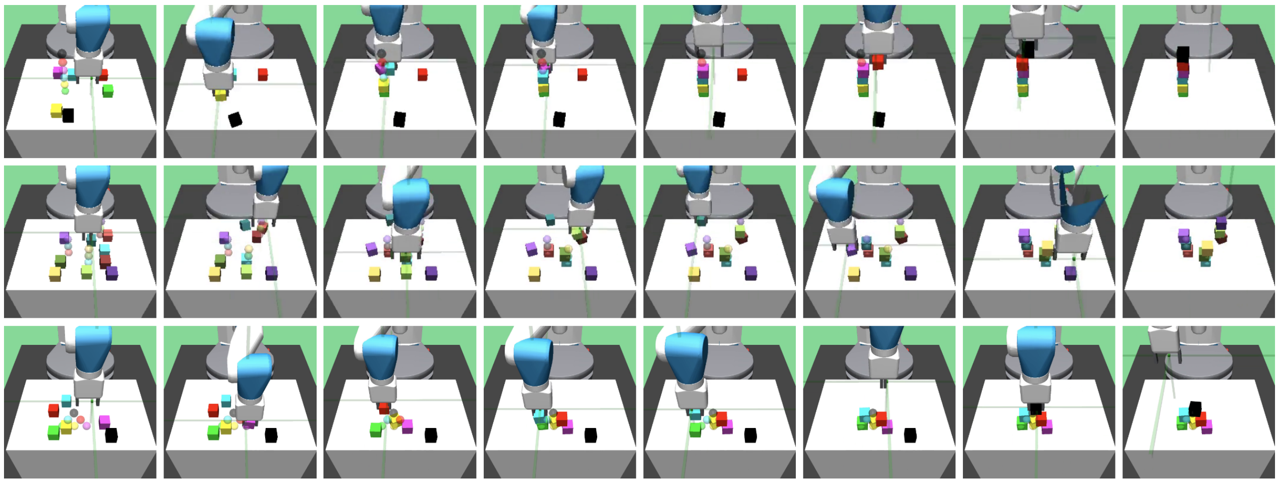
Towards Practical Multi-Object Manipulation using Relational Reinforcement Learning

Anonymous Authors[1]



We present a reinforcement learning system that can stack 6 blocks without requiring any demonstrations or task-specific assumptions. The last two rows show zero-shot generalization results of configuring blocks into unseen configurations of multiple towers and pyramids without additional training. See the videos here: https://richardrl.github.io/relational-rl.

Abstract

Learning robotic manipulation tasks using reinforcement learning with sparse rewards is currently impractical due to the outrageous data requirements. Many practical tasks require manipulation of multiple objects, and the complexity of such tasks increases with the number of objects. Learning from a curriculum of increasing object cardinality appears to be a natural solution, but unfortunately, does not work for many scenarios. We show that a graph-based relational architecture enables learning from this curriculum, and demonstrate our method on a simulated block stacking task. Despite using step-wise sparse rewards, our method is orders of magnitude more data-efficient and achieves much higher success rate than the existing state-of-the-art method that utilizes human demonstrations. Furthermore, the learned policy exhibits zero-shot generalization, successfully stacking blocks into taller towers and previously unseen configurations such as pyramids, without any further training.

1. Introduction

Sparse reward, which may be provided either after the agent completes the overall task (terminal reward) or intermittently when the agent completes critical steps (step-wise rewards), significantly simplifies reward design in reinforcement learning. However, because many tasks require execution of a long sequence of actions, sparse rewards drastically complicate the challenges of exploration and credit-assignment. Training with sparse rewards, therefore, either completely fails or requires massive amounts of data.

In this work, we utilize a curriculum over object cardinality to overcome the sparse reward problem. We show that training a policy represented by an attention-based graph neural network (GNN) enables transfer between tasks in this curriculum. Our agent learns to stack six or more blocks from scratch (see Figure). To the best of our knowledge, ours is the first work to solve the problem of stacking six

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

or more blocks using RL and without requiring any expert demonstrations. Our method is orders of magnitude more efficient than the previous state-of-the-art method relying on human-provided demonstrations (Nair et al., 2017).

Our system can build towers that are taller than that experienced at training time. It also succeeds at placing blocks in different configurations such as pyramids without any additional training (i.e., *zero-shot generalization*). While we present results on the task of stacking blocks in various arrangements, the approach developed in this work does not make any task-specific assumption and is therefore applicable to a wide range of tasks involving manipulation of multiple objects.

2. Related Work

Our work is broadly related to techniques for scaling reinforcement learning algorithms to more complex robotic manipulation settings, as well as the use of relational and curricular inductive biases in machine learning.

Relational Inductive Bias: Related to our work is past research combining GNNs with policy learning for manipulation tasks. However, these works either rely on tens or hundreds of thousands of expert demonstrations (Duan et al., 2017b; Janner et al., 2019) or simplify the task by teleporting objects around in the simulator instead of actuating a manipulator (Janner et al., 2019).

Block Stacking: Prior work on block stacking either heavily relied on human demonstrations (Nair et al., 2017; Duan et al., 2017a), or required significant reward engineering (Popov et al., 2018). The work of (Deisenroth et al., 2011) stacked blocks using a low-cost robot. However, they assumed the blocks were already picked and used a dense reward function. Other lines of work (Kroemer et al., 2018; Toussaint, 2015) achieved impressive results on stacking objects, but relied on human-designed representations. In contrast, we present a simple but effective method for stacking blocks using RL that makes minimal assumptions about task structure or the environment.

3. Experimental Setup

Our environment consists of a robot arm manipulating a variable number of cubic blocks on a table. The agent observes gripper state features (including gripper velocity and position) and block state features (including block position and velocity) for each of N blocks. The block features are denoted by $X^f : x^f_1, x^f_2, ..x^f_N$, where $N \in [1, 9]$ and x^f_i is the feature representation of the i^{th} block. The goal is expressed as set of 3D block positions, $X^g : x^g_1, x^g_2, ..x^g_N$. The overall input to the agent is therefore $\{X^{ee}, X^f, X^g\}$. At the start of every episode, the initial block positions are randomly initialized on the table and the goal positions are sampled using a hand-designed goal distribution. The maximum length of every episode is $50 * N$ steps, where N is the number of blocks.

4. Preliminaries

4.1. Goal-Conditioned RL

While the above formulation is appropriate for a single goal, for solving multiple tasks, it is necessary to provide a task description as input (Schaul et al., 2015; Agrawal et al., 2016; Andrychowicz et al., 2017). Goal conditioned policies are expressed as $a_t = \pi(s_t, s_g)$, where s_g represents the goal state. The learning problem is expressed as:

$$\max_{\pi} \mathbb{E}_{s_g \sim \rho(s_g), a \sim \pi, s \sim \mathcal{T}}[\sum_{i=t}^{T} \gamma^{(t-i)} r(s_t, a_t, s_g)] \quad (1)$$

where goal s_g is sampled from a goal distribution $\rho(s_g)$.

4.2. Graph Neural Networks (GNN)

The central computation in a GNN is message passing between 1-hop vertices of a graph, performed by a *graph-to-graph* module. This module takes as input a variable-size vertex set $\mathbf{v} = \{\vec{v}_i\}_{i=1}^{N_v}$ and outputs an updated set $\mathbf{v}' = \{\vec{v}'_i\}_{i=1}^{N_v}$, where N^v is the number of vertices in the input graph. $\vec{v}_i, \vec{v}'_i$ denote feature vectors of the i^{th} node before and after a round of message passing. In each message passing round, each vertex sends a message to every other vertex. In attention-based GNNs, the incoming messages are weighted by a scalar coefficient (computed by attention) according to their relevance to the receiving vertex. The new feature representation of the vertex is the weighted sum of incoming messages. Message passing is typically performed multiple times.

5. Method

We present a simple, but effective method combining curriculum with a grpah neural network architecture for solving long-horizon, sparse reward tasks using reinforcement learning. We use Soft-Actor Critic (SAC; (Haarnoja et al., 2018)) and Hindsight Experience Replay (Andrychowicz et al., 2017) as our base learning algorithm.

We construct an attention-based GNN architecture for our actor and critic called *ReNN*[1]. We compare the performance of *ReNN* against the baseline system that uses a multilayer perceptron (MLP) architecture.

Training Curriculum: We trained the robot to stack multiple blocks using three different curricula of tasks:

[1]See Appendix A.1.

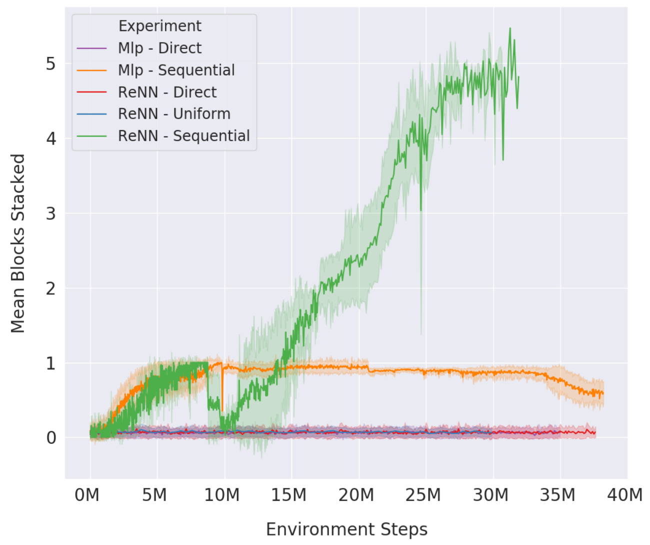


Figure 1. Comparing task performance measured as the mean number of blocks stacked per timestep during training. We report the mean and standard deviation across multiple workers and/or seeds. The performance of relational (*ReNN*) and the usual multi-layer (*MLP*) architectures are reported when they are subjected to different training curricula described in Section 5. Both *ReNN* and *MLP* fail to stack blocks, when the robot was directly trained for stacking 6 blocks ($MLP - Direct$, $ReNN - Direct$). Only *ReNN* trained with sequential curriculum ($ReNN - Sequential$) succeeds at stacking six blocks.

- **Direct**: The robot was directly tasked to learn a policy to stack six blocks starting from scratch.
- **Uniform:** At every episode, the number of blocks was uniformly sampled between 1 and 6.
- **Sequential**: The robot was tasked to pick and place one block, then to pick and place two blocks. Thereafter, the robot was tasked with stacking blocks in a single tower configuration from 2 to 6 blocks. The transition points in this curriculum were manually chosen based on the success rates on stacking.

5.1. Testing Details

We evaluated the generalization of the policy trained for stacking a single tower by evaluating its performance on the following tests[2]:

- **Single Tower:** A single point was uniformly sampled on the table to serve as the base of a block tower. The goal positions of the blocks corresponded to translation along the z-axis from the base.
- **Multiple Towers:** A few points ($k \in \{2, 3\}$) were sampled on the table to serve as the base location of multiple towers. Each block was randomly assigned to a tower to produce towers of approximately equal height.
- **Pyramid:** A uniformly sampled point on the table served as a corner point for pyramid configuration.

We report performance of *ReNN- Sequential* (referred to as *ReNN* in later text) across three seeds. For other methods we report performance on a single seed. Success rate is reported as accuracy of completing a task averaged over 100 episodes. An episode is counted as successful when each block is within its goal position at the final time step.

6. Results

Table 1. Comparing the performance of our method against the previous state-of-the art (Nair et al., 2017) that makes use of human demonstrations on the block stacking task. Each entry, $p\%$ (s), denotes accuracy of $p\%$ after s number of environment steps. Our method is both more sample efficient and outperforms prior work.

Task	Single Tower 4	Single Tower 5	Single Tower 6
Nair '17	91% (850M)	50% (1000M)	32% (2300M)
Ours	**93%(23M)**	**84% (27M)**	**75% (30M)**

Figure 1 shows that *ReNN* trained with the sequential curriculum (green line; section 5) succeeds at stacking six blocks into a tower. All other combinations of task distribution and architecture fail.

We report quantitative performance of our method and baselines in Table 1. Our method achieves a success rate of 75% at stacking 6 blocks in 30 million timesteps. In comparison, the existing state-of-art method (Nair et al., 2017), that makes use of human demonstrations and resets, achieves only a success rate of 32% after over 2.3 billion timesteps. While the base learning algorithm used by (Nair et al., 2017) is DDPG + HER, in comparison to SAC + HER used by us, the orders of magnitude difference in performance cannot be attributed to the choice of using SAC instead of DDPG, as shown by our ablation illustrating the importance of sequential curriculum.

Careful analysis of Figure 1 reveals that there are several dips in performance as the training progresses. Many of the significant dips correspond to a task change where an additional block is added to the environment. In most cases, the dip in performance is overcome after relatively little additional experience. The only notable exception is the performance dip at 9M steps that corresponds to transitioning from 1 to 2 blocks. This was the first time the agent observed multiple objects.

[2] See Appendix for visualizations of the different goal configurations.

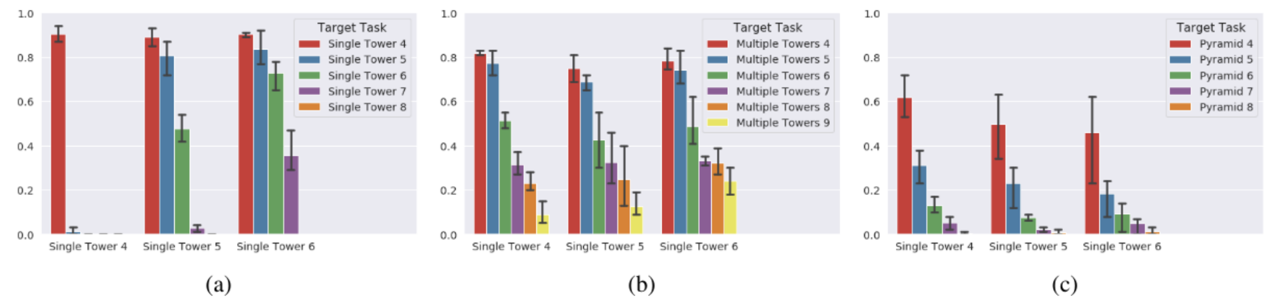


Figure 2. Quantitative evaluation of zero-shot generalization results of policies trained to stack i blocks in a single tower. These policies were evaluated, without any further training, on (a) single (but taller) tower (shown in shades of red); (b) multiple towers; (c) pyramid configurations. Details of these testing setups can be found in Section 5.1. The results show that robot is capable of zero-shot generalization to many of these tasks.

6.1. Zero-shot Generalization

If our *ReNN* architecture indeed provides a good inductive bias, then it should be possible solve different block configuration tasks with high-accuracy. To test this, we evaluated the performance of the learned policy, without any fine-tuning, on previously unseen block configurations (i.e. *zero-shot generalization*) described in Section 5.1. The results of this analysis are summarized in Figure 2.

Single Tower Evaluation: Figure 2 shows that a policy learned to stack N blocks generalizes to stacking $N+1$ blocks without any training. The performance on stacking $N+k$ blocks, where $k>1$ drops significantly. One possible explanation is that it becomes progressively harder to stabilize larger number of blocks in a tower and the robot needs to substantially refine its strategy to stack more blocks. An analysis of failure modes is presented in Appendix B.

Multiple Towers Evaluation: The previous experiments tested generalization to a larger number of blocks, but on the same task. To test if the learned policy generalizes to new tasks, we evaluated the performance on stacking multiple towers instead of a single tower. Results in Figure 2(b) show that the agent trained for stacking a single tower of N blocks can successfully stack multiple towers $N+k$ blocks. Generalization to $k \geq 2$ is better on the multiple towers task as compared to the single tower task. This suggest that *ReNN* generalizing over the total number of objects is easier than generalizing over the height of the tallest tower.

Pyramid Evaluation: To stress test our system further, we evaluated its performance on placing blocks in a pyramid configuration. Stacking blocks in pyramid is different than a tower, because now blocks may need to be balanced on two supporting blocks instead of only being stacked vertically. Figure 2(c) shows that our system is able to generalize and manipulate larger number of blocks than seen in training into pyramid configurations.

Emergent Strategies: The accompanying videos[3] show that our agent automatically learns to singulate individual blocks, roll blocks, grasps two blocks at a time, and other complex behaviors. These strategies emerge automatically as a consequence of optimizing a sparse reward function.

To the best of our knowledge, ours is the first work that reports such zero-shot generalization on the block stacking task using RL. At the same time, we acknowledge, there is substantial room for improving the zero-shot results and the stacking performance. Some future directions are described in Section 7.

7. Discussion

We have presented a framework for learning long-horizon, sparse reward tasks using deep reinforcement learning, relational graph architecture and curriculum learning. In the current work, the curriculum is manually-designed based on the principle that smaller sets of objects are easier to learn to manipulate than larger sets of objects. However, more complicated and effective curricula could exist beyond just the object cardinality, and discovering these curricula automatically is an interesting direction for future research. Developing computationally efficient methods is important to scale relational graph architectures (which have quadratic runtime in object cardinality) to much larger numbers of objects. Finally, while we have presented results from state observation, in the future we would like to scale our system to work from visual and other sensory observations.

References

Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. Learning to poke by poking: Experiential learning of intuitive physics. *NIPS*, 2016.

[3]https://richardrl.github.io/relational-rl

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5048–5058. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7090-hindsight-experience-replay.pdf.

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics Science and Systems*, pp. 57–64, 2011.

Duan, Y., Andrychowicz, M., Stadie, B., Ho, O. J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Advances in neural information processing systems*, pp. 1087–1098, 2017a.

Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1087–1098. Curran Associates, Inc., 2017b. URL http://papers.nips.cc/paper/6709-one-shot-imitation-learning.pdf.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Janner, M., Levine, S., Freeman, W. T., Tenenbaum, J. B., Finn, C., and Wu, J. Reasoning about physical interactions with object-oriented prediction and planning. In *International Conference on Learning Representations*, 2019.

Kroemer, O., Leischnig, S., Luettgen, S., and Peters, J. A kernel-based approach to learning contact distributions for robot manipulation tasks. *Autonomous Robots*, 42(3): 581–600, 2018.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089, 2017. URL http://arxiv.org/abs/1709.10089.

Popov, I., Heess, N., Lillicrap, T. P., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Erez, T., Tassa, Y., and Riedmiller, M. Data-efficient deep reinforcement learning for dexterous manipulation, 2018. URL https://openreview.net/forum?id=SJdCUMZAW.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Toussaint, M. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.