HIERARCHICAL ABSTRACTION FOR COMBINATORIAL GENERALIZATION IN OBJECT REARRANGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Object rearrangement is a challenge for embodied agents because solving these tasks requires generalizing across a combinatorially large set of underlying entities that take the value of object states. Worse, these entities are often unknown and must be inferred from sensory percepts. We present a hierarchical abstraction approach to uncover these underlying entities and achieve combinatorial generalization from unstructured inputs. By constructing a factorized transition graph over clusters of object representations inferred from pixels, we show how to learn a correspondence between intervening on states of entities in the agent's model and acting on objects in the environment. We use this correspondence to develop a method for control that generalizes to different numbers and configurations of objects, which outperforms current offline deep RL methods when evaluated on a set of simulated rearrangement and stacking tasks. [1]

1 INTRODUCTION

A core property of intelligence is the ability to re-purpose previously acquired knowledge for solving new problems, but identifying symmetries from sensorimotor data for factorizing modeling and control into independent components has been a challenge for artificial agents, which generally fall into one of two paradigms. One paradigm runs orthogonal to the above problem by assuming human-defined abstractions to begin with (Ghallab et al., 1998; Fikes & Nilsson, 1971), including the abstraction of natural language that powers large pre-trained models (Bommasani et al., 2021). Approaches in the other paradigm do learn purely from sensorimotor data, but generally learn monolithic representations that offer no clear mechanism for reuse (Hafner et al., 2019; Schrittwieser et al., 2020).

Object rearrangement (Batra et al., 2020) offers an intuitive setting for studying this problem of learning reusable abstractions. Because the space of object configurations is combinatorially large, solving novel rearrangement problems requires the agent to recognize that the same action for moving an object from one location to another can be reused for different objects in different contexts. We can formulate this as an offline goal-conditioned reinforcement learning (RL) problem, where the agent is trained on a dataset of sensorimotor interactions and is evaluated on rearranging objects to satisfy constraints depicted in a goal image.

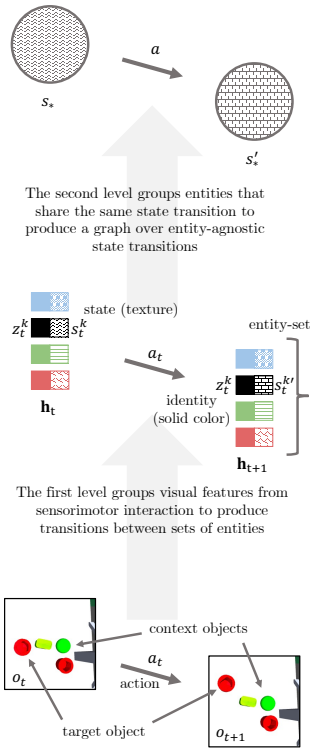

Figure 1: NCS uses a two-level hierarchy to abstract video interactions into a graph over individual state transitions. The entity representing the target object is in black.

Solving rearrangement problems is challenging because it requires long horizon planning in combinatorial task spaces. Offline RL approaches (Kostrikov et al., 2021) that do not explicitly infer factorized representations of objects – which we call entities – struggle to generalize to novel problems with more objects. On the other hand, planning with object-centric methods that do infer entities (Veerapaneni et al., 2020) is not easy because the difficulties of planning with learned single-step models (Janner et al., 2019) are exacerbated in combinatorial spaces.

[1] A step-by-step explanatory video of our method can be found in the supplementary material.

Instead of planning with parametric models, our work takes inspiration from non-parametric planning methods that have shown success in combining neural networks with graph search to generate long-horizon plans. These methods (Yang et al., 2020; Zhang et al., 2018; Lippi et al., 2020; Emmons et al., 2020) explicitly construct a transition graph from their buffer of experience and plan by searching through the actions recorded in this transition graph with a learned distance metric. The advantage of searching over transition graphs constructed from offline data is the ability to stitch together different path segments in the offline data to solve new problems. The disadvantage is that the non-parametric nature of such methods requires transitions that will be used to solve new problems to have already been recorded in the buffer, which make conventional methods that store observations monolithically ill-suited for combinatorial generalization. For example, Fig. 2b shows that the same state transition can manifest for different objects and in different contexts, but monolithic non-parametric methods do not have learning constraints that enable them to recognizes that all scenarios exhibit the same state transition at an abstract level, which may induce an blowup in the number of nodes of the search graph. To overcome this problem, we devise a method that explicitly represents the similarity among such state transitions in different contexts.

Our method, **Neural Constraint Satisfaction** (NCS), marries the strengths of non-parametric planning with the strengths of object-centric representations. The key idea is to factorize each entity h^k extracted from an object-centric method into two parts, its identity z^k and its state s^k, and enforce that the identity is the part that is not affected by action and the state is the part that is. The modeling component of our approach abstracts the dataset of sensorimotor interactions into a graph over state transitions of individual entities. By construction, a state transition is agnostic to the identity of the entity or the context entities present in the sensorimotor interaction from which it was learned, enabling it to be reused across multiple entities and contexts. The planning and control component of our approach then re-composes sequences of state transitions to solve new rearrangement problems by inferring what state transition can be taken given only the current and goal image observations. This is done by inferring goal entities as *constraints* on the states that the current entities should satisfy, *binding* a current entity and its corresponding goal constraint to nodes in the transition graph, and using the state transition between the two nodes to retrieve what action to take.

Our contributions are: (1) We present NCS as a neurosymbolic planning method that plans over entities as emergent symbols. (2) The modeling component of NCS is the first object-centric method with a two-level hierarchy of discrete abstractions tailored for control (Fig. 1). (3) The planning and control component of NCS presents a non-parametric planning approach for combinatorial task spaces. (4) Our results show that NCS outperforms both state-of-the-art offline RL methods and object-centric shooting-based planning methods across three simulated rearrangement problems.

2 Goal-Conditioned Reinforcement Learning with Entities

This section introduces a set of modifications to the standard goal-conditioned partially observed Markov decision process (POMDP) problem formulation that explicitly exposes the combinatorial structure of object rearrangement tasks of the following kind: "given an image o_1 of a set of objects on a tabletop and a goal image o_g that depicts a subset (or all) of the objects in new locations, execute a series of pick-and-move action primitives to rearrange the objects to satisfy the constraints depicted in the goal image." We assume an offline RL setting, where the agent is trained on a dataset of transitions $\{(o_1, a_1, ...a_{T-1}, o_T)\}_{n=1}^{N}$ and evaluated on tasks specified as (o_1, o_g).

The standard problem formulation is defined as $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{H}, \mathcal{G}, E, P, R)$, with observation space $\mathcal{O}$, action space $\mathcal{A}$, latent space $\mathcal{H}$, goal space $\mathcal{G}$, observation function $E : \mathcal{H} \to \mathcal{O}$, transition function $P : \mathcal{H} \times \mathcal{A} \to \mathcal{H}$, and reward function $R : \mathcal{H} \times \mathcal{G} \to \mathbb{R}$. The issue with monolithically modeling the latent space this way is that it exposes no mechanism for modeling commonalities among different scenes, such as scenes that share the same object, scenes that contain objects in the same location, or a scene with multiple instances of the same type of object. Not modeling this structure means each scene in the combinatorial space of object configurations is treated as unrelated from any other scene. This prevents us from designing control algorithms that exploit commonalities among scenes to reduce the intractability of searching the combinatorial space.

We thus introduce three structural assumptions to the standard formulation. The first is *independence*, that the latent space decomposes into independent subspaces as $\mathcal{H} = \mathcal{H}^1 \times ... \times \mathcal{H}^K$, each representing an independent degree of freedom, one for each object we intend to model. The second is *symmetry*, that the latent subspaces are not only independent but also symmetric, i.e. $\mathcal{H}^1 = ... = \mathcal{H}^K$, which manifests as the permutation invariance of E and R and the permutation

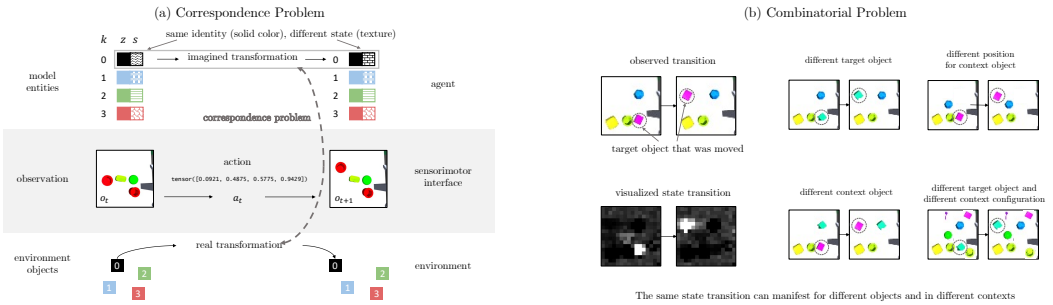

Figure 2: **Solving object rearrangement requires solving two challenging problems**. (a) The **correspondence problem** is the problem of abstracting raw sensorimotor signal into representations of entities such that there is a correspondence between how an agent intervenes on an entity and how its action affects an object in the environment. k denotes the index of the entity, z denotes its identity (shown with solid colors), and s denotes its state (shown with textures). The entity representing the target object that was moved is in black. (b) The **combinatorial problem** is the problem of representing the combinatorial space of tasks in a way that captures common state transitions in a way that is agnostic to the identity of the target object that was moved (indicated by the dotted circle) and to the configuration of context objects in the rest of the scene.

equivariance of P. We define an **entity**[2] $h \in \mathcal{H}^k$ as an independent degree of freedom that is processed symmetrically, and an **entity-set** as the set of entities $\mathbf{h} = (h^1, ..., h^K)$ that explain an observation. The third is *factorization*, that each entity h is factorized into an identity $z \in \mathcal{Z}$ and a state $s \in \mathcal{S}$, i.e. $\mathcal{H}^k = \mathcal{Z} \times \mathcal{S}$, where z is the part that is not affected by action (e.g. its color) and s is the part that is (e.g. its location). These assumptions – independence, symmetry, and factorization – model intuitive properties of physical objects: objects can be independently acted upon without affecting other objects, the same physical laws apply to all objects, and acting on an object does not change its identity. With such a formulation, it is natural to express goals as a set of constraints $\mathbf{h}_g = (h_g^1, ..., h_g^k)$, and to solve a task is to take actions that transforms the subset of entities in the initial observation o_1 whose identities are given by $\mathbf{z}_g$ to new states specified by $\mathbf{s}_g$.

This new formulation solves the problem of modeling the commonalities among different scenes stated above. To model scenes that share the same object, we assign an entity from each scene to all share the same identity z. To model scenes that contain objects in the same location, we assign an entity from each scene to all share the same state s. To model a scene with multiple instances of the same type of object, we assign multiple entities in the scene to share same identity z.

Exposing this structure in our problem formulation enables us to exploit it. To do so requires solving two problems. The first is the **correspondence problem**, which is the problem of learning to represent the space $\mathcal{H}$ (by training the weights of models of E, P, and R) such that there is a correspondence between how the agent models how actions affect entities and how actions actually affect objects in the environment. The source of the correspondence problem is that the agent only receives information about the environment through the sensorimotor interface and does not know what each $\mathcal{H}^k$ should semantically mean *a priori* (e.g. "h^1 and h^2 refer to things, but I do not yet know what"), unlike standard POMDP formulations like object-oriented MDPs, relational MDPs, and factorized MDPs that do attach human-defined semantics to $\mathcal{H}^k$ (e.g. "h^1 refers to this cube, h^2 refers to that ball"). The second is the **combinatorial problem**, which is the problem of efficiently searching the combinatorial space of $|\mathcal{Z}|^K \times \frac{|\mathcal{S}|!}{(\mathcal{S}-K)!}$ entity configurations. This leads to a natural evaluation criterion, which is to test to what extent an agent can zero-shot-generalize to solve rearrangement tasks involving a new set of object configurations that is disjoint from the object configurations observed in training, assuming that the training configurations have collectively covered $\mathcal{Z}$ and $\mathcal{S}$.

To focus on the combinatorial nature of rearrangement, we are not interested in low-level manipulation, so we represent the action space as a set of primitives $(w, \Delta w)$, with Cartesian coordinates $w = (x, y, z)$, where actions sparsely affect one entity at a time and how an action affects an object's state does not depend on its identity. We are not interested in representing occluded objects, so we assume that objects are all constrained to the xy plane or xz plane and are directly visible to the camera. Following prior work (Hansen-Estruch et al., 2022; Castro et al., 2009), we make a *bisimu-*

[2]We use "object" to refer to an independent degree of freedom in the environment, and "entity" to refer to the agent's representation of the object.

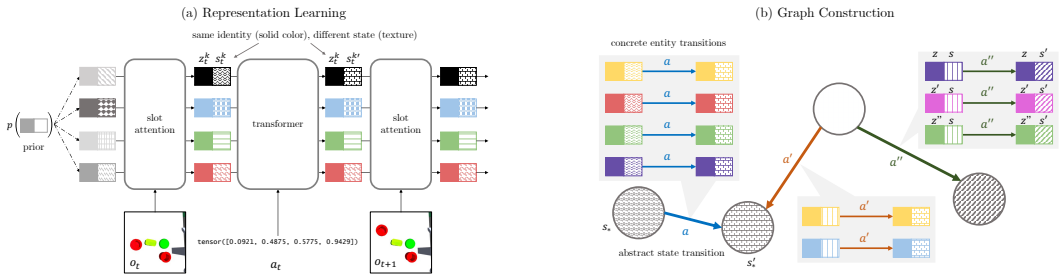

Figure 3: **Modeling**. The modeling component of NCS constructs a two-level abstraction hierarchy. (a) **Level one:** NCS learns to infer a set of entities from sensorimotor transitions with pick-and-move actions, in which one entity is moved per transition. We explicitly enforce that the identity z (shown with solid colors) of an entity remains unchanged between time-steps. The transformer dynamics model learns to sparsely predict the states s (shown with textures) of the entities at the next time-step. Empirically, NCS learns to modify only the state of the entity (shown in black) that it uses to represent the target object that was moved in the environment. *This solves the correspondence problem by forcing the network to use predict and reconstruct observations through the entity bottleneck.* (b) **Level two:** NCS abstracts transitions over entity-sets into transitions over individual states, constructing a graph where states are nodes and transitions between them are edges. This is done by clustering entity transitions that share similar initial states and similar final states. *This solves the combinatorial problem by constructing the state transitions to be agnostic to the identities or contexts of the entities from which the transitions were derived.*

lation assumption that the state space can be partitioned into a finite set of equivalence classes, and that there is one action primitive that transitions between each pair of equivalence classes.

3 Neural Constraint Satisfaction

We present our method, Neural Constraint Satisfaction (NCS), for solving object rearrangement problems from pixels. Our main idea is that generalizing over such a combinatorial space requires the agent to recognize that the same action for moving an object from one location to another can be reused for different objects and in different contexts. Concretely, given a transition $o, a \rightarrow o'$ from the training set in which a single object k has been moved, the modeling problem is to represent the state transition $s^k \rightarrow s^{k\prime}$ as decoupled from the identity z^k of the affected entity and as decoupled from the other context entities $h^{\neq k}$ in the scene. The control problem is then to apply a composition of previously seen state transitions to satisfy goal constraints on new object configurations.

3.1 Modeling

Our modeling approach abstracts the training dataset of action-conditioned videos into a factorized graph over state transitions that can be reused across different rearrangement problems. Constructing such a graph decomposes into two representation learning problems: the first concerns how to represent objects in a way that exposes their commonalities and the second concerns how to represent actions in a way that enables them to be reused across these commonalities. Note that these two problems are both different instances of the same fundamental problem of partitioning a space into discrete groups, but at different levels of abstraction: solving the correspondence problem reduces to learning how to cluster visual features into entities and solving the correspondence problem reduces to learning how to cluster transitions over entity-sets into transitions over individual states. Our model thus implements a two-level abstraction hierarchy.

Level 1: abstracting visual features into sets of entities The goal of the first level (Fig. 3a) is to map a video transition $o_1, a_1 \rightarrow o_2$ into a transition over entity-sets $(h_1^1, ..., h_1^K) \rightarrow (h_2^1, ..., h_2^K)$. Following our assumptions in §2, the criteria our entity-set transitions should satisfy are (1) entities represent different objects in the scene, i.e. they are *independent* and *symmetric* and (2) only the state s^k of entity k changes, not its identity z^k, i.e. the entity is *factorized*. Because each video transition depicts only a single object being moved, we also want (3) the transition to be *sparse*, i.e. only one entity h^k changes in the transition and the rest $h^{\neq k}$ remain unchanged.

We manually enforce independence, symmetry, and factorization but let sparsity emerge. We formulate the inference of entity-sets as the application of an equivariant action-conditioned sequential

Bayesian filter with a mixture model as the latent state, where entity representations are the parameters of the mixture components. Given the connection between the slots produced by slot attention (SA) (Locatello et al., 2020b) and mixture components Chang et al. (2022), we concretely implement this filter by treating the slots produced by the slot attention module of the state-of-the-art SLATE architecture (Singh et al., 2022) as entities and temporally evolving them with a transformer decoder (GPT) (Vaswani et al., 2017; Radford et al., 2018). We split the entity $h \in \mathbb{R}^n$ into an identity half $z \in \mathbb{R}^{\frac{n}{2}}$ and a state half $s \in \mathbb{R}^{\frac{n}{2}}$. With $\mathbf{h} = (h^1, ..., h^K)$,

$$\mathbf{h}_{t+1} = SA(\hat{\mathbf{h}}_t, o_{t+1}) \qquad \hat{\mathbf{s}}_{t+1} = GPT\left(\text{queries} = \mathbf{s}_t, \text{keys/values} = [\mathbf{s}_t, a_t]\right), \tag{1}$$

where $\hat{\mathbf{h}}_{t+1} = [\mathbf{z}_t, \hat{\mathbf{s}}_{t+1}]$ and $\hat{\mathbf{h}}_0$ are independent samples from a Gaussian. This approach enforces independence because $\hat{\mathbf{h}}_0$ are independent, symmetry because the transformer is equivariant, and factorization because $\hat{\mathbf{z}}_{t+1}$ is a copied from $\mathbf{z}_t$. Empirically, we find that this architecture models sparse changes in the slots (Fig. 4c). We call this implementation **dynamic SLATE** (dSLATE), but our contribution of how we structurally enforce the above three criteria remains invariant to *how* the sequential Bayesian filter is implemented, which we expect will advance beyond using SA and GPT.

Level 2: abstracting transitions over sets of entities into individual state transitions The goal of the second level (Fig. 3b) is to construct a state transition graph from a buffer of entity-set transitions $\{(\mathbf{h}_1, ...\mathbf{h}_T)\}_{n=1}^N$ produced from the first level. Specifically, we seek each edge of this graph to represent a state transition that has been shared across multiple entities in different contexts in the training dataset, such that we can reuse such state transitions when choosing actions for solving new rearrangement problems. Given our bisimulation assumption that observed locations can be partitioned into a finite number of groups, this goal translates into a problem of clustering state transitions, and the problem of clustering state transitions reduces to clustering the states before and after the transition. We treat each cluster centroid as a node in the state transition graph, and an edge between nodes is tagged with the single action that transforms one node's state to another's. The nodes represent equivalence classes among the states and the edges represent actions that transform a state from one class to anothere. The algorithm for constructing the graph is shown in Alg. 1.

Algorithm 1 Building the Graph

input `model`, `buffer`
for $\{(o_t, a_t, o_{t+1})\}_n$ in `buffer` **do**
  # infer entities from transition
  $\{(\mathbf{h}_t, a_t, \mathbf{h}_{t+1})\}_n \leftarrow$ `model` $(\{o_t, a_t, o_{t+1}\}_n)$.
  # identify which entity changed in transition
  $\{(h_t^k, a_t, h_{t+1}^k)\}_n \leftarrow$ `isolate` $(\{(\mathbf{h}_t, a_t, \mathbf{h}_{t+1})\}_n)$
end for
# partition transitions by clustering entities
$\{s_*\}_{m=1}^M \leftarrow$ `cluster` $\left(\{(s_t^k, a_t, s_{t+1}^k)\}_{n=1}^N\right)$
transitions between clusters are edges
initialize `graph` with nodes $s_*^{[m]}$, for $m \in [1:M]$
for each $\{(h_t^k, a_t, h_{t+1}^k)\}_n$ **do**
  # infer cluster assignments
  $[i], [j] \leftarrow$ `bind` $\left(h_t^k\right)$, `bind` $\left(h_{t+1}^k\right)$
  # tag edge with action a_t
  `graph.edges`$[i, j] \leftarrow$ `create-edge` $\left(s_*^{[i]} \xrightarrow{a_t} s_*^{[j]}\right)$
end for
return `graph`

To create the nodes, the first step is to identify the entity h^k that dSLATE predicted was affected by a_t in each transition $(\mathbf{h}_t, a_t, \mathbf{h}_{t+1})$. Our `isolate` procedure achieve this by solving $k = \arg\max_{k' \in \{1,...,K\}} d(s_t^{k'}, s_{t+1}^{k'})$ to identify the index of the entity whose state has most changed during the transition, where $d(\cdot, \cdot)$ is a distance function, detailed in the Table 5 of the Appendix. Having converted the buffer of transitions over entity sets $\{(\mathbf{h}_1, ...\mathbf{h}_T)\}_{n=1}^N$ to a buffer of individual entity transitions $\{(h_t^k, a_t, h_{t+1}^k)\}_{n=1}^{N \times (T-1)}$, and thus a buffer of individual state transitions $\{(s_t^k, a_t, s_{t+1}^k)\}_{n=1}^{N \times (T-1)}$, the second step is to cluster similar transitions together by clustering the states before and after each transition. Our `cluster` procedure achieves this by clustering over all states together, which is sufficient since the end state of transition is the starting state of another. A proper clustering should return centroids $\{s_*\}_{m=1}^M$ that all correspond to actual states of entities, which we hypothesize will correspond to actual locations in the training set based on how dSLATE is trained.

The purpose of the edges between the nodes is to record an action that transforms an entity from one state to another. For each entity transition (h_t^k, a_t, h_{t+1}^k) we `bind` entity h_t^k and h_{t+1}^k to their associated nodes $s_*^{[i]}$ and $s_*^{[j]}$ and create an edge between $s_*^{[i]}$ and $s_*^{[j]}$ tagged with action a, overwriting previous edges based on the assumption that with a proper clustering there should only be one action primitive per pair of nodes. Applying the `bind` procedure to an entity h^k returns the index of the cluster of the centroid s_* that is nearest to the entity's state s^k. For our experiments `cluster`

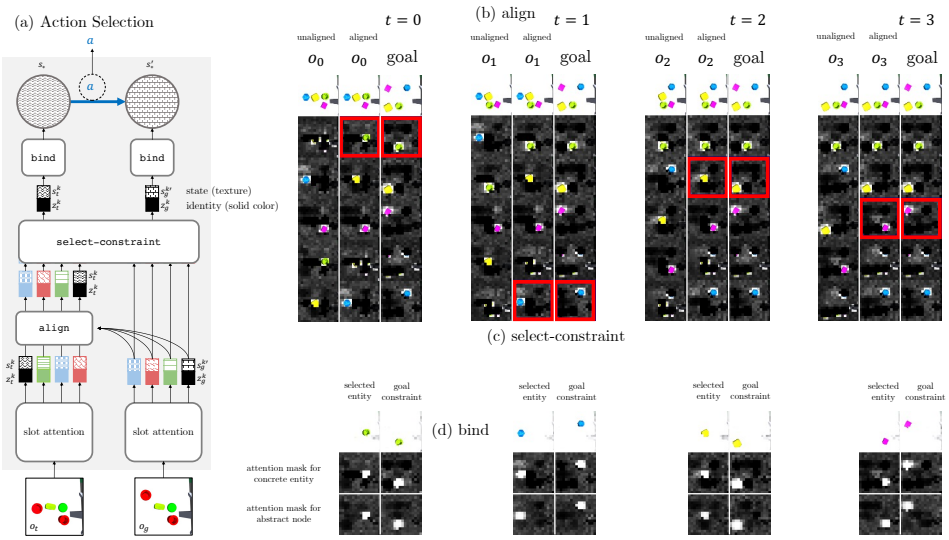

Figure 4: **Planning and control**. Given a rearrangement problem specified only by the current and goal observations (o_0, o_g), NCS decomposes the rearrangement problem into one subproblem (o_t, o_g) per entity. (a) shows the computations NCS uses to solve each subproblem and (b-d) show these steps in context. For each subproblem (o_t, o_g), NCS infers entities from both the current and goal observations. The states of the goal entities indicate constraints on the desired locations of the current entities. (b) NCS first aligns the indices of current and goal entities to have corresponding identities. (c) It selects the index k of the next goal constraint s_g^k to satisfy, as indicated by the red box. The selected goal constraint and current entity are also colored black in (a), and note that their identities are the same but states are different; we want to choose the action to transform the state of the current entity to the state of the goal constraint. (d) It binds the selected goal constraint and its corresponding current entity to nodes s_* and s'_* in the transition graph. Lastly, it identifies the edge connecting those two nodes and executes the action tagged to that edge in the environment.

is implemented with K-means and `bind` is implemented with nearest neighbors using the same distance metric used for K-means (see Table 5 in the Appendix), but other clustering algorithms and distance metrics can also be used. In our experiments (Fig. 8), we also show that it is also possible to have more than one action primitive per pair of nodes as long as these actions all map between states bound to the same pair of nodes.

3.2 Planning and Control

Having constructed a graph of state transitions, our approach for control re-composes sequences of state transitions to solve new rearrangement problems. Specifically, the agent decomposes the rearrangement problem into a set of per-entity subproblems, searches the transition graph for a transition that transforms the current entity's state to its goal state, and executes the action tagged with this transition in the environment. This problem decomposition is possible because the transitions in our graph are constructed to be agnostic to identity and context, enabling different rearrangement problems to share solutions to the same subproblems. The core challenge in deciding which transitions to compose is in determining which transitions are *possible* to compose. That is, the agent must determine which nodes in the graph correspond to the given goal constraints and which nodes correspond to the entities in the current observation, but the current entities $\mathbf{h}_t$ and goal constraints $\mathbf{h}_g$ must themselves be inferred from the current and goal observations o_t and o_g, requiring the agent to infer both what to do and how to do it purely from its sensorimotor interface.

Algorithm 2 Action Selection

given `model`, `graph`
input goal o_g, observation o_t
infer goal constraints and current entities
$\mathbf{h_g}, \mathbf{h_t} \leftarrow$ `model` (o_g), `model` (o_t)
align entity indices of $\mathbf{h_t}$ with those of $\mathbf{h_g}$
$\pi \leftarrow$ `align` $(\mathbf{h_t}, \mathbf{h_g})$
permute indices of $\mathbf{h_t}$ according to π
$\mathbf{h_t} \leftarrow (h_t^{\pi[1]}, ..., h_t^{\pi[K]})$
identify kth goal constraint to satisfy next
$k \leftarrow$ `select-constraint` $(\mathbf{h_t}, \mathbf{h_g})$
infer cluster assignments
$[i], [j] \leftarrow$ `bind` (h_t^k), `bind` (h_g^k)
action that transforms node $[i]$ to node $[j]$
return `graph.edges`$[i, j]$`.action`

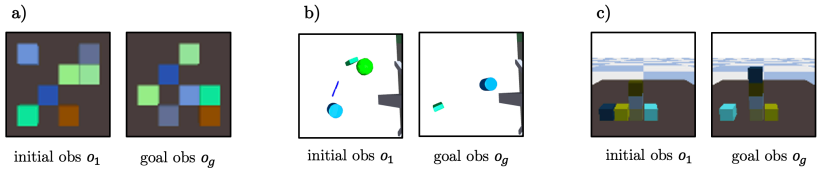

Figure 5: **Environments** Our environments are (a) *block-rearrange*, (b) *robogym-rearrange*, and (c) *block-stacking*. (a) shows a complete specification of goal constraints; (b) shows a partial specification that only specifies the desired locations for two objects; (c) shows the case where some goal constraints are already satisfied in the initial observation, which in this example are the three blocks in the middle column of o_1.

Our approach is summarized in Alg. 2 and Fig. 4 and takes four steps. The first step applies dSLATE to infer $\mathbf{h}_t$ and $\mathbf{h}_g$ from o_t and o_g. The second step permutes the indices of $\mathbf{h}_t$ and $\mathbf{h}_g$ to `align` the identities of h_t^k with h_g^k (Fig. 4b). This step is necessary because the permutation symmetry of the entities does not guarantee that the entities with the same identities will have the same index k, i.e. z_t^k might not correspond to z_g^k. We implement this using the Hungarian algorithm that minimizes the matching cost between $\mathbf{z}^t$ and $\mathbf{z}^g$. The third step selects which goal constraint h_g^k to satisfy next (Fig. 4c).. In the case where objects can move independently, we implement this `select-constraint` procedure by determining which constraint h_g^k has the highest difference in state with its counterpart h_t^k, which reduces to solving the same argxmax problem as in `isolate`. We discuss how we handle more complex dependencies in Appendix A.3. Lastly, we `bind` h_t^k and h_t^g to the graph and return the action tagged to the edge between their respective nodes (Fig. 4d). If an edge does not exist between the nodes, then we simply take a random action.

4 Experiments

This section evaluates to what extent NCS can generalize over combinatorially-structured task spaces from perceptual input. The crucial test for combinatorial generalization is to solve a set of tasks from a part of the combinatorial space of transitions over object configurations that is disjoint from the part of the space given for training. The most straightforward way to partition this space is to evaluate on different numbers of objects from training. Concretely, we compare NCS to several offline reinforcement learning baselines on three challenging rearrangement environments and show a significant gap in performance between our method and the next best method.

Environments. *block-rearrange* is our simplest environment; *robogym-rearrange* and *block-stack* each add complexity along different axes. In *block-rearrange* (Fig. 5a), all objects are the same size, shape, and orientation. $\mathcal{S}$ covers 16 locations in a grid. $\mathcal{Z}$ is the continuous space of red-green-blue values from 0 to 1. *robogym-rearrange* (Fig. 5b) adapts the OpenAI (2020) rearrange environment and removes the assumption from *block-rearrange* that all objects have the same size, shape, orientation, and predefined locations. *block-stack* (Fig. 5c) adds preconditions and postconditions on whether objects can be moved: blocks can only be picked if from the top of a stack, and blocks can only be placed at a given height if there is an object beneath to support it (otherwise it falls).

Experimental setup. We evaluate two settings: *complete* and *partial*. In the *complete* setting, all objects have an associated constraint, whereas in the *partial* setting, only a subset of objects have associated constraints, and this is reflected in the goal image (Fig. 5b). In *block-rearrange*, all constraints are unsatisfied in the start state. In *robogym-rearrange*, four constraints are unsatisfied in the start state. In *block-stacking*, three constraints are unsatisfied in the start state. We measure as the "Fractional Success Rate" the mean change in the number of satisfied constraints divided by the number of initially unsatisfied constraints for each episode.

Baselines. We implement a version of object-centric MPC (Veerapaneni et al., 2020) by replacing our factorized graph search with CEM. We investigate a monolithic ablation of NCS that constructs a graph over transitions between sets of entity states rather than individual entity states (MGS). We compare with state-of-the-art pixel-based behavior cloning (BC) and implicit Q-learning (IQL) implementations based off of Kostrikov (2021). Our last baseline just takes random actions (Rand).

4.1 Quantitative Results

Figure 6 shows that NCS performs significantly better than the baselines in combinatorial generalization (about a 5-10x improvement), suggesting that modeling the properties of independence, symmetry, and factorization of entities enables NCS to collapse the combinatorial structure of the

task space. We trained all methods on an offline dataset of 5000 trajectories showing only 4 objects. We evaluate on 4-7 objects for 100 episodes across 10 seeds. These task spaces are enormous: if we discretize the number of colors to 256, the number of possible configurations of k objects in the simplest environment of *block-rearrange* is $256^k \times \frac{16!}{(16-k)!}$ which for $k = 7$ is 4.2×10^{24}. Most of the baselines perform no better or only slightly better than random. Although in theory an equivariant relational dynamics model as in (Veerapaneni et al., 2020) also collapses the combinatorial structure of the task space, in practice it is difficult to perform shooting-based planning with a model trained to predict a single step forward (Janner et al., 2019). As a result, our MPC baseline performs poorly because its rollouts are poor, and it is significantly more computationally expensive to run (11 hours instead of 20 minutes). Additional results are in Appendix.

Figure 6: Combinatorial generalization from training data collected from environments with 4 objects reporting fractional success rate. MGS stands for monolithic graph search. Run with 10 seeds, standard error computed.

Table 1: *block-rearrange*, complete specification.

Method	4	5	6	7
Ours	**0.94** ± 0.01	**0.93** ± 0.00	**0.93** ± 0.00	**0.89** ± 0.00
Rand	0.06 ± 0.02	0.07 ± 0.03	0.07 ± 0.03	0.08 ± 0.03
MPC	0.16 ± 0.06	0.12 ± 0.04	0.11 ± 0.04	0.10 ± 0.03
MGS	0.07 ± 0.03	0.06 ± 0.02	0.07 ± 0.02	0.08 ± 0.03
IQL	0.07 ± 0.01	0.03 ± 0.00	0.02 ± 0.00	0.02 ± 0.00
BC	0.03 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	0.01 ± 0.00

Table 2: *block-stacking*, complete specification.

Method	4	5	6	7
Ours	**0.58** ± 0.02	**0.48** ± 0.02	**0.35** ± 0.01	**0.24** ± 0.01
Rand	0.02 ± 0.01	0.01 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
MPC	0.03 ± 0.01	0.01 ± 0.01	0.02 ± 0.01	0.01 ± 0.01
MGS	0.13 ± 0.00	0.01 ± 0.01	0.02 ± 0.01	0.01 ± 0.01
IQL	0.19 ±0.02	0.12 ±0.01	0.13 ±0.02	0.07 ±0.01
BC	0.21 ±0.01	0.13 ±0.01	0.13 ±0.01	0.08 ±0.01

Table 3: *robogym-rearrange*, complete specification.

Method	4	5	6	7
Ours	**0.64** ± 0.01	**0.47** ± 0.01	**0.49** ± 0.01	**0.41** ± 0.01
Rand	0.01 ± 0.0	0.01 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
MPC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
MGS	0.01 ± 0.0	0.01 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
IQL	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
BC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

Table 4: *block-stacking*, partial specification.

Method	4	5	6	7
Ours	**0.34** ± 0.01	**0.27** ± 0.01	**0.21** ± 0.01	**0.15** ± 0.01
Rand	0.03 ± 0.01	0.03 ± 0.01	0.03 ± 0.01	0.03 ± 0.01
MPC	0.04 ± 0.02	0.03 ± 0.01	0.02 ± 0.01	0.04 ± 0.01
MGS	0.03 ± 0.01	0.03 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
IQL	0.14 ±0.01	0.06 ±0.01	0.04 ±0.00	0.03 ±0.01
BC	0.17 ±0.01	0.07 ±0.01	0.04 ±0.00	0.03 ±0.00

4.2 Qualitative Analysis

Fig. 4b provides a qualitative visualization of key steps in NCS for *robogym-rearrange*, visualizing the `align`, `select-constraint`, and `bind` procedures. We observe that NCS discovers that states and identities correspond to object locations and appearance respectively. Fig. 7 shows a t-SNE (Van der Maaten & Hinton, 2008) plot that clusters entities inferred from the *robogym* environment. One feature we notice about the t-SNE plot is that there are multiple cluster indices that map onto similar groups of points. This reveals that multiple different regions of $\mathcal{S}$ appear to be modeling similar states. We also tried merging redundant clusters, but found that this did not improve quantitative performance.

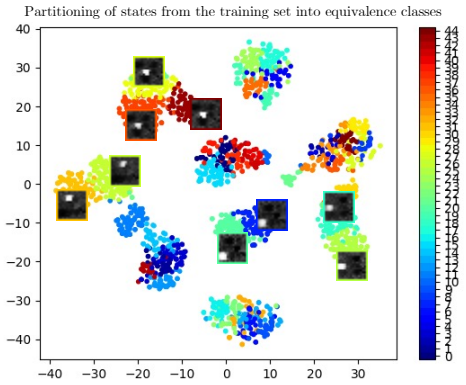

Figure 7: **Nodes as equivalent classes over states** This figure shows the clustering of states inferred for the *robogym* environment, where each cluster centroid is treated as a node in our transition graph. Each cluster is labeled with an attention mask computed by averaging the attention masks that slot attention produces for the entities associated with the cluster.

4.3 More Challenging Settings

We analyzed NCS in more challenging settings that reflect the noisy nature of real-world robotics. As Fig. 8 (left) shows, NCS is robust to the addition of Gaussian noise to the action at every time step, up until the noise variance is comparable to the maximum distance for successful picking and goal placements. The performance remains high given significantly fewer interaction steps (Fig. 8, right).

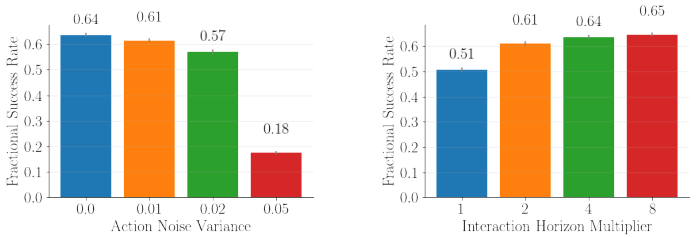

Figure 8: **Stress testing NCS** This figure shows the performance of NCS on *robogym-rearrange* as we vary the amount of noise added to the actions (left) and vary the interaction horizon, defined as a multiple of the minimum steps needed to complete the task (right).

5 Related Work

The problem of discovering re-composable representations is generally motivated by combinatorial task spaces. The traditional approach to enforcing this compositional inductive bias is to compactly represent the task space with MDPs that have a graph-structured latent space, such as factored MDPs Boutilier et al. (1995; 2000); Guestrin et al. (2003a), relational MDPs Wang et al. (2008); Guestrin et al. (2003b); Gardiol & Kaelbling (2003), and object-oriented MDPs Diuk et al. (2008); Abel et al. (2015). The drawback of these neurosymbolic approaches is that they generally assume entities have manually-defined semantics (Chang et al., 2016; Zadaianchuk et al., 2022; Bapst et al., 2019; Zhang et al., 2018), whereas our work discovers the entity semantics through data.

Other works that discover such entity semantics are so-called object-centric methods (Greff et al., 2017; Van Steenkiste et al., 2018; Greff et al., 2019; 2020; Locatello et al., 2020a; Kipf et al., 2021; Zoran et al., 2021; Singh et al., 2021). However, these approaches evaluate the quality of inferred entities via segmentation metrics, which does not assesses what we want the object representations to *do*. Instead, we directly evaluate our inferred entities by *how well they can reused for solving tasks*. Kulkarni et al. (2019) considers how object representations improve exploration, but we consider the offline setting which requires zero-shot generalization. Veerapaneni et al. (2020) also evaluate on control tasks, but their shooting-based planning method in suffers from compounding errors as other learned single-step models do (Janner et al., 2019), while our non-parametric approach of composing previously seen transitions enables us to plan for longer horizons.

Non-parametric approaches have become popular for long horizon planning (Yang et al., 2020; Zhang et al., 2018; Lippi et al., 2020; Emmons et al., 2020), but the drawback of these other approaches is they represent the entire observation monolithically, which causes a blowup of nodes in combinatorial task spaces, making it infeasible for these methods to be applied in rearrangement tasks that require generalizing to novel object configurations with different numbers of objects. Similar to our work, Huang et al. (2019) also tackles rearrangement problems by searching over a constructed latent task graph, but they require a demonstration during deployment time, whereas NCS does not because it reuses context-agnostic state transitions that were constructed during training.

6 Discussion

We have deconstructed the object rearrangement into the correspondence and combinatorial problems and proposed NCS as a solution to both. By drawing upon the principles of symmetry, independence, and factorization, NCS implements the insight that same action for moving an object from one location to another can be reused for different objects and in different contexts. Thus NCS outperforms various offline RL methods in combinatorial generalization across three simulated object rearrangement problems. NCS can be seen as physically solving a embodied constraint satisfaction probem, where states are variables, identities are variable values, and actions carry out variable assignments. Crucially the variables, their domains, the assignment operator, and the constraints are all learned from the sensorimotor interface, hence the name Neural Constraint Satisfaction.

Limitations and future work. NCS relies on a nonparametric, non-learning-based approach for control to highlight the generalization capability of our representation of the combinatorial task space, but this limits NCS to only composing previously seen transitions for previously seen entities. Collapsing the combinatorial space along state transitions already provides significant gains but does not adapt to the introduction of novel objects at test time. NCS is currently implemented with tools such as SLATE and K-means that have much potential for improvement. We expect future variations of NCS will improve upon our results by replacing SLATE and K-means with their future successors.

REFERENCES

David Abel, D. Ellis Hershkowitz, Gabriel Barth-Maron, Stephen Brawner, Kevin O'Farrell, James MacGlashan, and Stefanie Tellex. Goal-based action priors. In *ICAPS*, pp. 306–314. AAAI Press, 2015.

Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica Hamrick. Structured agents for physical construction. In *International conference on machine learning*, pp. 464–474. PMLR, 2019.

Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

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.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pp. 1104–1111, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artif. Intell.*, 121(1-2):49–107, 2000.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pp. 510–517. IEEE, 2015.

Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Equivalence relations in fully and partially observable markov decision processes. In *Twenty-First International Joint Conference on Artificial Intelligence*. Citeseer, 2009.

Michael Chang, Thomas L Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. *arXiv preprint arXiv:2207.00787*, 2022.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pp. 240–247. ACM, 2008.

Scott Emmons, Ajay Jain, Misha Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. *Advances in Neural Information Processing Systems*, 33:5251–5262, 2020.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf.

Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(71)90010-5. URL https://www.sciencedirect.com/science/article/pii/0004370271900105.

Natalia Gardiol and Leslie Kaelbling. Envelope-based Planning in Relational MDPs. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL https://proceedings.neurips.cc/paper/2003/file/4a06d868d044c50af0cf9bc82d2fc19f-Paper.pdf.

M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL—The Planning Domain Definition Language, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. *arXiv preprint arXiv:1708.03498*, 2017.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433. PMLR, 2019.

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

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, October 2003a. ISSN 1076-9757. doi: 10.1613/jair.1000. URL https://jair.org/index.php/jair/article/view/10341. tex.ids: guestrin2003EfficientSolutionAlgorithms.

Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pp. 1003–1010, San Francisco, CA, USA, 2003b. Morgan Kaufmann Publishers Inc.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. *arXiv preprint arXiv:2204.13060*, 2022.

De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8565–8574, 2019.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL https://github.com/ikostrikov/jaxrl.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *NeurIPS*, 2019.

Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5619–5626. IEEE, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020a.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/8511df98c02ab60aea1b2356c013bc0f-Abstract.html.

OpenAI. Robogym. https://github.com/openai/robogym, 2020.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL·E learns to compose. *arXiv preprint arXiv:2110.11405*, 2021.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-e learns to compose. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=h0OYV0We3oh.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pp. 1439–1456. PMLR, 2020.

C. Wang, S. Joshi, and R. Khardon. First order decision diagrams for relational mdps. *Journal of Artificial Intelligence Research*, 31:431–472, Mar 2008. ISSN 1076-9757. doi: 10.1613/jair.2489. URL http://dx.doi.org/10.1613/jair.2489.

Ge Yang, Amy Zhang, Ari S. Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2vec: Unsupervised Representation Learning by Latent Plans. In *Proceedings of The 2nd Annual Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 1–12, 2020.

Andrii Zadaianchuk, Georg Martius, and Fanny Yang. Self-supervised reinforcement learning with independently controllable subgoals. In *Conference on Robot Learning*, pp. 384–394. PMLR, 2022.

Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. Composable Planning with Attributes. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80, pp. 5837–5846. JMLR.org, 2018.

Daniel Zoran, Rishabh Kabra, Alexander Lerchner, and Danilo J Rezende. Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10439–10447, 2021.