Proceedings of Machine Learning Research vol 168:1–15, 2022 4th Annual Conference on Learning for Dynamics and Control

Modeling Partially Observable Systems using Graph-Based Memory and Topological Priors

Steven D. Morad* SM2558@CAM.AC.UK
Stephan Liwicki† STEPHAN.LIWICKI@CRL.TOSHIBA.CO.UK
Ryan Korvelesy* RK627@CAM.AC.UK
Roberto Mecca† ROBERTO.MECCA@CRL.TOSHIBA.CO.UK
Amanda Prorok* ASP45@CAM.AC.UK

**Deptartment of Computer Science and Technology, University of Cambridge, Cambridge, UK*
†Toshiba Europe Limited, Cambridge, UK

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

Solving partially observable Markov decision processes (POMDPs) is critical when applying reinforcement learning to real-world problems, where agents have an incomplete view of the world. Recurrent neural networks (RNNs) are the defacto approach for solving POMDPs in reinforcement learning (RL). Although they perform well in supervised learning, noisy gradients reduce their capabilities in RL. This leads researchers to hand-design task-specific memory models to stabilize learning, based on their prior knowledge of the task at hand. In this paper, we present graph convolutional memory (GCM)[1], the first RL memory framework with swappable task-specific priors, enabling users to inject expertise into their models. GCM uses human-defined topological priors to form graph neighborhoods, combining them into a larger network topology. We query the graph using graph convolution, coalescing relevant memories into a context-dependent summary of the past. Results demonstrate that GCM outperforms state of the art methods on control, memorization, and navigation tasks while using fewer parameters.

Keywords: Reinforcement learning, POMDP, memory, graph neural networks

1. Introduction

RL is designed to solve *fully observable* Markov decision processes (MDPs) (Sutton and Barto, 2018, Chapter 3), where an agent knows its true state – a property that rarely holds in the real world. Problems where agent state is ambiguous, incomplete, noisy, or unknown violate the Markov property of MDPs (Russell and Norvig, 2010, Chapter 2.3.2), but can be modeled as POMDPs (Kaelbling et al., 1998). Recent literature even suggests that test-time domain shifts (e.g. simulation to reality) induce partial observability in otherwise fully observable domains (Ghosh et al., 2021). Operating on partially observable states strips optimal policy convergence guarantees from traditional RL methods like Q-learning or value iteration (Cassandra et al., 1994). By conditioning decisions on the *trajectory*, everything the agent has seen and done, we can restore the Markov property and convergence guarantees (Sutton and Barto, 2018, Chapter 17.3). The summarization of the ever-growing trajectory into a fixed-sized Markov state is known as *memory*.

1. Source code available at `https://github.com/proroklab/graph-conv-memory`

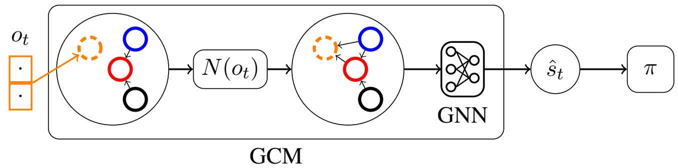


Figure 1: GCM flowchart for an incoming observation o_t. GCM places o_t as a node in a graph, and computes its neighborhood $N(o_t)$, and then updates the edge set. Task-specific topological priors are represented via the neighborhood. A convolutional GNN queries the graph for summary $\hat{s}_t$. A policy π uses the summary for decision making. Compared to modern memory models, GCM is conceptually simple.

In RL, memory-based models are either general or task-specific. Rooted in sequence learning, general memory consists of RNNs, transformers, or memory augmented neural networks (MANNs). Such models learn associations between observations and can be applied to any POMDP. Their drawback is longer training times, further exacerbated by the noisy learning signal in RL.

The substantial cost of training general memory drives many to design task-specific memory, like Chaplot et al. (2020); Parisotto and Salakhutdinov (2017); Gupta et al. (2017); Lenton et al. (2021) which build maps for navigation, or Li et al. (2018a) which utilizes past dosing information for hospital treatment. Other applications of task-specific memory include behavioral ecology, policy-making, questionnaire design (Cassandra, 1997), and robotics (Morad et al., 2021). Such memory is implemented from the ground up for each specific task, because there is no general memory framework to build upon. This puts task-specific memory out of reach of most non-experts. Our framework allows practitioners to create memory tailored to their specific task in just a few lines of code (see *Example Prior* in Sec. 3.2).

In this paper, we propose Graph Convolutional Memory (GCM), a general approach to leveraging task-specific prior knowledge for any partially observable task. The user embeds their task-specific knowledge into *topological priors*, which serve to accelerate and stabilize learning. GCM builds a graph and defines local neighborhoods using said priors, resulting in an expressive graph topology. Unlike past graph-based memory representations, we leverage the computational efficiency and representational power of graph neural networks (GNNs) to extract contextualized trajectory summaries from the graph. In our experiments, we show that with human-defined priors, GCM reliably solves tasks that RNNs, MANNs, and transformers cannot, while using significantly fewer parameters.

1.1. Contributions

- We propose the first task-agnostic, GNN-based memory architecture to solve partially observable RL tasks
- We are the first to suggest the use of swappable, human-designed memory priors in partially observable RL
- We explore the effect of memory priors, demonstrating the importance of selecting suitable priors for the task at hand

2. Related Work

General Memory in Reinforcement Learning We classify RNNs, MANNs, transformers, and related memory models as general memory. RNN-based architectures, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (GRU) (Chung et al., 2014) are used heavily in RL to solve POMDP tasks (Oh et al., 2016; Mnih et al., 2016; Mirowski et al., 2017). RNNs update a recurrent state by combining an incoming observation with the previous recurrent state. Compared to transformers and similar methods, RNNs fail to retain information over longer episodes due to vanishing gradients (Li et al., 2018b). By connecting relevant experiences directly and aggregating memories in a single forward pass, GCM sidesteps the vanishing gradient issue.

MANNs address limited temporal range of RNNs (Graves et al., 2014). Unlike RNNs, MANNs have addressable external memory. The differentiable neural computer (Graves et al., 2016) (DNC) is a fully-differentiable general-purpose computer that coined the term MANN. In the DNC, an RNN-based memory controller uses content-based addressing to read and write to specific memory addresses. The MERLIN MANN (Wayne et al., 2018) outperformed DNCs on navigation tasks. The implementations of the MANNs are much more complex than RNNs. In contrast to transformers or RNNs, MANNs are much slower to train, and benefit from more compute.

The transformer is the most ubiquitous implementation of self-attention (Vaswani et al., 2017). Until the gated transformer XL (GTrXL), transformers had mixed results in RL due to their brittle training requirements (Mishra et al., 2018). The GTrXL outperforms MERLIN, and by extension, DNCs in Parisotto et al. (2019). We can approximate the self-attention module in a transformer using a single graph attention layer over a fully-connected graph (Joshi, 2020). Unlike self-attention in transformers, GCM connectivity is discrete, sparse, and hierarchical.

Similar to our work, Savinov et al. (2018) build an observation graph, but specifically for navigation tasks, and do not use GNNs. Wu et al. (2019) use a probabilistic graphical model to represent spatial locations during indoor navigation. Eysenbach et al. (2019); Emmons et al. (2020) build a state-transition graph similar to our memory graph for model-based RL, but use A* to evaluate the graph. Unlike these methods, we evaluate the memory graph using GNNs, which are more efficient.

Graph Neural Networks GNNs are most easily understood using a message-passing scheme (Gilmer et al., 2017), where each vertex in a graph sends and receives latent messages from its neighborhood. Each layer in the GNN learns to aggregate incoming messages into a hidden representation, which is then shared with the neighborhood. Convolutional graph neural networks (Kipf and Welling, 2017) are a subcategory of GNNs and a generalization of convolutional neural networks (CNNs) to the graph domain. Convolutional GNNs tend to be efficient in both the computational and parameter sense due to their use of sliding filters and reliance on GPU-optimized instructions like batched sums and matrix multiplies.

Graph RNNs (Ruiz et al., 2020) are a generalization of RNNs to graph inputs with a fixed number of time-varying vertices, and tackle an entirely different problem than GCM. Chen et al. (2019); Li et al. (2019); Chen et al. (2020) apply GNNs to RL for task-specific problems. Beck et al. (2020) implement feature aggregation for RL in a similar fashion to GNNs. Zweig et al. (2020) combines a GNN with an RNN to tackle tasks with graph-based observations using reinforcement learning. To date, GCM is the only *task-agnostic* RL memory model to utilize GNNs.

3. Graph Convolutional Memory

We model a POMDP following Kaelbling et al. (1998) with tuple $(S, A, \mathcal{T}, R, \Omega, \mathcal{O})$. At time t we enter hidden state $s_t \in S$ and receive observation $o_t \sim \mathcal{O}(s_t) : S \to \Omega$. We sample action $a_t \in A$ from policy π and follow transition probabilities $\mathcal{T}(s_t, a_t) : S \times A \to S$ to the next state

s_{t+1}, receiving reward $R(s_t, a_t) : S \times A \rightarrow \mathbb{R}$. We learn π to maximize the expected cumulative discounted reward subject to discount factor γ: $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$. In an MDP, the policy uses the true state $\pi(s_t) : S \rightarrow A$, but in a POMDP we are not given s_t. Rather, we must construct an approximation $\hat{s}_t$ from the trajectory $\tau = o_1, \dots o_t$, and train some policy $\pi(\hat{s}_t) : S \rightarrow A$.

Our goal in this paper is summarize τ into $\hat{s}_t$. Note that o_t contains previous action a_{t-1} when necessary. Reasoning over all observations at each timestep is intractable, so we define a recurrent memory function M with the help of memory state m_t

$$(\hat{s}_t, m_t) = \mathrm{M}(o_t, m_{t-1}). \tag{1}$$

3.1. Model Description

We implement GCM following Eq. 1 using Alg. 1. GCM stores a collection of experiences over an episode, with each experience represented by an observation vertex o and associated neighborhood $N(o)$. We query the set of experiences using a graph neural network (GNN) to produce a context-dependent summary $\hat{s}_t | o_t$.

Algorithm 1 Graph Convolutional Memory

```
1: procedure M(o_t, m_{t-1})
2:     V, E ← m_{t-1}                        ▷ Unpack memory
3:     V ← V ∪ o_t                           ▷ Add observation to graph
4:     E ← E ∪ {(o_t, o_i)}_{i∈N(o_t)}       ▷ Add obs edges
5:     Z ← GNN_θ(V, E)                       ▷ Get embedding
6:     ŝ_t ← Z[t]                            ▷ At current vertex
7:     m_t ← V, E                            ▷ Pack into memory
8:     return ŝ_t, m_t                       ▷ Summary and memory
9: end procedure
```

In detail, at time t, we insert vertex o_t into the graph, producing $m_t = (V_t, E_t)$ where $V_t = (o_1, \dots o_t)$ and $E_t : 2^{V_t}$.[2] We determine the neighborhood $N(o_t)$ using *topological priors* defined in Sec. 3.2, and update the edges following:

$$E_t = E_{t-1} \cup \{(o_t, o_i) \mid i \in N(o_t)\} \tag{2}$$

We query the graph for context-dependent information using a GNN with layers $h \in \{1 \dots \ell\}$. We convolve over $o_1, \dots o_t$ to produce hidden representations $z_1^h, \dots z_t^h$ for each hidden layer, propagating information from the h^{th}-degree neighbors of o_t into z_t^h. After collecting and integrating data across the ℓ^{th}-degree neighborhood, we output z_t^ℓ as the summary $\hat{s}_t$. This provides a mechanism for fast and relevant feature aggregation over memory graphs, depicted in Fig. 2. As an example, let us consider some control task where the observation is agent pose, and the neighborhood consists of the previous observation $N(o_t) = \{t-1\}$. Then, the first GNN layer combines agent poses o_1, o_2 and o_2, o_3 to estimate velocities z_2^1 and z_3^1 respectively. The second GNN layer combines velocities z_2^1, z_3^1 to output acceleration z_2^3 as the summary.

We found the 1-GNN defined in Morris et al. (2019) empirically outperformed graph isomorphism networks (Xu et al., 2019) and the original graph convolutional network (Kipf and Welling, 2017). GCM can utilize any GNN, but our GNNs are built from the 1-GNN convolutional layer defined as:

$$z_t^h = \sigma\left[W_1^h z_t^{h-1} + b^h + W_2^h \texttt{agg}\left(\left\{z_i^{h-1} | i \in N(o_t)\right\}\right)\right] \tag{3}$$

with σ representing a nonlinearity and $z_t^0 = o_t, z_i^0 = o_i$ for the base case. At each layer h, weights and biases W_1^h, b^h produce a root vertex embedding while W_2^h generates a neighborhood embedding using vertex aggregation function $\texttt{agg}$. Separate weights allow the 1-GNN to independently weigh each h^{th}-degree neighborhood's contribution to the summary, ignoring the neighborhood and

2. For an episode one thousand timesteps long, an adjacency matrix would use $1000^2 \cdot 4\text{B} = 4\text{MB}$, while an edgelist representation with a neighborhood size of 10 would use $1000 \cdot 10 \cdot 2 \cdot 4\text{B} = 160\text{kB}$. Using observations of 128 dimensions, the vertex matrix V in both cases would be $1000 \cdot 128 \cdot 4\text{B} = 512\text{kB}$

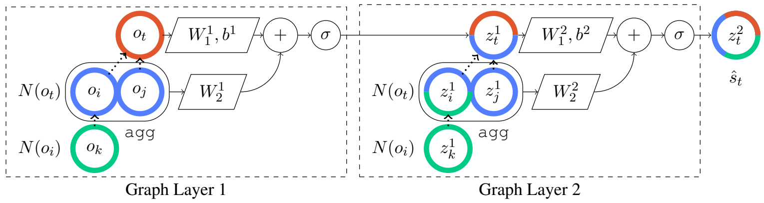


Figure 2: The two-layer 1-GNN used in all our experiments. Colors denote mixing of vertex information and dashed lines denote directed edges, forming neighborhoods $N(o_t), N(o_i)$. The current observation o_t and aggregated neighboring observations o_i, o_j pass through fully-connected layers $(W_1^1, b^1), (W_2^1)$ before summation and nonlinearity σ, resulting in the first hidden state z_t^1 (Eq. 3). We repeat this process at o_j, o_k, o_l to form hidden states z_j^1, z_k^1, z_l^1. The second layer combines embeddings of the first layer and the second-layer hidden state z_t^2 is output as the summary $\hat{s}_t$. Additional layers increase the GNN receptive field.

decomposing into an MLP for empty or uninformative neighborhoods. The root and neighborhood embeddings are combined to produce the layer embedding z_t^h (Fig. 2). Notice, the weights W_1^h, b^h in Eq. 3 form an MLP, so GCM does not require an MLP preprocessor like other memory models (Mnih et al., 2016).

3.2. Topological Priors

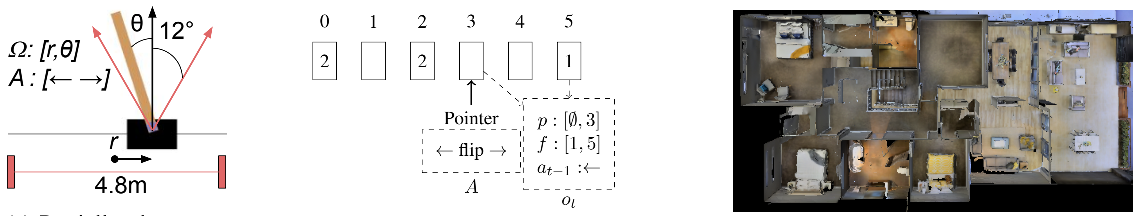
Topological priors determine the neighborhood at each specific vertex, which in turn determines the underlying graph connectivity. More formally, topological priors are a mapping from a vertex to a neighborhood. We use shorthand $N(o_t)$ to define the open neighborhood of o_t over vertices V_t, in edge-list format. We compute $N(o_t)$ using the union of k topological priors $\Phi_i : \Omega^t \to 2^{V_{t-1}}$, as in

$$N(o_t) : V \to 2^{V_{t-1}} := \bigcup_{i=1}^{k} \Phi_i(V_t). \tag{4}$$

Breaking down the graph connectivity problem into easier neighborhood-forming subtasks is reminiscent of dynamic programming. Tasks may require different priors – associating memories temporally is useful in control tasks, but spatial associations are more powerful in navigation tasks. We have implemented spatial, temporal, latent similarity, and other topological priors in Tab. 1.

Topological Prior Description	$\Phi(V_t)$ Definition	
Empty: o_t has no neighbors and GCM decomposes into an MLP.	$\emptyset$	(5)
Dense: Connects o_t to all other observations $o_1 \dots o_{t-1}$.	$\{1, 2, \dots t-1\}$	(6)
Temporal: Similar to the temporal prior of an LSTM, where each observation o_t is linked to some previous $t-c$ observation.	$\{t-c\}$	(7)
Spatial: Connects observations taken within c meters of each other, useful for problems like navigation. Let $\mathrm{p}(\cdot)$ extract the position from an observation.	$\left\{ i \,\middle\vert\, \begin{array}{c} \lVert\mathrm{p}(o_i) - \mathrm{p}(o_t)\rVert_2 \leq c \\ \text{and } 0 < i < t \end{array} \right\}$	(8)
Latent Similarity: Links observations in a non-human readable latent space (e.g. autoencoders). Various measures like cosine or L_2 distance may be used depending on the space. e is an encoder function, d is a distance measure, and c is user-defined.	$\left\{ i \,\middle\vert\, \begin{array}{c} \mathrm{d}\left(\mathrm{e}\left(o_i\right), \mathrm{e}\left(o_t\right)\right) < c \\ \text{and } 0 < i < t \end{array} \right\}$	(9)
Identity: Connects observations where two values are identical, useful in discrete domains where inputs are related. a, b are indexing functions ($\mathrm{a} = \mathrm{b}$ may hold).	$\left\{ i \,\middle\vert\, \begin{array}{c} \mathrm{a}(o_i) - \mathrm{b}(o_t) = 0 \\ \text{and } 0 < i < t \end{array} \right\}$	(10)

Table 1: Knowledge-based priors for GCM



(a) Partially obs. cartpole

(b) Concentration card game

(c) Habitat 3D simulation

Figure 3: Visualizations of our experiments. (a) The classic cartpole control problem, but where $\dot{r}, \dot{\theta}$ are hidden. (b) An example state from the long-term non-sequential recall environment with six cards. The observation space o_t contains the value and index of pointer card p and last flipped card f, as well as previous action a_{t-1}. (c) The top-down view of the 3D scene used in our navigation experiment.

Example Prior To demonstrate how easy it is to write task-specific topological priors, we provide an example in Pytorch. Assume we are learning a satellite control policy for collision avoidance in Low Earth Orbit. Each time we detect a new piece of space debris, we receive an observation containing the estimated orbital parameters of said debris, and may act to perturb our orbit.

```
1  import torch
2  class OrbitalPrior(torch.nn.Module):
3    tolerable_risk = 1e-4 # Probability of collision
4
5    def forward(self, V, *args, **kwargs):
6      coll_probs = self.compute_collision_risk(V)
7      risky_debris = coll_probs > self.tolerable_risk
8      neighborhood = risky_debris.nonzero().squeeze()
9      return neighborhood
10
11 gcm.edge_selectors = OrbitalPrior()
```

We omit the batch dimension for clarity. Line 6 computes future orbits and returns collision probabilities with each piece of tracked debris (each row in V). Line 7 determines pieces of debris outside the acceptable collision risk – these are the objects we want to focus on. Line 8 returns the indices of these risky object in V, which serve as the neighborhood (Eq. 4). Line 11 adds our custom prior to GCM. The ESA estimates there are 36,500 pieces of orbital debris bigger than 10cm, making naive memory approaches intractable. We can inject our knowledge of orbital mechanics into GCM in just a few lines of code, drastically reducing the search space.

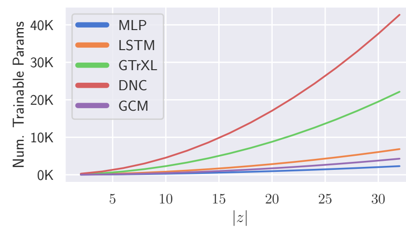
4. Experiments

We evaluate GCM on control, card games, and indoor navigation. We run three trials for each memory model across all experiments and report reward mean and standard deviation per train batch. We test five contrasting models, and base our evaluation on the hidden size of the memory models, denoted as $|z|$ in Fig. 4. Nearly all hyperparameters are Ray RLlib defaults, tuned for RLlib's built-in models (Tab. 2).[3] Tab. 2 contains all training hyperparameters.

3. The MLP, LSTM, DNC, and GTrXL are standard Ray RLlib (Liang et al., 2018) implementations written in Pytorch (Paszke et al., 2019). We implement GCM using Pytorch Geometric (Fey and Lenssen, 2019), and integrate it into RLlib.

We compare GCM to an MLP and three alternative memory models in all our experiments. The **MLP** model is a two-layer feed-forward neural network using $\tanh$ activation. It has no memory, and forms a performance lower bound for the memory models. The **LSTM** memory model is a MLP followed by an LSTM cell, the standard model for solving POMDPs (Mnih et al., 2016). **GTrXL** is a MLP followed by a single-head GRU-gated transformer XL. The **DNC** is an MLP followed by a neural computer with an LSTM-based memory controller. Our memory model, **GCM**, uses a two-layer 1-GNN using $\tanh$ activation with sum (cartpole and concentration) and mean (navigation) neighborhood aggregation.

Partially Observable Cartpole Our first experiment evaluates memory in the control domain. We use a partially observable form of cartpole-v0 (Barto et al., 1983; Brockman et al., 2016), where the velocities are hidden and only positions are visible (Fig. 3a). We optimize our policy using proximal policy optimization (PPO) Schulman et al. (2017). The equations of motion for the cartpole system are a set of second-order differential equations containing the position, velocity, and acceleration of the system (Barto et al., 1983). Using this information, we use GCM with temporal priors, i.e., $N(o_t) = \{t-1, t-2\}$ (Tab. 1) and present results in Fig. 5a.



(a)

Model	Meaning of $\lvert z\rvert$
MLP	Layer size
LSTM	Size of hidden and cell states
GTrXL	Size of the K, Q, V MLPs
DNC	LSTM size, word width, and number of memory cells
GCM	Size of the graph layers

(b)

Figure 4: (a) The number of trainable parameters per memory model, based on the hidden size $|z|$. (b) The meaning of $|z|$ with respect to each memory model, as used in all our experiments.

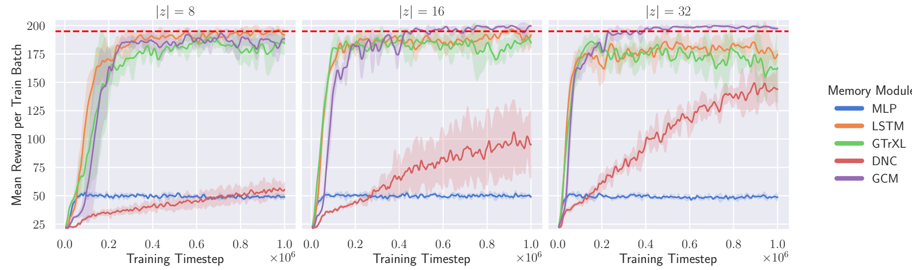
Concentration Card Game Our next experiment evaluates non-sequential and long-term recall with the concentration card game.[4] Unlike reactionary cartpole, this experiment tests memorization and recall over longer time periods. The agent is given $n/2$ pairs of shuffled face-down cards, and must flip two cards face up. If the cards match, they remain face up, otherwise they are turned back over again. Once the player has matched all the cards, the game ends. We model the game of memory using a *pointer*, which the player moves to read and flip cards (Fig. 3b). The observation space consists of the pointer (card index and card value), the last flipped (if any) face-up card, and the previous action. Cards are represented as one-hot vectors. The agent receives a reward for each pair it matches, with a cumulative reward of one for matching all the cards. We vary the number of cards $n \in \{8, 10, 12\}$ with episodes lengths of $50, 75, 100$ respectively. All models have $|z| = 32$ and train using PPO. We use GCM with temporal priors for short-term memory and an additional value identity prior between the face-up card and the card at the pointer, using function $v : \Omega \to \mathbb{N}$:

$$N(o_t) = \{t-1, t-2\} \cup \{i \mid v(o_t) = v(o_i)\}. \tag{11}$$

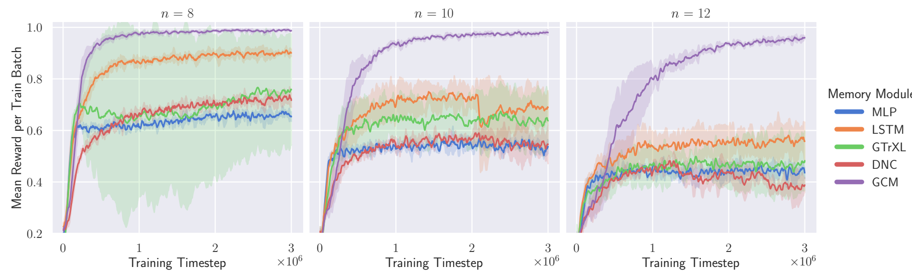
We present the results in Fig. 5b.

Navigation The final experiment evaluates spatial reasoning with a navigation task. We use the Habitat 3D simulator with the validation scene from the 2020 Habitat Challenge (Fig. 3c). The observation space consists of an autoencoded depth image, agent coordinates and angle relative to

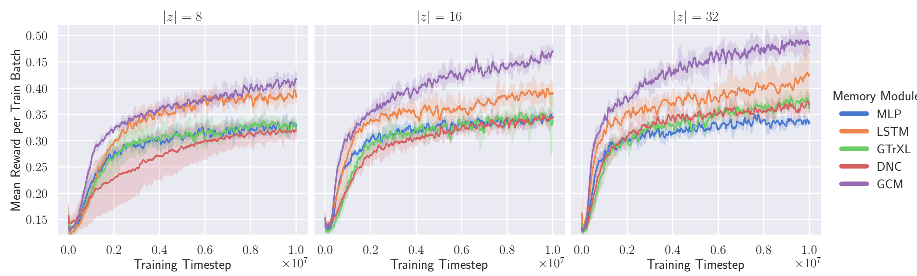
4. Rules for concentration are available at: https://en.wikipedia.org/wiki/Concentration_(card_game)



(a) Partially observable cartpole. The red line denotes OpenAI's success criteria for cartpole-v0.



(b) Concentration card game. n is the number of cards and $|z| = 32$ for all three plots.



(c) 3D navigation

Figure 5: Comparison of GCM to other memory models across three different environments. $|z|$ denotes the hidden size used across all models. Lines represent the mean and shaded areas represent one standard deviation over three trials.

start, and the previous action. We train for 10M timesteps using IMPALA (Espeholt et al., 2018), examining $z \in \{8, 16, 32\}$ across all models. Fig. 6 is a GCM ablation study across multiple topological priors. We evaluate the effectiveness of empty, dense, temporal, spatial, and learned priors (formally defined in Tab. 1). We also examine whether we k^{th} order neighbors are helpful using the FlatSpatial entry. FlatSpatial is the spatial prior, but with the second graph layer replaced with a fully-connected layer of equal size. This helps us determine whether GCM benefits from the broader graph topology or just relies on local neighborhoods.

5. Discussion

The versatility of GCM compared to other models stems from its representation of experiences as a graph. This allows it to access specific observations from the past, bypassing the limited temporal range of LSTM. By using a multilayer GNN to reason over this graph of experiences, GCM can build embeddings hierarchically, unlike transformers. The importance of hierarchical reasoning is demonstrated experimentally in Fig. 6, where the GCM outperforms the FlatSpatial GCM, which

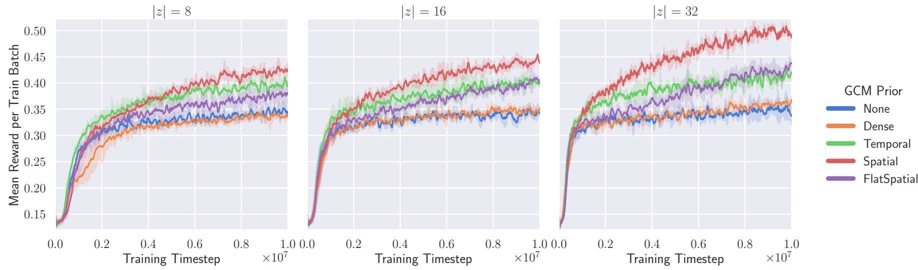


Figure 6: We compare various GCM priors across hidden sizes $|z|$ for the navigation problem. Since navigation is a spatial problem, the spatial prior performs best. This shows the importance of selecting good priors. The solid lines denote the mean reward per batch and the shaded regions represent the standard deviation.

Table 2: Hyperparameters for the experiments and their RLlib defaults. Non-default values are underlined.

Term	IMPALA Default	Navigation	PPO Default	Cartpole	Concentration
Decay factor γ	0.99	0.99	0.99	0.99	0.99
Value fn. loss coef.	0.5	0.5	1.0	1e-5	1.0
Entropy loss coef.	0.01	0.001	0.0	0.0	0.0
Gradient clipping	40.0	40.0	-	-	-
Value function clipping	-	-	10	10	10
Learning rate	0.005	0.005	5e-5	5e-5	3e-4
Num. SGD iters	1.0	1.0	30	30	30
Exp. replay ratio	0:1	1:1	-	-	-
Batch size	500	1024	4000	4000	4000
Minibatch size	-	-	128	128	4000
GAE λ	1.0	1.0	1.0	1.0	1.0
V-trace ρ	1.0	1.0	-	-	-
KL target	-	-	0.01	0.01	0.01
KL coefficient	-	-	0.2	0.2	0.2
PPO clipping	-	-	0.3	0.3	0.3
Value clipping	-	-	0.3	0.3	0.3

only utilizes the first degree neighborhood from the spatial prior. This implies the second-order neighbors meaningfully contribute to the summary.

Like Beck et al. (2020), we find sequence learning is much harder in RL than supervised learning. This is particularly clear in Fig. 5b, where introducing one more pair of cards significantly decreases reward. Although general memory models can learn optimal policies in theory, this was not the case given our timescales. The LSTM performs well but does not reliably solve (i.e. reach 195 reward) stateless cartpole, even with small 2-dimensional observation and action spaces, and a large number of inner and outer PPO iterations (Heess et al. (2015), Fig. 5a). Even though transformers significantly outperform LSTMs in supervised learning (Vaswani et al., 2017), their added complexity seems to hinder them in RL, at least at single-GPU scales. The memory search space over all past observations is huge, and determining which observations are useful greatly reduces what the memory model must learn.

GCM's graph structure can utilize external information about which experiences are relevant, greatly reducing the search space. Human intuition is an incredibly useful tool that cannot be easily leveraged by transformers, RNNs, or MANNs. This is the key contribution of our work – a prior defined by a few lines of code can accelerate and stabilize learning. GCM provides an easy way to embed this intuition, using more general priors (Tab. 1) or task-specific priors (Sec. 3.2).

In our experiments, we use simple environments to demonstrate how model-dependent memory connectivity affects performance. Models like LSTM work nearly as well as GCM on problems like cartpole where a temporal prior makes sense (Fig. 5a), but the gap widens on the concentration environment where non-temporal priors are more suitable (Fig. 5b). The navigation ablation study (Fig. 6) demonstrates how using a suboptimal topological prior can negatively impact performance – the dense prior (a fully-connected graph) performs nearly as poorly as the empty prior (no edges at all) in Fig. 6.

A drawback of our approach compared to general models is that it requires human input in form of a topological prior. However, the temporal prior in Fig. 6 performs similarly to LSTM in Fig. 5c across all hidden sizes on the navigation task, even though navigation is primarily a spatial task. This suggests that we could apply the temporal GCM to arbitrary sequential decision making tasks without having prior knowledge, similar to LSTM. In the future, we plan to learn topological priors from data – a relatively difficult task due to the large space of possible edges and their discrete, non-differentiable nature.

GCM is significantly more interpretable than RNNs, transformers, or MANNs. RNNs mutate a hidden state over time, making it unclear which observations contribute to the hidden state. MANNs, which utilize an RNN in the memory controller as well as external latent memory, are even more opaque. In transformers, the softmaxed attention weights mean all past observations contribute a non-zero amount to the current decision. Slight perturbations of attention weights produce completely different results (Wiegreffe and Pinter, 2020). The graph structure of GCM makes interpretability simple. The observations which contribute to a specific decision are precisely the ℓ-degree neighborhood of vertex o_t. The observations V are not modified, so we are left with a small subgraph of unmodified observations at each timestep directly responsible for the current decision.

Across all experiments, GCM with human expertise received significantly more reward than all tested models. We believe that this is remarkable, considering that GCM uses notably fewer parameters than the other models (Fig. 4). Caveat emptor: we tackled simple tasks using smaller models, due to our limited computational capacity. These conclusions might not hold for those who train markedly larger models for billions of timesteps.

6. Conclusion

In this paper, we introduced GCM – the first general, GNN-based memory architecture for RL. GCM provides a framework to easily embed task-specific priors into memory in just a few lines of code. Surprisingly, we found that the transformer, DNC, and LSTM struggle to learn effective memory representations even in simple tasks, such as the concentration card game. We empirically demonstrated the importance of selecting good priors, with unsuitable memory priors performing similarly to memory-free models. We also found that the hierarchical properties of multilayer GNNs were a significant contributor to model performance. When little is known about the task at hand, general memory models like RNNs are a good choice. But when even basic domain knowledge is available (e.g., when the problem is spatial, or when it follows Newton's laws) GCM outperforms transformers, LSTM, and DNCs, while using significantly fewer parameters.

7. Acknowledgements

Steven Morad, Stephan Liwicki, and Roberto Mecca gratefully acknowledge the support of Toshiba Europe Ltd. Ryan Kortvelesy was supported by Nokia Bell Labs through their donation for the Centre of Mobile, Wearable Systems and Augmented Intelligence to the University of Cambridge. Amanda Prorok was supported by ERC Project 949940 (gAIa). We thank Jan Blumenkamp for their helpful discussions.

References

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Jacob Beck, Kamil Ciosek, Sam Devlin, Sebastian Tschiatschek, Cheng Zhang, and Katja Hofmann. AMRL: Aggregated Memory For Reinforcement Learning. *International Conference on Learning Representations (ICLR)*, pages 1–14, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. jun 2016.

Anthony R Cassandra. A Survey of POMDP Applications. *Uncertainty in Artificial Intelligence*, pages 472–480, 1997.

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the National Conference on Artificial Intelligence*, volume 2, pages 1023–1028, 1994.

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to Explore using Active Neural SLAM. In *International Conference on Learning Representations (ICLR)*. arXiv, apr 2020.

Fanfei Chen, John D. Martin, Yewei Huang, Jinkun Wang, and Brendan Englot. Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. arXiv, jul 2020.

Kevin Chen, Juan Pablo de Vicente, Gabriel Sepúlveda, Fei Xia, Alvaro Soto, Marynel Vázquez, and Silvio Savarese. A behavioral approach to visual navigation with graph localization networks. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, 2019. doi: 10.15607/rss.2019.xv.010.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

Scott Emmons, Ajay Jain, Michael Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *Advances in Neural Information Processing Systems*, volume 2020-Decem, 2020.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Boron Yotam, Firoiu Vlad, Harley Tim, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *35th International Conference on Machine Learning, ICML 2018*, volume 4, 2018.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. *arXiv*, 2019. URL http://arxiv.org/abs/1903.02428.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. 2021. URL http://arxiv.org/abs/2107.06277.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, pages 2053–2070. PMLR, jul 2017. ISBN 9781510855144. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. oct 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, oct 2016. doi: 10.1038/nature20101.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 7272–7281. Institute of Electrical and Electronics Engineers Inc., nov 2017. doi: 10.1109/CVPR.2017.769.

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. dec 2015. URL https://arxiv.org/abs/1512.04455v1http://arxiv.org/abs/1512.04455.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Chaitanya Joshi. Transformers are Graph Neural Networks, feb 2020. URL https://graphdeeplearning.github.io/post/transformers-are-gnns/.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998. ISSN 00043702. doi: 10.1016/s0004-3702(98)00023-x.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Daniel Lenton, Stephen James, Ronald Clark, and Andrew J. Davison. End-to-End Egospheric Spatial Memory. In *International Conference on Learning Representations*, sep 2021. URL http://arxiv.org/abs/2102.07764.

Dong Li, Qichao Zhang, Dongbin Zhao, Yuzheng Zhuang, Bin Wang, Wulong Liu, Rasul Tutunov, and Jun Wang. Graph attention memory for visual navigation, may 2019.

Luchen Li, Matthieu Komorowski, and Aldo A. Faisal. The Actor Search Tree Critic (ASTC) for Off-Policy POMDP Learning in Medical Decision Making. *arXiv preprint arXiv:1805.11548*, 2018a. URL http://arxiv.org/abs/1805.11548.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018b. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00572.

Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *35th International Conference on Machine Learning, ICML 2018*, volume 7, pages 4768–4780, 2018.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Volodymyr Mnih, Adria Puigdomenech Badia, Lehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4, pages 2850–2869, 2016. ISBN 9781510829008.

Steven D. Morad, Roberto Mecca, Rudra P.K. Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied Visual Navigation with Automatic Curriculum Learning in Real Environments. *IEEE Robotics and Automation Letters*, 6(2):683–690, apr 2021. doi: 10.1109/LRA.2020.3048662.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602.

Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of Memory, Active Perception, and Action in Minecraft. Technical report, jun 2016.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Vancouver, feb 2017. arXiv.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, oct 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated Graph Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020. ISSN 19410476. doi: 10.1109/TSP.2020.3033962.

Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach Third Edition*. 2010. doi: 10.1017/S0269888900007724.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009, 2017.

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, and Timothy Lillicrap. Unsupervised Predictive Memory in a Goal-Directed Agent. *arXiv*, mar 2018.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2020. doi: 10.18653/v1/d19-1002.

Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 2769–2779. Institute of Electrical and Electronics Engineers Inc., oct 2019. doi: 10.1109/ICCV.2019.00286.

Keyulu Xu, Stefanie Jegelka, Weihua Hu, and Jure Leskovec. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Aaron Zweig, Nesreen K Ahmed, Ted Willke, and Guixiang Ma. Neural Algorithms for Graph Navigation. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*, pages 1–11, oct 2020. URL https://openreview.net/pdf?id=sew79Me0W0c.