Graph Embedding Priors for Multi-task Deep Reinforcement Learning

Neev Parikh[†][*] **Zachary Horvitz**[‡][†] **Naveen Srinivasan**[‡][†] **Aansh Shah**[‡]

George Konidaris[§]

Abstract

Humans appear to effortlessly generalize knowledge of similar objects and relations when learning new tasks. For example, humans playing Minecraft can learn how to use a tool to mine one block, then rapidly generalize that skill to mine others. We leverage graph-encoded object priors to capture this property and improve the performance of reinforcement learning agents across multiple tasks. We introduce a novel, flexible architecture that utilizes graph convolutional networks (GCNs), which provide a natural method to combine relational information over connected nodes. We evaluate our approach on a procedurally-generated, multi-task environment: *Symbolic Procgen*. Our experiments demonstrate that the method generalizes across many tasks and scales to domains with hundreds of objects and relations. Additionally, we perform ablation studies that demonstrate robustness to noisy graph priors, suggesting that the method is suitable for leveraging graphs generated from large, unstructured sources of knowledge in real-world settings.

1 Introduction

Deep reinforcement learning (RL) has achieved tremendous success over the last few years, with advances in Go [22], Chess, and Shogi [23], as well as video game domains like the Atari 2600 suite [17]. However, these incredible advances require an equally incredible amount of data: millions of frames of training data and many hundreds of hours of compute resources. Humans, on the other hand, are far more efficient at learning how to solve novel tasks [2, 16]. Prior work has hypothesized that this gap can be attributed to prior knowledge and built-in inductive biases. Humans do not learn new tasks from scratch [4, 5]; instead, they rely on millions of years of evolution and a wealth of collective knowledge: universal, task-agnostic priors [9]. Effectively capturing such universal priors is integral to closing the sample-efficiency gap.

Owing to their ability to represent concepts and relations, knowledge graphs (KGs) are an effective vehicle for modeling these priors [1, 30, 31]. These graph representations can capture shared structure across tasks, allowing the agent to reuse information learned from one task and generalize efficiently to others. Prior work in multi-task RL has emphasized the utility of using shared representations across tasks [6, 7, 13, 27, 28, 29]. We seek to explore the intersection of these two complementary approaches: using universal priors in multi-task settings to capture task-agnostic structure.

To best leverage universal priors, we desire a model with several critical properties. First, we believe our approach should induce *task invariant* object representations that encode universal relationships between objects rather than task-specific state information. Secondly, we prioritize *parameter*

[*]Corresponding author: `neev_parikh@brown.edu`

[†]Equal contribution

[‡]`{first_last}@alumni.brown.edu` [§]`gdk@cs.brown.edu`

efficiency: our approach should scale to the hundreds of objects that characterize real-world, multi-task settings. Lastly, we emphasize *modularity*: to be maximally useful, our embedding approach should flexibly augment existing methods. To this end, we introduce a novel architecture that exploits graph-based priors to solve hundreds of simultaneous tasks. We evaluate our approach on domains with hundreds of relationships and observe robust, sample-efficient generalization. Our approach offers a promising route to incorporating universal priors in multi-task RL (MTRL) domains.

2 Background

Taylor & Stone [27] identify that there are many different possible formulations for what constitutes a multi-task reinforcement learning setting. For our purposes, we consider an MTRL environment to comprise a broader “universe" defined by a set of states, consistent transition dynamics, and a single set of actions. Each distinct task is defined by a unique reward function.

Many real world domains are best described as a collection of objects and their interactions. One approach to modeling these environments is via object-oriented Markov decision processes (OO-MDPs). Diuk et al. [8] extend Markov decision processes (MDPs) [20] by defining a collection of classes and a set of attributes. The environment is composed of objects: specific instances of classes. Our model simultaneously learns over multiple OO-MDPs that share objects and transition dynamics.

Given a collection of objects, a natural way to include prior knowledge is via a knowledge graph (KG), a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex can represent an object in the environment and each edge represents a relationship between two objects. A common approach is to store information as a node embedding $h_v \in \mathbb{R}^k$ that is associated with a vertex $v \in \mathcal{V}$. Agents can utilize these embeddings to reason about their environment at the level of objects.

GraphDQN [30] applies knowledge graphs and relational priors to generalize known relationships across various objects to solve symbolic environments. The agent maintains node embeddings for objects in the world, updating these representations using a combination of convolutional filters over the state and a series of pooling, broadcast, and graph convolution operations [15] over the knowledge graph. While GraphDQN achieves impressive, sample-efficient results in single-task symbolic domains, we show that such an architecture cannot scale to domains with many tasks.

3 Graph Embedding Priors for Multi-task RL

In the multi-task setting, learning an embedding for objects using state information can result in destructive interactions between different task-specific updates [29]. To efficiently scale up to domains with many possible object relations, we introduce a model, GEM-RL, that encourages a task-invariant object representation, jointly learning weights to combine node embeddings via a GCN-based encoder (see Fig. 1) [1]. In addition to improved parameter efficiency, GEM-RL's simplicity and flexibility render it ideal for use with abstract knowledge priors and on more complex, diverse MTRL tasks.

Our GCN-based encoder is initialized with a set of objects and an adjacency matrix that describes the relationships between objects in our environment. We explore both random and DeepWalk [19] initializations of our object embeddings (See Appendix). The encoder computes the mean of the neighborhood node embeddings to generate the aggregated node embedding $a_v^{(l)}$. The encoder then multiplies aggregated node embedding by a learnable weight matrix, then applies a non-linear activation (ReLU) to yield the combined node embedding $h_v^{(l+1)}$ [15]. Concretely, $a_v^{(l)} = \text{mean}\left(\left\{h_u^{(l)}, u \in \mathcal{N}(v) \cup \{v\}\right\}\right)$ and $h_v^{(l+1)} = \text{ReLU}\left(W^{(l)} a_v^{(l)}\right)$, where each $W^{(l)} \in R^{k \times k}$ is a learnable weight matrix, and $\mathcal{N}(v)$ represents the set of nodes connected to v by an edge. This is followed by a linear layer, resulting in a set of object embeddings. In contrast to GraphDQN, these object embeddings are *not* a function of state observations. Instead, each object embedding is only a function of the learned node embedding and those of its neighbors. Then, we replace the object symbol in the symbolic state with its object embedding vector. The resulting 3D embedded state is passed into a set of convolutional and linear layers, which output $Q(s, a)$ for $a \in \mathcal{A}$.[2]

[1]Code available: `https://github.com/zacharyhorvitz/Hierarchical-Graph-Priors`

[2]We separate state and inventory embeddings, passing them directly to the DQN head.

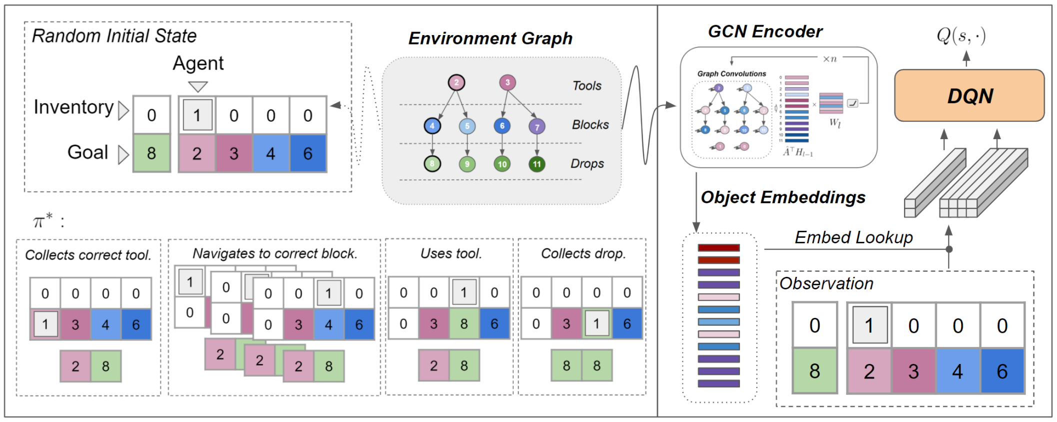


Figure 1: Left: An example Symbolic Procgen environment. Following an optimal policy π^*, the agent collects the right tool, navigate to the right block, use the tool, and collect the drop. Right: Model architecture.

This architecture is flexible and can, in principle, sit atop any deep RL method, including actor-critic or distributed methods [10, 18, 26]. In addition to modularity, our method of updating embeddings encourages the model to learn general representations. Further, we strive to be conscious of parameter count; our encoder is comparatively efficient, and it grows linearly with the number of distinct objects.

4 Experiments on Symbolic Procgen

Inspired by Malmo [14] and the Warehouse environments [25], we design a symbolic environment, *Symbolic Procgen*, that allows us to adjust the number of objects and relations, and by extension, the number of tasks. A Symbolic Procgen environment is initialized with **tools** and **blocks**. During initialization, each block is assigned a single tool and one unique **drop**—a refined resource resulting from 'mining' a block. The relationships between objects are stored in an environment graph, with directed edges between related objects (tool $\rightarrow$ block, block $\rightarrow$ drop). Each episode, the agent is randomly assigned a goal object that uniquely identifies a task. To solve each task, the agent must navigate to the correct tool, use it on the correct block, and collect the resulting goal object.

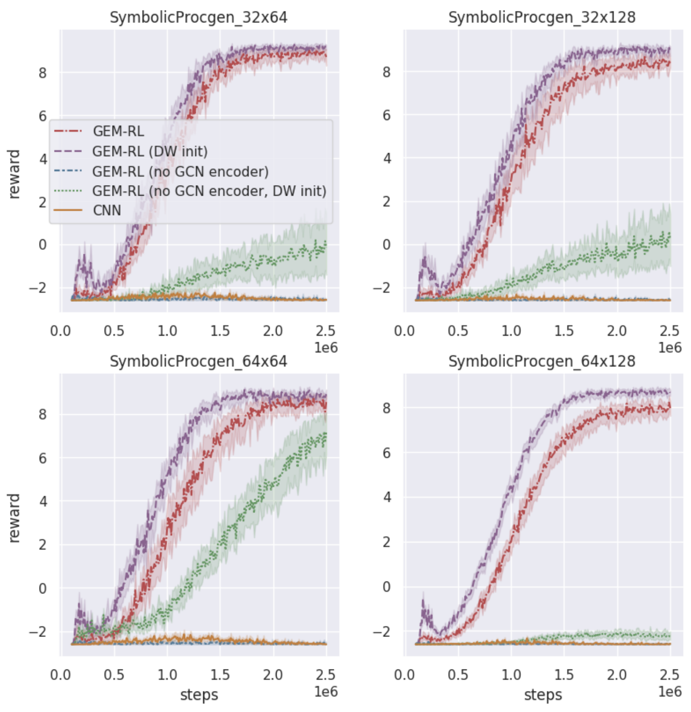
We run experiments on a number of *SymbolicProcgen* $t \times b$ configurations, varying the number of tools t and blocks b. We compare our model against GraphDQN and several baseline models: a CNN pixel-based architecture and a symbolic baseline with an embedding lookup (i.e. GEM-RL without the GCN encoder). The graph-based approaches utilize the environment graph.[3] GEM-RL outperforms GraphDQN and CNN baseline on all domain sizes, and the encoder is indispensable for larger domains. Additional experiments with noisy graphs demonstrate that our method's performance gracefully degrades as the prior becomes less reliable (See Appendix).

5 Related Work

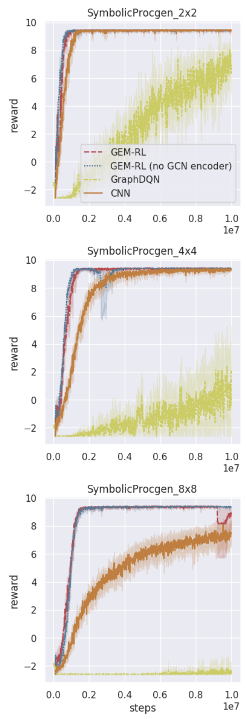
Recent work on relational knowledge in RL has emphasized relational inductive biases coupled with attention-based mechanisms [3]. With impressive results on complex environments like Starcraft [21], this line of work shows that relational information can be successfully exploited in RL. These methods focus on discovering relational structure from interactions with the environment, whereas we aim to include such information as a prior, inspired by how humans approach unseen tasks. However, these approaches could be combined, and we leave investigating the benefit of graph priors in learning relational structure for future work.

Prior work has explored encoding information via graphical priors in specific settings. Yang et al. [31] investigate using linguistic relations encoded in a graph—Scene Priors—for a visual navigation task. The agent uses co-occurrence relations with a pretrained object detection model [12] to improve

[3]Additionally, we add the identity matrix to our environment graph to include self-loops for each node.



(a)



(b)

Figure 2: (a) Results on scaling up domain sizes. Note that GEM-RL outperforms our two baselines: CNN and GEM-RL without the GCN encoder. The relative boost provided by the GCN-encoder dramatically increases in more complex domains. Additionally, DeepWalk based initializations consistently improve performance. (b) Results comparing GraphDQN to GEM-RL on small domains. Note that GraphDQN and CNN baselines require more samples to converge as complexity increases.

object search. However, their agent constructs its object representations from image states, and their work remains to be extended to more general RL domains. We compare to another approach, used by Vijay et al. [30], that uses graph convolution operations to structure the latent representation of a DQN, but only examines single-task, symbolic domains.

Canonical MTRL agents do not incorporate graph information and do not operate on OO-MDPs. IMPALA and PopArt [10, 13] are A3C-based agents designed to scale training across many machines and optimize resource utilization. PopArt and DISTRAL [28] attempt to address the 'distraction dilemma'—a challenge in MTRL where certain tasks are prioritized by the agent over others—and they present an excellent base for extensions to GEM-RL.

6 Discussion

Graph-based embedding priors provide a flexible, intuitive mechanism for encoding relational knowledge. We introduce a novel and modular method, GEM-RL, designed to exploit relational information for improved performance over many tasks with large numbers of objects, as well as a new multi-task domain of variable complexity, *Symbolic Procgen*. Our approach demonstrably outperforms competing methods, such as GraphDQN and CNN baselines.

We plan to use GEM-RL to investigate utilizing graphs from structured knowledge bases, like ConceptNet or WordNet [11, 24], and from unstructured sources, like natural language. Another promising direction is using actions in object relations, expanding the realm of possible common-sense information to use as a prior. Additionally, exploring sample-efficiency gains in multi-task domains with complex dynamics and rich observations is of great interest and practical relevance. We hope to combine our work with existing object-detection/semantic segmentation methods to ground objects in pixel-based observations.

References

[1] Prithviraj Ammanabrolu and Mark O. Riedl. Playing text-adventure games with graph-based deep reinforcement learning. *ArXiv*, abs/1812.01628, 2019.

[2] Adrià Puigdomènech Badia, B. Piot, Steven Kapturowski, P. Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *ArXiv*, abs/2003.13350, 2020.

[3] P. Battaglia, Jessica B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, Mateusz Malinowski, Andrea Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Çaglar Gülçehre, H. Song, A. Ballard, J. Gilmer, G. Dahl, Ashish Vaswani, Kelsey R. Allen, C. Nash, V. Langston, Chris Dyer, N. Heess, Daan Wierstra, Pushmeet Kohli, M. Botvinick, Oriol Vinyals, Y. Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *ArXiv*, abs/1806.01261, 2018.

[4] H. Beeck and C. Baker. The neural basis of visual object learning. *Trends in Cognitive Sciences*, 14:22–30, 2010.

[5] M. Brants, J. Bulthé, N. Daniels, J. Wagemans, and H. Beeck. How learning might strengthen existing visual object representations in human object-selective cortex. *NeuroImage*, 127:74–85, 2016.

[6] R. Caruana. Multitask learning. In *Encyclopedia of Machine Learning and Data Mining*, 1998.

[7] Carlo D'Eramo, Davide Tateo, A. Bonarini, Marcello Restelli, and J. Peters. Sharing knowledge in multi-task deep reinforcement learning. In *ICLR*, 2020.

[8] Carlos Diuk, A. Cohen, and M. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML '08*, 2008.

[9] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, T. Griffiths, and Alexei A. Efros. Investigating human priors for playing video games. In *ICML*, 2018.

[10] Lasse Espeholt, Hubert Soyer, R. Munos, K. Simonyan, V. Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, T. Harley, Iain Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *ArXiv*, abs/1802.01561, 2018.

[11] C. Fellbaum. Wordnet : an electronic lexical database. *Language*, 76:706, 2000.

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[13] Matteo Hessel, Hubert Soyer, Lasse Espeholt, W. Czarnecki, Simon Schmitt, and H. V. Hasselt. Multi-task deep reinforcement learning with popart. In *AAAI*, 2019.

[14] Matthew Johnson, Katja Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, 2016.

[15] Thomas Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

[16] B. Lake, Tomer D. Ullman, J. Tenenbaum, and Samuel Gershman. Building machines that learn and think like people. *The Behavioral and brain sciences*, 40:e253, 2018.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[18] V. Mnih, Adrià Puigdomènech Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[19] Bryan Perozzi, Rami Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. In *KDD '14*, 2014.

[20] M. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

[21] Mikayel Samvelyan, Tabish Rashid, C. S. Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, P. Torr, Jakob N. Foerster, and S. Whiteson. The starcraft multi-agent challenge. In *AAMAS*, 2019.

[22] D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[23] D. Silver, T. Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, A. Guez, Marc Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv*, abs/1712.01815, 2017.

[24] Robyn Speer, J. Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *ArXiv*, abs/1612.03975, 2017.

[25] Tom Stepleton, 2019. URL `https://github.com/deepmind/pycolab`.

[26] R. Sutton and A. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.

[27] Matthew E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, 2009.

[28] Y. Teh, V. Bapst, W. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, N. Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *NIPS*, 2017.

[29] Nelson Vithayathil Varghese and Q. Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9:1363, 2020.

[30] V. K. Vijay, A. Ganesh, Hanlin Tang, and Arjun K. Bansal. Generalization to novel objects using prior relational knowledge. *ArXiv*, abs/1906.11315, 2019.

[31] Wei Yang, X. Wang, Ali Farhadi, A. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. *ArXiv*, abs/1810.06543, 2019.

A Additional Experiments

A.1 Noise Ablation Results

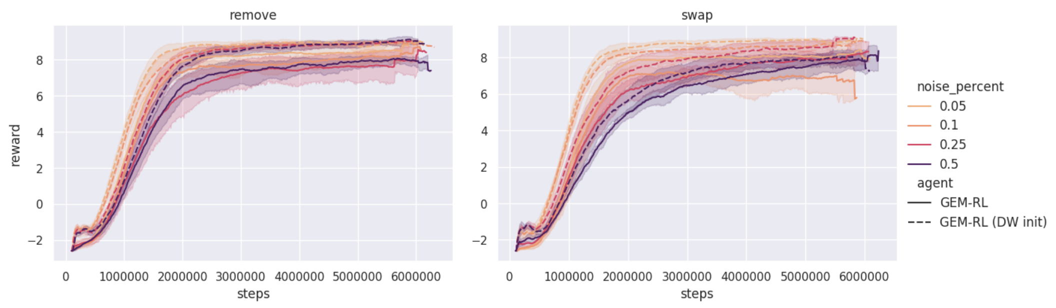


Figure 3: Noise ablation results on an instance of *Symbolic Procgen* with 32 tools and 128 blocks. Note the graceful degradation in performance as up to 50% of the graph has noise applied.

To investigate the robustness of the graphical prior, we performed two distinct ablations using noisy graphs. First, we conducted a series of "remove" noise experiments, where we randomly selected $k\%$ of the edges in our knowledge graph and removed these edges. This ablation tested how our method performs when provided with an incomplete graphical prior.

Second, we conducted a series of "swap" noise experiments, where we randomly shuffled $k\%$ of the edges connecting tools and blocks and blocks and drops. This ablation tested how our method performs when provided with a partially incorrect graphical prior.

Our method performance degrades gracefully as we increase the ablation degree for both remove and swap noise variants. Furthermore, for every ablation tier $t \in \{0.05, 0.1, 0.25, 0.5\}$, we observe that remove noise performance dominates swap noise performance. We hypothesize that swap noise, unlike remove noise, introduces an incorrect prior which leads to elements of negative transfer. This negative transfer takes additional experience to overcome. In contrast, increasing the degree of remove noise decreases the degree of positive transfer induced by the graphical prior and does not directly lead to negative transfer.

The knowledge graph can be utilized in myriad ways to encourage learning task-invariant object representations. One of these mechanisms is using the graphical prior to construct informed object embedding initializations. In this ablation, we focus only on DeepWalk node embeddings [19]. We observe that DeepWalk embedding initializations decrease the sensitivity of all methods to noisy graphs (both remove and swap noise), while also yielding benefits in sample efficiency. Further ablations are required to investigate other initialization schemes.

A.2 Full Small Experiments

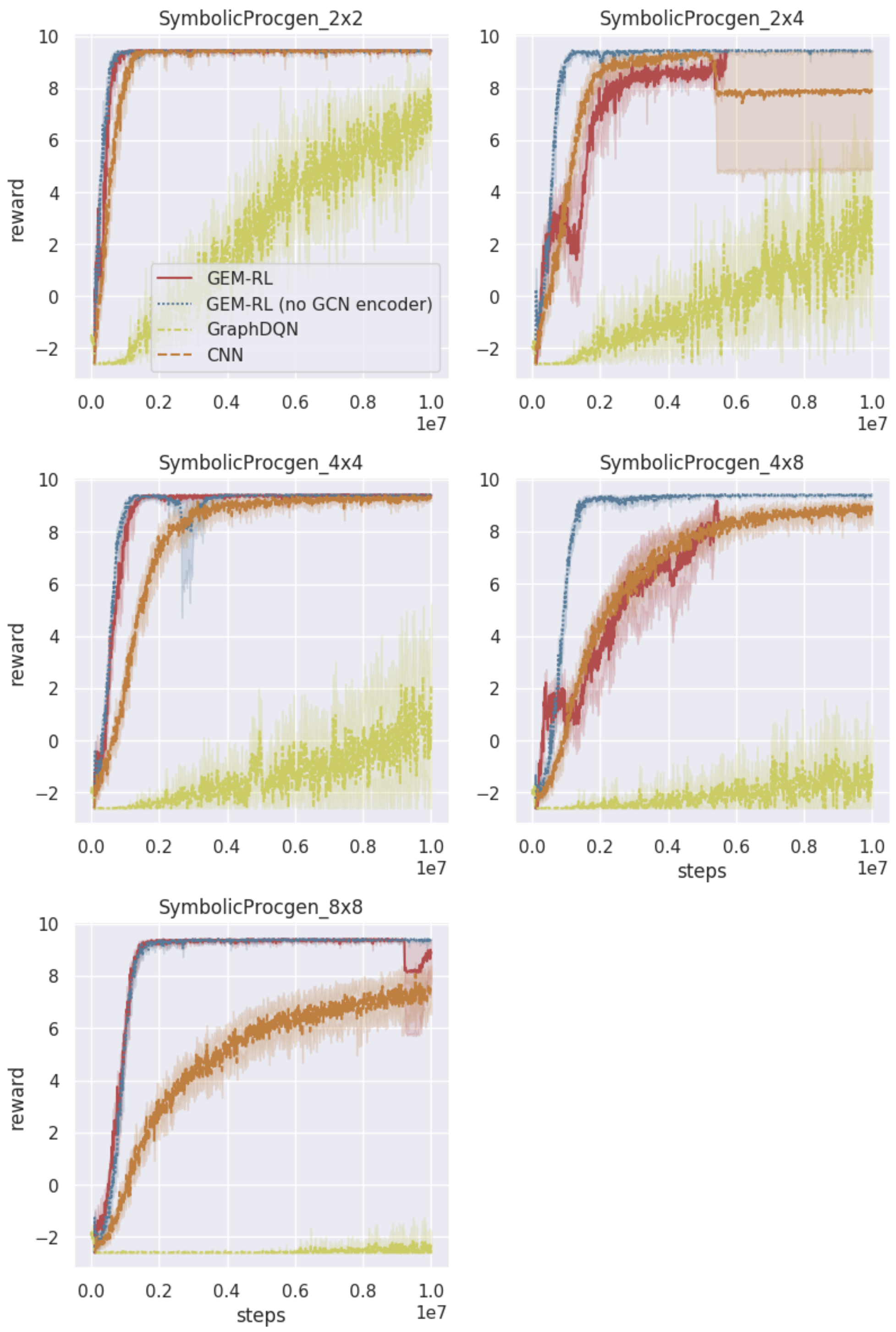


Figure 4: Results on more small domains in *Symbolic Procgen*.

A.3 Full GraphDQN Experiments

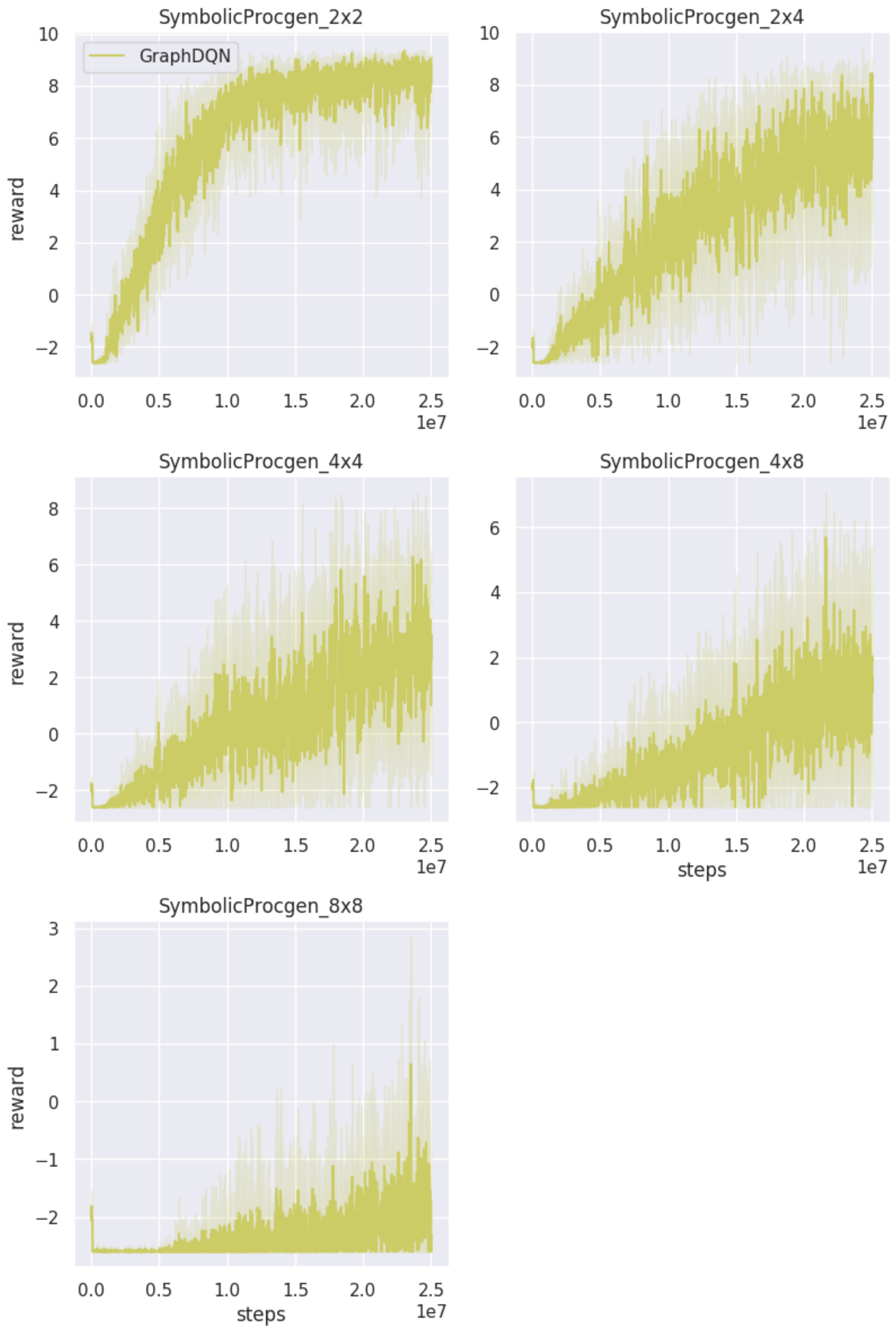


Figure 5: Results of GraphDQN for 25M steps on *Symbolic Procgen*.

B Experimental Details

B.1 Architecture

```
model(
  (body): Sequential(
    (0): Conv2d(32, 32, kernel_size=(2, 2), stride=(1, 1))
    (1): ReLU()
  )
  (head): Sequential(
    (0): Linear(in_features=160, out_features=300, bias=True)
    (1): ReLU()
    (2): Linear(in_features=300, out_features=300, bias=True)
    (3): ReLU()
    (4): Linear(in_features=300, out_features=5, bias=True)
  )
  (gcn): GCN(
    ((0, 0)): Linear(in_features=32, out_features=32, bias=False)
    ((0, 1)): Linear(in_features=32, out_features=32, bias=False)
    ((0, 2)): Linear(in_features=32, out_features=32, bias=False)
    (final_mapping): Linear(in_features=32, out_features=32, bias=True)
  )
  (embeds): Embedding([num_objects], 32)
)
```

B.2 Hyperparameters

Hyperparameter	Value
batchsize	32
embedding size	32
gradient updates per step	1
target update param	5×10^{-3}
gamma	5×0.99
warmup	100000
epsilon initial	1
epsilon final	0.05
epsilon linear decay length	1×10^{6}
double DQN update	yes

Figure 6: Hyperparameters used for GEM-RL

We tuned the learning rate for GraphDQN and GEM-RL on *Symbolic Procgen* for each experimental setting on our domain. A learning rate of 1.0×10^{-5} was used for the (2 tools, 4 blocks) and (4 tools, 8 blocks) settings, while a learning rate of 8.333×10^{-5} was used for all other experimental settings.

B.3 Symbolic Procgen

A Symbolic Procgen state comprises a 2×4 gridworld and two additional variables: the 'inventory' and 'goal'. The bottom row of the grid is then populated with four objects in random order: the block assigned to that drop, the tool required to break the block, one random tool, and one random block. The agent has five possible actions: *North*, *South*, *East*, *West*, and *Use Tool*. The agent can only collect tools and drop items. In order to convert a block to its drop, the agent must use the correct tool on that block. As the agent moves about the gridworld, it can store a drop or tool in its single inventory slot. The episode ends after a fixed number of steps, or when the the agent adds the goal object to its inventory. Every episode, one drop is randomly chosen as the ‘goal’. That object’s corresponding block and that block’s tool, along with one random block and one random tool, are added in random order to the bottom row of the gridworld.