Structured agents for physical construction

Victor Bapst [*1] **Alvaro Sanchez-Gonzalez** [*1] **Carl Doersch** [1] **Kimberly L. Stachenfeld** [1] **Pushmeet Kohli** [1]
Peter W. Battaglia [1] **Jessica B. Hamrick** [1]

Abstract

Physical construction—the ability to compose objects, subject to physical dynamics, to serve some function—is fundamental to human intelligence. We introduce a suite of challenging physical construction tasks inspired by how children play with blocks, such as matching a target configuration, stacking blocks to connect objects together, and creating shelter-like structures over target objects. We examine how a range of deep reinforcement learning agents fare on these challenges, and introduce several new approaches which provide superior performance. Our results show that agents which use structured representations (e.g., objects and scene graphs) and structured policies (e.g., object-centric actions) outperform those which use less structured representations, and generalize better beyond their training when asked to reason about larger scenes. Model-based agents which use Monte-Carlo Tree Search also outperform strictly model-free agents in our most challenging construction problems. We conclude that approaches which combine structured representations and reasoning with powerful learning are a key path toward agents that possess rich intuitive physics, scene understanding, and planning.

1. Introduction

Humans are a "construction species"—we build forts out of couch cushions as children, pyramids in our deserts, and space stations that orbit hundreds of kilometers above our heads. What abilities do artificial intelligence (AI) agents need to possess to perform such achievements? This question frames the high-level purpose of this paper: to explore a range of tasks more complex than those typically studied in AI, and to develop approaches for learning to solve them.

[*]Equal contribution [1]DeepMind, London, UK. Correspondence to: Jessica Hamrick <jhamrick@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Physical construction involves composing multiple elements under physical dynamics and constraints to achieve rich functional objectives. We introduce a suite of simulated physical construction tasks (Fig. 1), similar in spirit to how children play with toy blocks, which involve stacking and attaching together multiple blocks in configurations that satisfy functional objectives. For example, one task requires stacking blocks around obstacles to connect target locations to the ground. Another task requires building shelters to cover up target blocks and keep them dry in the rain. These tasks are representative of real-world construction challenges: they emphasize problem-solving and functionality rather than simply replicating a given target configuration, reflecting the way human construction involves forethought and purpose.

Real-world physical construction assumes many forms and degrees of complexity, but a few basic skills are typically involved: spatial reasoning (e.g. concepts like "empty" vs "occupied"), relational reasoning (e.g. concepts like "next to" or "on top of"), knowledge of physics (e.g., predicting physical interactions among objects), and planning the allocation of resources to different parts of the structure. Our simulated task environment (Fig. 1) is designed to exercise these skills, while still being simple enough to allow careful experimental control and tractable agent training.

While classic AI studied physical reasoning extensively (Chen, 1990; Pfalzgraf, 1997), construction has not been well-explored using modern learning-based approaches. We draw on a number of techniques from modern AI, combining and extending them in novel ways to make them more applicable and effective for construction. Our family of deep reinforcement learning (RL) agents can support: (1) vector, sequence, image, and graph-structured representations of scenes; (2) continuous and discrete actions, in absolute or object-centric coordinates; (3) model-free learning via deep Q-learning (Mnih et al., 2015), or actor-critic methods (Heess et al., 2015; Munos et al., 2016); and (4) planning via Monte-Carlo Tree Search (MCTS) (Coulom, 2006).

We find that graph-structured representations and reasoning, object-centric policies, and model-based planning are crucial for solving our most difficult tasks, outperforming standard approaches which combine unstructured represen-

tations with policies that take absolute actions. Our results demonstrate the value of integrating rich structure and powerful learning approaches as a key path toward complex construction behavior.

2. Related Work

Physical reasoning has been of longstanding interest in AI. Early work explored physical concepts with an emphasis on descriptions that generalize across diverse settings (Winston, 1970). Geometric logical reasoning was a major topic in symbolic logic research (Chou, 1987; Arnon, 1988), leading to geometric theorem-provers (Bouma et al., 1995), rule-based geometric constraint solvers for computer-aided design (Aldefeld, 1988; Schreck et al., 2012), and logic-based optimization for open-ended objectives in robotics (Toussaint, 2015). Classic work often focused on rules and structured representations rather than learning because the sample complexity of learning was often prohibitive for contemporary computers.

Modern advances in learning-based approaches have opened new avenues for using vector and convolutional representations for physical reasoning (Wu et al., 2015; 2016; 2017; Mottaghi et al., 2016; Fragkiadaki et al., 2016; Finn et al., 2016; Agrawal et al., 2016; Lerer et al., 2016; Li et al., 2016; Groth et al., 2018; Bhattacharyya et al., 2018; Ebert et al., 2018). A common limitation, however, is that due to their relatively unstructured representations of space and objects, these approaches tend not to scale up to complex scenes, or generalize to scenes with different numbers of objects, etc.

Several recent studies have explored learning construction, including learning to stack blocks by placing them at predicted stable points (Li et al., 2017), learning to attach blocks together to stabilize an unstable stack (Hamrick et al., 2018), learning basic block-stacking by predicting shortest paths between current and goal states via a transition model (Zhang et al., 2018), and learning object representations and coarse-grained physics models for stacking blocks (Janner et al., 2019). Though promising, in these works the physical structures the agents construct are either very simple, or provided explicitly as an input rather than being *designed* by the agent itself. A key open challenge, which this paper begins to address, is how to learn to design and build complex structures to satisfy rich functional objectives.

A main direction we explore is object-centric representations of the scene and agent's actions (Diuk et al., 2008; Scholz et al., 2014), implemented with graph neural networks (Scarselli et al., 2009; Bronstein et al., 2017; Gilmer et al., 2017; Battaglia et al., 2018). Within the domain of physical reasoning, graph neural networks have been used as forward models for predicting future states and images (Battaglia et al., 2016; Chang et al., 2017; Watters et al., 2017; van Steenkiste et al., 2018), and can allow efficient learning and rich generalization. These models have also begun to be incorporated into model-free and model-based RL, in domains such as combinatorial optimization, motor control, and game playing (Dai et al., 2017; Kool & Welling, 2018; Hamrick et al., 2018; Wang et al., 2018; Sanchez-Gonzalez et al., 2018; Zambaldi et al., 2019). There are several novel aspects to our graph network policies beyond these existing works, including the use of multiple actions per edge and graphs that change size during an episode.

3. Physical Construction Tasks

Our simulated task environment is a continuous, procedurally-generated 2D world implemented in Unity (Juliani et al., 2018) with the Box2D physics engine (Catto, 2013). Each episode contains unmoveable obstacles, target objects, and floor, plus movable rectangular blocks which can be picked up and placed.

On each step of an episode, the agent chooses an available block (from below the floor), and places it in the scene (above the floor) by specifying its position. In all but one task (*Covering Hard*—see below), there is an unlimited supply of blocks of each size, so the same block can be picked up and placed multiple times. The agent may also attach objects together by assigning the property of "stickiness" to the block it is placing. Sticky objects form unbreakable, nearly rigid bonds with objects they contact. In all but one task (*Connecting*) the agent pays a cost to make a block sticky. After the agent places a block, the environment runs physics forward until all blocks come to rest.

An episode terminates when: (1) a movable block makes contact with an obstacle, either because it is placed in an overlapping location, or because they collide under physical dynamics; (2) a maximum number of actions is exceeded; or (3) the task-specific termination criterion is achieved (described below). The episode always yields zero reward when a movable block makes contact with an obstacle.

***Silhouette* task** (Fig. 1a). The agent must place blocks to overlap with target blocks in the scene, while avoiding randomly positioned obstacles. The reward function is: $+1$ for each placed block which overlaps at least 90% with a target block of the same size; and -0.5 for each block set as sticky. The task-specific termination criterion is achieved when there is at least 90% overlap with all targets.

This is similar to the task in Janner et al. (2019), and challenges agents to reason about physical support of complex arrangements of objects and to select, position, and attach sequences of objects accordingly. However, by fully specifying the target configuration, *Silhouette* does not require the agent to design a structure to satisfy a functional objective, which is an important component of our other tasks.

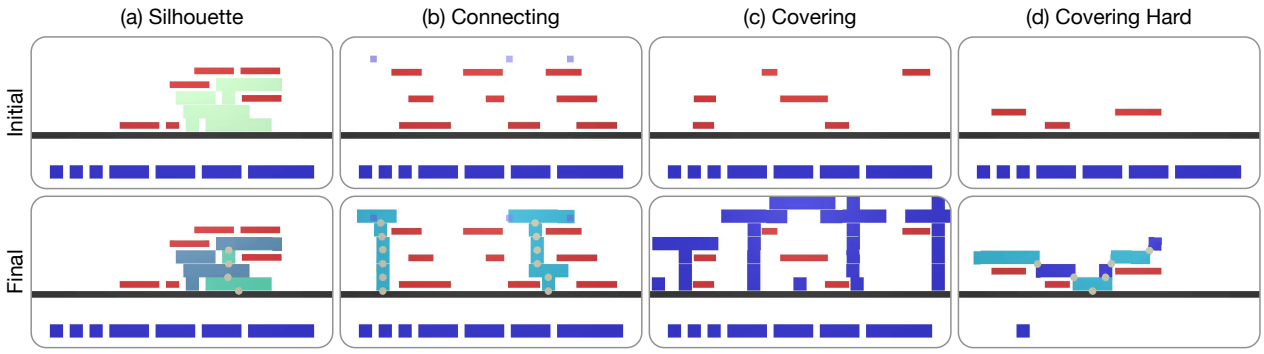

Figure 1. Construction task suite. In all tasks, dark blue objects are regular blocks, light blue blocks are "sticky", red objects are obstacles which cannot be touched, and grey circles indicate points where blocks have stuck together. The black line indicates the floor separating the scene above, which is subject to physics, from the blocks below, which can be picked up and placed. (a) *Silhouette* task. The agent stacks blocks to match the target blocks (depicted as light green blocks). (b) *Connecting* task. The agent stacks blocks to connect the small blue target objects to the floor. (c) *Covering* task. The agent stacks blocks to shelter the obstacles from above. (d) *Covering Hard* task. Similar to *Covering*, but crucially the agent has a limited supply of movable blocks. Videos of agent behaviors in these tasks are available at https://tinyurl.com/y7wtfen9 and in Supplemental Table G.1.

***Connecting* task** (Fig. 1b). The agent must stack blocks to connect the floor to three different target locations, avoiding randomly positioned obstacles arranged in layers. The reward function is: $+1$ for each target whose center is touched by at least one block, and 0 (no penalty) for each block set to sticky. The task-specific termination criterion is achieved when all targets are connected to the floor.

By not fully specifying the target configuration, the *Connecting* task requires the agent to *design* a structure with a basic function—connecting targets to the floor—rather than simply implementing it as in the *Silhouette* task. A wider variety of structures could achieve success in *Connecting* than *Silhouette*, and the solution space is much larger because the task is tailored so that solutions usually require many more blocks.

***Covering* task** (Fig. 1c). The agent must build a shelter that covers all obstacles from above, without touching them. The reward function is: $+L$, where L is the sum of the lengths of the top surfaces of the obstacles which are sheltered by blocks placed by the agent; and -2 for each block set as sticky. The task-specific termination criterion is achieved when at least 99% of the summed obstacle surfaces are covered. The layers of obstacles are well-separated vertically so that the agent can build structures between them.

The *Covering* task requires richer reasoning about function than the previous tasks: the purpose of the final construction is to provide shelter to a separate object in the scene. The task is also demanding because the obstacles may be elevated far from the floor, and the cost of stickiness essentially prohibits its use.

***Covering Hard* task** (Fig. 1d). Similar to *Covering*, the agent must build a shelter, but the task is modified to encourage longer term planning: there is a finite supply of movable blocks, the distribution of obstacles is denser, and the cost of stickiness is lower (-0.5 per sticky block). It thus incorporates key challenges of the *Silhouette* task (reasoning about which blocks to make sticky), the *Connecting* task (reasoning about precise block layouts), and the *Covering* task (reasoning about arch-like structures). The limited number of blocks necessitates foresight in planning (e.g. reserving long blocks to cover long obstacles). The reward function and termination criterion are the same as in *Covering*.

4. Agents

With our suite of construction tasks, we can now tackle the question we posed at the top of the Introduction: what would an agent need to perform complex construction behaviors? We expect agents which have explicit structured representations to perform better, due to their capacity for relational reasoning, compositionality, and combinatorial generalization. We implement seven construction agents which vary in the degree of structure in their observation types, internal representations, learning algorithms, and action specifications, as summarized in Table 1 and Fig. 2.

4.1. Observation formats

Each construction task (Sec. 3) provides object state and/or image observations. Both types are important for construction agents to be able to handle: we ultimately want agents that can use symbolic inputs, e.g., the representations in computer-aided design programs, as well as raw sensory inputs, e.g., photographs of a construction site.

Agent	Observation	Encoder	Policy	Planning	Learning alg.	Action space
RNN-RS0	Object	RNN	MLP/vector	-	RS0	Continuous
CNN-RS0	Image	CNN	MLP/vector	-	RS0	Continuous
GN-RS0	Object	-	GN/graph	-	RS0	Continuous
GN-DQN	Object	-	GN/graph	-	DQN	Discrete
GN-DQN-MCTS	Object	-	GN/graph	MCTS	DQN	Discrete
CNN-GN-DQN	Seg. image	Per-object CNN	GN/graph	-	DQN	Discrete
CNN-GN-DQN-MCTS	Seg. image	Per-object CNN	GN/graph	MCTS	DQN	Discrete

Table 1. Full agent architectures. Each component is as described in Sec. 4 and also illustrated in Fig. 2. All agents can be trained with either relative or absolute actions.

Object state: These observations contain a set of feature vectors that communicate the objects' positions, orientations, sizes, types (e.g., obstacle, movable, sticky, etc.). Contact information between objects is also provided, as well as the order in which objects were placed in the scene (see Supplemental Sec. C).

Image: Observed images are RGB renderings of the scene, with (x, y) coordinates appended as two extra channels.

Segmented images: The RGB scene image is combined with a segmentation mask for each object, thus comprising a set of segmented images (similar to Janner et al., 2019).

4.2. Encoders

We use two types of internal representations for computing policies from inputs: fixed-length vectors and directed graphs with attributes.

CNN encoder: The convolutional neural network (CNN) embeds an input image as a vector representation.

RNN encoder: Object state input vectors are processed sequentially with a recurrent neural network (RNN)—a gated recurrent unit (GRU) (Cho et al., 2014)—in the order they were placed in the scene, and the final hidden state vector is used as the embedding.

Graph encoder: To convert a set of state input vectors into a graph, we create a node for each input object, and add edges either between all nodes or a subset of them (see Supplemental Sec. C.2).

Per-object CNN encoder: To generate a graph-based representation from images, we first split the input image into segments, and generate new images with only single objects. Each of these are passed to a CNN, and the output vectors are used as nodes in a graph, with edges added as above.

4.3. Policies

MLP policy: Given a vector representation, we obtain a policy using a multi-layer perceptron (MLP), which outputs actions or Q-values depending on the learning algorithm.

GN policy: Given a graph-based representation from a graph encoder or a per-object CNN, we apply a stack of three graph networks (GN) (Battaglia et al., 2018) arranged in series, where the second net performs some number of recurrent steps, consistent with the "encode-process-decode" architecture described in Battaglia et al. (2018). Unless otherwise noted, we used three recurrent steps.

4.4. Actions

In typical RL and control settings that involve placing objects, the agent takes *absolute* actions in the frame of reference of the observation (e.g. Silver et al., 2016; 2018; Zhang et al., 2018; Ganin et al., 2018; Janner et al., 2019). We implement this approach in our "absolute action" agents, where, for example, the agent might choose to "place block D at coordinates $(5.3, 7.2)$". However, learning absolute actions scales poorly as the size of the environment grows, because the agent must effectively re-learn its construction policy at every location.

To support learning compositional behaviors which are more invariant to the location in the scene (e.g. stacking one block on top of another), we develop an object-centric alternative to absolute actions which we term *relative* actions. With relative actions, the agent takes actions in a reference frame relative to one of the objects in the scene. This is a natural way of expressing actions, and is similar to how humans are thought to choose actions in some behavioral domains (Ballard et al., 1997; Botvinick & Plaut, 2004).

The different types of actions are shown at the bottom of Fig. 2, with details in Supplemental Sec. B.

Continuous absolute actions are 4-tuples (X, x, y, s), where X is a horizontal cursor to choose a block from the available blocks at the bottom of the scene, "snapping" to the closest one, (x, y) determines its placement in the scene and the sign of s indicates stickiness (see Sec. 3).

Continuous relative actions are 5-tuples, $(X, x, y, \Delta x, s)$,

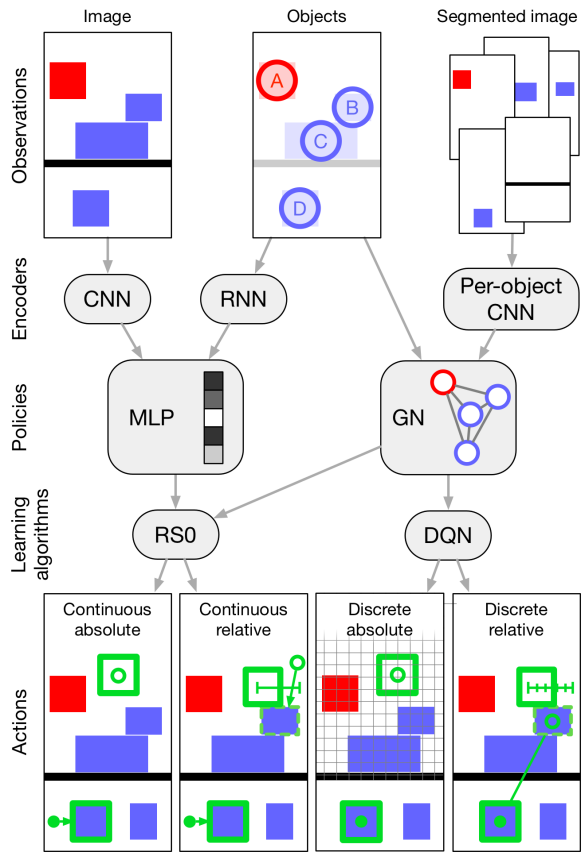

Figure 2. Summary of our construction agents' components. Each agent is defined by an observation format, encoder, policy, learning algorithm, and output action space. We evaluated many of the compatible combinations of these components, as indicated by the grey arrows. For "continuous absolute" actions, the agent picks a block from the bottom (solid green circle), and places it in the scene (empty green circle). For "continuous relative" actions, the agent picks a block from the bottom (solid green circle), and places it in the scene (empty green circle) with a relative horizontal offset from the nearest block it snaps to. "Discrete absolute" actions are similar to continuous absolute actions, except with discrete coordinates. For "discrete relative" actions, the agent picks an edge between a block at the bottom (solid green circle) and block in the scene (empty green circle), and a relative horizontal offset.

where X and s are as before, (x, y) is used to choose a reference block (by snapping to the closest one), and Δx determines where to place the objects horizontally relatively to the reference object, the vertical positioning being automatically adjusted.

Discrete absolute actions are 4-tuples (u, i, j, s) where u is an index over the available objects, i, j indicate the discrete index at which to place the object in a grid-like 2D discretization of space, and s indicates stickiness.

Absolute actions and continuous relative actions are easily

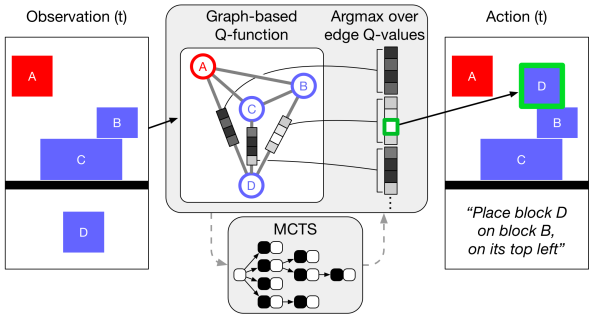

Figure 3. Structure of the GN-DQN agents. The agent takes in a graph-structured representation of the scene where each object corresponds to a node in the graph, and passes this representation through a GN. The GN produces a vector of Q-values for each edge, corresponding to relative actions for picking a block (the start node of the edge) and placing it on another object (the end node of the edge) at a given offset (the edge attribute). To choose actions, the agent takes an $\arg\max$ across all edges' Q-values and then converts the edge and offset into (x, y) positions.

implemented by any agent that outputs a single fixed-length continuous vector, such as that output by an MLP or the global output feature of a GN.

Discrete relative actions are triplets, (e, i, s), where $e := (u, v)$ is an edge in the input graph between the to-be-placed block u and the selected reference block v, i is an index over finely discretized horizontal offsets to place the chosen block relatively to the reference block's top surface, and s is as before.

Discrete relative actions are straightforward to implement with a graph-structured internal representation: if the nodes represent objects, then the edges can represent pairwise functions over the objects, such as "place block D on top of block B" (see Fig. 3).

4.5. Learning algorithms

The internal vector and graph representations are used to produce actions either by an explicit policy or a Q-function.

RS0 learning algorithm: For continuous action outputs, we use an actor-critic learning algorithm that combines retrace with stochastic value gradients (denoted RS0) (Munos et al., 2016; Heess et al., 2015; Riedmiller et al., 2018).

DQN learning algorithm: For discrete action outputs, we use Q-learning implemented as a deep Q network (DQN) from Mnih et al. (2015), with Q-values on the edges, similar to Hamrick et al. (2018). See Sec. 4.4 and Fig. 3.

MCTS: Because the DQN agent outputs discrete actions, it is straightforward to combine it with standard planning techniques like Monte-Carlo Tree Search (Coulom, 2006; Silver et al., 2016) (see Fig. 3). We use the base DQN

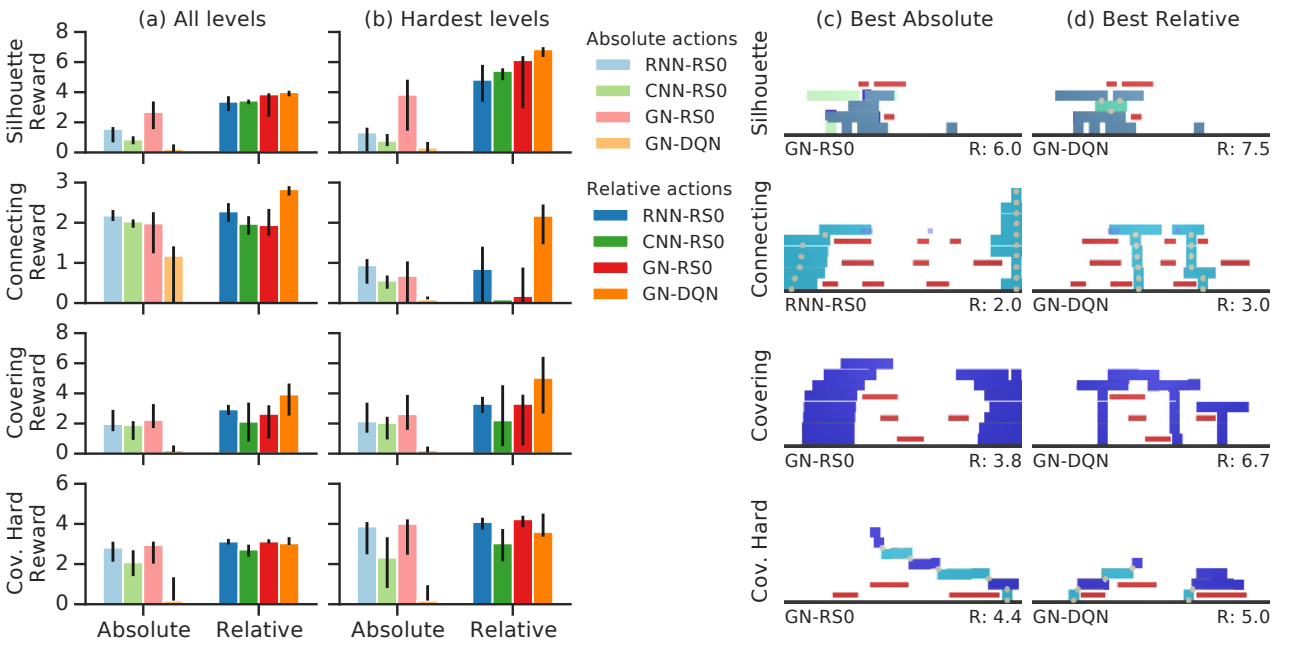

Figure 4. Comparison of absolute and relative actions for model-free agents. (a) Comparison of rewards, averaged across all levels of the curricula. (b) The same as in (a), but for the hardest level of each curricula. (c-d) Qualitative comparison between the best-performing absolute and relative seeds at the hardest curriculum levels in *Silhouette*, *Connecting*, and *Covering*.

agent as a prior for MCTS, and use MCTS with various budgets (either only at test time, only during training, or both), thereby modifying the distribution of experience fed to the learner. As a baseline, we also perform MCTS without the model-free policy prior. In all results reported in the main text, we use the environment simulator as our model; we also explored using learned models with mixed success (see Supplemental Sec. E.3).

5. Experiments and results

We ran experiments to evaluate the effectiveness of different agent architectures (see Table 1) on our construction tasks; we also tested several heuristic baselines to estimate bounds on our tasks (see Supplemental Sec. A.2). We focused on quantifying the effect of structured actions (Sec. 5.1), the effect of planning both during training and at decision time (Sec. 5.2), zero-shot generalization performance on larger and more complex scenes (Sec. 5.3). In all experiments, we report results for 10 randomly initialized agents (termed "seeds") which were trained until convergence. Each seed is evaluated on 10,000 scenes, and in all figures we report median performance across seeds as well as errorbars indicating worst and best seed performance.

For efficient training, we found it was important to apply a curriculum which progressively increases the complexity of the task across training episodes. In *Silhouette*, the curriculum increases the number of targets. In *Connecting*, it increases the elevation of the targets. In the *Covering* tasks, it increases the elevation of the obstacles. Details are available in Supplemental Sec. A.2. In our analysis, we evaluated each seed on scenes generated either uniformly at random across all difficulty levels, or only at the hardest difficulty level for each task.

5.1. Relative versus absolute actions

We find that agents which use relative actions consistently outperform those which use absolute actions. Across tasks, almost every relative action agent converges at a similar or higher median performance level (see Fig. 4a), and the best relative agents achieve up to 1.7 times more reward than the best absolute agents when averaging across all curriculum levels. When considering only the most advanced level, the differences are larger with factors of up to 2.4 (Fig. 4b).

Fig. 4c shows examples of the best absolute agents' constructions. These outcomes are qualitatively worse than the best relative agents' (Fig. 4d). The absolute agents do not anticipate the long term consequences of their actions as well, sometimes failing to make blocks sticky when necessary, or failing to place required objects at the base of a structure, as in Fig. 4c's *Silhouette* example. They also fall into poor local minima, building stacks of blocks on the

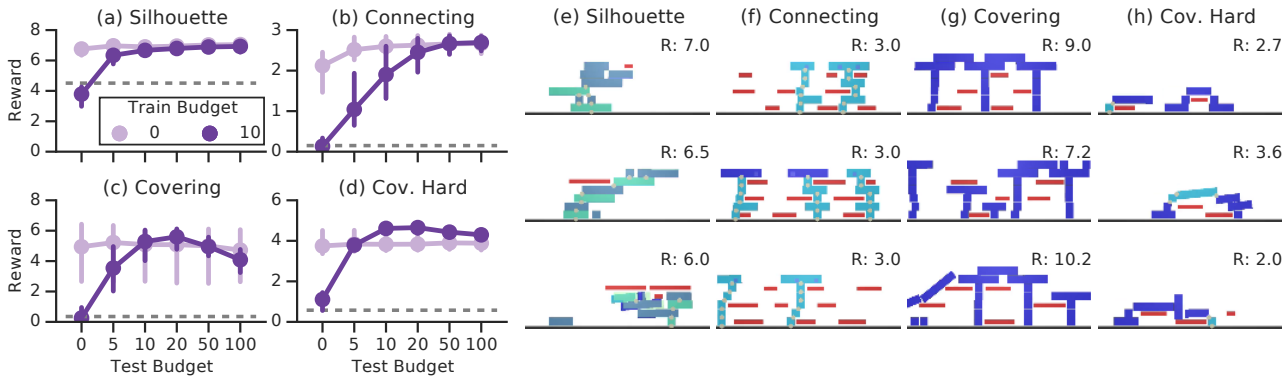

Figure 5. (a-d) Comparison of different training and testing budgets for the model-based GN-DQN-MCTS agents on the hardest curricula levels. The gray dashed line corresponds to a pure-planning agent with a search budget of 1000. (e-h) Representative structures built by GN-DQN-MCTS agents, chosen from a set of 10 random episodes for each task. The *Silhouette* and *Connecting* agents use training budgets of 0 and test budgets of 50; the *Covering* agent uses a training budget of 0 and test budget of 5, and the *Covering Hard* agent uses a train and test budget of 10. An example of sub-optimal behavior has been chosen for the third row when available. The entire set of random episodes are shown in Supplemental Sec. H.

sides of the scene which fail to reach or cover objects in the center, as in Fig. 4c's *Connecting* and *Covering* examples.

By contrast, the best relative agents (which, across all tasks, were GN-DQN) construct more economical solutions (e.g., Fig. 4d, *Connecting*) and discover richer strategies, such as building arches (Fig. 4d, *Covering*). The GN-DQN agent's superior performance suggests that structured representations and relative, object-centric actions are powerful tools for construction. Our qualitative results suggest that these tools provide invariance to dimensions such as spatial location, which can be seen in cases where the GN-DQN agent re-uses local block arrangements at different heights and locations, such as the T structures in Fig. 4g.

Most agents achieve similar levels of performance of *Covering Hard*: GN-RS0 has the best median performance, while GN-DQN has the best overall seed. But inspecting the qualitative results (Fig. 4), even the best relative agent does not give very strong performance. Though *Covering Hard* involves placing fewer blocks than other tasks because of their limited supply, reasoning about the sequence of blocks to use, which to make sticky, etc. is indeed a challenge, which we will address in the next section with our planning agent.

Interestingly, the GN-RS0 and GN-DQN agents have markedly different performance despite both using the same structured GN policy. There are a number of subtle differences, but notably, the object-centric information contained in the graph of the GN-RS0 agent must be pooled and passed through the global attribute to produce actions, while the GN-DQN agent *directly* outputs actions via the graph's edges. This may allow its policy to be more analogous to the actual structure of the problem than the GN-RS0 agent.

The CNN-RS0 agent's performance is generally poorer than the GN-based agents', but the observation formats are also different: the CNN agent must learn to encode images, and it does not receive distinct, parsed objects. To better control for this, we train a GN-based agent from pixels, labelled CNN-GN-DQN, described in Sec. 4. The CNN-GN-DQN agent achieves better performance than the CNN-RS0 agent (see Supplemental Fig. C.2). This suggests that parsing images into objects is valuable, and should be investigated further in future work.

5.2. Model-based versus model-free

Generally, complex construction should require longer-term planning, rather than simply reactive decision-making. Given a limited set of blocks, for example, it may be crucial to reserve certain blocks for roles they uniquely satisfy in the future. We thus augment our GN-DQN agent with a planning mechanism based on MCTS (see Sec. 4.5) and evaluate its performance in several conditions, varying the search budget at training and testing time independently (a search budget of 0 corresponds to no planning).

Our results (Fig. 5) show that planning is generally helpful, especially in *Connecting* and *Covering Hard*. In *Connecting*, planning with a train budget of 10 and test budget of 100 improves the agent's median reward from 2.17 to 2.72 on the hardest scenes, or from 72.5% to 90.6% of the optimal reward of 3. In *Covering Hard*, planning with a train and test budget of 10 improves the agent's median reward from 3.60 to 4.61. Qualitatively, the planning agent appears to be close to ceiling (Fig. 5h). Note that a pure-planning agent (Fig. 5a-d, gray dashed line) with a budget of 1000 still performs poorly compared to learned policies, underscoring

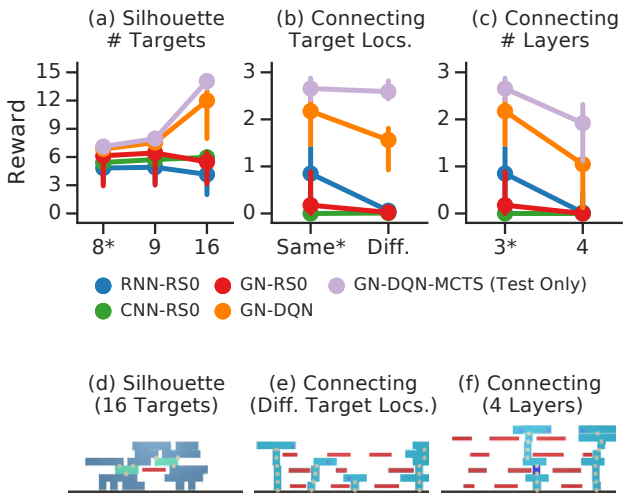

Figure 6. Zero-shot generalization performance of various agents. In all cases, asterisks indicate values seen during training. (a) In *Silhouette*, we varied the number of targets from 8 to 16. (b) In *Connecting*, we first varied the location of the target locations to be either on the same level or on different levels. (c) In *Connecting*, we also varied the number of obstacle layers from three to four. (d-f) Examples of the GN-DQN-MCTS generalizing to new scenes. In each case, the agent has a train budget of 0 and a test budget of 50. (d) Generalization to 16 targets in *Silhouette*. (e) Generalization to multi-level targets in *Connecting*. (f) Generalization to 4 layers of obstacles and higher targets in *Connecting*.

the difficulty of the combinatorially large search space in construction. In Supplemental Sec. E, we discuss of the trade-offs of planning during training, testing, or both.

5.3. Generalization

We next ask: how do our agents generalize to conditions beyond those on which they were trained? In *Silhouette*, our agents only experience 1-8 targets during training, so we test them on 9 and 16 targets. In *Connecting*, agents always experience targets at the same elevation within one scene during training, so we test them on targets appearing at multiple different levels in the same scene (in one condition) and all at a higher elevation than experienced during training.

We find that the GN-DQN and especially GN-DQN-MCTS agents with relative actions generalize substantially better than others. In *Silhouette*, the GN-DQN-* agents cover nearly twice as many targets as seen during training, while the other agents' performances plateau or fall off dramatically (Fig. 6a). In *Connecting* with targets at multiple different levels, the GN-DQN and GN-DQN-MCTS agents' performances drops only slightly, while other agents' performance drops to near 0 (Fig. 6b). With increased numbers of obstacle layers in *Connecting*, both agents' performances drop moderately but remain much better than the less structured agents (Fig. 6c). Fig. 6d-f show the qualitative generalization behavior of the GN-DQN-MCTS agent. Overall, these generalization results provide evidence that structured agents are more robust to scenarios which are more complex than those in their training distribution. This is likely a consequence of their ability to recognize structural similarity and re-use learned strategies.

5.4. Iterative relational reasoning

Recurrent GNs support iterative relational reasoning by propagating information across the scene graph. We vary the number of recurrent steps in our GN-DQN agent to understand how its relational reasoning capacity affects task performance. We find that increasing the number of propagation steps from 1 to 3 to 5 generally improves performance (to a point) across all tasks: in *Silhouette*, the median rewards were 3.75, 4.04 and 4.06; in *Connecting*, 2.49, 2.84, and 2.81; in *Covering*, 3.41, 3.96, and 4.01; and in *Covering Hard*, 2.62, 3.03, and 3.02, respectively.

6. Discussion

We introduced a suite of representative physical construction challenges, and a family of RL agents to solve them. Our results suggest that graph-structured representations, model-based planning under model-free search policies, and object-relative actions are valuable ingredients for achieving strong performance and effective generalization. We believe this work is the first to demonstrate agents that can learn rich construction behaviors in complex settings with large numbers of objects (up to 40-50 in some cases), and can satisfy challenging functional objectives that go beyond simply matching a pre-specified goal configuration.

Given the power of object-centric policies, future work should seek to integrate methods for detecting and segmenting objects from computer vision with learned relational reasoning. Regarding planning, this work only scratches the surface, and future efforts should explore learned models and more sophisticated search strategies, perhaps using policy improvement (Silver et al., 2018) and gradient-based optimization via differentiable world models (Sanchez-Gonzalez et al., 2018). Finally, procedurally generating problem instances that require complex construction solutions is challenging, and adversarial or other learned approaches may be promising future directions.

Our work is only a first step toward agents which can construct complex, functional structures. However we expect approaches that combine rich structure and powerful learning will be key making fast, durable progress.

7. Acknowledgements

We would like to thank Yujia Li, Hanjun Dai, Matt Botvinick, Andrea Tacchetti, Tobias Pfaff, Cédric Hauteville, Thomas Kipf, Andrew Bolt, Piotr Trochim, Victoria Langston, Nicole Hurley, Tejas Kulkarni, Vlad Mnih, Catalin Ionescu, Tina Zhu, Thomas Hubert, and Vinicius Zambaldi for helpful discussions, input, and feedback on this work.

References

Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. Learning to poke by poking: Experiential learning of intuitive physics. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, 2016.

Aldefeld, B. Variation of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, 1988.

Arnon, D. Geometric reasoning with logic and algebra. *Artificial Intelligence*, 37(1–3):37–60, 1988.

Azizzadenesheli, K., Yang, B., Liu, W., Lipton, E. B. Z. C., and Anandkumar, A. Surprising negative results for generative adversarial tree search. *arXiv preprint arXiv:1806.05780*, pp. 1–25, 2018.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., and Rao, R. P. Deictic codes for the embodiment of cognition. *Behavioral and Brain Sciences*, 20(4):723–742, 1997.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, 2016.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, pp. 1–38, 2018.

Bhattacharyya, A., Malinowski, M., Schiele, B., and Fritz, M. Long-term image boundary prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

Botvinick, M. and Plaut, D. C. Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action. *Psychological review*, 111(2):395, 2004.

Bouma, W., Fudosa, I., Hoffmann, C., Cai, J., and Paige, R. Geometric constraint solver. *Computer-Aided Design*, 27 (6):487–501, 1995.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Catto, E. Box2D. https://box2d.org/, 2013.

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. A compositional object-based approach to learning physical dynamics. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 2017.

Chen, S.-s. *Advances in Spatial Reasoning*, volume 2. Intellect Books, 1990.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Chou, S.-C. *Mechanical geometry theorem proving*. Kluwer Academic, 1987.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Dai, H., Khalil, E., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

Diuk, C., Cohen, A., and Littman, M. L. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 240–247. ACM, 2008.

Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A., and Levine, S. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Finn, C., Goodfellow, I., and Levine, S. Unsupervised learning for physical interaction through video prediction. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, 2016.

Fragkiadaki, K., Agrawal, P., Levine, S., and Malik, J. Learning visual predictive models of physics for playing billiards. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, 2016.

Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S., and Vinyals, O. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Groth, O., Fuchs, F. B., Posner, I., and Vedaldi, A. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3338–3346. Curran Associates, Inc., 2014.

Hamrick, J. B., Ballard, A. J., Pascanu, R., Vinyals, O., Heess, N., and Battaglia, P. W. Metacontrol for adaptive imagination-based optimization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 2017.

Hamrick, J. B., Allen, K. R., Bapst, V., Zhu, T., McKee, K. R., Tenenbaum, J. B., and Battaglia, P. W. Relational inductive bias for physical construction in humans and machines. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, 2018.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Tassa, Y., and Erez, T. Learning continuous control policies by stochastic value gradients. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS 2015)*, 2015.

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.

Juliani, A., Berges, V.-P., Vckay, E., Gao, Y., Henry, H., Mattar, M., and Lange, D. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M. (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Kool, W. and Welling, M. Attention solves your TSP. *arXiv preprint arXiv:1803.08475*, 2018.

Lerer, A., Gross, S., and Fergus, R. Learning physical intuition of block towers by example. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, 2016.

Li, W., Azimi, S., Leonardis, A., and Fritz, M. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, pp. 1–20, 2016.

Li, W., Leonardis, A., and Fritz, M. Visual stability prediction and its application to manipulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mottaghi, R., Bagherinezhad, H., Rastegari, M., and Farhadi, A. Newtonian image understanding: Unfolding the dynamics of objects in static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Pascanu, R., Li, Y., Vinyals, O., Heess, N., Buesing, L., Racanière, S., Reichert, D., Weber, T., Wierstra, D., and Battaglia, P. Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*, pp. 1–13, 2017.

Pfalzgraf, J. On geometric and topological reasoning in robotics. *Annals of Mathematics and Artificial Intelligence*, 19:279, 1997.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4344–4353, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pp. 4967–4976, 2017.

Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

Scholz, J., Levihn, M., Isbell, C., and Wingate, D. A physics-based model prior for object-oriented mdps. In *International Conference on Machine Learning*, pp. 1089–1097, 2014.

Schreck, P., Mathis, P., and Narboux, J. Geometric construction problem solving in computer-aided learning. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, volume 24, pp. 1139–1144, 2012.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377.

Toussaint, M. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, volume 24, pp. 1930–1936, 2015.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: unsupervised discovery of objects and their interactions. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, 2018.

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Watters, N., Tacchetti, A., Weber, T., Pascanu, R., Battaglia, P., and Zoran, D. Visual interaction networks: Learning a physics simulator from video. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

Winston, P. Learning structural descriptions from examples. *AI Technical Reports (1964 - 2004)*, 1970.

Wu, J., Yildirim, I., Lim, J. J., Freeman, W. T., and Tenenbaum, J. B. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS 2015)*, 2015.

Wu, J., Lim, J. J., Zhang, H., Tenenbaum, J. B., and Freeman, W. T. Physics 101: Learning physical object properties from unlabeled videos. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

Wu, J., Lu, E., Kohli, P., Freeman, W. T., and Tenenbaum, J. B. Learning to see physics via visual de-animation. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., and Battaglia, P. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

Zhang, A., Lerer, A., Sukhbaatar, S., Fergus, R., and Szlam, A. Composable planning with attributes. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.