SELF-SUPERVISED VISUAL REINFORCEMENT LEARNING WITH OBJECT-CENTRIC REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous agents need large repertoires of skills to act reasonably on new tasks that they have not seen before. However, acquiring these skills using only a stream of high-dimensional, unstructured, and unlabeled observations is a tricky challenge for any autonomous agent. Previous methods have used variational autoencoders to encode a scene into a low-dimensional vector that can be used as a goal for an agent to discover new skills. Nevertheless, in compositional/multi-object environments it is difficult to disentangle all the factors of variation into such a fixed-length representation of the whole scene. We propose to use *object-centric representations* as a modular and structured observation space, which is learned with a compositional generative world model. We show that the structure in the representations in combination with *goal-conditioned attention policies* helps the autonomous agent to discover and learn useful skills. And these skills can be further combined to solve complex compositional tasks like the manipulation of several different objects.

https://sites.google.com/view/smorl-iclr2021

1 INTRODUCTION

Reinforcement learning (RL) includes a promising class of algorithms that have shown capability to solve challenging tasks when those tasks are well specified by suitable reward functions. However, in the real world, people are rarely given a well-defined reward function. Indeed, humans are excellent at setting their own abstract goals and achieving them. Agents that exist persistently in the world should likewise prepare themselves to solve diverse tasks by first constructing plausible goal spaces, setting their own goals within these spaces, and then trying to achieve them. In such a way, they can learn about the world around them.

In principle, the goal space for an autonomous agent could be any arbitrary function of the state space. However, when the state space is high-dimensional and unstructured, such as only images, it is desirable to have goal spaces which allow efficient exploration and learning, where the factors of variation in the environment are well disentangled. Recently, unsupervised representation learning (Nair et al., 2018; 2019; Pong et al., 2020) has been proposed to learn such goal spaces. All existing methods based on this use variational autoencoders (VAEs) to map observations into a low-dimensional latent space that can later be used for sampling goals and reward shaping.

However, for complex compositional scenes consisting of multiple objects, the inductive bias of VAEs could be harmful. In contrast, representing perceptual observations in terms of entities has been shown to improve data efficiency and transfer performance on a wide range of tasks (Burgess et al., 2019). Recent research has proposed a range of methods for unsupervised scene and video decomposition (Kosiorek et al., 2018; Burgess et al., 2019; Greff et al., 2019; Weis et al., 2020; Locatello et al., 2020; Jiang et al., 2019). These methods learn object representations and scene decomposition jointly. Majority of them are in part motivated by the fact that the learned representations could be useful for downstream tasks such as image classification, object detection, or semantic segmentation. In this work, we show that such learned representations are also beneficial for autonomous control and reinforcement learning.

We propose to combine these *object-centric unsupervised representation* methods that represent the scene as a set of potentially structured vectors with goal-conditional visual RL. In our method

(illustrated in Figure 1), dubbed SMORL (for self-supervised multi-object RL), a representation of raw sensory inputs is learned by a compositional latent variable model based on the SCALOR architecture (Jiang et al., 2019). We show that using object-centric representations simplifies the goal space learning. Autonomous agents can use those representations to learn how to achieve different goals with a reward function that utilizes the structure of the learned goal space. Our main contributions are as follows:

- We show that structured object-centric representations learned with generative world models can significantly improve the performance of the self-supervised visual RL agent.
- We develop SMORL, an algorithm that uses learned representations to autonomously discover and learn useful skills in compositional environments with several objects using only images as inputs.
- We show that even with fully disentangled ground-truth representation there is a large benefit from using SMORL in environments with complex compositional tasks such as rearranging many objects.

We validate our proposed method in several multi-object visual environments for robotic manipulation with varying difficulty of object-related tasks.

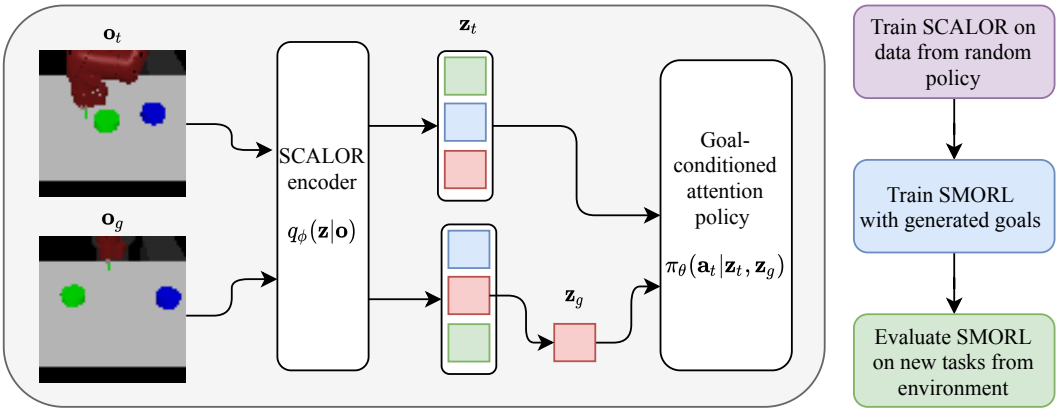


Figure 1: Our proposed SMORL architecture. Representations $\mathbf{z}_t$ are obtained from observations $\mathbf{o}_t$ through the object-centric SCALOR encoder q_ϕ, and processed by the goal-conditional attention policy $\pi_\theta(\mathbf{a}_t|\mathbf{z}_t, \mathbf{z}_g)$. During training, representations of goals are sampled conditionally on the representations of the first observation $\mathbf{z}_1$. At test time, the agent is provided with an external goal image $\mathbf{o}_g$ that is processed with the same SCALOR encoder to a set of potential goals $\{\mathbf{z}_n\}_{n=1}^N$. After this, the goal $\mathbf{z}_g$ is sequentially chosen from this set. This way, the agent attempts to solve all the discovered sub-tasks one-by-one, not simultaneously.

2 Background

Our method combines goal-conditional RL with unsupervised object-oriented representation learning for multi-object environments. Before we describe each technique in detail, we briefly state some RL preliminaries. We consider a Markov decision process defined by $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ and $\mathcal{A}$ are the continuous state and action spaces, $p\colon \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, \infty)$ is an unknown probability density representing the probability of transitioning to state $\mathbf{s}_{t+1} \in \mathcal{S}$ from state $\mathbf{s}_t \in \mathcal{S}$ given action $\mathbf{a}_t \in \mathcal{A}$, and $r\colon \mathcal{S} \mapsto \mathbb{R}$ is a function computing the reward for reaching state $\mathbf{s}_{t+1}$. The agent's objective is to maximize the expected return $R = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{s}_t \sim \rho_\pi, \mathbf{a}_t \sim \pi, \mathbf{s}_{t+1} \sim p} \left[r(\mathbf{s}_{t+1}) \right]$ over the horizon T, where $\rho_\pi(\mathbf{s}_t)$ is the state marginal distribution induced by the agent's policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

2.1 Goal-conditional Reinforcement Learning

In the standard RL setting described before, the agent only learns to solve a single task, specified by the reward function. If we are interested in an agent that can solve multiple tasks (each with a

different reward function) in an environment, we can train the agent on those tasks by telling the agent which distinct task to solve at each time step. But how can we describe a task to the agent? A simple, yet not too restrictive way is to let each task correspond to an environment state the agent has to reach, denoted as the goal state g. The task is then given to the agent by conditioning its policy $\pi(a_t \mid s_t, g)$ on the goal g, and the agent's objective turns to maximize the expected goal-conditional return:

$$\mathbb{E}_{\mathbf{g}\sim\mathbf{G}}\left[\sum_{t=1}^{T}\mathbb{E}_{\mathbf{s}_t\sim\rho_\pi,\mathbf{a}_t\sim\pi,\mathbf{s}_{t+1}\sim p}\left[r_g(\mathbf{s}_{t+1})\right]\right] \tag{1}$$

where G is some distribution over the space of goals $\mathcal{G} \subseteq \mathcal{S}$ the agent receives for training. The reward function can, for example, be the negative distance of the current state to the goal: $r_{\mathbf{g}}(\mathbf{s}) = -\|\mathbf{s}-\mathbf{g}\|$. Often, we are only interested in reaching a partial state configuration, e.g. moving an object to a target position, and want to avoid using the full environment state as the goal. In this case, we have to provide a mapping $m\colon \mathcal{S} \mapsto \mathcal{G}$ of states to the desired goal space; the mapping is then used to compute the reward function, i.e. $r_{\mathbf{g}}(\mathbf{s}) = -\|m(\mathbf{s}) - \mathbf{g}\|$.

As the reward is computed within the goal space, it is clear that the choice of goal space plays a crucial role in determining the difficulty of the learning task. If the goal space is low-dimensional and structured, e.g. in terms of ground truth positions of objects, rewards provide a meaningful signal towards reaching goals. However, if we only have access to high-dimensional, unstructured observations, e.g. camera images, and we naively choose this space as the goal space, optimization becomes hard as there is little correspondence between the reward and the distance of the underlying world states (Nair et al., 2018).

One option to deal with such difficult observation spaces is to *learn a goal space* in which the RL task becomes easier. For instance, we can try to find a low-dimensional latent space $\mathcal{Z}$ and use it both as the input space to our policy and the space in which we specify goals. If the environment is composed of independent parts that we intend to control separately, intuitively, learning to control is easiest if the latent space is also structured in terms of those independent components. Previous research (Nair et al., 2018; Pong et al., 2020) relied on the disentangling properties of representation learning models such as the β-VAE (Higgins et al., 2017) for this purpose. However, as we will show, these models quickly become insufficient when faced with multi-object scenarios due to the increasing combinatorial complexity of the scene. Instead, we use a model explicitly geared towards inferring object-structured representations, which we introduce in the next section.

2.2 Structured representation learning with SCALOR

SCALOR (Jiang et al., 2019) is a probabilistic generative world model for learning object-oriented representations of a video or stream of high-dimensional environment observations. SCALOR assumes that the environment observation $\mathbf{o}_t$ at step t is generated by the background latent variable $\mathbf{z}_t^{\text{bg}}$ and the foreground latent variable $\mathbf{z}_t^{\text{fg}}$. The foreground is further factorized into a set of object representations $\mathbf{z}_t^{\text{fg}} = \{\mathbf{z}_{t,n}\}_{n\in\mathcal{O}_t}$, where $\mathcal{O}_t$ is the set of recognised object indices. To combine the information from previous time steps, a propagation-discovery model is used (Kosiorek et al., 2018). In SCALOR, an object is represented by $\mathbf{z}_{t,n} = (z_{t,n}^{\text{pres}}, \mathbf{z}_{t,n}^{\text{where}}, \mathbf{z}_{t,n}^{\text{what}})$. The scalar $z_{t,n}^{\text{pres}}$ defines if the object is present in the scene, whereas the vector $\mathbf{z}_{t,n}^{\text{what}}$ encodes object appearance. The component $\mathbf{z}_{t,n}^{\text{where}}$ is further decomposed into the object's center position $\mathbf{z}_{t,n}^{\text{pos}}$, scale $\mathbf{z}_{t,n}^{\text{scale}}$, and depth $z_{t,n}^{\text{depth}}$. With this, the generative process of SCALOR can be written as:

$$p(\mathbf{o}_{1:T}, \mathbf{z}_{1:T}) = p(\mathbf{z}_1^{\mathcal{D}})(\mathbf{z}_1^{\text{bg}})\prod_{t=2}^{T}\underbrace{p(\mathbf{o}_t|\mathbf{z}_t)}_{\text{rendering}}\,\underbrace{p(\mathbf{z}_t^{\text{bg}}|\mathbf{z}_{<t}^{\text{bg}}, \mathbf{z}_t^{\text{fg}})}_{\text{background transition}}\,\underbrace{p(\mathbf{z}_t^{\mathcal{D}}|\mathbf{z}_t^{\mathcal{P}})}_{\text{discovery}}\,\underbrace{p(\mathbf{z}_t^{\mathcal{P}}|\mathbf{z}_{<t})}_{\text{propagation}}, \tag{2}$$

where $\mathbf{z}_t = (\mathbf{z}_t^{\text{bg}}, \mathbf{z}_t^{\text{fg}})$, $\mathbf{z}_t^{\mathcal{D}}$ contains latent variables of objects discovered in the present step, and $\mathbf{z}_t^{\mathcal{P}}$ contains latent variables of objects propagated from the previous step. Due to the intractability of the true posterior distribution $p(\mathbf{z}_{1:T}|\mathbf{o}_{1:T})$, SCALOR is trained using variational inference with the

following posterior approximation:

$$q(\mathbf{z}_{1:T}|\mathbf{o}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{o}_{\leq t}) = \prod_{t=1}^{T} q(\mathbf{z}_t^{\text{bg}}|\mathbf{z}_t^{\text{fg}}, \mathbf{o}_t)\, q(\mathbf{z}_t^{\mathcal{D}}|\mathbf{z}_t^{\mathcal{P}}, \mathbf{o}_{\leq t})\, q(\mathbf{z}_t^{\mathcal{P}}|\mathbf{z}_{<t}, \mathbf{o}_{\leq t}), \quad (3)$$

by maximizing the following evidence lower bound $\mathcal{L}(\theta, \phi) =$

$$\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{<t}|\mathbf{o}_{<t})} \Big[\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{o}_{\leq t})} \big[\log p_\theta(\mathbf{o}_t|\mathbf{z}_t)\big] - D_{\text{KL}}\big[q_\phi(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{o}_{\leq t}) \parallel p_\theta(\mathbf{z}_t|\mathbf{z}_{<t})\big] \Big], \quad (4)$$

where D_{KL} denotes the Kullback-Leibler divergence. As we are using SCALOR in an active setting, we additionally condition the next step posterior predictions on the actions taken by the agent. For more details and hyperparameters used to train SCALOR, we refer to App. A.2.1. In the next section, we describe how the structured representations learned by SCALOR can be used in downstream RL tasks such as goal-conditional visual RL.

3 Self-Supervised Multi-Object Reinforcement Learning

Learning from flexible representations obtained from unsupervised scene decomposition methods such as SCALOR creates several challenges for RL agents. In particular, these representations consist of sets of vectors, whereas standard policy architectures assume fixed-length state vectors as input. We propose to use a *goal-conditioned attention policy* that can handle sets as inputs and flexibly learns to attend to those parts of the representation needed to achieve the goal at hand.

To discover useful skills that can be used during evaluation tasks, the agent can use the discovered structure in the representations, namely object position and appearance. Previous VAE-based methods use latent distances to the goal state as the reward signal. However, for compositional goals, this means that the agent needs to master the simultaneous manipulation of all objects. In our experiments, we show that even with a fully disentangled, ground-truth representation of the scene this is a challenging setting to learn for state-of-the-art model-free RL agents when the number of the objects are larger than 2 (see Sec. 4.1). Instead, we propose to use the available structure in the learned goal and state spaces for learning, and solve only sub-tasks that correspond to manipulating individual components during training.

3.1 Policy with Goal-conditioned Attention

We use the multi-head attention mechanism (Vaswani et al., 2017) as the first stage of our policy π_θ to deal with the challenge of the set-based input representation. As the policy needs to flexibly vary its behavior based on the goal at hand, it appears sensible to steer the attention using a goal-dependent query $Q(\mathbf{z}_g) = \mathbf{z}_g W^q$. Each object is allowed to match with the query via an object-dependent key $K(\mathbf{z}_t) = \mathbf{z}_t W^k$ and contribute to the attention's output through the value $V(\mathbf{z}_t) = \mathbf{z}_t W^v$, which is weighted by the similarity between $Q(\mathbf{z}_g)$ and $K(\mathbf{z}_t)$. As inputs, we concatenate the representations for object l to vectors $\mathbf{z}_{t,n} = [\mathbf{z}_{t,n}^{\text{what}}; \mathbf{z}_{t,n}^{\text{where}}; z_{t,n}^{\text{depth}}]$, and similarly the goal representation to $\mathbf{z}_g = [\mathbf{z}_g^{\text{what}}; \mathbf{z}_g^{\text{where}}; z_g^{\text{depth}}]$. The attention head A_k is computed as

$$A_k = \text{softmax}\left(\frac{\mathbf{z}_g W^q (Z_t W^k)^T}{\sqrt{d_e}}\right) Z_t W^v, \quad (5)$$

where Z_t is a packed matrix of all $\mathbf{z}_{t,n}$'s, W^q, W^k, W^v constitute learned linear transformations and d_e is the common key, value and query dimensionality. The final attention output A is a concatenation of all the attention heads $A = [A_1; ...; A_K]$. The second stage of our policy is a fully-connected neural network that takes as inputs A and the goal representation $\mathbf{z}_g$ and outputs an action a_t. The full policy π_θ can thus be described by

$$\pi_\theta\left(\{\mathbf{z}_{t,n}\}_{n \in \mathcal{O}_t}, \mathbf{z}_g\right) = f(A, \mathbf{z}_g). \quad (6)$$

In general, we expect that it is beneficial for the policy to not always attend to entities conditional on the goal; we thus allow some heads to only attend to additional learned parametric queries (left out above for notational clarity). As goal images are compositional, their representation is also a set of

goal vectors. During our experiments, we assume that these goals are independent of each other and that we can thus try to sequentially achieve them. For future work, we will consider more complex planning and reasoning policies such as described in (Nasiriany et al., 2019) as a potential way to improve the overall performance of the final policy.

3.2 SELF-SUPERVISED TRAINING

In principle, our policy can be trained with any goal-conditioned model-free RL algorithm. For our experiments, we picked soft-actor critic (SAC) (Haarnoja et al., 2018b) as a state-of-the-art method for continuous action spaces, using hindsight experience replay (HER) (Andrychowicz et al., 2017) as a standard way to improve sample-efficiency in the goal-conditional setting.

The full training algorithm is summarized in Alg. 1. We first train SCALOR on data collected from a random policy and fit a distribution $p(\mathbf{z}^{\text{where}})$ to representations $\mathbf{z}^{\text{where}}$ of collected data. Each rollout, we generate a new goal for the agent by picking a random $\mathbf{z}^{\text{what}}$ from the initial observation $\mathbf{z}_1$ and sampling a new $\mathbf{z}^{\text{where}}$ from the fitted distribution $p(\mathbf{z}^{\text{where}})$. The policy is then rolled out using this goal. During off-policy training, we are relabeling goals with HER, and, similar to RIG Nair et al. (2018), also with "imagined goals" produced in the same way as the rollout goals.

A challenge with compositional representations is how to measure the progress of the agent towards achieving the chosen goal. As the goal always correspond to a single object, we have to extract the state of this object in the current observation in order to compute a reward. One way is to rely on the tracking of the objects, as was shown possible e.g. by SCALOR Jiang et al. (2019). However, as the agent learns, we noticed that it would discover some flaws of the tracking and exploit them to get a maximal reward that is not connected with environment changes, but rather with internal vision and tracking flaws (details in App. A.3).

An alternative approach is to use the $\mathbf{z}^{\text{what}}$ component of discovered objects and match them with the current goal representation $\mathbf{z}_g^{\text{what}}$. As the $\mathbf{z}^{\text{what}}$ space encodes the appearance of objects, two objects corresponding to the same object are close in this space. Thus, it is easy to match the object corresponding to the current goal object using the distance $\min_k ||\mathbf{z}_k^{\text{what}} - \mathbf{z}_g^{\text{what}}||$. In case of failure to discover close representation (e.g. that all of the representation of the components have the distance larger than some threshold α to goal component representation), we suggest to use a fixed negative reward $r_{\text{no goal}}$.

Our reward signal is thus

$$r(\mathbf{z}_k, \mathbf{z}_g) = \begin{cases} -||\mathbf{z}_{\hat{k}}^{\text{where}} - \mathbf{z}_g^{\text{where}}||, & \text{if} \quad \min_k ||\mathbf{z}_k^{\text{what}} - \mathbf{z}_g^{\text{what}}|| < \alpha \\ r_{\text{no goal}}, & \text{otherwise,} \end{cases} \tag{7}$$

where $\hat{k} = \arg\min_k ||\mathbf{z}_k^{\text{what}} - \mathbf{z}_g^{\text{what}}||$.

Algorithm 1 Self-Supervised Multi-Object RL (SMORL) training

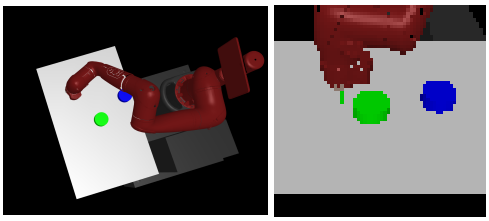
Require: SCALOR encoder q_ϕ, goal-conditioned policy π_θ, goal-conditioned SAC trainer, number of training episodes K.

Train SCALOR on sequences data uniformly sampled from $\mathcal{D}$ using loss described in Eq. 4.
Fit prior $p(\mathbf{z}^{\text{where}}|\mathbf{z}^{\text{what}})$ to the latent encodings of observations.
for $n = 1, ..., K$ episodes **do**
  Sample goal $\mathbf{z}_g = (\hat{\mathbf{z}}_g^{\text{where}}, \mathbf{z}_g^{\text{what}})$
  Collect episode data with policy $\pi_\theta(\mathbf{a}_t|q_\phi(\mathbf{o}_t), \mathbf{z}_g)$ and SCALOR representation of observations $q_\phi(\mathbf{z}_t|\mathbf{o}_t)$.
  Store transitions $(\mathbf{z}_t, \mathbf{a}_t, \mathbf{z}_{t+1}, \mathbf{z}_g)$ into replay buffer $\mathcal{R}$.
  Sample transitions from replay buffer $(\mathbf{z}, \mathbf{a}, \mathbf{z}', \mathbf{z}_g) \sim \mathcal{R}$
  Relabel $\mathbf{z}_g^{\text{where}}$ goal components to the combination of future states and $p(\mathbf{z}^{\text{where}}|\mathbf{z}^{\text{what}})$.
  Compute matching reward signal $r = r(\mathbf{z}', \mathbf{z}_g)$.
  Update policy $\pi_\theta(\mathbf{z}_t|q_\phi(\mathbf{o}_t), \mathbf{z}_g)$ with SAC trainer.
end for

4 EXPERIMENTS

We have done computational experiments to address the following questions:

- How well would our method scale challenging tasks with large number of objects in case when ground-truth representations are provided?
- How does our method perform compared to prior visual goal-conditioned RL methods on image-based, multi-object continuous control tasks?
- How suitable are the representations learned by the compositional generative world model for discovering and solving RL tasks?



(a) View from top (b) Agent observation

Figure 2: Multi-Object Visual Pusher and Rearrange environments with 2 objects and a Sawyer robotic arm.

To answer these questions, we constructed the *Multi-Object Visual Pusher* and *Multi-Object Visual Rearranger* environments. Both environments are based on MuJoCo (Todorov et al., 2012) and the Multiworld package for image-based continuous control tasks introduced by Nair et al. (2018), and contain a 7-dof Sawyer arm where the agent needs to be controlled to manipulate a variable number of small picks on a table. In the first environment, the objects are located on fixed positions in front of the robot arm that the arm must push to random target positions. In the second environment, the task is to rearrange the objects from random starting positions to random target positions. This task is more challenging for RL algorithms due to the randomness of initial object positions. For both environments, we measure the performance of the algorithms as the average distance of all pucks to their goal positions on the last step of the episode. Out code as well as multi-objects environments will be made public after the paper publication.

4.1 SMORL WITH GROUND-TRUTH (GT) STATE REPRESENTATION

We first compared SMORL with ground-truth representation with Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) with Hindsight Experience Replay (HER) relabeling (Andrychowicz et al., 2017) that takes an unstructured vector of all objects coordinates as input. We are using a one-hot encoding for objects identities $\mathbf{z}^{\text{what}}$ and object and arm coordinates as $\mathbf{z}^{\text{where}}$ components. With such a representation, the matching task becomes trivial, so our main focus was on the benefits of goal-conditioned attention policy and sequential solving of independent sub-tasks. While for 2 objects, SAC+HER is performing similarly, for 3 and 4 objects, SAC+HER fails to rearrange any of the objects. In contrast, SMORL equipped with ground-truth representation is still able to rearrange 3 and 4 objects, and it can solve the more simple sub-tasks of moving each object independently. This shows that provided with good representations, SMORL can use them for constructing useful sub-tasks and learning how to solve them.

4.2 VISUAL RL METHODS COMPARISON

We have compared the performance of our algorithm with two Self-Supervised Multi-task Visual RL algorithms on our two environments, with one and two objects. The first one, RIG (Nair et al., 2019), uses the VAE latent space to sample goals and to estimate the reward signal. The second one, Skew-Fit (Pong et al., 2020), also uses the VAE latent space, however, is additionally biased on rare observations that were not modeled well by the VAE from previously collected data. For more simple *Multi-Object Visual Pusher* environment, the performance of SMORL is comparable to the best performing baseline, while for the more challenging *Multi-Object Visual Rearranger* environment, SMORL is significantly better then both RIG and Skew-Fit. This shows that learning of object-oriented representations brings benefits for goal sampling and self-supervised learning of useful skills. However, out method is significantly worse that the SAC with ground-truth representations. One potential reason for this is that SMORL was not equipped with a reliable tracking from SCALOR. Because of this additional matching was needed to provide a meaningful feedback for RL agent.

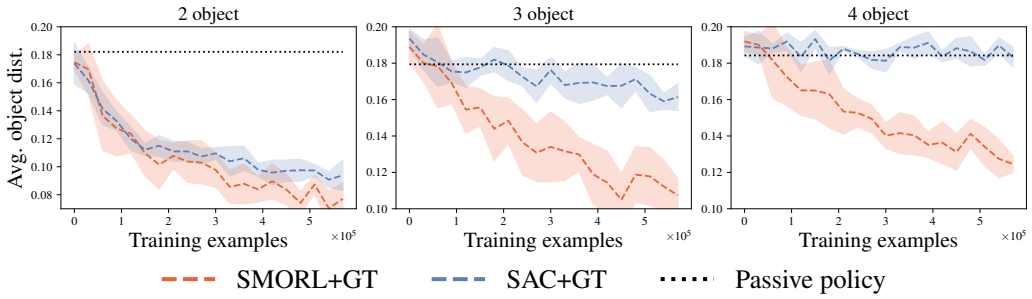


Figure 3: Average distance of objects to goal positions, comparing SMORL using ground truth representations to SAC with ground truth representations in the Rearrange environment with different number of objects. SAC struggles to improve performance when the combinatorial complexity of the scene rises. The dotted line indicates the performance of a *passive policy* that performs no movements. Results averaged over 5 random seeds, shaded region indicates one standard deviation.

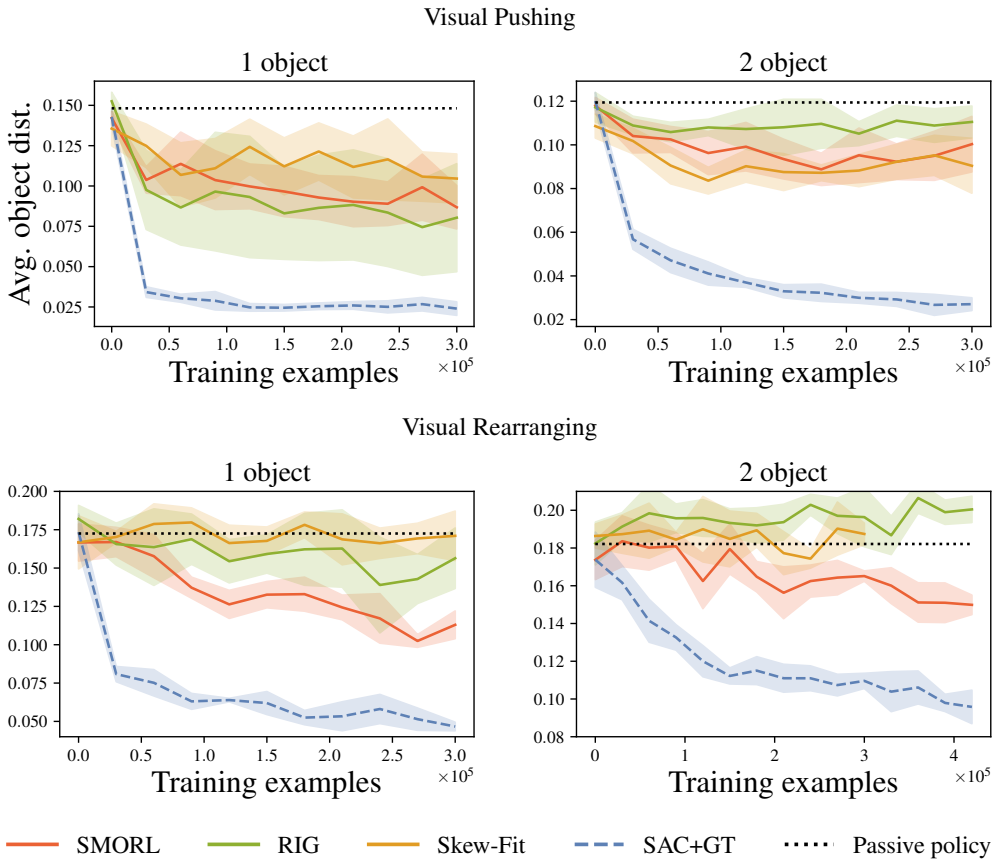


Figure 4: Average distance of objects to goal positions, comparing SMORL to Visual RL Baselines. In addition to the baselines, we show SAC performance with ground truth representations. Results averaged over 5 random seeds, shaded region indicates one standard deviation.

5 Related Work

There are several different lines of related work.

The first one (Nair et al., 2018; 2019; Pong et al., 2020; Ghosh et al., 2019; Warde-Farley et al., 2019) tackles visual self-supervised multi-tasks RL problems. However, they assume that the environment

observation can be encoded with single vector (like VAE representations). This assumption may cause the binding problem Greff et al. (2016). In addition, as the reward shaping is also based on this vector, the agent is incentived to solve tasks that are incompatible (like simultaneously moving all objects to the goal positions). In contrast, we learn object-centric representations and use them for reward shaping. Thus agent can learn to solve each task independently and then combine these skills during evaluation.

The second line of related work (Veerapaneni et al., 2020; Watters et al., 2019; Kipf et al., 2020) is learning similar object-oriented representations and using them to tackle RL tasks. However, these works assume a fixed task and given reward signal, whereas we are using only learned representation to discover potentially interesting goals and reward signal that helps to learn useful skills. In addition, these methods use scene-mixture models such as (Burgess et al., 2019; Greff et al., 2019), which do not contain disentangled and interpretable features like position and scale. These features can be used by agent for more efficient sampling from goal space.

Another line of research concerned with exploiting structure in the environment is concentrated on factored MDPs (Boutilier et al., 1995; Kearns & Koller, 1999; Osband & Roy, 2014). Methods working with factored MDPs also make use of compositionality of the state space, but on top of that require conditional independence within the transition and reward distributions that are not guaranteed in the robotics environments we target.

6 Conclusion

In this work, we have shown that discovering structure in the observations of the environment with compositional generative world models and using it for controlling different parts of the environment is crucial for solving tasks in compositional environments. The manipulation of different parts of learned object-centric representations is a powerful way to learn useful skills such as object manipulation. Our SMORL agent learns how to control different entities in the environment and can then combine learned skills to achieve more complex compositional goals such as rearranging several objects on a table using only the final image of the arrangement.

References

Marcin Andrychowicz, Dwight Crow, Alex Ray, J. Schneider, Rachel H Fong, P. Welinder, Bob McGrew, Josh Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *ArXiv*, abs/1707.01495, 2017.

Craig Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *IJCAI*, 1995.

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation, 2019.

D. Ghosh, A. Gupta, and S. Levine. Learning actionable representations with goal-conditioned policies. *ArXiv*, abs/1811.07819, 2019.

Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. Binding via reconstruction clustering, 2016.

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. *Proceedings of the 36nd International Conference on Machine Learning*, 2019.

T. Haarnoja, Aurick Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018b.

I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalable object-oriented sequential generative models. *arXiv preprint arXiv:1910.02384*, 2019.

Michael Kearns and Daphne Koller. Near-optimal reinforcement learning in factored mdps. In *IJCAI*, volume 16, pp. 740–747, 1999.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models, 2020.

Adam Roman Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, 2018. URL `https://arxiv.org/abs/1806.01794`.

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

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr H. Pong, G. Berseth, and S. Levine. Contextual imagined goals for self-supervised robotic learning. In *CoRL*, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pp. 14843–14854, 2019.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. In *NIPS*, 2014.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37nd International Conference on Machine Learning*, volume 42 of *JMLR Workshop and Conference Proceedings*. JMLR, 2020.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning, 2020.

David Warde-Farley, Tom Van de Wiele, T. Kulkarni, Catalin Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *ArXiv*, abs/1811.11359, 2019.

Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P. Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration, 2019.

Marissa A. Weis, Kashyap Chitta, Yash Sharma, Wieland Brendel, Matthias Bethge, Andreas Geiger, and Alexander S. Ecker. Unmasking the inductive biases of unsupervised object representations for video sequences, 2020.

A APPENDIX

A.1 FULL PSEUDO-CODE OF SMORL ALGORITHM

Algorithm 2 SMORL: Self-Supervised Multi-object RL (with details)

Require: SCALOR encoder q_ϕ, goal-conditioned policy π_θ, goal-conditioned value function Q_w, number of data points from random policy N, number of training episodes K.

Collect $\mathcal{D} = \{\mathbf{o}_i\}_{i=1}^N$ using random initial policy.
Train SCALOR on sequences data uniformly sampled from $\mathcal{D}$ using loss described in Eq. 4.
Fit prior $p(\mathbf{z}^{\text{where}})$ to the latent encodings of observations $\{(\mathbf{z}_i^{\text{where}}, \mathbf{z}_i^{\text{what}}) = q_\phi(\mathbf{o}_i)\}_{i=1}^N$.
for $n = 1, ..., K$ episodes **do**
  for $t = 1, ..., H$ steps **do**
    if $t = 1$ **then**
      Generate goal $\mathbf{z}_g = (\hat{\mathbf{z}}_g^{\text{where}}, \mathbf{z}_g^{\text{what}})$ using SCALOR and initial observation $\mathbf{o}_1$ (pick random detected object k and substitute $\mathbf{z}^{\text{where}}$ by sampled from prior $\hat{\mathbf{z}}_g^{\text{where}} \sim p(\mathbf{z}^{\text{where}})$).
    end if
    Get action $a_t \sim \pi_\theta(q_\phi(\mathbf{o}_t), \mathbf{z}_g)$.
    Get next state observation $\mathbf{o}_{t+1}$.
    Encode $\mathbf{z}_t = q_\phi(\mathbf{o}_t), \mathbf{z}_{t+1} = q_\phi(\mathbf{o}_{t+1})$.
    Store $(\mathbf{z}_t, a_t, \mathbf{z}_{t+1}, \mathbf{z}_g)$ into replay buffer $\mathcal{R}$.
    Sample transition $(\mathbf{z}, a, \mathbf{z}', \mathbf{z}_g) \sim \mathcal{R}$.
    Compute matching reward signal $r = r(\mathbf{z}', \mathbf{z}_g)$. See Eq. 7
    Minimize Bellman Error using $(\mathbf{z}, a, \mathbf{z}', \mathbf{z}_g, r)$.
  end for
  for $t = T, ..., H$ steps **do**
    for $i = 1, ..., k$ steps **do**
      Sample future state $\mathbf{o}_{h_i}$ that have matching component in observation representation set $\mathbf{z}_{h_i}$ to the original goal $\mathbf{z}_g$, $t < h_i \leq H - 1$.
      Store $(\mathbf{z}_t, a_t, \mathbf{z}_{t+1}, \mathbf{z}_{h_i,k})$ into $\mathcal{R}$ (for k such that $\mathbf{z}_{h_i,k}$ is matching the original goal $\mathbf{z}_g$).
    end for
  end for
end for

A.2 ARCHITECTURE AND HYPERPARAMETER DETAILS

In this section we describe the hyperparameters used for SCALOR and SMORL in our experiments. For SAC, RIG and Skew-Fit, we started from standard settings and made environment-specific tweaks to tune them for best performance. In particular, significant hyperparameter search effort (>500 runs) was spent on finding the best SAC parameters for *Multi-Object Visual Rearranger* 2, 3, and 4 objects.

A.2.1 SCALOR

We were using the implementation SCALOR from the original authors. As our background model is not important and environments have stable background for this work we were modeling it with small $z_{bg} = 1$. We were using $\mathbf{z}^{\text{what}}$ dimension equal to 4 for 2 objects and 8 for 8 for one object. We observed that smaller dimension of $\mathbf{z}^{\text{what}}$ makes the training more stable, if it is possible to train SCALOR with it. Add the parameters that we modified from default settings can be found in Table 1.

A.2.2 SMORL

A.3 SCALOR TRACKING SENSITIVITY DURING (ADVERSARIAL) RL LEARNING

During our experimentation of the reward specification, we consider several different natural choices. First, we wanted to relay on internal SCALOR tracking of objects and compute distance to the goal $\mathbf{z}_g$ in the sub-space of $\mathbf{z}^{\text{where}}$ component. However, during such reward specification, agent was easily finding the way to exploit the biases towards position in the propagation of the representation to next time step. This shows importance of the proper evaluation of the scene decomposition performance in downstream tasks.

Hyper-parameter	Value
Optimizer	Adam (Kingma & Ba, 2015) with default settings
Number of iterations	5000
Learning rate	0.0001
Batch size	11
Explained Ratio Threshold	0.1
z_{what} dim	4
Number of training points	10000
Number of cells	4
Size bias	0.22
Size variance	0.12
Ratio bias	1.0
Ratio variance	0.3

Table 1: SCALOR parameters

Hyper-parameter	Value
Optimizer	Adam (Kingma & Ba, 2015) with default settings
Exploration Noise	None (SAC policy is stochastic)
RL Batch Size	2048
Reward Scaling	1
Automatic SAC entropy tuning	yes
SAC Soft Update Rate	0.05
# Training Batches per Time Step	1
Hidden Activation	ReLU
Network Initialization	Xavier uniform (Glorot & Bengio, 2010)
Separate Attention for Policy & Q-Function	yes
Replay Buffer Size	100000
Relabeling Fractions Rollout/Future/Imagined Goals	0.1 / 0.4 / 0.5
Number of Initial Random Samples	10000

Table 2: General hyper-parameters used by SMORL.

Hyper-parameter	Push, 1 Obj.	Push, 2 Obj.	Rearrange, 1 Obj	Rearrange, 2 Obj
Training Path Length	15	15	20	20
Evaluation Path Length	45	75	60	100
Learning Rate	0.001	0.0007	0.001	0.0005
Discount Factor	0.925	0.95	0.925	0.925
Matching Threshold α	1.2	1.3	1.2	1.3
No Match Reward $r_{\text{no goal}}$	0.75	1.0	0.75	1.5
$\mathbf{z}^{\text{what}}$ Dim	8	4	8	4
Embedding Dim d_e	48	32	48	32
Number of Cond./Uncond. Heads	3/0	1/1	3/0	3/0
Number of Parametric Queries	0	3	0	3
Policy Hidden Sizes	$[128, 128]$	$[128, 128, 128]$	$[64, 64]$	$[128, 128, 128]$
Q-Function Hidden Sizes	$[256, 256, 256]$	$[128, 128, 128]$	$[128, 128, 128]$	$[128, 128, 128]$

Table 3: Environment specific hyper-parameters used by SMORL.