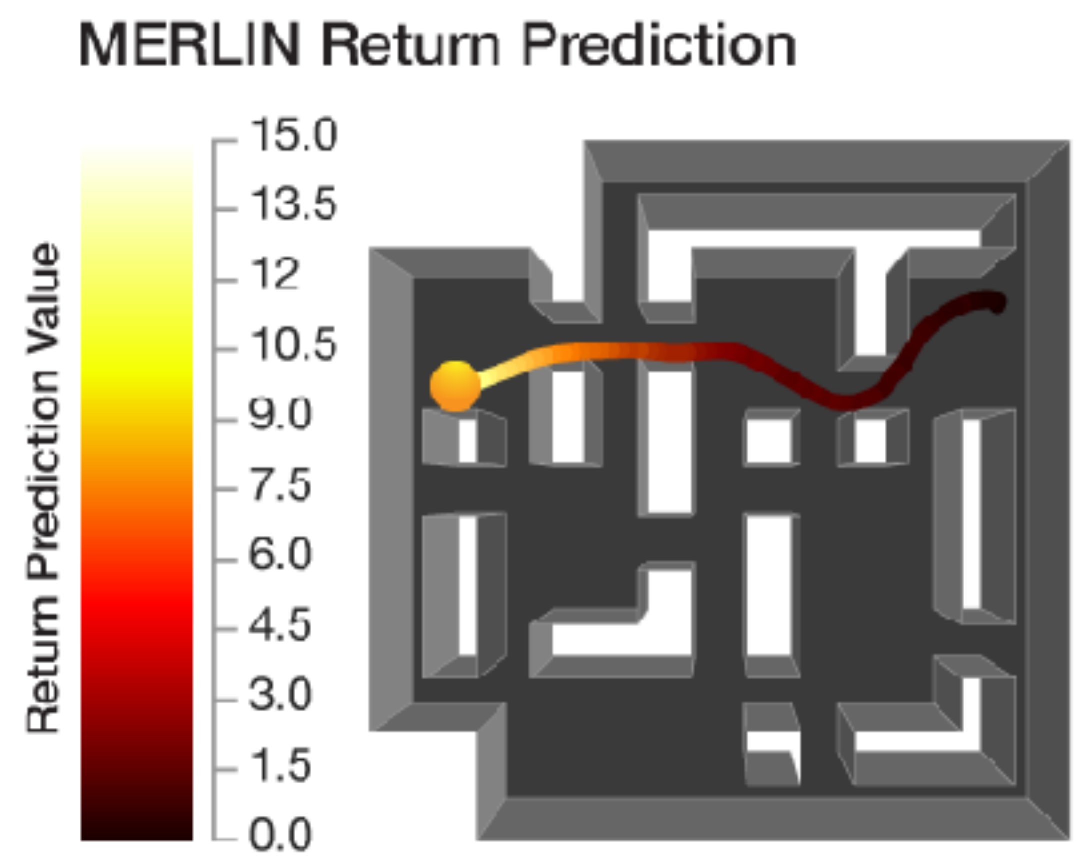

MERLIN
RL-LSTM
RL-MEM
Time to Reach Goal
Sub-Episodes


MERLIN MBP
MERLIN Convnet
RL-LSTM
RL-LSTM Convnet
RL-MEM
RL-MEM Convnet
Allocentric Goal Position Decoding Error
(Earth-Mover Manhattan Distance)
Sub-Episodes

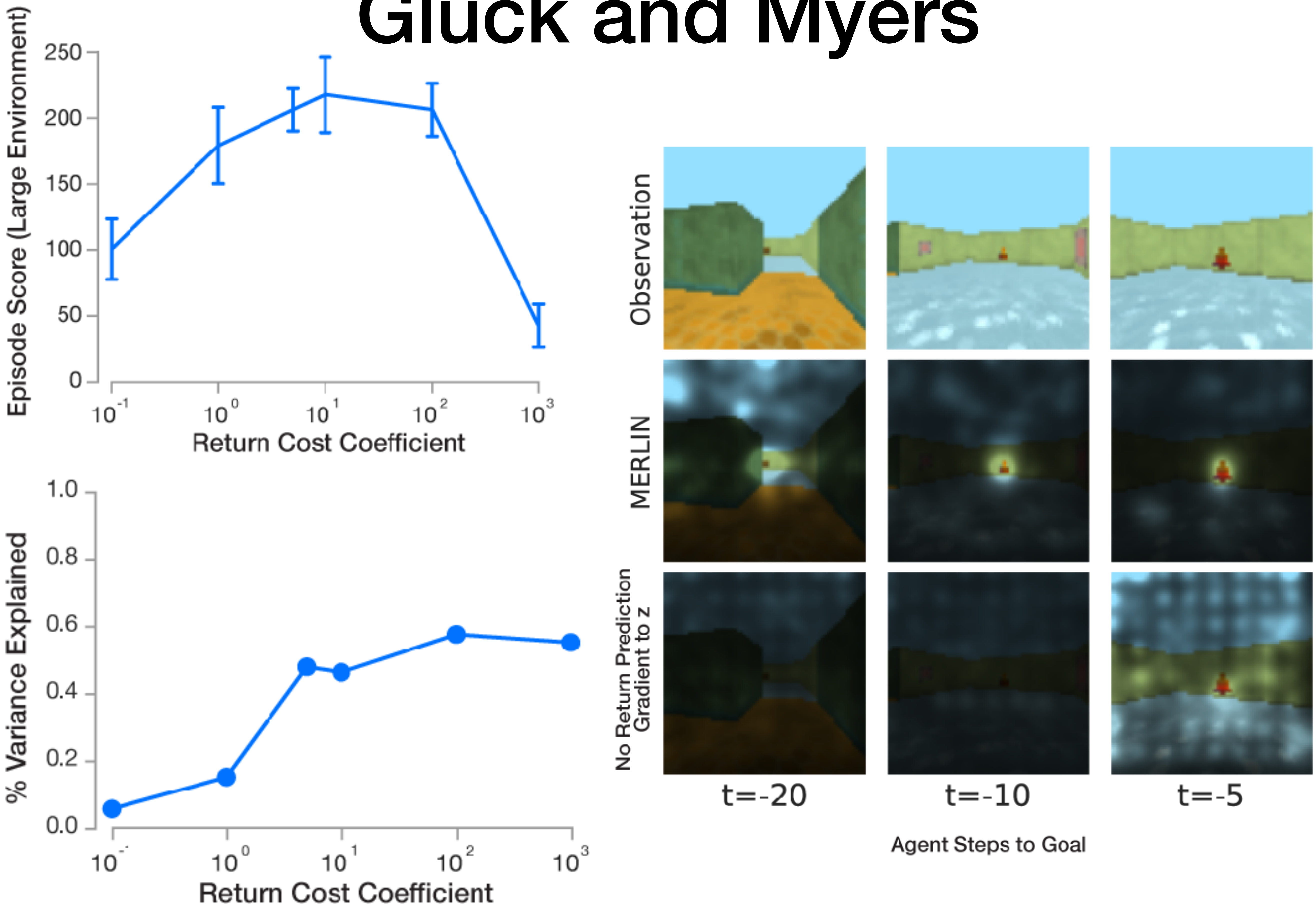
# Navigation

# Return Prediction

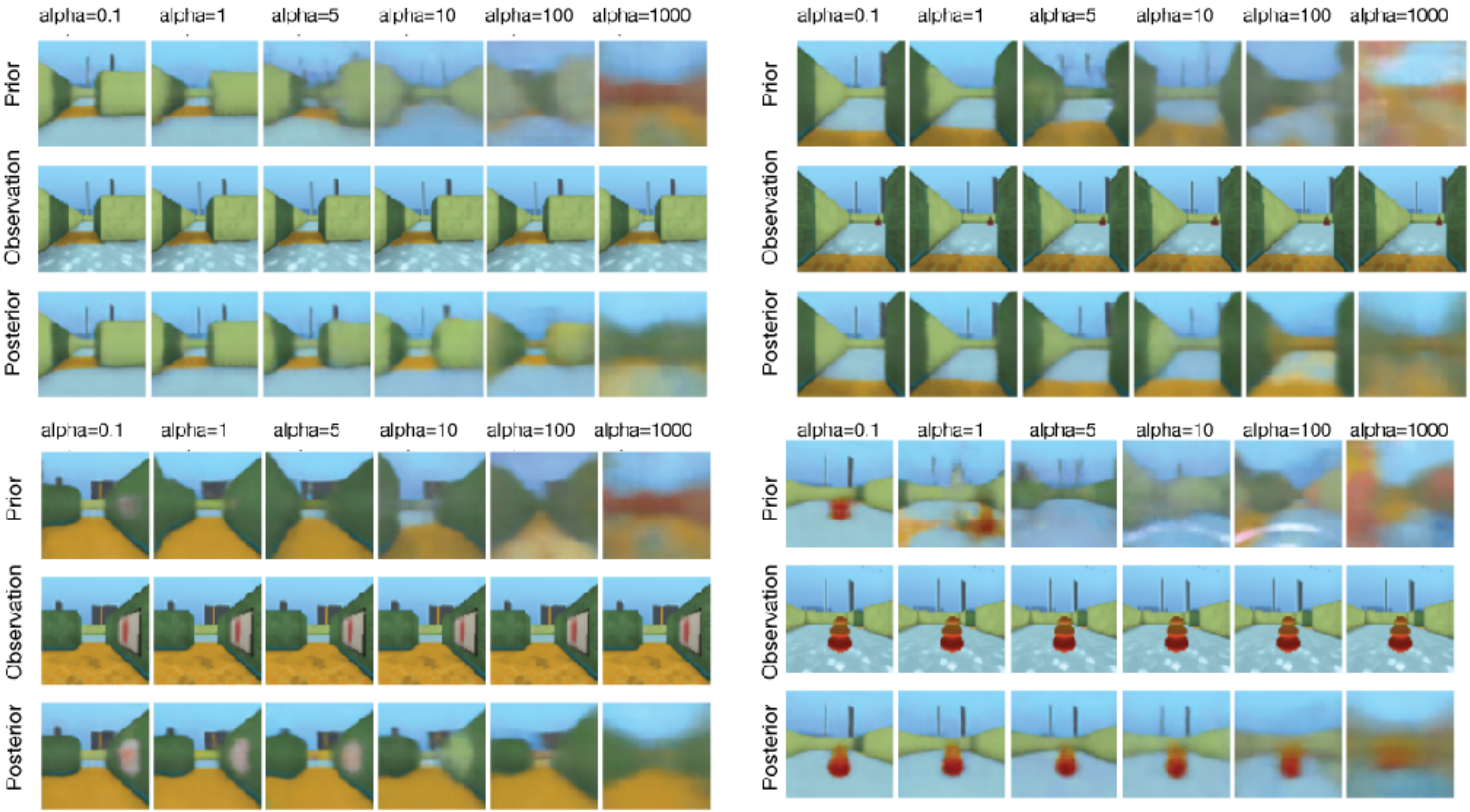


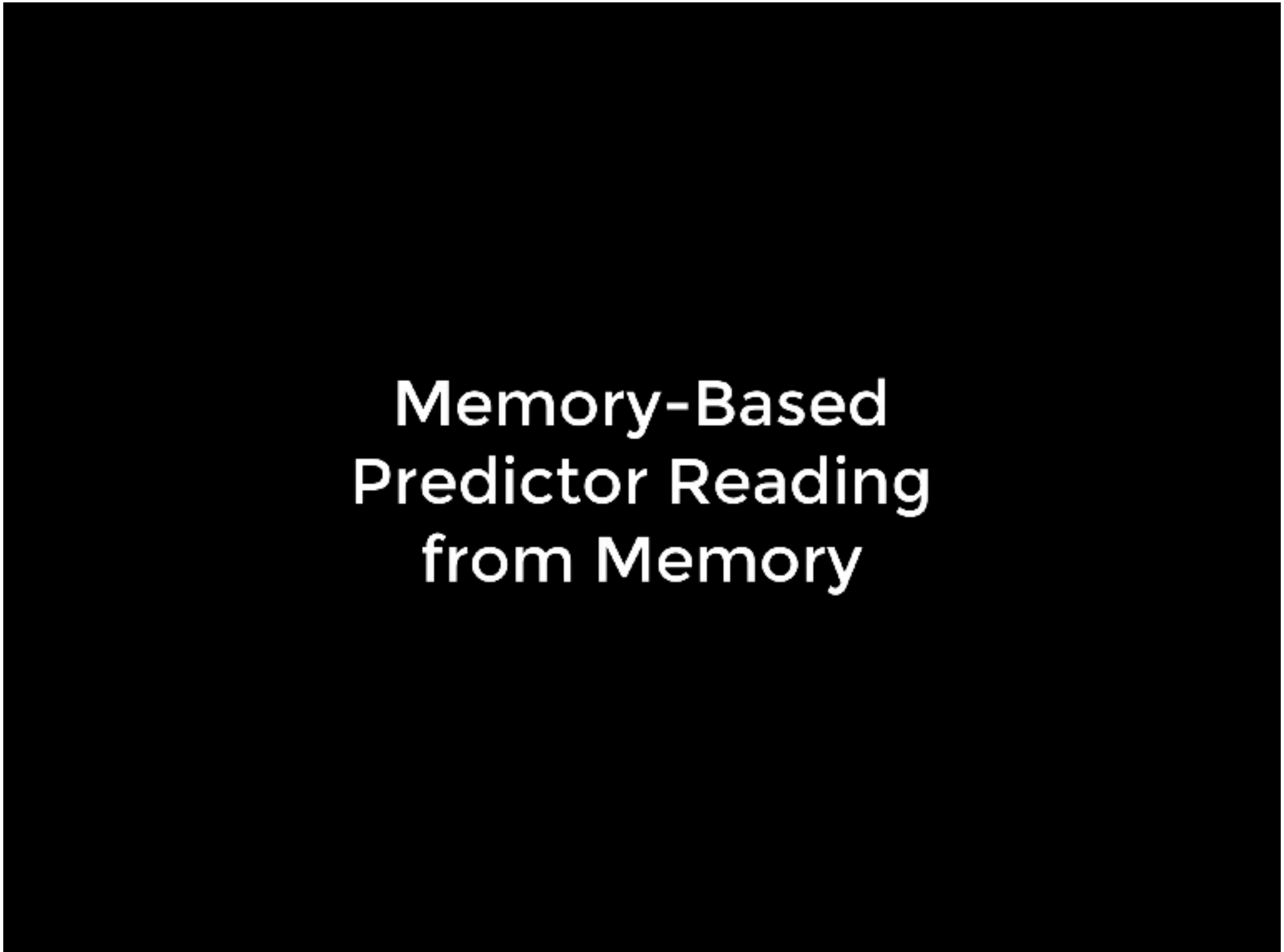
# Gluck and Myers

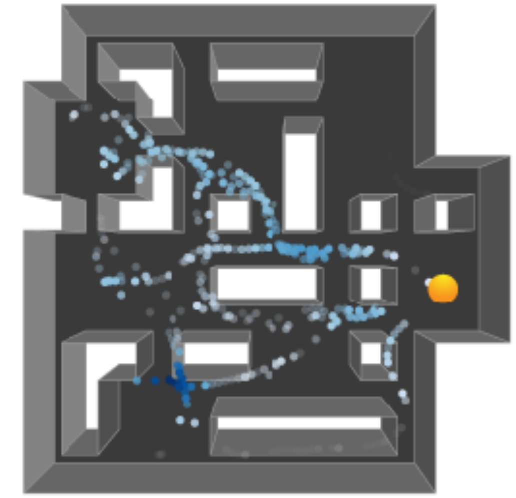


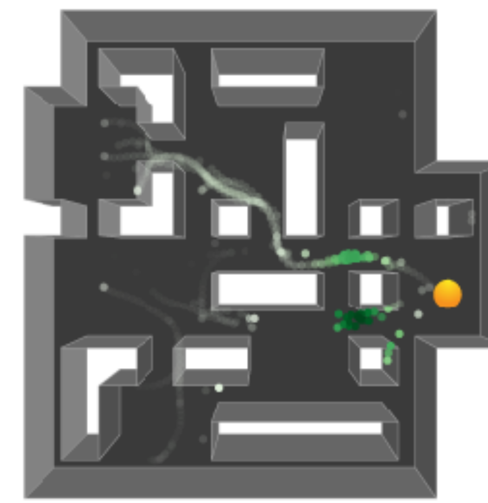
# Predictive Model (Prior vs. Posterior) for Different Values of Return Cost Coefficient

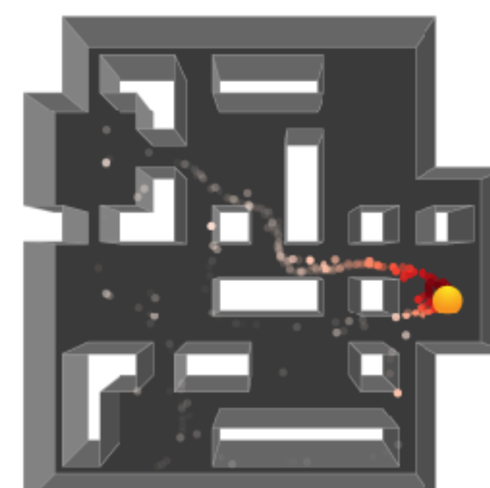


# Hierarchical Behavior

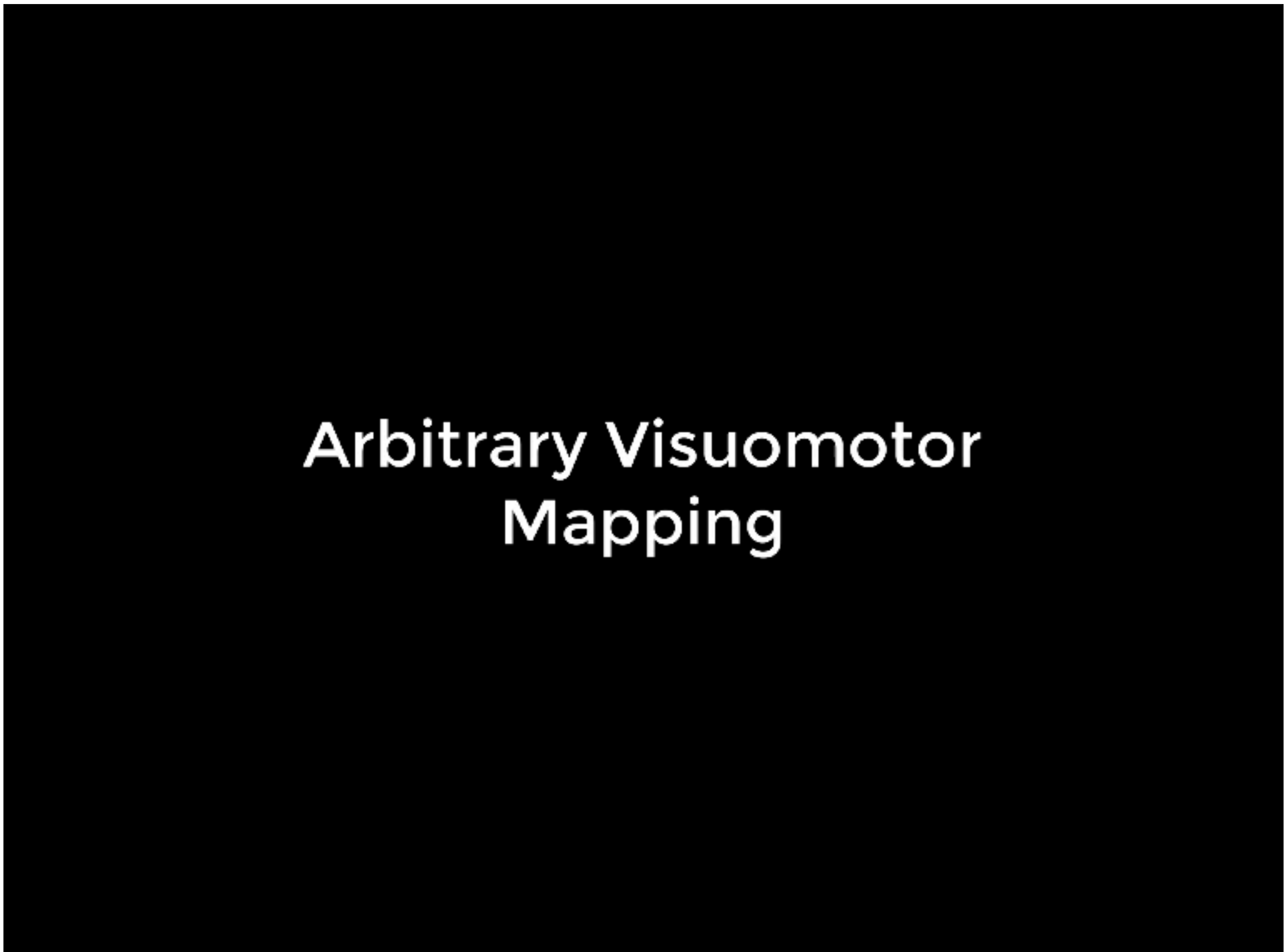




Read Head 1

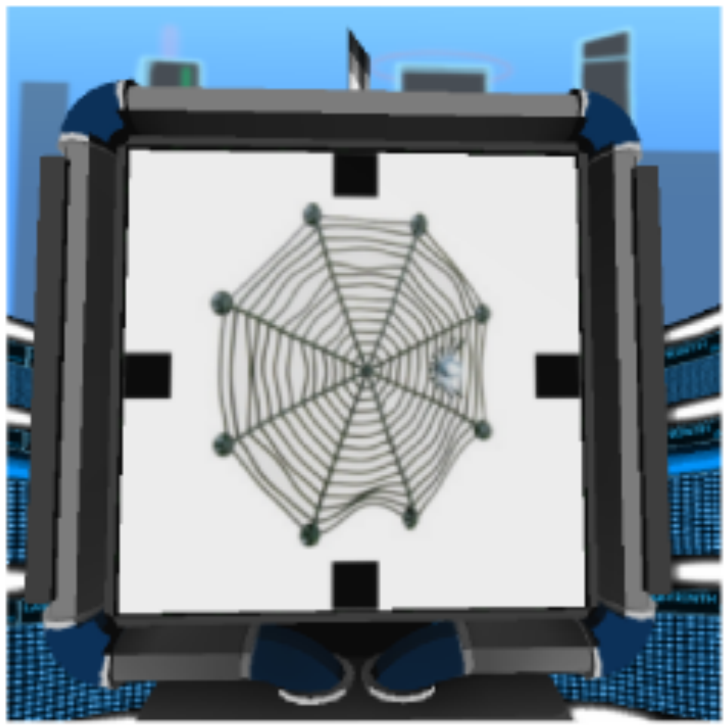

Read Head 2

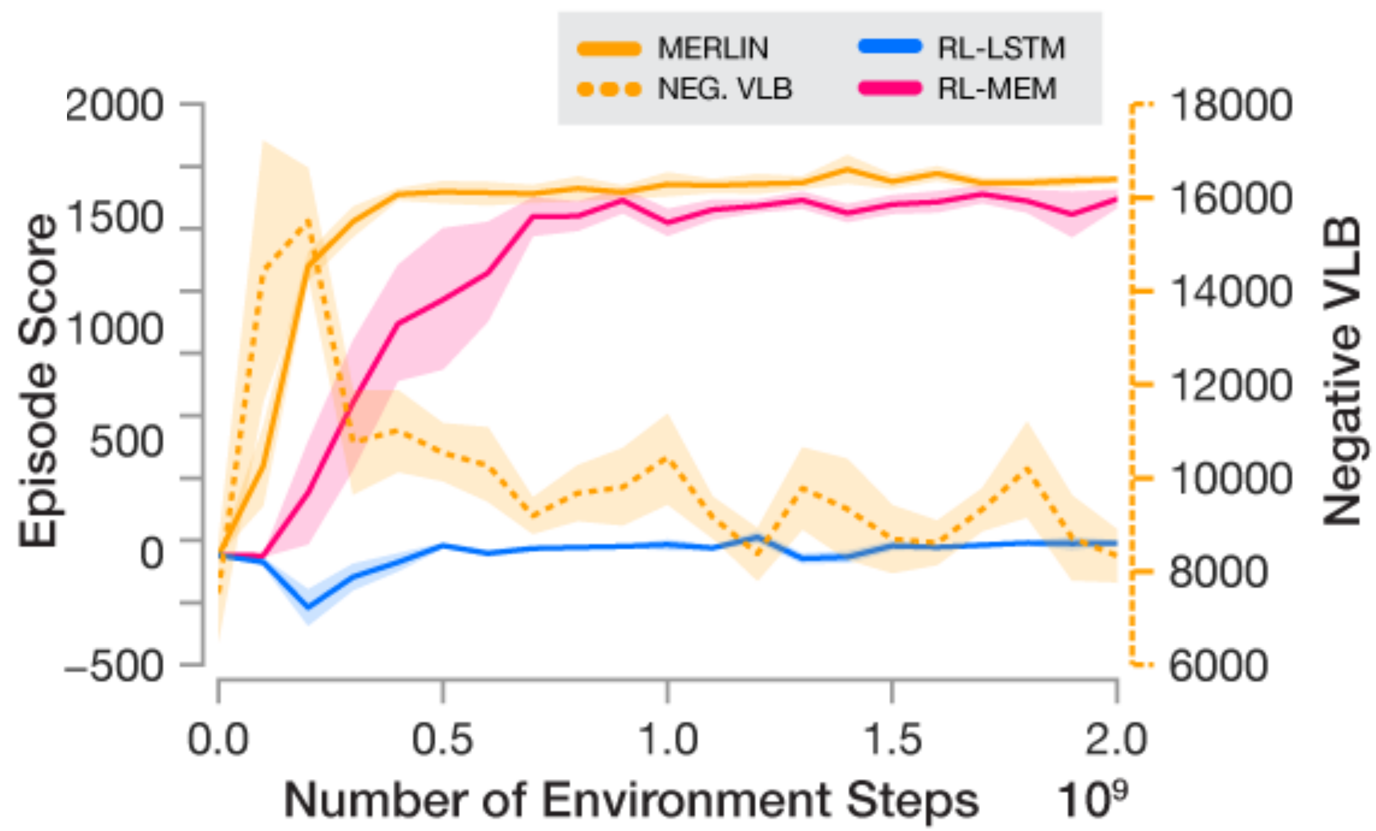

Read Head 3

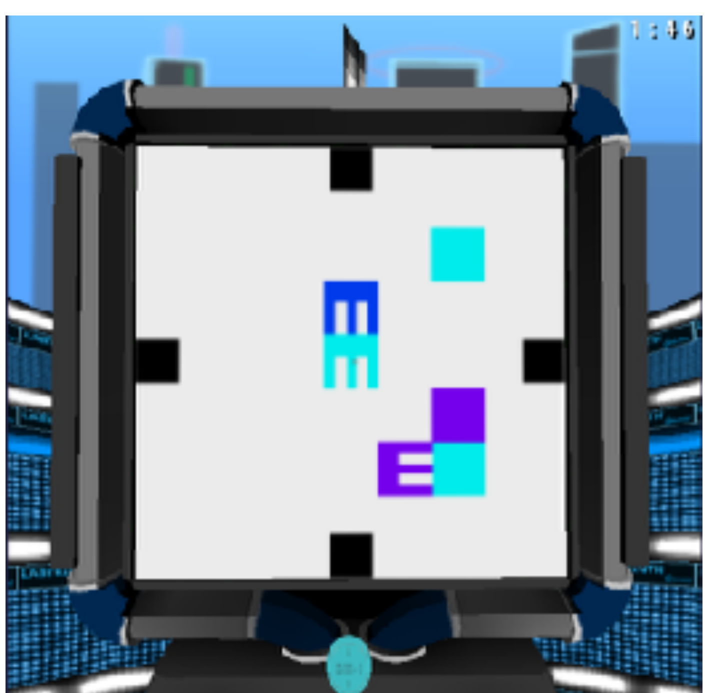
# Arbitrary Visuomotor Mapping (Wise and Murray, 2000)

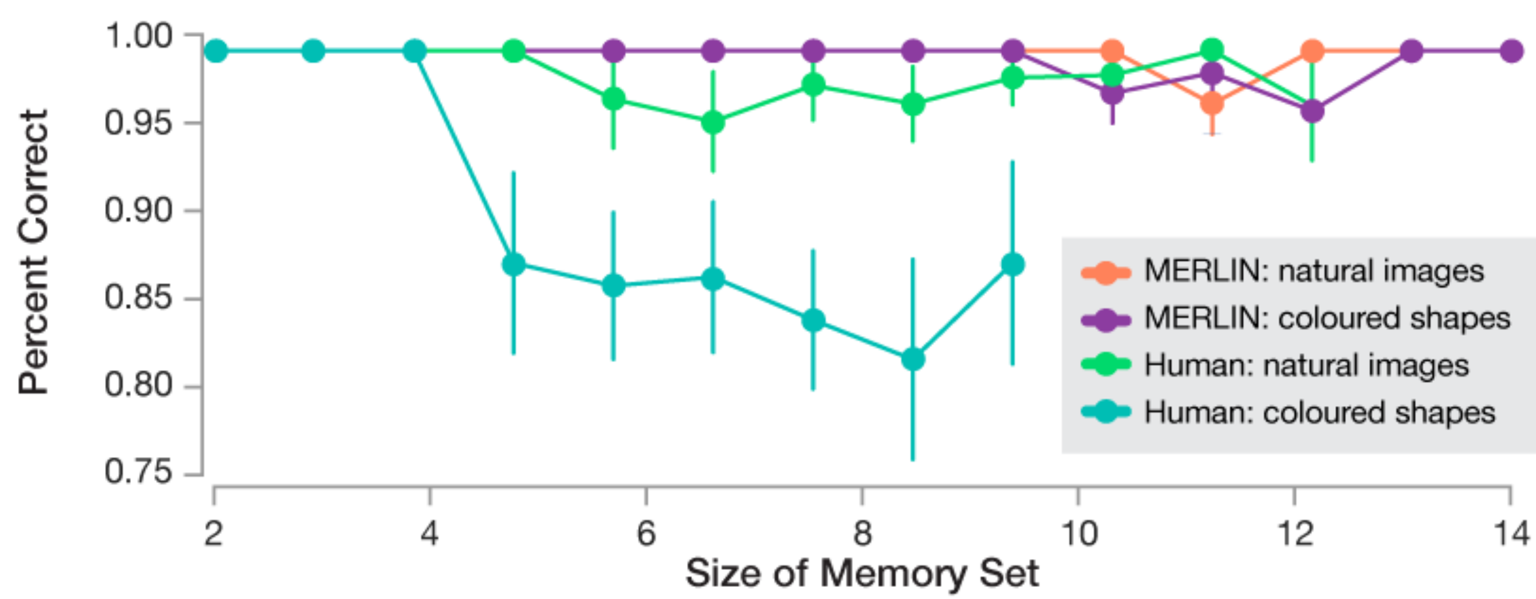


# Arbitrary Visuomotor Mapping (Wise and Murray, 2000)

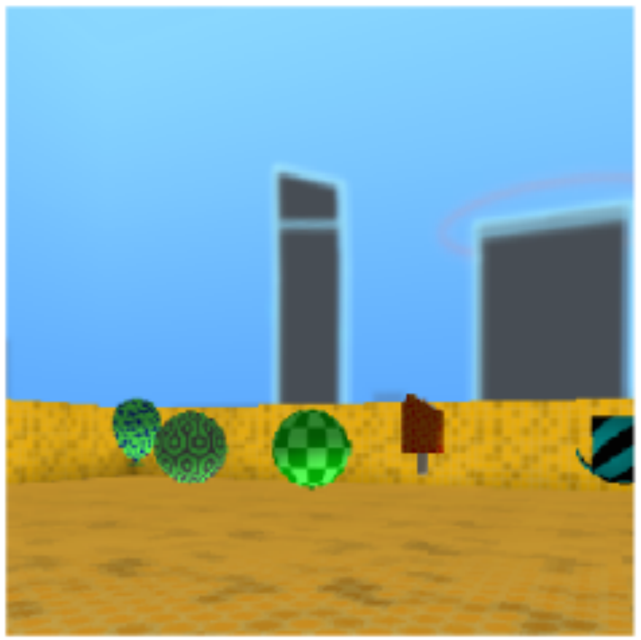


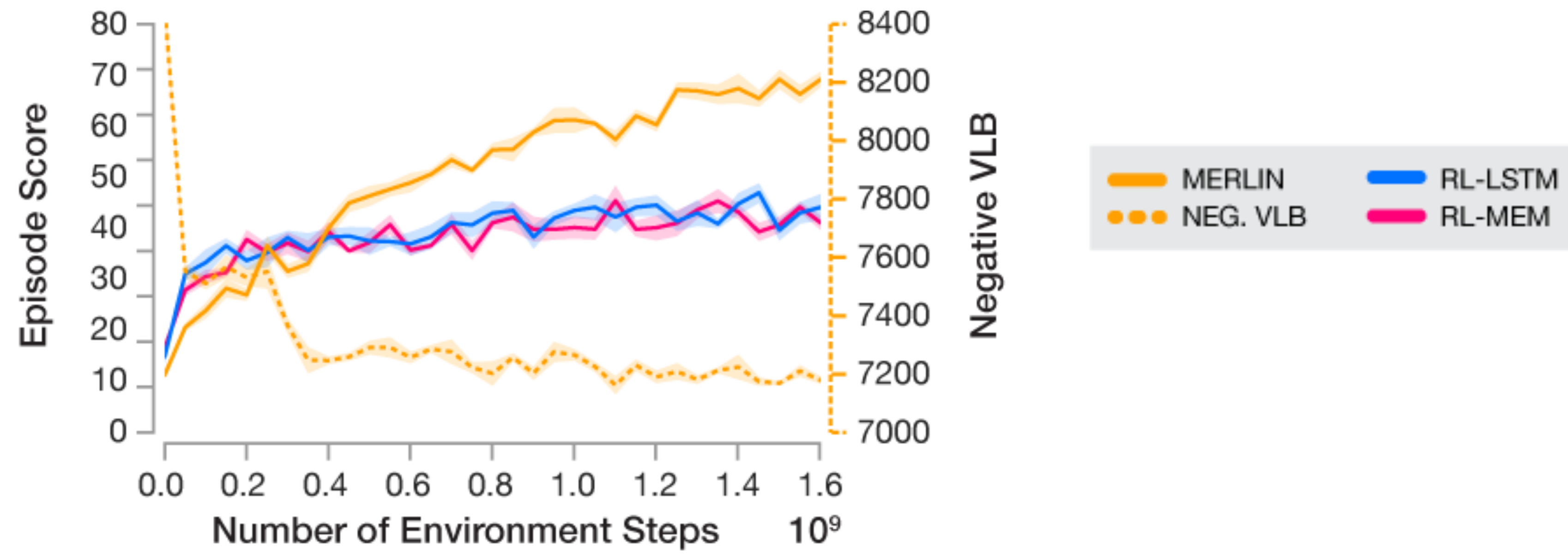


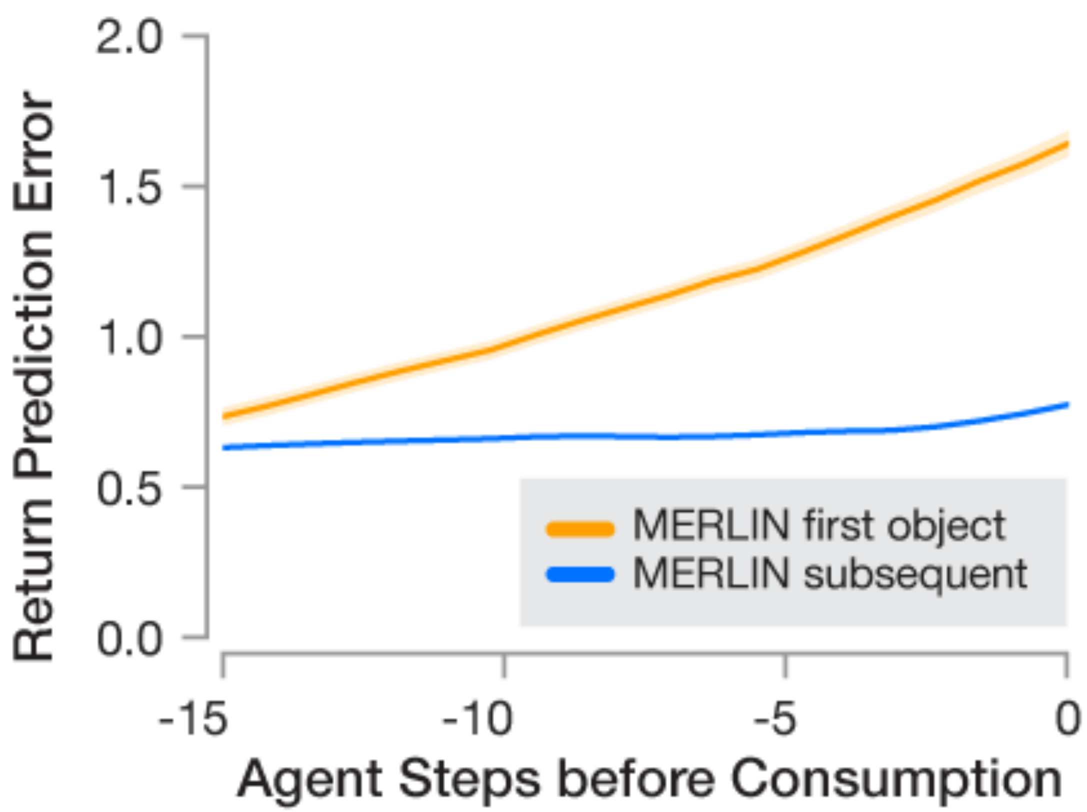



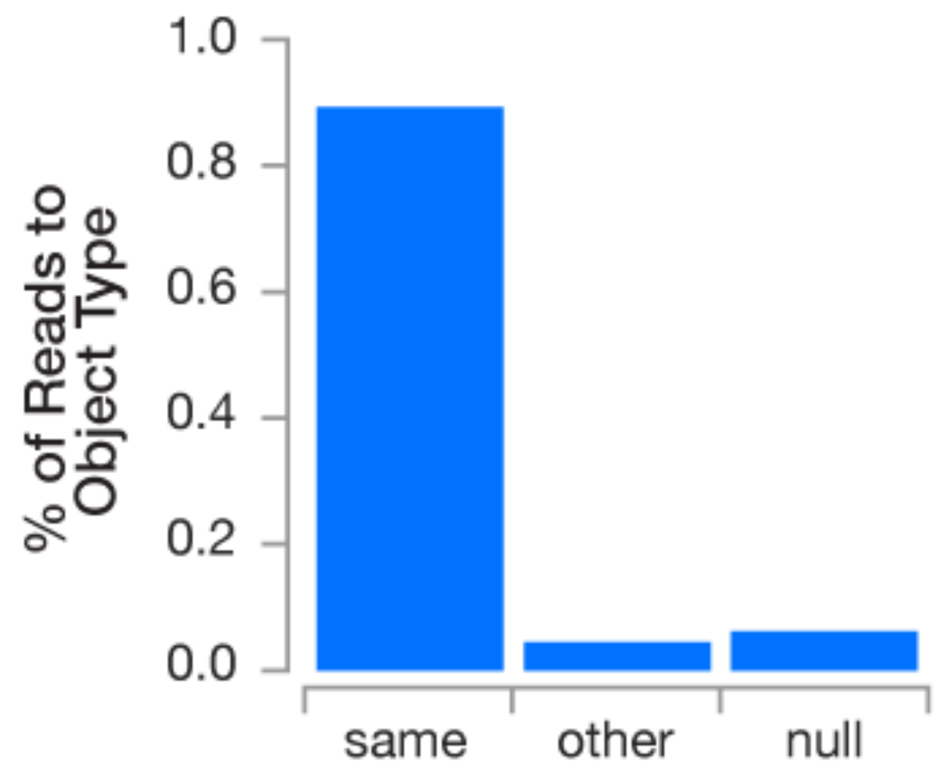


# Rapid Reward Valuation (One-Shot)





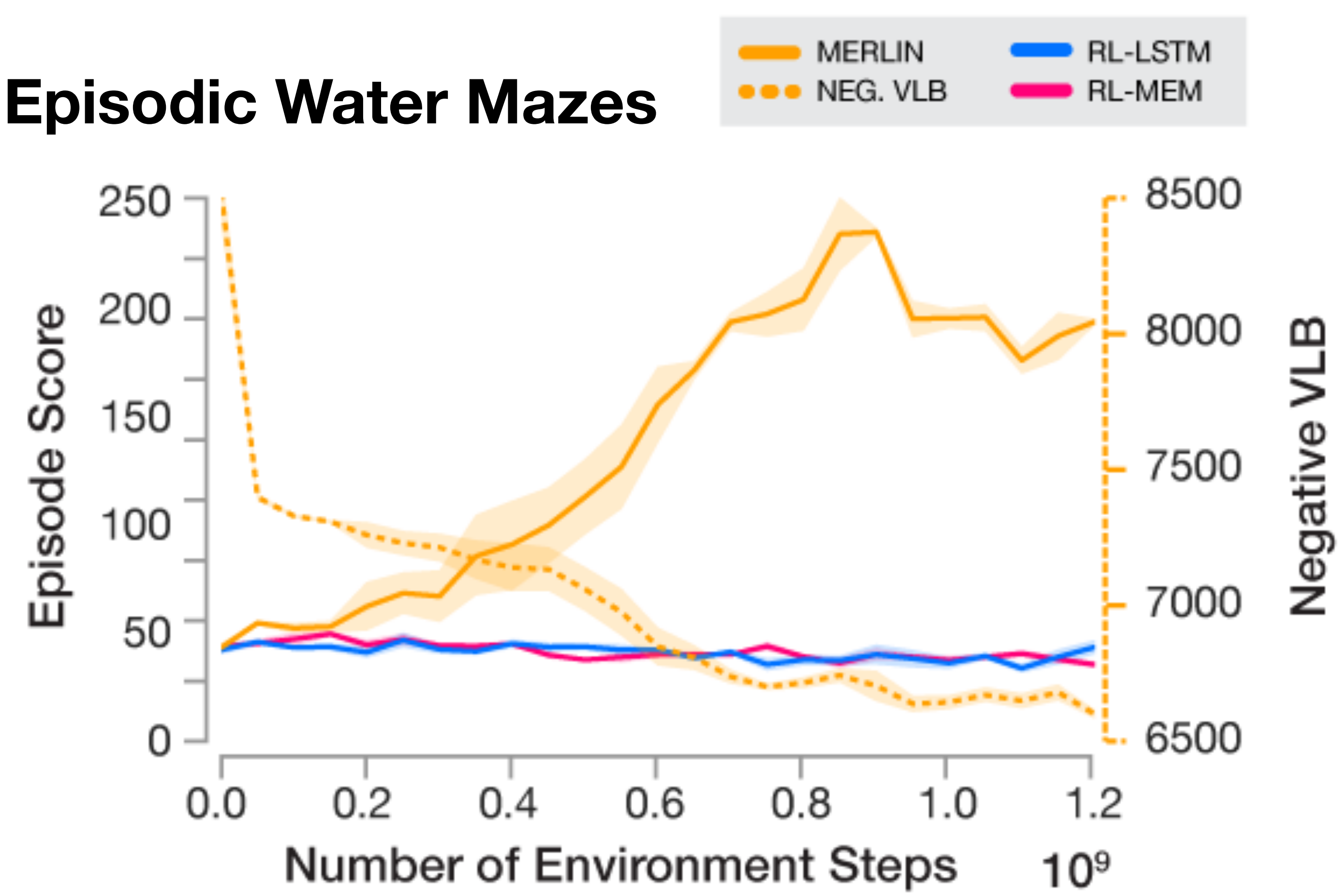



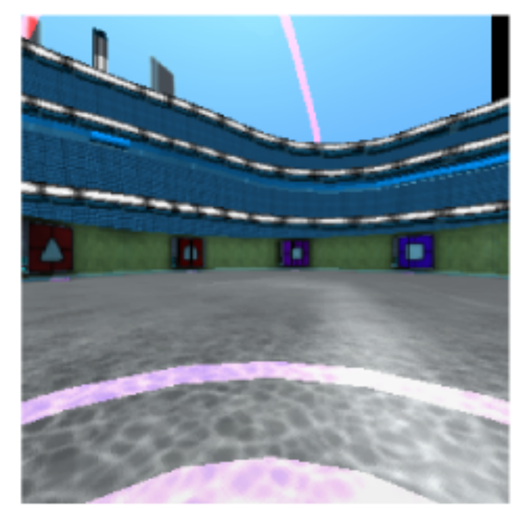



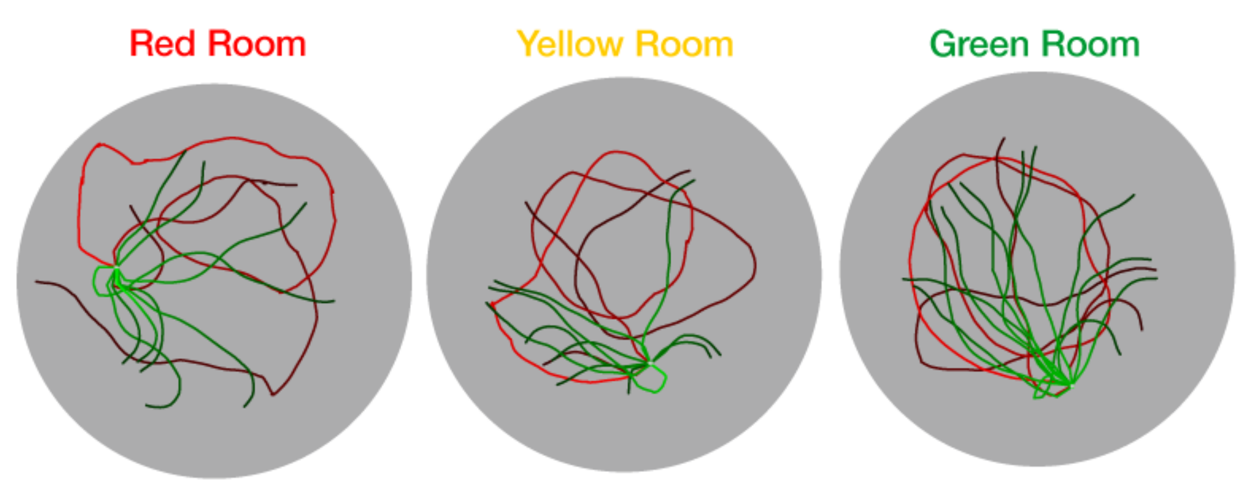
# Rapid Reward Valuation (One-Shot)

# An Episodic Task that Breaks Gradient Flow

**Block 1** **Block 2** **Block 3** **Block 1’**

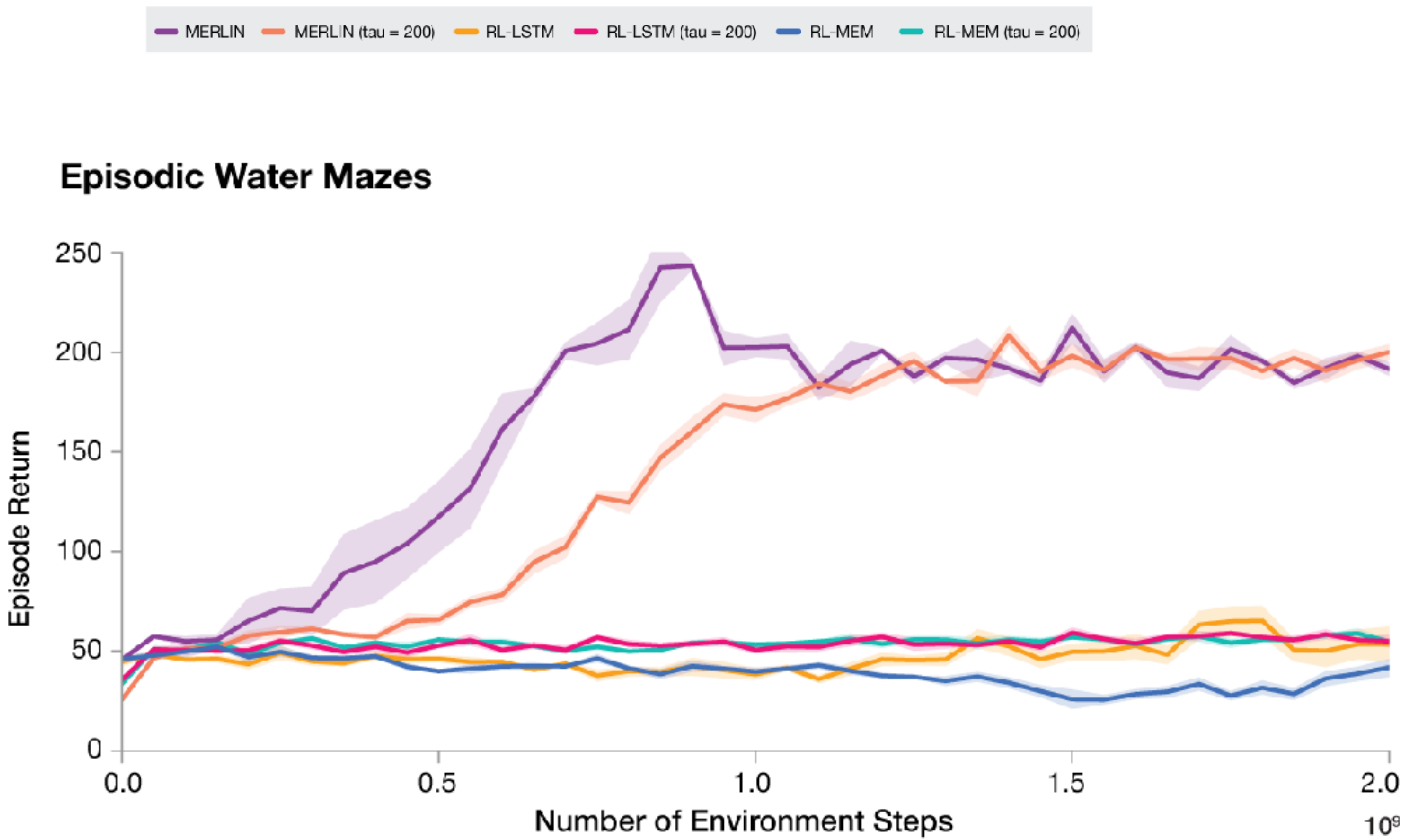


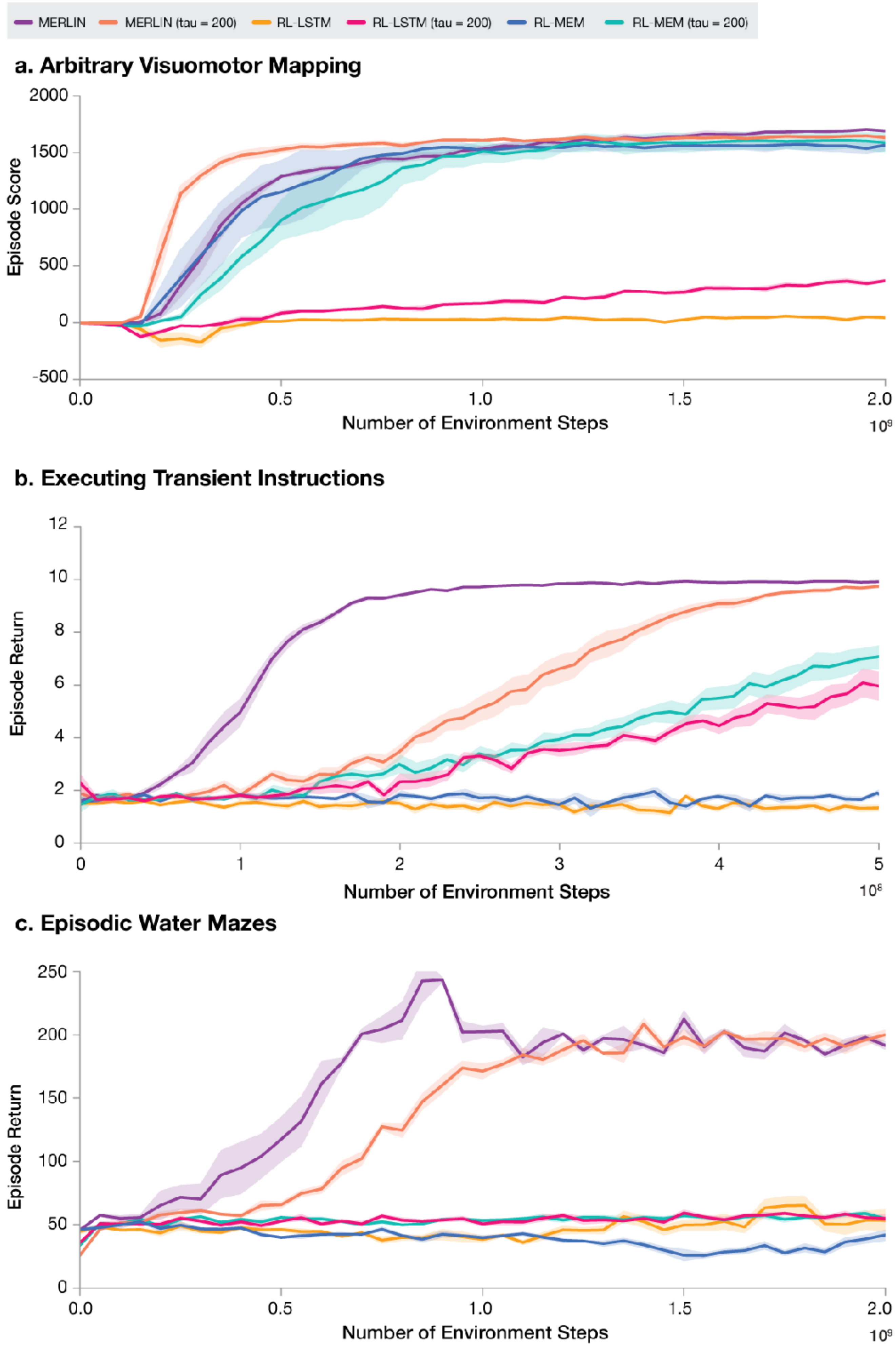


# An Episodic Task that Breaks Gradient Flow

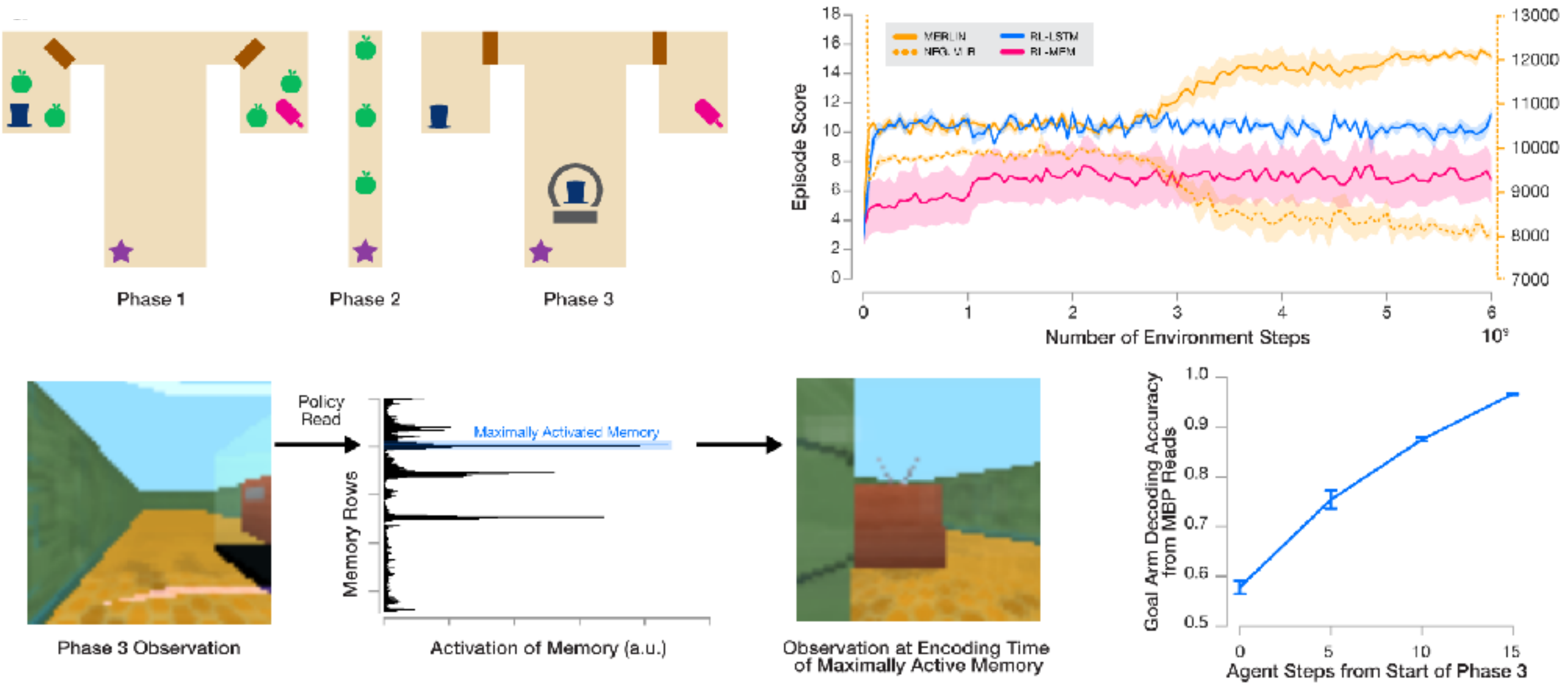
Learning Curves with Longer Temporal Credit Assignment Windows



Learning Curves with Longer Temporal Credit Assignment Windows

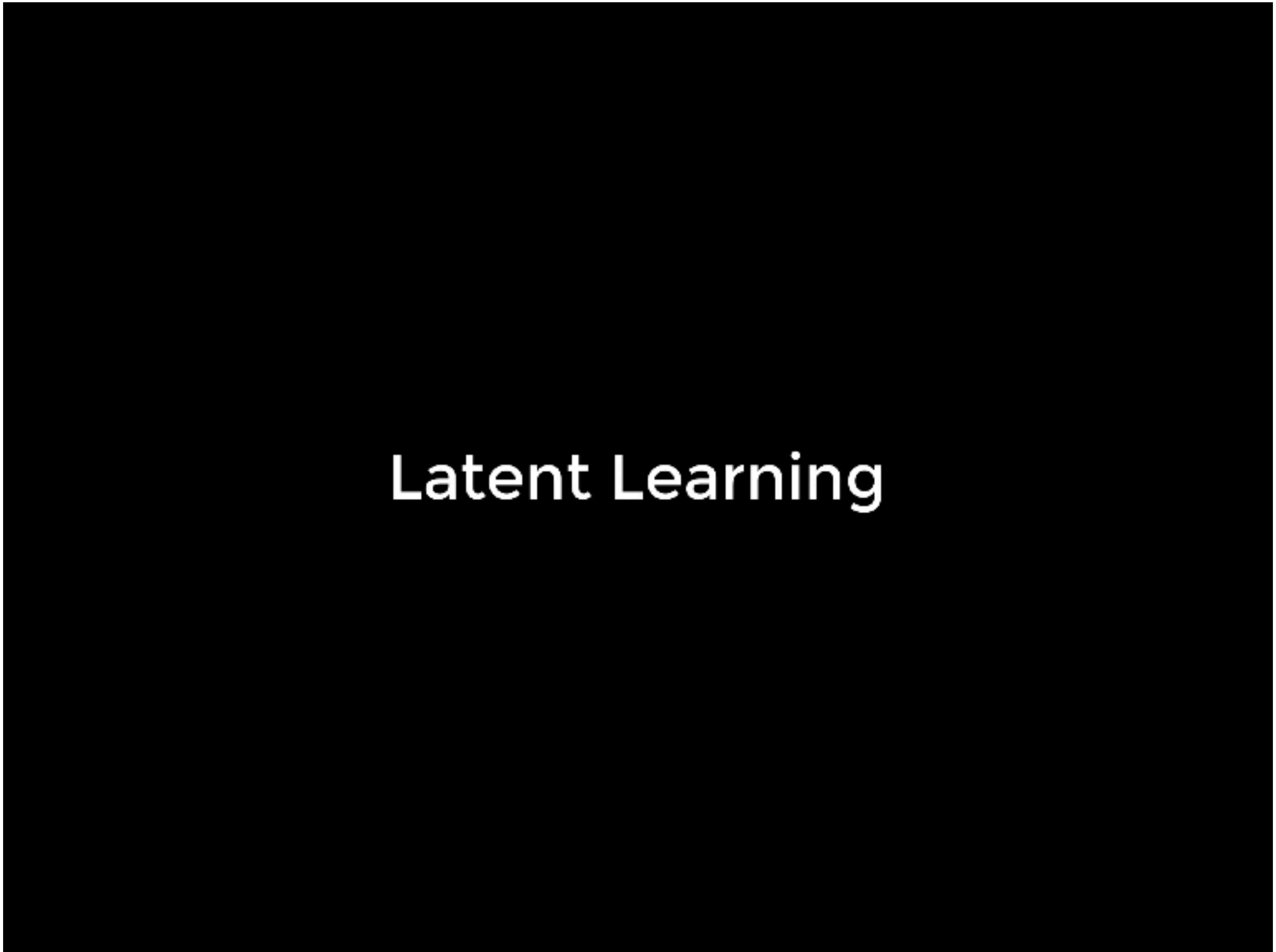


# Latent Learning




**Learning in the absence of direct task reward.** **Tolman and Honzik, 1930**

# Latent Learning



# Learning from Demonstrations

Programmer's Apprentice: (Waters and Rich, 1987)

The objective for programming is hard to write down!

Imitation is a useful proxy when the objective is hard to articulate.

Maybe that's why it was called the programmer's *apprentice*.

# Learning Human Behaviors

Given human input-output measurements, how to build a good imitation learner?

Simple approach is supervised learning or behavioral cloning:

$$\tau = (s_1, a_1, s_2, a_2, s_3, a_3, \dots) \qquad \max_\theta \sum_k \log p_\theta(a_k | s_k)$$

Problem with this approach: if you find yourself in a state unlike one the human demonstrated, the policy may do anything.

# Learning Human Behaviors

Another approach: make the achieved states / trajectories look like one's from demonstration data.

$$\max_\theta \sum_k \log[p(s_{k+1}|s_k, a_k)p_\theta(a_k|s_k)]$$

Advantage: if the agent gets away from the distribution of states, there is a motivation to get back.

Problem with this approach: It typically implies having a model of transitions that is good. This can be hard for complicated environments.

# Learning Human Behaviors

Another approach: make marginal distributions over states similar to the expert's.

$$\max_\theta \sum_k p(s_k)$$

Inverse Reinforcement Learning

Generative Adversarial Imitation Learning (Ho and Ermon, 2016) based on GANs (Goodfellow, 2014). Here, fool the discriminator:

$$\max_\theta p_{\text{discriminator}}(\text{demonstration}|s_k)$$

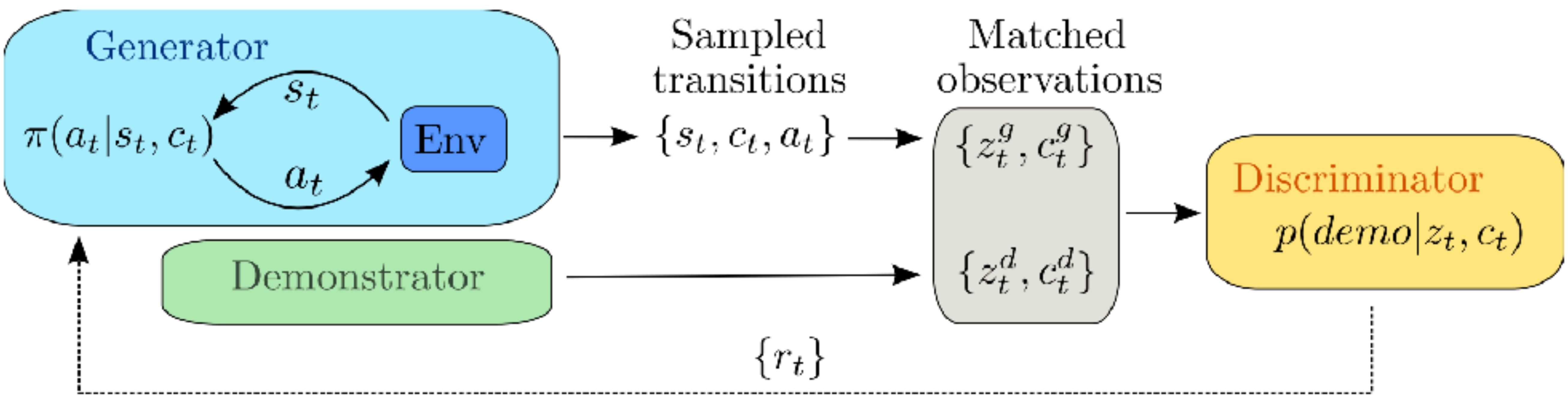
# Learning Human Behaviors

Generative Adversarial Imitation Learning (Ho and Ermon, 2016)

Learning Human Behaviors from Motion Capture by Adversarial Imitation (Merel, Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, Nicolas Heess, 2017)

# Conditional GAIL

Note: no action information strictly needed; important, as explained next



# Algorithm

**Algorithm 1**

**Input:** Set of demonstration observations $\{z_t^d, c_t^d\}_{t=1...T^d}$
Randomly initialize policy ($\pi_\theta$) and discriminator ($D_\phi$)
// Perform N training iterations of policy & discriminator updating
**for** $i$ in $1 \dots N$ **do**
  Execute policy rollouts to collect $T^g$ timestep observations, $\{z_t^g, c_t^g\}_{t=1...T^g}$
  Compute rewards $\{r_t = -\log(1 - D_\phi(z_t^g, c_t^g))\}_{t=1...T^g}$
  Update $\theta$ (e.g. by TRPO)
  // Perform M discriminator updates steps
  **for** $j$ in $1 \dots M$ **do**
    $\ell(\phi) = \sum_{t=1...T^g} \log(1 - D_\phi(z_t^g, c_t^g)) - \sum_{t=1...T^d} \log(D_\phi(z_t^d, c_t^d))$
    Update $\phi$ by a gradient method w.r.t. $\ell(\phi)$

**Return:** $\pi$

# Pipeline

Train low-level behavioral policy to match motion capture feature distribution

Train high-level task policy by RL using lower-level behaviors

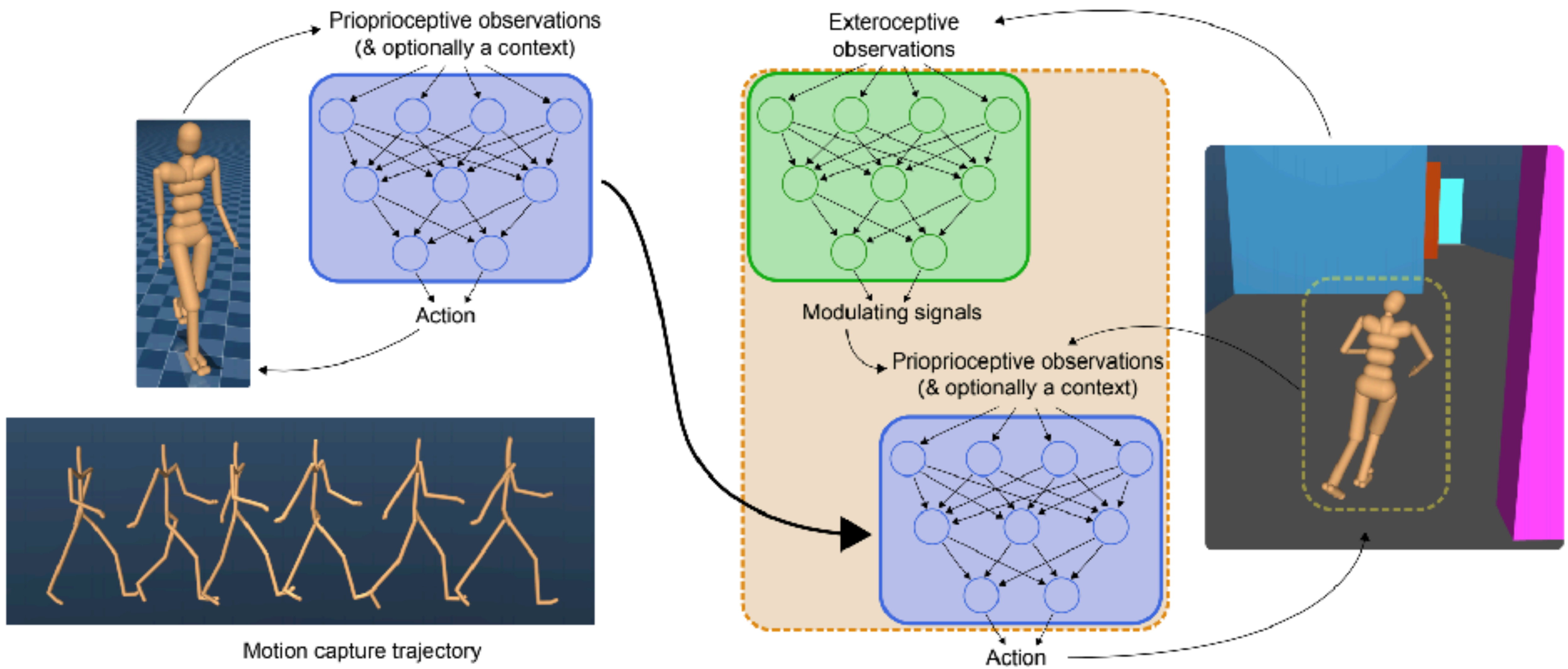
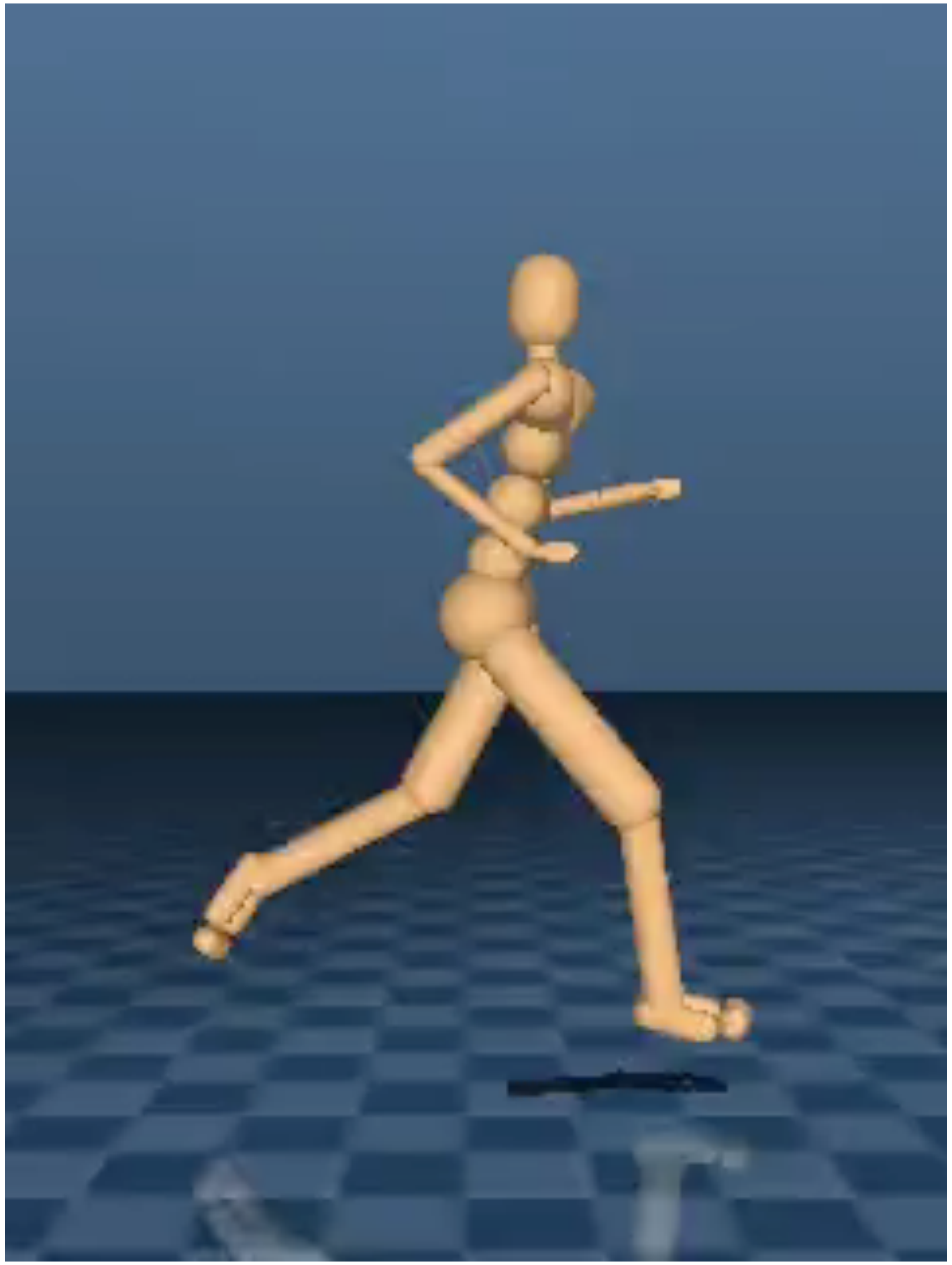
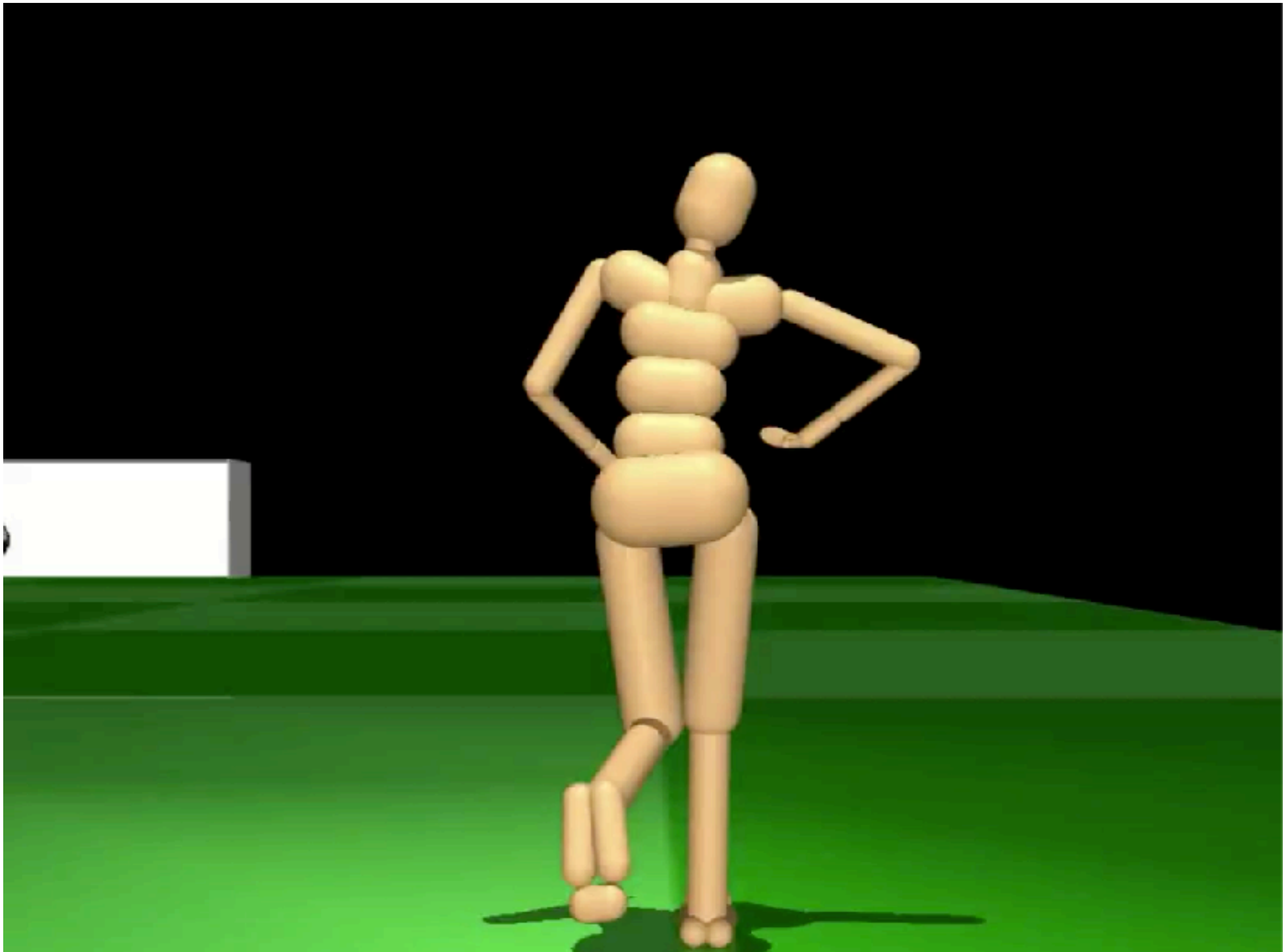



Figure 1: Overview of our approach: (Left) First train specific skills into low-level controller (LLC) policies by imitation learning from motion capture data. (Right) Train a high-level controller (HLC) by RL to reuse pre-trained LLCs.

# Running and Turning



# Running and Turning



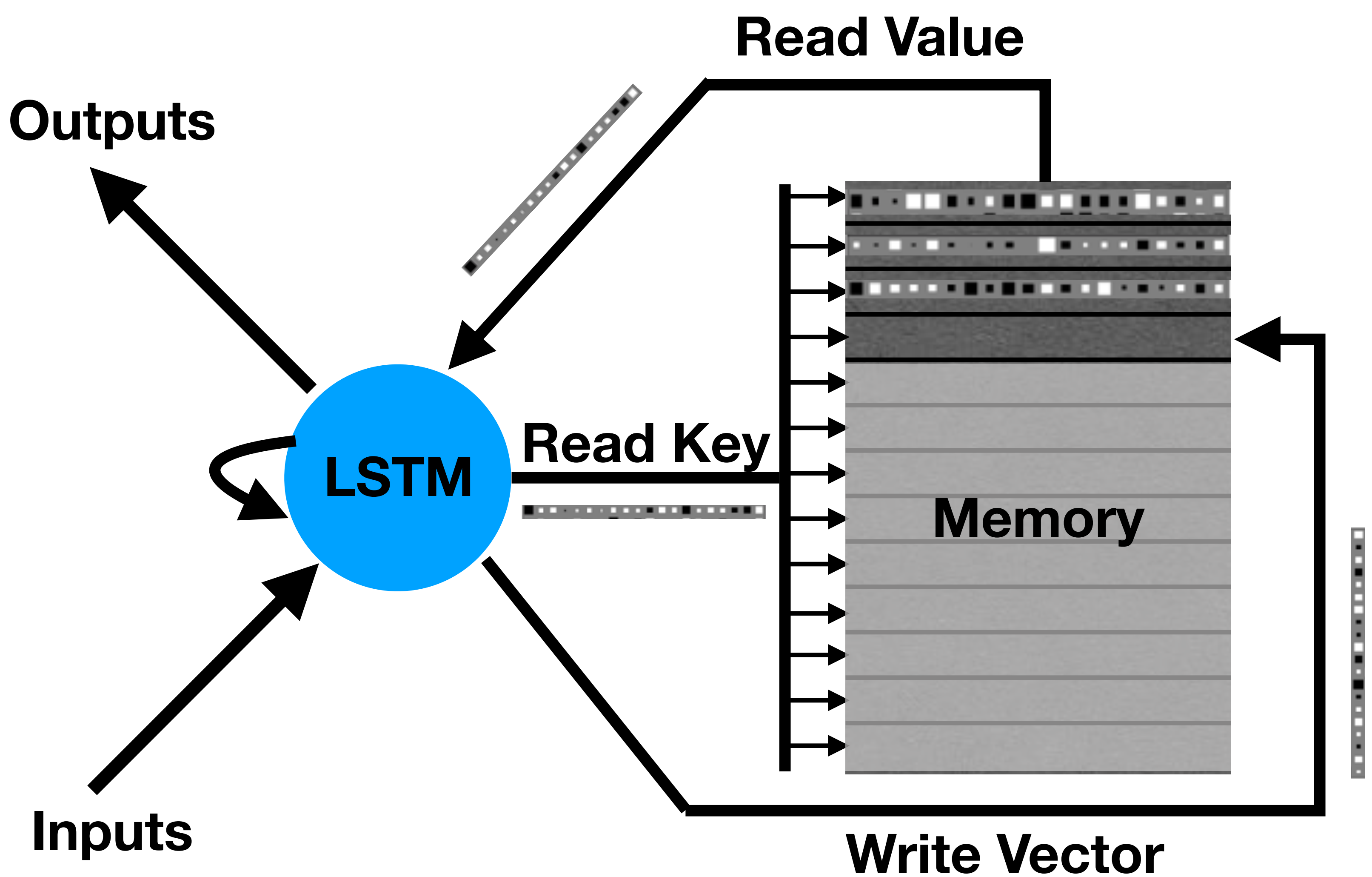
# Conclusions and Prospects

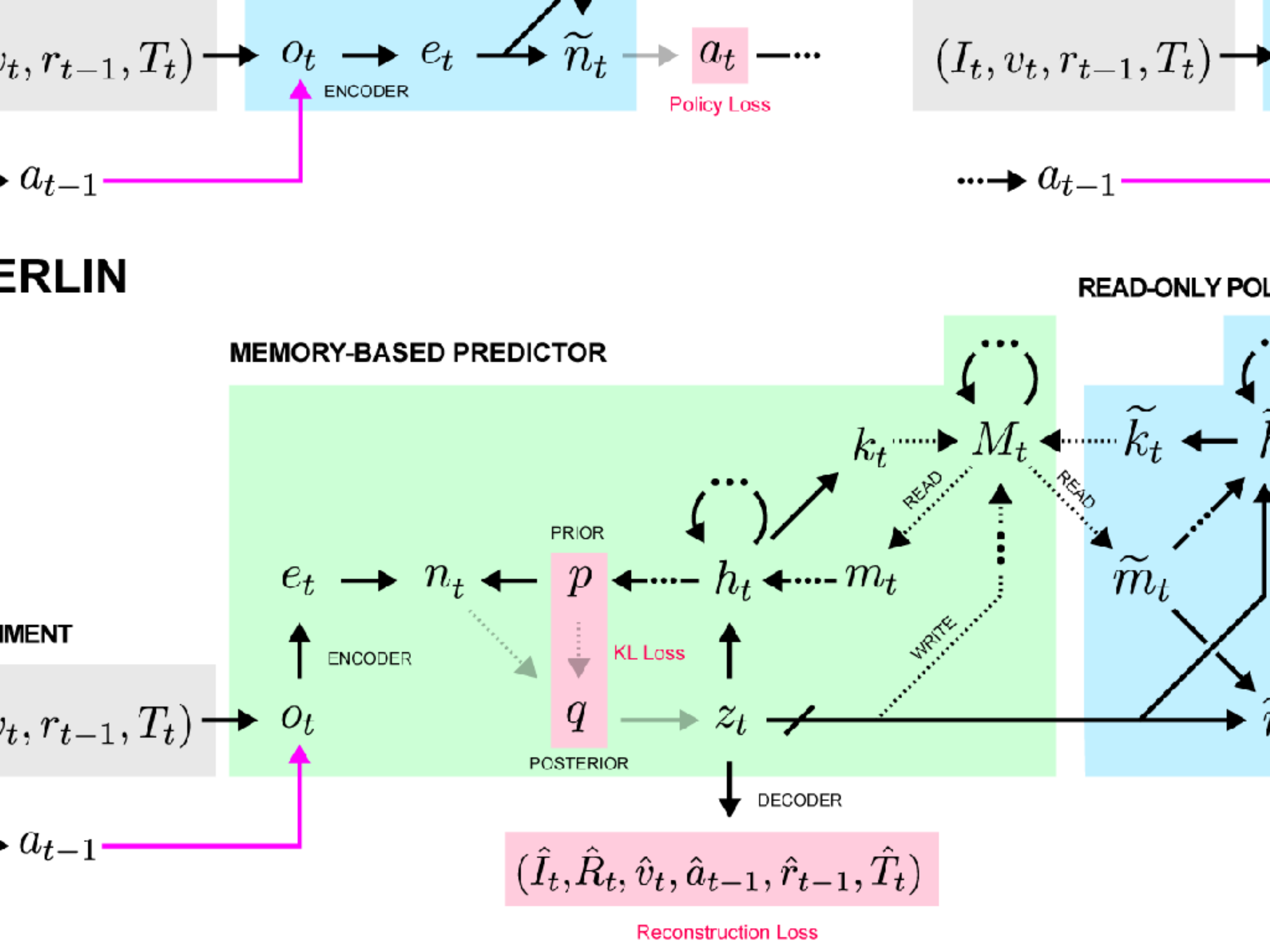
- MERLIN model can solve quite sophisticated tasks from raw sensory data in partially observed environments.
- Perception and memory formation are guided by a process of predictive modeling and compression, less by trial and error task success.
  - 200 dimensional z captures relevant features from approximately 10^4 sensory dimensions.
- Lessens the need for end-to-end gradient computation.
  - MERLIN uses a temporal credit assignment window of 1.3 seconds to solve memory tasks of 6 minutes.
- Can make use of information acquired without associated reward, exhibiting properties of latent learning.
- Provides a conceptual but **functional** model of the interaction of multiple neural systems in a complete, goal-directed cognitive architecture.
- GAIL can be scaled up to do imitation learning of non-obvious objectives.

# Noticed This

- Btw, sounds relevant: NEURAL SKETCH LEARNING FOR CONDITIONAL PROGRAM GENERATION (Murali et al., 2018)

# Backmatter


Read Value
Outputs
Read Key
LSTM
Memory
Inputs
Write Vector



ENCODER
Policy Loss
MEMORY-BASED PREDICTOR
READ
READ
PRIOR
KL Loss
WRITE
ENCODER
POSTERIOR
DECODER
$(\hat{I}_t, \hat{R}_t, \hat{v}_t, \hat{a}_{t-1}, \hat{r}_{t-1}, \hat{T}_t)$
Reconstruction Loss

Ext. Video 5 for Unsupervised Predictive Memory in a Goal-Directed Agent

**unsupervised learning / compressing sensory data**

**state variables — prior and posterior; instantiation in memory model**

**variational autoencoder framework**

**problem with pure unsupervised learning: bullet problem (gluck and myers)**

**scaling to challenging rl problems — typically use truncated backpropagation through time**

**Knock on benefit: can cut backpropagation time scales**

**figures: back matter learning curves for navigation tasks, one step prediction, unroll length ext. fig 10**