END-TO-END INVARIANCE LEARNING WITH RELATIONAL INDUCTIVE BIASES IN MULTI-OBJECT ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Although reinforcement learning has seen remarkable progress over the last years, solving robust dexterous object-manipulation tasks in multi-object settings remains a challenge. In this paper, we focus on models that can learn manipulation tasks in fixed multi-object settings *and* extrapolate this skill zero-shot without any drop in performance when the number of objects changes. We consider the generic task of moving a single cube out of a set to a goal position. We find that previous approaches, which primarily leverage attention and graph neural network-based architectures, do not exhibit this invariance when the number of input objects changes while scaling as K^2. We analyse effects on generalization of different relational inductive biases and then propose an efficient plug-and-play module that overcomes these limitations. Besides exceeding performances in their training environment, we show that our approach, which scales linearly in K, allows agents to extrapolate and generalize zero-shot to any new object number.

1 INTRODUCTION

Deep reinforcement learning (RL) has witnessed remarkable progress over the last years, particularly in domains such as video games or other synthetic toy settings (Mnih et al., 2015; Silver et al., 2016; Vinyals et al., 2019). On the other hand, applying deep RL on real-world grounded robotic setups such as learning seemingly simple dexterous manipulation tasks in multi-object settings is still confronted with many fundamental limitations being the focus of many recent works (Duan et al., 2017; Janner et al., 2018; Deisenroth et al., 2011; Kroemer et al., 2018; Andrychowicz et al., 2020; Rajeswaran et al., 2017; Lee et al., 2021; Funk et al., 2021).

The RL problem in robotics setups is much more challenging (Dulac-Arnold et al., 2019). Compared to discrete toy environments, state and action spaces are continuous, and solving tasks typically requires long-horizon time spans, where the agent needs to apply long sequences of precise low-level control actions. Accordingly, exploration under easy-to-define sparse rewards becomes only feasible with horrendous amounts of data. This is usually impossible in the real world but has been likewise hard for computationally demanding realistic physics simulators. To alleviate this, manually designing task-specific dense reward functions is usually required, but this can often lead to undesirable or very narrow solutions. Numerous approaches exist to alleviate this even further by e.g. imitation learning from expert demonstrations (Abbeel & Ng, 2004), curricula (Narvekar et al., 2020), or model-based learning (Kaelbling et al., 1996). Another promising path is to constrain the possible solution space of a learning agent by encoding suitable inductive biases in their architecture (Geman et al., 1992). Choosing inductive biases that leverage the underlying problem structure can help to learn solutions that facilitate desired generalization capabilities Mitchell (1980); Baxter (2000); Hessel et al. (2019).

In robotics, multi-object manipulation tasks naturally suit a compositional description of their current state in terms of symbol-like entities (such as physical objects, robot parts, etc.). These representations can be directly obtained in simulator settings and ultimately hoped to be inferred robustly from learned object perception modules Greff et al. (2020a); Kipf et al. (2021); Locatello et al. (2020). While such a compositional understanding is in principle considered crucial for any systematic generalization ability Greff et al. (2020b); Spelke (1990); Battaglia et al. (2018); Garnelo et al. (2016) it remains an open question how to design an agent that can process this type of input data to leverage this promise.

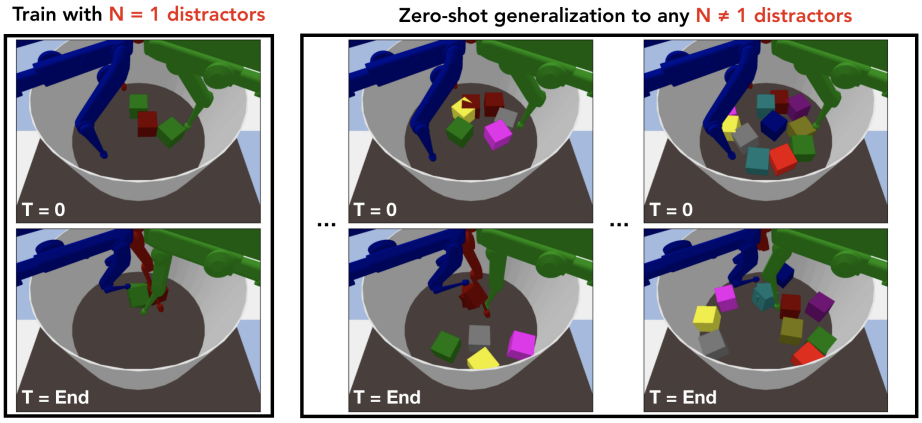


Figure 1: We consider the following **problem setting**: There are 2 identical cubes with unique identifiers in the arena. During an episode, a random cube (here green) is selected to be transported to a random episode-specific goal-location. The remaining cube (here dark red) acts as an unused distractor during that episode but could be the cube to be transported in the next episode. Can we train RL agents that learn such manipulation tasks within a fixed multi-object setting – training with only one distractor – and extrapolate this skill zero-shot when the number of distractors changes?

Consider transporting a cube to a specified target location and assume the agent has mastered solving this task with three available cubes (see Figure 1 for this problem setting which forms the basis of our work). We refer to cubes different from the ones to transport as distractors. If we now introduced two additional cubes, the solution to solve this task would remain the same, but the input distribution changed. So why not just train with much more cubes and use a sufficiently large input state? First, the learning problem requires exponentially more data when solving such a task with more available cubes (see Figure 2, left plot, red line), making it infeasible already for only half a dozen cubes. Second, we do not want to put any constraint on the possible number of cubes at test time but simply might not have this flexibility at train time. In such a task, the agent must be able to learn end-to-end a policy that is invariant to the number of distractors, while never observing at training time such input distribution shift. Therefore, we demand to learn challenging manipulation tasks in fixed multi-object settings and extrapolate this skill zero-shot without any drop in performance when the number of cubes changes. Achieving this objective will be the primary goal of this work.

An essential prerequisite for achieving this is to endow the agents with the ability to process a variable number of input objects by choosing a suitable inductive bias. One popular model class for these object encoding modules are graph neural networks (GNNs), and a vast line of previous approaches builds upon the subclass of attentional GNNs to achieve this (Zambaldi et al., 2018; Li et al., 2019; Wilson & Hermans, 2020; Zadaianchuk et al., 2020). Another possible model class - although largely ignored for robotic manipulation tasks so far - are relation networks that process input sets by computing relations for every object-object pair Santoro et al. (2017). As is the case for attentional GNNs, relation networks scale quadratically in the number of input objects (see Figure 2, right plot), which can become computationally impractical for training and inference with many objects. In this work, we are therefore primarily interested in learning invariances regarding the manipulable objects of a robotic agents' environment but not invariance regarding the objects' properties such as shape or weight which is an orthogonal robustness objective in machine learning. Without properly accounting for such modularity, task complexities can otherwise grow exponentially.

Main contributions. As a first main contribution, we will demonstrate that utilizing popular attention-based GNN approaches does not achieve the desired invariance. Instead, attention-based GNN approaches fail to extrapolate any learned behavior to a changing number of objects. As a solution, we will then present support to build agents upon relational reasoning inductive biases. In addition to a more traditional implementation of a relational network, we introduce a linearized relation network module (LRN) which improves the computational complexity in the number of objects from quadratic to linear. Finally, we show that agents based on this proposed module can extrapolate and therefore generalize zero-shot without any drop in performance when the number of objects changes. We will present supporting evidence for this currently under-explored type of generalization and necessary requirements for multi-object manipulation across two challenging robotics tasks.

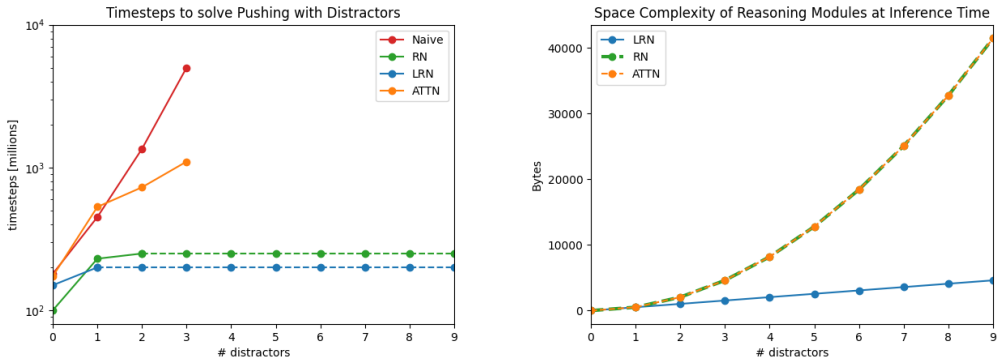


Figure 2: **Scaling challenges in multi-object manipulation**. Left: Scaling behavior of training frames required for learning to push a specified object to a target pose with additional distractors. Training agents with a naive fixed-input size MLP encoder is constrained to a predetermined object number and scales exponentially. Data points on dashed lines are extrapolated from our proposed models trained with only 2 distractors as they achieve zero-shot generalization for any more distractors. Right: Required Memory for action prediction with a different number of distractors using the ATTN object reasoning backbone vs. RN and LRN backbone.

2 Problem Setting

This work considers tasks where the goal is always to transport one of the N_o movable objects, i.e. cubes, to a target location, and all other $N_o - 1$ cubes act as distractors. Therefore, given their straightforward relation, we refer to each environment either using the total number of cubes or the number of distractors, the latter called N. Specifically, we define the objective of any learning agent as follows: First, the agent needs to learn to solve the task at train time under a *constant* and a minimal number of manipulable objects N_o^{train}. Second, the agent needs to be able to *zero-shot* extrapolate solving the equivalent task under $N_o^{test} \neq N_o^{train}$ objects in the scene. Therefore, the agent needs to learn an invariant behavior with respect to the episode-specific distractors.

Note that our task formulation (i.e. moving/picking a selected object out of a variable number of other objects) can be seen as a generic subtask of more general manipulation tasks. If we can solve this task, more complex multi-object goals could be solved more easily by specifying one object-goal pose after another. As an example, stacking tasks can be trivially decomposed as a sequence of lifting tasks where each goal is on top of the previous one. Crucially, we want to learn such tasks with a constant number of available objects because varying the number of objects during training might limit us in the real world. Possible real-world limitations include but are not limited to the increased complexity of designing automatic resetting systems able to deal with a variable number of objects which are necessary to train an agent over many episodes. We ultimately need adapting agents that can generalize when additional objects are added or removed from the environment scene.

Background on compositional multi-object manipulation tasks. Given the compositional environment structure of such real-world grounded manipulation tasks, we are concerned with input states $s_t \in S$ that comprise a set of objects describing individual rigid objects in the scene such as movable cubes $\{s_1^o, s_2^o, ..., s_{N_o}^o\}$ or the individual parts of a robot such as each finger in our setup $\{s_1^r, s_2^r, ..., s_{N_r}^r\}$. To facilitate a goal-conditioned formulation, we similarly feed additional environment-specific information in the form of a goal object s^g. Concerning our tasks, the goal information can be provided via the object state s^g that describes the target pose of a cube. To identify the object to be transported, we simply add a unique identifier feature to each object state and specify the object to be moved through the same identifier in the goal object. Translated to a robotic RL setting, the above state-space factorization requires a policy that is parameterized by a function $a_t = \pi(\{s_1^o, ..., s_{N_o}^o, s_1^r, ..., s_{N_r}^r, s^g\})$. Our demand is that this policy can deal with a variable number N_o of input objects independent of their order. In other words, the policy needs to be permutation-invariant, i.e. it produces the same output for any possible permutation of its input objects.

3 RELATED WORK

Flavors of this problem have a long history and are typically being approached by various suitable inductive biases that enable reasoning upon such object-oriented structured representations (Džeroski et al., 1998; Džeroski et al., 2001; Kaelbling et al., 2001; Van Otterlo, 2002). A popular class of neural architectures that offer these properties are various sub-types of graph neural networks (GNNs) with attention-based approaches or interaction networks being a special sub-case (Bronstein et al., 2021; Veličković et al., 2017; Vaswani et al., 2017; Battaglia et al., 2016; Santoro et al., 2017; Kipf & Welling, 2016). Zambaldi et al. (2018) was one of the first to leverage the self-attention approach in the context of deep reinforcement learning for toy and game environments from a constant number of features extracted from a CNN. The ATTN baseline in our experiments directly resembles their object-processing module, but we will also evaluate this approach when the number of input objects changes. Interestingly, most of the previous work for learning similar manipulation tasks in multi-object settings uses similar object-encoding modules based on multi-head dot product attention (MHDPA) layers to process the object-based input representations. On related stacking tasks using a robotic arm with a 1D gripper Li et al. (2019) propose a model architecture that uses an MHDPA-equivalent GNN. Compared to our work, their method heavily builds upon handcrafted curricula to achieve some degree of fading zero-shot adaptation to related multi-object tasks. Another recent approach is SMORL by Zadaianchuk et al. (2020), which similarly passes object-oriented representations to an attention-based and goal-conditioned policy that learns the rearrangement of objects. The method can adapt to related test tasks with fewer objects but needs to be fine-tuned on each of such tasks individually. Finally, Wilson & Hermans (2020) proposes another approach to learn to manipulate many objects with a focus on sim-to-real. The agent model is conditioned on a state embedding obtained through a GNN and builds upon an MHDPA-equivalent module, with a set of object poses as input. Due to a slightly different task specification, multi-object generalization is achieved more easily as the movement to solve their tasks for a changing number of objects remains the same. Recently, Zhou et al. (2022) studied a similar generalization setting as ours and again with a focus on attention-based architectures. Importantly, their results validate the persisting limitation with attentional backbones as they also see decreasing generalization capabilities in terms of success rate with an increasing number of objects in the OOD task. **In summary**, previous approaches suffer from various fundamental limitations regarding our desiderata. First, the compute of MHDPA-based architectures scales quadratically with the number of objects (see Figure 2), which is especially limiting for training and inference with $N_o \gg 1$. Second, to achieve any meaningful extrapolation for different object numbers, the training domain must cover a variable and typically large number of objects. Third, the performance at test time typically decreases for a new number of objects, and fine-tuning is required to recover the original performance.

Related work on relation networks. Our method is motivated by means of an alternative perspective on this problem via relation networks Raposo et al. (2017); Santoro et al. (2017). Relation networks process input sets by computing relations for every object-object pair, followed by a permutation invariant aggregation function. Relation networks scale quadratically in the number of input objects Santoro et al. (2017). They are predominantly used for various non-RL tasks such as visual and text-based question answering, few-shot image comparison, or object detection Santoro et al. (2017); Sung et al. (2018); Hu et al. (2018). Importantly, we are not aware of RL applications approaching this problem utilizing relation networks as relational inductive biases. In addition, standard relation networks typically operate on a fixed number of input objects and suffer from the same scalability issues as attention. Our linear relation network module alleviates this limitation.

4 AGENTS WITH OBJECT REASONING MODULES

We will now describe the agent model used in this work and then present the linear relation network (LRN) that we propose as an alternative object reasoning module for learning compositional multi-object manipulation skills (see Figure 3 for a summary). In Section 5, we will benchmark this module against an attention-based module (ATTN) aiming to resemble the prevailing choice of architecture in previous approaches. Moreover, we will also compare our LRN module against a naive relation network implementation (RN) with inferior scaling in computational costs to further highlight the sole benefits of relations over naive attention. We refer to the appendix for further implementation details.

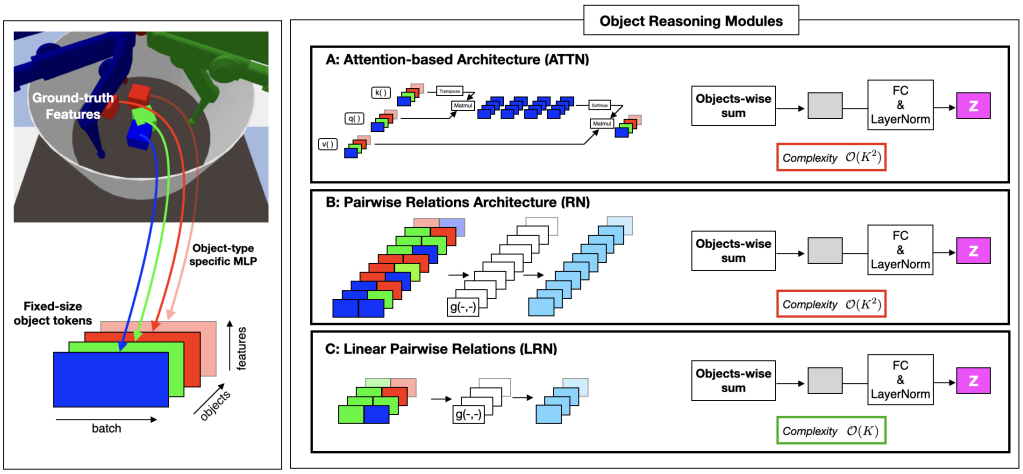


Figure 3: Summary of object reasoning modules studied in this work. Left: In a first step, ground-truth feature states are encoded into fixed-size object tokens via object-type specific small MLPs. Right: An object-encoder module processes this token set and outputs a fixed-size representation (highlighted in pink). This representation forms the input of any downstream predictor required to learn the optimal policy. Besides a naive attention-based approach chosen as an object encoder module by most previous works (Box A), we propose to use architectures based on relational reasoning and discuss two variants. The first presented approach is a relation network architecture that combines all possible pairwise relations, and we refer to this module as RN (Box B). The second and main method is our Linear Relation Network architecture which only uses relations involving the goal object, thereby scaling linearly in the number of input tokens (Box C). We refer to this model as LRN.

4.1 Agent Model

Our agents will build upon the following standard architecture scaffold. In a first step, ground-truth simulator state features represented by objects are encoded into fixed-size object tokens $\mathbf{o}_i \in \mathbb{R}^d$ via object-type specific small MLPs. In our case, there are three robotic states for each finger, goal objects, and physical objects for the cube. Next, an object-reasoning encoder module processes this token set and outputs a fixed-size representation $\mathbf{z}$ (highlighted in pink in Figure 3). Finally, this representation forms the input of any downstream predictor required to learn the optimal policy. Here, we will assume a small MLP to predict the action (policy); however, we can likewise use the representation to predict the value (value network) or Q-Value (when combined with an action) when using Q-Learning or actor-critic approaches. The primary focus of this work is on proposing a simple and flexible object-encoder module that overcomes the limitations of previous approaches.

4.2 Linear Relation Network Reasoning Module

Similar to the original Relation Network from Santoro et al. (2017) used for visual question-answering tasks, our module builds upon the idea of reasoning about future actions from learned relations between the given objects. However, in the context of goal-conditioned RL, it seems reasonable that the next optimal action is primarily governed by the relations of the available objects with the particular goal of interest. In contrast, most other relations might be only important in a sparse and context-dependent way. We, therefore, argue to focus on these relations, and propose the following adjusted composite representation function

$$\mathbf{z} = \text{LayerNorm}\left(\mathbf{f}_\phi\left(\sum_i^K \mathbf{g}_\theta(\mathbf{o}_i, \mathbf{o}_g)\right)\right), \tag{1}$$

where the $\mathbf{o}_i \in \mathbb{R}^d$ are common-size embeddings of the K input objects. The objects are the scene's moveable elements in our setting, which means cubes and the three robotic finger states. The goal information of the task is being embedded into a separate object $\mathbf{o}_g \in \mathbb{R}^d$. First, the shared learnable function $\mathbf{g}_\theta$ computes the K embeddings $\mathbf{r}_i \in \mathbb{R}^d$ for each object-goal relation. For our experiment $\mathbf{g}_\theta$ is a 4-layer MLP. Next, a dimension-wise sum operation over the K relations is applied, which

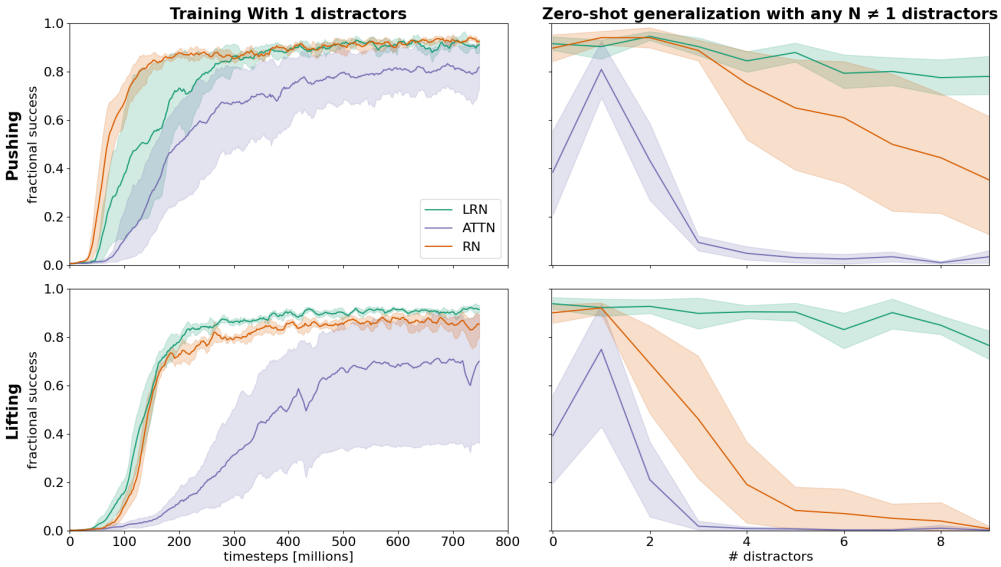


Figure 4: **Training curves and zero-shot generalization from fixed multi-object train domains.** Left: Training curves for PushingCube (top) and LiftingCube (bottom) tasks in one-distractor environments (5 random seeds per model). Our RN and LRN modules perform favorably against attention-based implementation in terms of final success, sample efficiency, and variance across different seeds. Right: Evaluating the corresponding zero-shot generalization on equivalent N-distractor environments for both tasks. Our LRN and RN modules show a strong extrapolation ability of their learned manipulation skills to environments with many more objects.

serves as a permutation-invariant aggregation function. The final representation $\mathbf{z} \in \mathbb{R}^{d_z}$ is obtained after processing this intermediate representation by another learnable function $\mathbf{f}_\phi$ followed by a layer normalization operation (Ba et al., 2016). The function $\mathbf{f}_\phi$ can serve the purpose of learning further necessary relations from the K object-goal relations. We will further discuss the models' empirical ability to learn any other needed relations through the combination of this minimal set of goal-object relations in Section 5.5. In our experiments, we will use a simple linear transformation for $\mathbf{f}_\phi$. The layer normalization ensures that the final representation $\mathbf{z}$ remains in distribution for a changing number of input objects. The overall complexity of this goal-conditioned linear relation network reasoning module is linear in K, i.e. $\mathcal{O}(K)$. See Figure 3 (Box C) for a summary. To assess the impact of inferring the K object-goal relations only, we will also implement an equivalent variant that considers *all* possible K^2 relations, similar to the naive implementation of relation networks. To the best of our knowledge, they have been largely ignored for any similar robotic manipulation tasks and generalization so far. Inspired by the original relation network module, we will refer to this module as RN.[1]

5 Experiments

We now aim to evaluate the previously discussed architectures for learning challenging manipulation tasks in fixed multi-object settings *and* their ability to extrapolate this skill zero-shot when the number of objects changes. We will start by outlining the experimental setup in Section 5.1 for two particularly generic tasks of pushing and lifting objects. We will then discuss the learning behavior in terms of sample efficiency and performance of our proposed modules against the common attention-based approach from previous works in Section 5.2. We close this section by demonstrating the models' zero-shot OOD extrapolation skills in Section 5.3, then by analyzing the impact of the train domain in section 5.4, and, finally, by looking at the ability of the model of reasoning about higher order relations section 5.5. Additional ablations on the architecture together with a detailed analysis of the models' OOD generalization and learned relations are discussed in the appendix (9.2.1 - 9.2.4).

[1]Note that the original relation network module for question answering tasks computes relations that are additionally conditioned on the question and operates on a fixed number of input features such that normalization is not required for these problem domains.

5.1 EXPERIMENTAL SETUP.

Both tasks are derived from the simulated robotic manipulation setup and tasks proposed in Causal-World (Ahmed et al., 2020) which builds upon the robotic trifinger design (Wüthrich et al., 2020; Bauer et al., 2021). Compared to previous work on this setup, we attempt to solve tasks in scenes that contain more than 1 cube Dittadi et al. (2021); Ahmed et al. (2020); Allshire et al. (2021). Specifically, the goal is to move any of the 2 available cubes to a freely specifiable target position. We cover a simpler sub-task where the target position is constrained to the floor and the more general task where the target position is also allowed to be above the ground. Such a lifting task involves learning to hold the cube stable in the air at a variable height. We refer to these tasks as PushingCube and LiftingCube, respectively. As discussed in Section 2, we specify the object to be rearranged through a matching object identifier in one of the goal objects' features. For both tasks, we use a generic reward function designed to work for any single-goal task in this environment. The reward is composed of three terms: (1) a reward term that directly measures the success at each time step, which is defined as the volumetric overlap of the cube with the goal cube; (2) a second reward term that encourages moving the target object closer to the goal; (3) a third curiosity-inspired reward term for efficient exploration that gives a positive reward signal for moving the target object when being further away from the goal. Across all tasks, we will report the factional success as the universal evaluation metric at the last timestep of an episode. Tasks can be visually considered solved with a score around 80%, and we can therefore ensure an objective and interpretable performance measure (Ahmed et al., 2020; Dittadi et al., 2021). All models are being trained under the described reward with PPO (Schulman et al., 2017) on distributed workers with 5 random seeds each. Initial object poses and the goal pose are randomly sampled at the beginning of each episode, and the episodic task needs to be solved within 30 simulation seconds or 2500 frames (10 seconds per object). We compare the instantiation of our proposed RN and LRN network architectures against a representative attention-based baseline derived from Zambaldi et al. (2018), referred to as ATTN. We ensured a fair comparison of this baseline through extensive hyperparameter optimization (see Section 9.2.5). We refer to the supplement for a comprehensive account of further implementation details.

5.2 LEARNING TO MANIPULATE OBJECTS FROM FIXED MULTI-OBJECT TRAIN DOMAINS.

Quantitative results for learning to manipulate objects from the 2-cubes (i.e. 1 distractor) train domain are summarized in Figure 4 for PushingCube in the top panel and LiftingCube in the lower panel. We refer to the project-site[2] for additional videos of trained agents. Across tasks, we can see that our proposed backbone performs very favorably over the attention-based ATTN baseline. Specifically, both relation network variants (RN and LRN) reliably succeed in learning to solve the tasks almost perfectly with very high scores. On the other hand, ATTN exhibits much higher variance across different seeds and takes longer to train. Interestingly, RN and LRN reveal very similar robust learning behavior in terms of training time steps. A significant difference, however, is the required compute. For environments with 9 cubes, the LRN module requires computing only $K = 12$ relations (9 cube-goal and 3 finger-goal relations), whereas RN combines $(K + 1)^2 = 169$ relations to process its observation set, highlighting the favorable properties of our LRN module.

Summary. The relation network reasoning modules and our LRN model represent a promising alternative to attention-based architectures commonly used in prior works for learning to manipulate an object on a multi-object train domain. Our module outperforms these approaches in final success, sample efficiency, and variance across different seeds.

5.3 ZERO-SHOT GENERALIZATION TO N-OBJECT ENVIRONMENTS.

We now evaluate the trained models' generalization ability when the number of objects is different from training. Our main result to assess this question is summarized in Figure 4 (right column). Remember that all agents have been trained for pushing and lifting an object in 2-cube environments, synonymous of 1 distractor. Testing these agents' generalization in N-object environments with $N_o \neq 2$ reveals a clear separation between the relation network modules and the attention-based architecture. Agents with the ATTN module cannot generalize to a changing number of objects and experience a strong and consistent fading ability to solve the task the more the number of objects deviates from the training environment. ATTN-based agents fail to generalize under our test tasks

[2]https://sites.google.com/view/compositional-rrl/

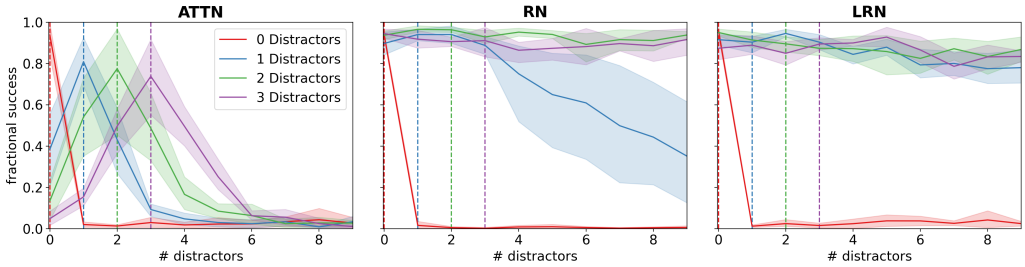


Figure 5: **Zero-shot generalization across different numbers of training distractors.** All three modules (left: ATTN; middle: RN; right: LRN) were retrained under 0, 1, 2, and 3 distractors with 5 seeds each. The ATTN baseline always peaks in performance for its train domain, quickly vanishing generalization to any new number of cubes. Crucially, our RN module can generalize to any number of objects in the scene if trained on at least 2 distractors (3 cubes in total). The LRN module achieves this with as little as 1 distractor at train time.

completely when adding more than 2 additional objects. Next, a naive relation network module (RN) achieves significantly improved performances in generalization over the ATTN baseline. Importantly, RN still suffers from fading generalization under more distractors, especially in the more complex task of LiftingCube. In contrast, our LRN module can maintain strong train-time performance for any tested number of objects. As we observe no relevant deteriorating trend, we suspect that this generalization capability could possibly even hold beyond the 9-distractor environments tested here. Interestingly, the same trends can be seen for environments with only 1 cube and no further distractor. Even though it seems intuitive that an agent should be able to solve pushing or lifting an object when no further distractors are in the scene, the ATTN module seems to overfit on the specific 1-distractor input train distribution strongly and cannot solve these tasks without exactly one distractor. On the other hand, RN and LRN module-based agents can robustly maintain their respective train-time performances, presumably due to their relation-based processing that better facilitates invariance w.r.t. the input objects.

Summary. Agents based on the relation network modules show a strong extrapolation ability of their learned manipulation skills to environments with many more objects. In contrast, we do not observe such zero-shot generalization with commonly employed attention-based architectures, which experience a substantial decrease in performance when deployed out-of-distribution.

5.4 Role of changing the train domains.

Previous generalization results obtained by the LRN were unexpected given that they were obtained training with 1 distractor only. In particular, we believed that at least a 2-distractor environment would be necessary since it also includes informative edge cases that the agent has to deal with to solve tasks with a large number of distractors. An example is when the selected cube is stuck between 2 distractor cubes blocking its grasping. We found this to be a frequent scenario when the arena gets crowded. To verify if the training environment has an effect on generalization performances, we repeated our experiments in environments with up to 3 distractors and additionally covered the limiting case of no distractors at train time at all. Our results are presented in Figure 5 which shows how the zero-shot generalization under each architecture is affected by its training distribution. Concerning the ATTN module, we observe the effect of the train distribution on generalization capabilities to be generally limited. Performances always peak in the training environment and monotonically decrease for $N_o^{test} \neq N_o^{train}$. On the other hand, a steady improvement can be observed for the RN module when changing the training environment up until 2 distractors. Note that the RN already achieves some level of generalization when training with a single distractor. It appears a phase change to robust performances happens when 2 distractors are used for training which yields a strong generalization for any test environment. Finally, we can observe that our LRN module - while already achieving a strong generalization when trained with 1 distractor only - can maintain the same level of performance when trained with 2 and 3 distractors. We, therefore, conclude that richer training distributions allow the RN to achieve generalization, whereas it does not actually change the overall behavior of the ATTN module. However, it is important to recall that, even if the RN achieves good generalization when trained on 2 distractors, the required number

of training steps improves significantly with respect to the 1 distractor environment as highlighted in Figure 2 (left plot). **Summary.** The training domain significantly affects generalization performances. Our LRN module appears to be the most robust architecture to changes in the training domain as it is the only architecture that already achieves strong generalization when trained with a single distractor.

5.5 Capturing higher-order relations

The Linear Relation Network (LRN) does not explicitly compute most of the possible relations, allowing it to scales only linearly with the number of objects. A question that naturally arises is what happens when a particular relation that the LRN does not compute is strictly required to solve the task. In fact, an example of this situation is even present in the pushing task: Here, the agent must reason about where the fingers are with respect to the cube to touch it and then grasp it. However, since the LRN only computes relations to the goal, the fingers-cube relation is not explicitly available to the agent. The only way to manipulate the cube using available relations is to combine the cube-goal and fingers-goal relations using the goal as a reference. This capability means the agent can extrapolate and use higher-order relations. The LRN successfully solves the pushing task, which therefore shows its ability to infer essential relations for the downstream task. More precisely, the higher-order reasoning is handled by the subsequent layers of the architecture after computing the relations, which is the MLP policy head in our case.

6 Limitations

Choice of environments. With a limited compute budget in mind and significant engineering effort necessary in robotics applications, we decided to specifically focus only on the TriFinger platform because successfully learning multi-object policies on this platform is still a long-standing open challenge given its highly flexible mechanical design which implicates a large and factored action space at the same time. Importantly, the platform comprises multiple manipulation challenges like learning coordinated dexterity such that it is more challenging than many other robotic environments where gripping is defined as a 1d space representing the finger distance Brockman et al. (2016).

Task formulation. The here proposed architecture is designed to handle single-goal environments. While being an important next step towards the long-term goal of multi-goal manipulation policies, we focused on suitable architecture that can deal with a variable number of input objects. As discussed in Section 2 our tasks mirror generic subtasks of general manipulation tasks such that future work can build upon such robust backbones and test how different (hierarchical) planning algorithms can master multi-goal tasks Yang et al. (2021); Kroemer et al. (2015).

RL algorithm. In our experiments, we decided to use PPO Schulman et al. (2017) due to it being a popular choice in RL and favourable robustness to different hyper-parameter choices. Extensive results on pushing and lifting from ground-truth features on the same setup with a single cube in Ahmed et al. (2020) indicate that methods like TD3 Fujimoto et al. (2018) or SAC Haarnoja et al. (2018) perform very similarly to PPO under the same reward structure and observation space.

7 Conclusion

In this work, we presented a new object encoder module based on relation networks, enabling reinforcement learning agents to learn challenging compositional manipulation tasks in multi-object robotics environments. We extensively studied a newly proposed linear relation network module on the generic task of transporting one particular cube out of N cubes to any specified goal position. While previous approaches mostly build upon quadratically scaling attention-based architectures to process the set of input objects, our approach scales only linear with the number of objects. At the same time, our method outperforms these previous approaches at train time. Most importantly, we can show that our agents can extrapolate and generalize zero-shot without any drop in performance when the number of objects changes which is a major limitation of most previous approaches. As next steps we also envision using the proposed module for solving similar tasks directly from pixels, i.e. combining our architecture with learned object perception modules. Locatello et al. (2020); Greff et al. (2020a); Kipf et al. (2021).

8 REPRODUCIBILITY STATEMENT

The main results of this paper come from the implementation of 3 object reasoning modules: LRN, RN, ATTN. Each module is extensively described respectively in section 4.2, section 9.1.5, section 9.1.4. These modules, as described in section 4, are sub-components of an agent architecture which is detailed in section 4.1. These agent architectures are trained optimizing for the reward structure highlighted in section 9.1.1, which is unique across the 2 tasks (PushingCube, and LiftingCube) studied in this work. The robotic manipulation tasks we work on are easily reproducible using the CausalWorld environment Ahmed et al. (2020), which provides out-of-the-box support for multi-object environments. The details for the RL algorithm used in all experiments are described in section 9.1.2.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning, 2020.

Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. *arXiv preprint arXiv:2108.09779*, 2021.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Stefan Bauer, Felix Widmaier, Manuel Wüthrich, Niklas Funk, Julen Urain De Jesus, Jan Peters, Joe Watson, Claire Chen, Krishnan Srinivasan, Junwu Zhang, et al. A robot cluster for reproducible research in dexterous manipulation. *arXiv preprint arXiv:2109.10957*, 2021.

Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12: 149–198, 2000.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems VII*, volume 7, pp. 57–64, 2011.

Andrea Dittadi, Frederik Träuble, Manuel Wüthrich, Felix Widmaier, Peter Gehler, Ole Winther, Francesco Locatello, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. The role of pretrained representations for the ood generalization of rl agents. *arXiv preprint arXiv:2107.05686*, 2021.

Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Sašo Džeroski, Luc De Raedt, and Hendrik Blockeel. Relational reinforcement learning. In *International Conference on Inductive Logic Programming*, pp. 11–22. Springer, 1998.

Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine Learning*, 43(1):7–52, 2001. doi: 10.1023/A:1007694015589. URL https://doi.org/10.1023/A:1007694015589.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Niklas Funk, Charles Schaff, Rishabh Madan, Takuma Yoneda, Julen Urain De Jesus, Joe Watson, Ethan K Gordon, Felix Widmaier, Stefan Bauer, Siddhartha S Srinivasa, et al. Benchmarking structured policies and policy optimization for real-world dexterous object manipulation. *arXiv preprint arXiv:2105.02087*, 2021.

Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference, 2020a.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020b.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Matteo Hessel, Hado van Hasselt, Joseph Modayil, and David Silver. On inductive biases in deep reinforcement learning. *arXiv preprint arXiv:1907.02908*, 2019.

Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection, 2018.

Michael Janner, Sergey Levine, William T Freeman, Joshua B Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-oriented prediction and planning. *arXiv preprint arXiv:1812.10972*, 2018.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Leslie Pack Kaelbling, Tim Oates, Natalia Hernandez, and Sarah Finney. Learning in worlds with objects. In *Working Notes of the AAAI Stanford Spring Symposium on Learning Grounded Representations*, pp. 31–36, 2001.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1503–1510, 2015. doi: 10.1109/ICRA.2015.7139389.

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

Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning, 2019.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020.

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

David Raposo, Adam Santoro, David Barrett, Razvan Pascanu, Timothy Lillicrap, and Peter Battaglia. Discovering objects and their relations from entangled scene representations. *arXiv preprint arXiv:1702.05068*, 2017.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Elizabeth S Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

Martijn Van Otterlo. Relational representations in reinforcement learning: Review and open problems. In *Proceedings of the ICML*, volume 2, 2002.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Matthew Wilson and Tucker Hermans. Learning to manipulate object collections using grounded state representations, 2020.

Manuel Wüthrich, Felix Widmaier, Felix Grimminger, Joel Akpo, Shruti Joshi, Vaibhav Agrawal, Bilal Hammoud, Majid Khadiv, Miroslav Bogdanovic, Vincent Berenz, et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.

Xintong Yang, Ze Ji, Jing Wu, Yu-Kun Lai, Changyun Wei, Guoliang Liu, and Rossitza Setchi. Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations, 2020.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Relational deep reinforcement learning, 2018.

Allan Zhou, Vikash Kumar, Chelsea Finn, and Aravind Rajeswaran. Policy architectures for compositional generalization in control. 2022.

	type	size	position	orientation	linear velocity	angular velocity	identifier	end-effector position
Cube	1	3	3	4	3	3	1	-
Goal	1	3	3	4	-	-	1	-
Finger	-	-	3	-	3	-	1	3

Table 1: Number or variables used to describe each element of the state for each object type

9 APPENDIX

9.1 IMPLEMENTATION DETAILS

9.1.1 REWARD DESIGN

We used the following same reward for both tasks:

$$\begin{aligned} R_t = \alpha_1 \rho_t & \\ & + \alpha_2 \left[d(o_t, g_t) - d(o_{t-1}, g_{t-1}) \right] \\ & + \alpha_3 (1 - \rho_t)^\beta \log \left[d(o_t, o_{t-1}) + 1e-5 \right] \end{aligned} \tag{2}$$

where t denotes the time step, $\rho_t \in [0, 1]$ is the fractional overlap with the goal cube at time t, $e_t \in \mathbf{R}^3$ is the end-effector position, $o_t \in \mathbf{R}^3$ the cube position of the object that is supposed to be used for the task, $g_t \in \mathbf{R}^3$ the goal position, and $d(\cdot, \cdot)$ denotes the Euclidean distance. Across all experiments we set $\boldsymbol{\alpha} = [100, 250, 10]$ and $\beta = 0.05$.

9.1.2 TRAINING SCHEME FOR AGENTS USING PPO

We use the distributed PPO implementation from `rlpyt` (Stooke & Abbeel, 2019) with discount factor 0.98, entropy loss coefficient 0.01, learning rate 0.00025, value loss coefficient 0.5, gradient clipping norm 0.5, 40 minibatches, gae lambda 0.95, clip ratio 0.5 and 4 epochs. We applied a linear learning rate scheduling and Adam as optimizer Kingma & Ba (2014).

9.1.3 SIMULATOR INPUT FEATURES

The ground truth representations for each object type are described in Table 1. Cubes are described using their position and orientation and each corresponding velocity. In addition, every cube is marked with a unique identifier which makes the set of cubes heterogeneous. Similarly, the goal is specified via its cartesian position, orientation, and an identifier variable used to specify which cube has to be moved. Both cubes and the goal include features describing their shape and size; however, we always use a cube shape and fixed size in this work. Each finger of the robot has 3 degrees of freedom, and it is described by a 9-dimensional vector composed by its joint positions, joint velocities, and end-effector position.

9.1.4 ATTENTION MODULE - BASELINE

The baseline we use in this work exploits a MHDPA module (Vaswani et al., 2017) operating on the set $\mathbf{o} \in \mathbb{R}^{d \times (K+1)}$ composed of common-size embedding of the K input objects $\mathbf{o}_i \in \mathbb{R}^d$, and the goal $\mathbf{o}_g \in \mathbb{R}^d$ with

$$\mathbf{o}^{attn} = \text{softmax}\left(\frac{q(\mathbf{o}) k(\mathbf{o})^T}{\sqrt{d}} \right) v(\mathbf{o}) \tag{3}$$

where $\mathbf{o}^{attn}$ is the set of $K + 1$ embeddings scaled from the MHDPA module. In this module, the goal-conditioning element is obtained by dealing with the goal as an object. This set is then mapped to the fixed-size embedding by an aggregation function and normalized to avoid changes in scale when the number of objects differs between training to testing.

$$\mathbf{z} = \text{LayerNorm}\left(\mathbf{f}_\phi \left(\sum_i^{K+1} \mathbf{o}_i^{attn} \right) \right) \tag{4}$$

This reasoning module resembles the one from Zambaldi et al. (2018), and it is used to introduce relational inductive biases into the learning process. The same self-attention layer is also used by

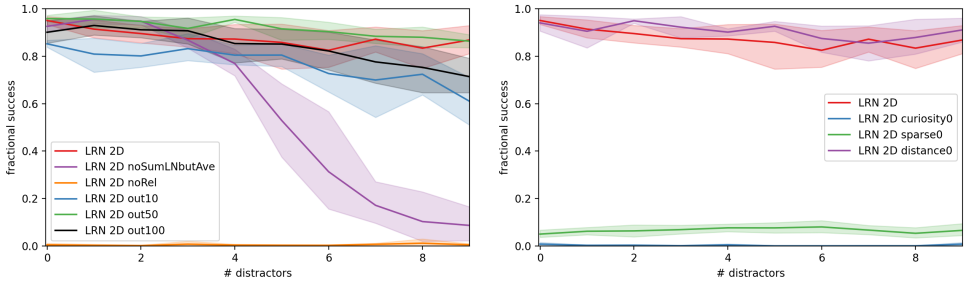


Figure 6: Zero-shot generalization of the linear relation network module on the PushingCube task concerning various architecture choices and the proposed reward structure. Left: We find that we cannot solve the task when dropping the object-goal relations (purple curve). We also observe that even though our agents can learn the task using an average instead of the LayerNorm, our model experiences a fading extrapolation performance (orange curve). Finally, we find a much weaker fading in generalization ability when providing too few or too many representational dimensions (green and red curves). Right: When ablating on the individual terms of our reward function, we first find that removing the distance term does not affect the overall performance (red curve). Whereas the sparse term only combined with the curiosity term is sufficient to solve the task successfully.

Zadaianchuk et al. (2020) to build a goal-conditioned policy that is able to handle set-based input representations. Similarly, it also describes approaches that rely on Graph Attention Networks (GAT) (Veličković et al., 2017) to build policies while using the common assumption in multi-object manipulation of a fully connected graph (Li et al., 2019; Wilson & Hermans, 2020). Self-attention when computing the attention matrix takes into consideration each object-object pair, which makes it scale quadratically with the number of objects in the input set, as highlighted in Figure 3 (A).

9.1.5 Relation Network Module

To have a further element of comparison we introduced Relation Networks as relational reasoning module in the context of reinforcement learning.

$$\mathbf{z} = \text{LayerNorm}\left(\mathbf{f}_\phi\left(\sum_{i}^{K+1}\sum_{j}^{K+1}\mathbf{g}_\theta(\mathbf{o}_i, \mathbf{o}_j)\right)\right) \tag{5}$$

Relation Networks have been proposed in Santoro et al. (2017) and they compute $(K+1)^2$ embeddings $\mathbf{r}_i \in \mathbb{R}^d$, referred to as relations, using the shared function $\mathbf{g}_\theta$. Relations are aggregated using a sum and then normalized to obtain a fixed-size embedding used as input for the MLP policy head. Similarly to self-attention, Relation Networks need to consider every object-object interaction, making them scale quadratically in the number of objects. A key aspect of this architecture is that $\mathbf{g}_\theta$ always takes as input a pair of objects no matter the total number of objects in the input set. This property makes the architecture robust to changes in the number of objects as long as the objects' statistics do not change.

9.2 Additional Results

9.2.1 Assessing architectural choices of our LRN module.

We also tested the impact of various design choices for our proposed linear relation network module. Results are summarized in Figure 6 (left). Specifically, we focus on 3 questions: First, do we need relations at all? To answer, we repeated the LRN experiments but replaced the relations with equivalent non-relational single-object feedforward MLPs, i.e. $\mathbf{g}_\theta(\mathbf{o}_i)$. We found that the module cannot even learn the task on the training domain without relations. Second, we want to understand if we could use a simple average instead of the LayerNorm after summing over all relations. Our experiments show that even though we can learn the task in the train environments, the agents reveal deteriorating generalization the more objects we add to the task. We hypothesize this to be due to the changing variance of $\mathbf{z}$ which is kept constant when doing a layer normalization but not an average.

Lastly, we analyzed the role of the representations' dimension, for which we used $d_z = 25$. When repeating the experiments with $d_z = 10$, the zero-shot generalization slowly decreases to about 60 % success with 9 objects. On the other hand, when allowing for $d_z = 50$ or even $d_z = 100$, the agent achieves slightly elevated train time performance with a slight drop of about 5 % and a more pronounced drop of about 20 % respectively when testing with up to 9 distractors. This result suggests that the representation dimension is optimally being adjusted for the right capacity requirements of the task.

9.2.2 Role of the reward function.

We finally wanted to understand the impact of each term on the reward function we proposed. Figure 6 (right) shows the generalization performance of corresponding agents trained with each variation of the reward function. We can distinguish reward terms based on their effect: sparse, and distance terms can be used to define the task; the curiosity term guides the exploration. The first notable observation is that removing the distance term does not affect the overall performance. Therefore, we can conclude that the sparse term is sufficient to solve the task successfully. On the other hand, removing the sparse term while keeping the distance term stops the agent from learning to solve the task, underlying its importance.

9.2.3 Distribution Shift for OOD tasks

To better understand the difference between the architectures, we visualized the representations of each Reasoning module with the help of lower-dimensional embeddings. Each model has been tested for 200 episodes on environments with the number of distractors ranging from 0 up to 9. The representations obtained as output have been mapped to a low-dimensional space using t-SNE (van der Maaten & Hinton, 2008). These low-dimensional points let us compare the support of the embedding distribution between different test domains. It is important to observe that these points represent the input of the MLP policy head; therefore, to have generalization capabilities, they should have similar statistics across different tasks. In Figure 7(a) we can observe the result obtained for the attention-based module when visualized in a 2-dimensional space. The embedding distribution changes with the task, which causes the MLP policy head to operate out-of-distribution justifying the drop in performances observed in Figure 4. In Figure 7(b) and Figure 7(c), respectively the RN and LRN output embeddings are visualized. For both, we can say that the approximated support of the distribution appears to be invariant with respect to the task.

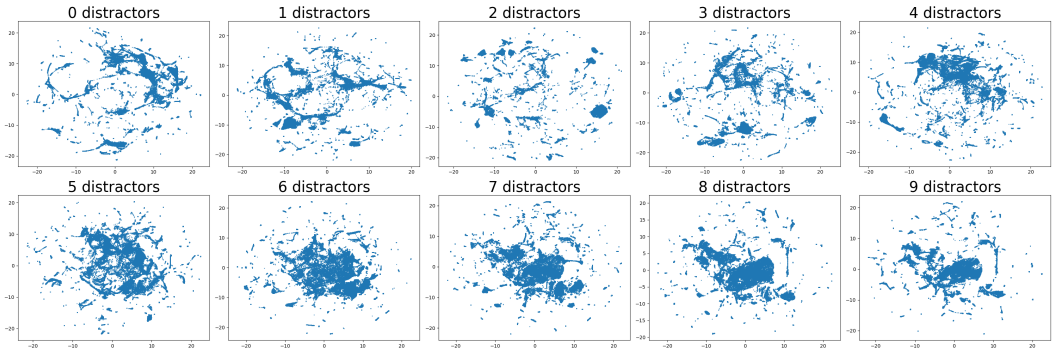
9.2.4 Relations analysis

Relations Sub-Spaces We decided to visualize 2-dimensional embeddings computed using t-SNE color-coding different relations to understand whether relations with different objects are qualitatively different. We used relations generated from 200 episodes with a number of distractors varying from 3 to 6. The obtained plot is presented in Figure 8, where cube-goal relations are shown in green, distractors-goal relations are visualized in blue, and fingers relations are colored respectively in black, brown, and red. Different relations are mapped to different sub-spaces of the representation space, suggesting that invariance to the number of distractors might be obtained via a gating property learned by the architecture to filter out relations embedded into certain sub-spaces when computing the action.

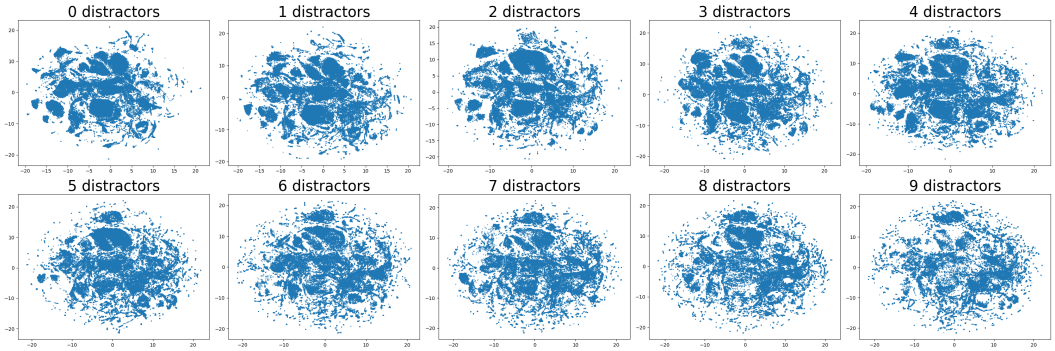
Relations over time We also analyzed how the norm of each relation type changes over time while the agent is solving the task. In Figure 9 we show the result when testing on 5 distractors a LRN-based agent and normalizing each type of relation by the number of relations belonging to that type. Since we used the sum as the aggregation function, the norm can be associated with each relation's weight in the representation after the sum. Interestingly, the most important relations are those describing fingers and the target cube, whereas those linked to distractors have a much smaller norm. This qualitative result suggests that the cube-goal relation might be correlated with the cube-goal distance.

9.2.5 Limitations of the Attention-Based Network

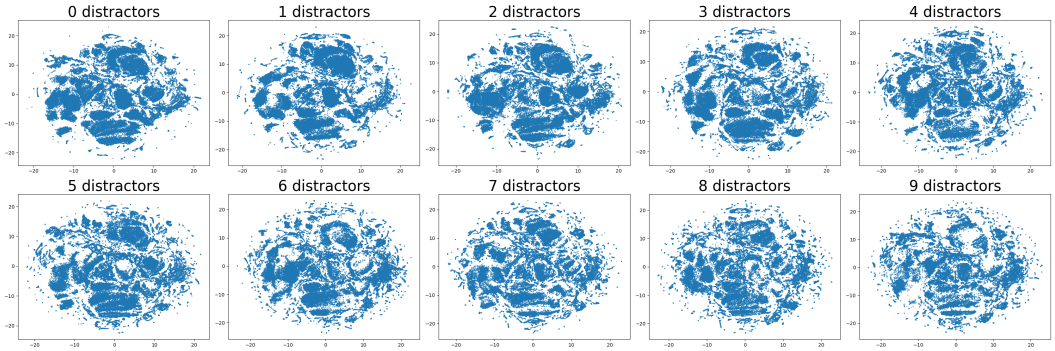
Given the observed poor generalization results and the very high variance during training for the attention-based baseline, we performed extensive hyperparameter tuning to ensure a fair comparison against our proposed RN and LRN modules. We used Bayesian search (Biewald, 2020) across more



(a) Distribution Shift Attention Network (training distribution: 2 distractors)



(b) Distribution Shift Relation Network (training distribution: 2 distractors)



(c) Distribution Shift Linear Relation Network (training distribution: 2 distractors)

Figure 7: OOD Distribution shifts

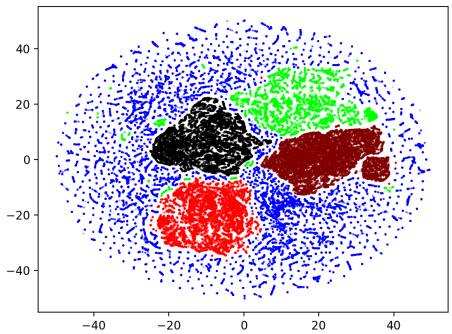


Figure 8: Low-dimensional t-SNE embedding of the relations generated over 200 episodes from the Linear Relation Network. Different relations are color-coded (Black: Finger0-Goal. Brown: Finger1-Goal. Red: Finger2-Goal. Green: Cube-Goal. Blue: Distractors-Goal). The architecture learns to map different relations in different sub-spaces of the representation space.

than 550 hyperparameter configurations. We found that none of those improves over our set of hyperparameters meaningfully.

To find potential bottlenecks in both the RL algorithm (PPO) and the attention modules, we used the hyperparameters search algorithm from Biewald (2020) to check elements of both. Regarding PPO, we tunned:

- Gradient clipping norm
- Discount factor
- Entropy loss coefficient
- Value loss coefficient
- GAE lambda
- Learning rate
- Ratio clip

For the architecture of the module, we explored different configurations of:

- Embedding dimension
- Number of heads
- $\mathbf{f}_\phi$ output dimension

Due to the long training time, we evaluated each model after 50 million timesteps. This training length should be sufficiently long for the purpose of observing notable differences between models' fractional success and reward.

Additionally, we tried to add a skip connection to the self-attention layer following what is done by Vaswani et al. (2017). This change reduced the variance across seeds but did not affect the overall result in terms of generalization.

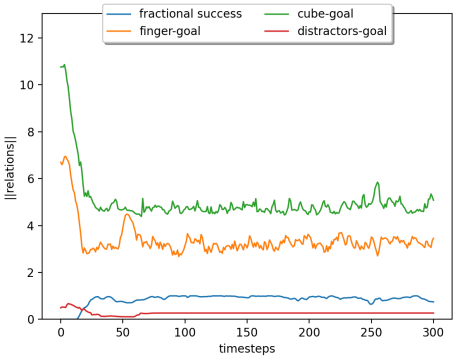


Figure 9: Relations norm changing throughout the task in LRN when testing on 4 distractors.