Generalization to New Actions in Reinforcement Learning

Ayush Jain [* 1] **Andrew Szot** [* 1] **Joseph J. Lim** [1]

Abstract

A fundamental trait of intelligence is the ability to achieve goals in the face of novel circumstances, such as making decisions from new action choices. However, standard reinforcement learning assumes a fixed set of actions and requires expensive retraining when given a new action set. To make learning agents more adaptable, we introduce the problem of zero-shot generalization to new actions. We propose a two-stage framework where the agent first infers action representations from action information acquired separately from the task. A policy flexible to varying action sets is then trained with generalization objectives. We benchmark generalization on sequential tasks, such as selecting from an unseen tool-set to solve physical reasoning puzzles and stacking towers with novel 3D shapes. Videos and code are available at `https://sites.google.com/view/action-generalization`.

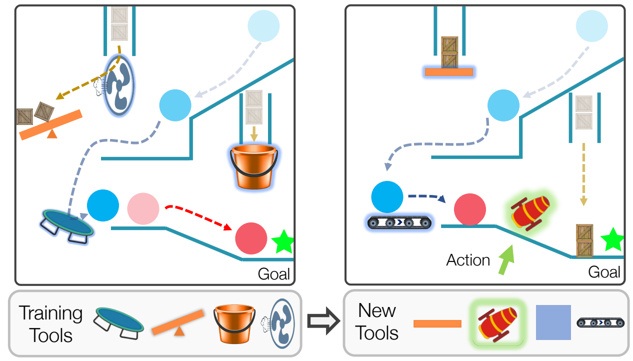


Figure 1. An illustration of zero-shot generalization to new actions in a sequential decision-making task, CREATE. (Left) Learning to select and place the right tools for reaching the goal. (Right) Generalizing the learned policy to a previously unseen set of tools.

1. Introduction

Imagine making a salad with an unfamiliar set of tools. Since tools are characterized by their behaviors, you would first inspect the tools by interacting with them. For instance, you can observe a blade has a thin edge and infer that it is sharp. Afterward, when you need to cut vegetables for the salad, you decide to use this blade because you know sharp objects are suitable for cutting. Like this, humans can make selections from a novel set of choices by observing the choices, inferring their properties, and finally making decisions to satisfy the requirements of the task.

From a reinforcement learning perspective, this motivates an important question of how agents can adapt to solve tasks with previously unseen actions. Prior work in deep reinforcement learning has explored generalization of policies over environments (Cobbe et al., 2018; Nichol et al., 2018), tasks (Finn et al., 2017; Parisi et al., 2018), and agent morphologies (Wang et al., 2018; Pathak et al., 2019). However, zero-shot generalization of policies to new discrete actions has not yet been explored. The primary goal of this paper is to propose the problem of generalization to new actions. In this setup, a policy that is trained on one set of discrete actions is evaluated on its ability to solve tasks zero-shot with new actions that were unseen during training.

Addressing this problem can enable robots to solve tasks with a previously unseen toolkit, recommender systems to make suggestions from newly added products, and hierarchical reinforcement learning agents to use a newly acquired skill set. In such applications, retraining with new actions would require prohibitively costly environment interactions. Hence, zero-shot generalization to new actions without retraining is crucial to building robust agents. To this end, we propose a framework and benchmark it on using new tools in the CREATE physics environment (Figure 1), stacking of towers with novel 3D shapes, reaching goals with unseen navigation skills, and recommending new articles to users.

We identify three challenges faced when generalizing to new actions. Firstly, an agent must observe or interact with the actions to obtain data about their characteristics. This data can be in the form of videos of a robot interacting with various tools, images of inspecting objects from different

[*]Equal contribution [1]Department of Computer Science, University of Southern California, California, USA. Correspondence to: Ayush Jain <ayushj@usc.edu>, Andrew Szot <szot@usc.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

viewpoints, or state trajectories observed when executing skills. In present work, we assume such action observations are given as input since acquiring them is domain-specific. The second key challenge is to extract meaningful properties of the actions from the acquired action observations, which are diverse and high-dimensional. Finally, the task-solving policy architecture must be flexible to incorporate new actions and be trained through a procedure that avoids overfitting (Hawkins, 2004) to training actions.

To address these challenges, we propose a two-stage framework of representing the given actions and using them for a task. First, we employ the hierarchical variational autoencoder (Edwards & Storkey, 2017) to learn action representations by encoding the acquired action observations. In the reinforcement learning stage, our proposed policy architecture computes each given action's utility using its representation and outputs a distribution. We observe that naive training leads to overfitting to specific actions. Thus, we propose a training procedure that encourages the policy to select diverse actions during training, hence improving its generalization to unseen actions.

Our main contribution is introducing the problem of generalization to new actions. We propose four new environments to benchmark this setting. We show that our proposed two-stage framework can extract meaningful action representations and utilize them to solve tasks by making decisions from new actions. Finally, we examine the robustness of our method and show its benefits over retraining on new actions.

2. Related Work

Generalization in Reinforcement Learning: Our proposed problem of zero-shot generalization to new discrete action-spaces follows prior research in deep reinforcement learning (RL) for building robust agents. Previously, state-space generalization has been used to transfer policies to new environments (Cobbe et al., 2018; Nichol et al., 2018; Packer et al., 2018), agent morphologies (Wang et al., 2018; Sanchez-Gonzalez et al., 2018; Pathak et al., 2019), and visual inputs for manipulation of unseen tools (Fang et al., 2018; Xie et al., 2019). Similarly, policies can solve new tasks by generalizing over input task-specifications, enabling agents to follow new instructions (Oh et al., 2017), demonstrations (Xu et al., 2017), and sequences of subtasks (Andreas et al., 2017). Likewise, our work enables policies to adapt to previously unseen action choices.

Unsupervised Representation Learning: Representation learning of high-dimensional data can make it easier to extract useful information for downstream tasks (Bengio et al., 2013). Prior work has explored downstream tasks such as classification and video prediction (Denton & Birodkar, 2017), relational reasoning through visual representation of objects (Steenbrugge et al., 2018), domain adaptation in RL by representing image states (Higgins et al., 2017b), and goal representation in RL for better exploration (Laversanne-Finot et al., 2018) and sample efficiency (Nair et al., 2018). In this paper, we leverage unsupervised representation learning of action observations to achieve generalization to new actions in the downstream RL task.

Learning Action Representations: In prior work, Chen et al. (2019); Chandak et al. (2019); Kim et al. (2019) learn a latent space of discrete actions during policy training by using forward or inverse models. Tennenholtz & Mannor (2019) use expert demonstration data to extract contextual action representations. However, these approaches require a predetermined and fixed action space. Thus, they cannot be used to infer representations of previously unseen actions. In contrast, we learn action representations by encoding action observations acquired independent of the task, which enables zero-shot generalization to novel actions.

Applications of Action Representations: Continuous representations of discrete actions have been primarily used to ease learning in large discrete action spaces (Dulac-Arnold et al., 2015; Chandak et al., 2019) or exploiting the shared structure among actions for efficient learning and exploration (He et al., 2015; Tennenholtz & Mannor, 2019; Kim et al., 2019). Concurrent work from Chandak et al. (2020) learns to predict in the space of action representations, allowing efficient finetuning when new actions are added. In contrast, we utilize action representations learned separately, to enable zero-shot generalization to new actions in RL.

3. Problem Formulation

In order to build robust decision-making agents, we introduce the problem setting of generalization to new actions. A policy that is trained on one set of actions is evaluated on its ability to utilize unseen actions without additional retraining. Such zero-shot transfer requires additional input that can illustrate the general characteristics of the actions. Our insight is that action choices, such as tools, are characterized by their general behaviors. Therefore, we record a collection of an action's behavior in diverse settings in a separate environment to serve as action observations. The action information extracted from these observations can then be used by the downstream task policy to make decisions. For instance, videos of an unseen blade interacting with various objects can be used to infer that the blade is sharp. If the downstream task is cutting, an agent can then reason to select this blade due to its sharpness.

3.1. Reinforcement Learning

We consider the problem family of episodic Markov Decision Processes (MDPs) with discrete action spaces. MDPs

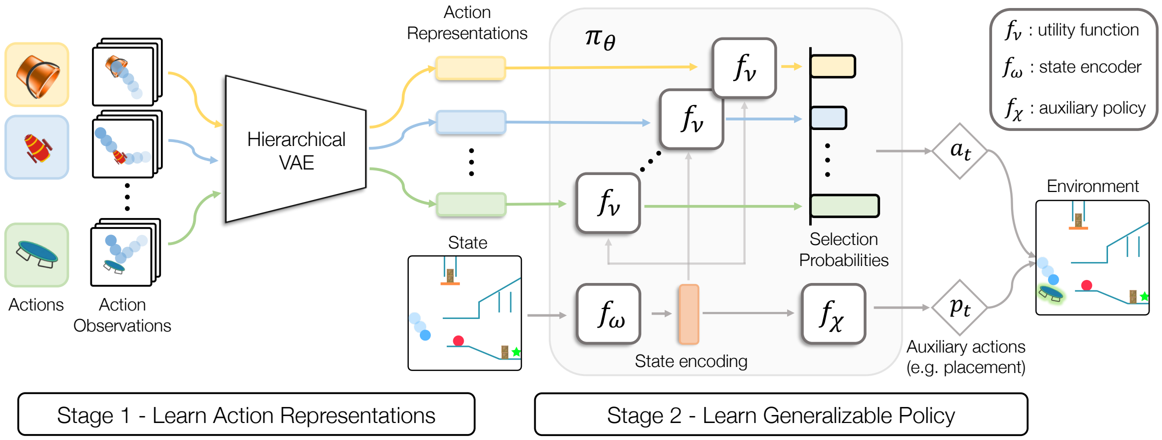


Figure 2. Two-stage framework for generalization to new actions through action representations. (1) For each available action, a hierarchical VAE module encodes the action observations into action representations and is trained with a reconstruction objective. (2) The policy π_θ encodes the state with state encoder $f_\omega(s)$ and pairs it with each action representation using the utility function f_ν. The utility scores are computed for each action and output to a categorical distribution. The auxiliary network takes the encoded state and outputs environment-specific auxiliary actions such as tool placement in CREATE. The policy architecture is trained with policy gradients.

are defined by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$ of states, actions, transition probability, reward function, and discount factor. At each time step t in an episode, the agent receives a state observation $s_t \in \mathcal{S}$ from the environment and responds with an action $a_t \in \mathcal{A}$. This results in a state transition to s_{t+1} and a state-conditioned reward $\mathcal{R}(s_{t+1})$. The objective of the agent is to maximize the expected discounted reward $R = \sum_{t=1}^{T} \gamma^{t-1} \mathcal{R}(s_t)$ in an episode of length T.

3.2. Generalization to New Actions

The setting of generalization to new actions consists of two phases: training and evaluation. During training, the agent learns to solve tasks with a given set of actions $\mathbb{A} = \{a_1, \ldots, a_N\}$. During each evaluation episode, the trained agent is evaluated on a new action set $\mathcal{A}$ sampled from a set of unseen actions $\mathbb{A}'$. The objective is to learn a policy $\pi(a|s, \mathcal{A})$, which maximizes the expected discounted reward using any given action set $\mathcal{A} \subset \mathbb{A}'$,

$$R = \mathbb{E}_{\mathcal{A} \subset \mathbb{A}',\, a \sim \pi(a|s,\mathcal{A})} \left[\sum_{t=1}^{T} \gamma^{t-1} \mathcal{R}(s_t) \right]. \quad (1)$$

For each action $a \in \mathbb{A} \cup \mathbb{A}'$, the set of acquired action observations is denoted with $\mathcal{O} = \{o_1, \ldots, o_n\}$. Here, each $o_j \in \mathcal{O}$ is an observation for the action like a state-trajectory, a video, or an image, indicating the action's behavior. For the set of training actions $\mathbb{A}$, we denote the set of associated actions observations as $\mathbb{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$.

4. Approach

Our approach for generalization to new actions is based on the intuition that humans make decisions from new options by exploiting prior knowledge about the options (Gershman & Niv, 2015). First, we infer the properties of each action from the action observations given as prior knowledge. Second, a policy learns to make decisions based on these inferred action properties. When a new action set is given, their properties are inferred and exploited by the policy to solve the task. Formally, we propose a two-stage framework:

1. **Learning Action Representations**: We use unsupervised representation learning to encode each set of action observations into an action representation. This representation expresses the latent action properties present in the set of diverse observations (Section 4.1).
2. **Learning Generalizable Policy**: We propose a flexible policy architecture to incorporate action representations as inputs, which can be trained through RL (Section 4.2). We provide a training procedure to control overfitting to the training action set, making the policy generalize better to unseen actions (Section 4.3).

4.1. Unsupervised Learning of Action Representations

Our goal is to encode each set of action observations into an action representation that can be used by a policy to make decisions in a task. The main challenge is to extract the shared statistics of the action's behavior from high-dimensional and diverse observations.

To address this, we employ the hierarchical variational autoencoder (HVAE) by Edwards & Storkey (2017). HVAE first summarizes the entire set of an action's observations into a single action latent. This action latent then conditions the encoding and reconstruction of each constituent observation through a conditional VAE. Such hierarchical conditioning ensures that the observations for the same action are organized together in the latent space. Furthermore, the action latent sufficiently encodes the diverse statistics of the action. Therefore, this action latent is used as the action's representation in the downstream RL task (Figure 2).

Formally, for each training action $a_i \in \mathbb{A}$, HVAE encodes its associated action observations $\mathcal{O}_i \in \mathbb{O}$ into a representation c_i by mean-pooling over the individual observations $o_{i,j} \in \mathcal{O}_i$. We refer to this action encoder as the action representation module $q_\phi(c_i|\mathcal{O}_i)$. The action latent c_i sampled from the action encoder is used to condition the encoders $q_\psi(z_{i,j}|o_{i,j}, c_i)$ and decoders $p(o_{i,j}|z_{i,j}, c_i)$ for each individual observation $o_{i,j} \in \mathcal{O}_i$. The entire HVAE framework is trained with reconstruction loss across the individual observations, along with KL-divergence regularization of encoders q_ϕ and q_ψ with their respective prior distributions $p(c)$ and $p(z|c_i)$. For additional details on HVAE, refer to Appendix D.3.1 and Edwards & Storkey (2017). The final training objective requires maximizing the ELBO:

$$\mathcal{L} = \sum_{\mathcal{O}\in\mathbb{O}} \Big[\mathbb{E}_{q_\phi(c|\mathcal{O})}\Big[\sum_{o\in\mathcal{O}} \mathbb{E}_{q_\psi(z|o,c)} \log p(o|z,c) - D_{KL}(q_\psi||p(z|c))\Big] - D_{KL}(q_\phi||p(c))\Big]. \tag{2}$$

For action observations consisting of sequential data, $o = \{x_0, \ldots, x_m\}$ like state trajectories or videos, we augment HVAE to extract temporally extended behaviors of actions. We accomplish this by incorporating insights from trajectory autoencoders (Wang et al., 2017; Co-Reyes et al., 2018) in HVAE. Bi-LSTM (Schuster & Paliwal, 1997) is used in the encoders and LSTM is used as the decoder $p(x_1, \ldots, x_m|z, c, x_0)$ to reconstruct the trajectory given the initial state x_0. Explicitly for video observations, we also incorporated temporal skip connections (Ebert et al., 2017) by predicting an extra mask channel to balance contributions from the predicted and first frame of the video.

We set the representation for an action as the mean of the inferred distribution $q_\phi(c_i|\mathcal{O}_i)$ as done in Higgins et al. (2017a); Steenbrugge et al. (2018).

4.2. Adaptable Policy Architecture

To enable decision-making with new actions, we develop a policy architecture that can adapt to any available action set $\mathcal{A}$ by taking the list of action representations as input. Since the action representations are learned independently of the downstream task, a task-solving policy must learn to extract the relevant task-specific knowledge.

The policy $\pi(a|s, \mathcal{A})$ receives a set of available actions $\mathcal{A} = \{a_1, \ldots, a_k\}$ as input, along with the action representations$\{c_1, \ldots, c_k\}$. As shown in Figure 2, the policy architecture starts with a state encoder f_ω. The utility function f_ν is applied to each given action's representation c_i and the encoded state $f_\omega(s)$ (Eq. 3). The utility function estimates the score of an action at the current state, through its action representation, just like a Q-function (Watkins & Dayan, 1992). Action utility scores are converted into a probability distribution through a softmax function:

$$\pi(a_i|s, \mathcal{A}) = \frac{e^{f_\nu[c_i, f_\omega(s)]}}{\sum_{j=1}^{k} e^{f_\nu[c_j, f_\omega(s)]}}. \tag{3}$$

In many physical environments, the choice of a discrete action is associated with auxiliary parameterizations, such as the intended position of tool usage or a binary variable to determine episode termination. We incorporate such hybrid action spaces (Hausknecht & Stone, 2015), through an auxiliary network f_χ, which takes the encoded state and outputs a distribution over the auxiliary actions [1]. An environment action is taken by sampling the auxiliary action from this distribution and the discrete action from Eq. 3. The policy parameters $\theta = \{\nu, \omega, \chi\}$ are trained end-to-end using policy gradients (Sutton et al., 2000).

Algorithm 1. Two-stage Training Framework

Inputs: Training actions $\mathbb{A}$, action observations $\mathbb{O}$
Randomly initialize HVAE and policy parameters
for epoch = 1, 2, ... **do**
  Sample batch of action observations $\mathcal{O}_i \sim \mathbb{O}$
  Train HVAE parameters with gradient ascent on Eq. 2
end for
Infer action representations: $c_i = q^\mu_\phi(\mathcal{O}_i), \forall a_i \in \mathbb{A}$
for iteration = 1, 2, ... **do**
  while episode not done **do**
    Subsample action set $\mathcal{A} \subset \mathbb{A}$ of size m
    Sample action $a_t \sim \pi_\theta(s, \mathcal{A})$ using Eq. 3
    $s_{t+1}, r_t \leftarrow \text{ENV}(s_t, a_t)$
    Store experience (s_t, a_t, s_{t+1}, r_t) in replay buffer
  end while
  Update and save policy θ using PPO on Eq. 4
end for
Select θ with best validation performance

Algorithm 2. Generalization to New Actions

Inputs: New actions $\mathcal{A} = \{a_1, \ldots a_M\}$, observations $\{\mathcal{O}_1, \ldots \mathcal{O}_M\}$. Trained networks q_ϕ and π_θ
Infer action representations: $c_i = q^\mu_\phi(\mathcal{O}_i), \forall a_i \in \mathcal{A}$
while not done **do**
  Sample action $a_t \sim \pi_\theta(s, \mathcal{A})$ using Eq. 3
  $s_{t+1}, r_t \leftarrow \text{ENV}(s_t, a_t)$
end while

[1] Alternatively, the auxiliary network can take the discrete selection as input as tested in Appendix C.4

4.3. Generalization Objective and Training Procedure

Our final objective is to find policy parameters θ to maximize reward on held-out action sets $\mathcal{A} \subset \mathbb{A}'$ (Eq. 1), while being trained on a limited set of actions $\mathbb{A}$. We study this generalization problem based on statistical learning theory (Vapnik, 1998; 2013) in supervised learning. Particularly, generalization of machine learning models is expected when their training inputs are independent and identically distributed (Bousquet et al., 2003). However, in RL, a policy typically acts in the environment to collect its own training data. Thus when a policy overexploits a specific subset of the training actions, this skews the policy training data towards those actions. To avoid this form of overfitting and be robust to diverse new action sets, we propose the following regularizations to approximate i.i.d. training:

- **Subsampled action spaces**: To limit the actions available in each episode of training, we randomly subsample action sets, $\mathcal{A} \subset \mathbb{A}$ of size m, a hyperparameter. This avoids overfitting to any specific actions by forcing the policy to solve the task with diverse action sets.
- **Maximum entropy regularization**: We further diversify the policy's actions during training using the maximum entropy objective (Ziebart et al., 2008). We add the entropy of the policy $\mathcal{H}[\pi_\theta(a|s)]$ to the RL objective with a hyperparameter weighting β. While this objective has been widely used for exploration, we find it useful to enable generalization to new actions.
- **Validation-based model selection**: During training, the models are evaluated on held-out validation sets of actions, and the best performing model is selected. Just like supervised learning, this helps to avoid overfitting the policy during training. Note that the validation set is also used to tune hyperparameters such as entropy coefficient β and subsampled action set size m. There is no overlap between test and validation sets, hence the test actions are still completely unseen at evaluation.

The final policy training objective is:

$$\max_\theta \mathbb{E}_{\mathcal{A} \subset \mathbb{A}, a \sim \pi_\theta(.|s,\mathcal{A})} [R(s) + \beta \mathcal{H}[\pi_\theta(a|s,\mathcal{A})]]. \quad (4)$$

The training procedure is described in Algorithm 1. The HVAE is trained using RAdam optimizer (Liu et al., 2019), and we use PPO (Schulman et al., 2017) to train the policy with Adam Optimizer (Kingma & Ba, 2015). Additional implementation and experimental details, including the hyperparameters searched, are provided in Appendix D. The inference process is described in Algorithm 2. When given a new set of actions, we can infer the action representations with the trained HVAE module. The policy can also generalize to utilize these actions since it has learned to map a list of action representations to an action probability distribution.

5. Experimental Setup

5.1. Environments

We propose four sequential decision-making environments with diverse actions to evaluate and benchmark the proposed problem of generalization to new actions. These test the action representation learning method on various types of action observations. The long-horizon nature of the environments presents a challenge to use new actions correctly to solve the given tasks consistently. Figure 3 provides an overview of the task, types of actions, and action observations in three environments. In each environment, the train-test-validation split is approximately 50-25-25%. Complete details on each environment, action observations, and train-validation-test splits can be found in Appendix A.

5.1.1. Grid World

In the Grid world environment (Chevalier-Boisvert et al., 2018), an agent navigates a 2D lava maze to reach a goal using predefined skills. Each skill is composed of a 5-length sequence of left, right, up or down movement. The total number of available skills is 4^5. Action observations consist of state sequences of an agent observed by applying the skill in an empty grid. This environment acts as a simple demonstration of generalization to unseen skill sets.

5.1.2. Recommender System

The Recommender System environment (Rohde et al., 2018) simulates users responding to product recommendations. Every episode, the agent makes a series of recommendations for a new user to maximize their click-through rate (CTR). With a total of 10,000 products as actions, the agent is evaluated on how well it can recommend previously unseen products to users. The environment specifies predefined action representations. Thus we only evaluate our policy framework on it, not the action encoder.

5.1.3. CREATE

We develop the Chain REAction Tool Environment (CREATE) as a challenging benchmark to test generalization to new actions[2]. It is a physics-based puzzle where the agent must place tools in real-time to manipulate a specified ball's trajectory to reach a goal position (Figure 3). The environment features 12 different tasks and 2,111 distinct tools. Moreover, it tests physical reasoning since every action involves selecting a tool and predicting the 2D placement for it, making it a hybrid action-space environment. Action observations for a tool consist of a test ball's trajectories interacting with the tool from various directions and speeds. CREATE tasks evaluate the ability to understand complex

[2]CREATE environment: https://clvrai.com/create

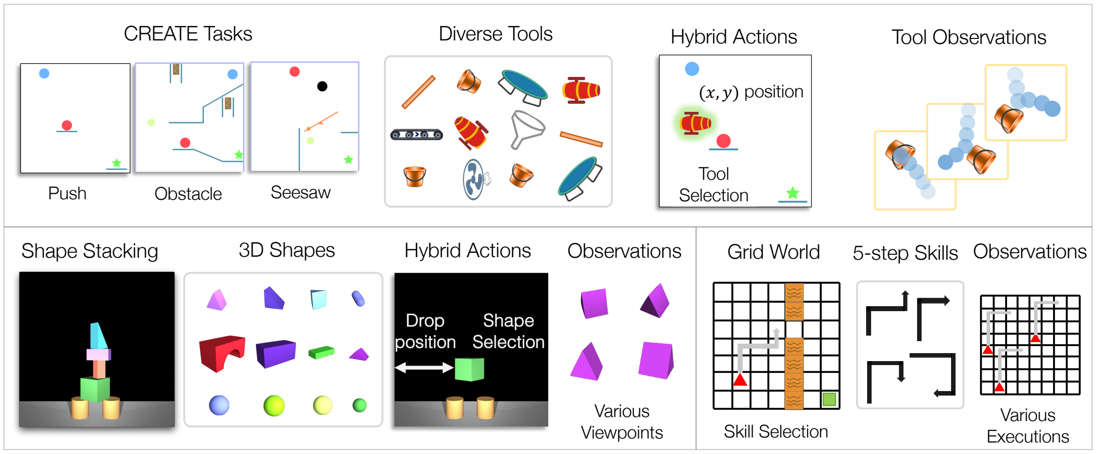


Figure 3. Benchmark environments for evaluating generalization to new actions. (Top) In CREATE, an agent selects and places various tools to move the red ball to the goal. Other moving objects can serve as help or obstacles. Some tasks also have subgoals to help with exploration (Appendix C.3 shows results with no subgoal rewards). The tool observations consist of trajectories of a test ball interacting with the tool. (Left) In Shape Stacking, an agent selects and places 3D shapes to stack a tower. The shape observations are images of the shape from different viewpoints. (Right) In Grid World, an agent reaches the goal by choosing from 5-step navigation skills. The skill observations are collected on an empty grid in the form of agent trajectories resulting from skill execution from random locations.

functionalities of unseen tools and utilize them for various tasks. We benchmark our framework on all 12 CREATE tasks with the extended results in Appendix C.4.

5.1.4. Shape Stacking

We develop a MuJoCo-based (Todorov et al., 2012) Shape Stacking environment, where the agent drops blocks of different shapes to build a tall and stable tower. Like in CREATE, the discrete selection of shape is parameterized by the coordinates of where to place the selected shape and a binary action to decide whether to stop stacking. This environment evaluates the ability to use unseen complex 3D shapes in a long horizon task and contains 810 shapes.

5.2. Experiment Procedure

We perform the following procedure for each action generalization experiment[3].

1. *Collect action observations* for all the actions using a supplemental play environment that is task-independent.
2. *Split the actions* into train, validation, and test sets.
3. *Train HVAE* on the train action set by autoencoding the collected action observations.
4. *Infer action representations* for all the actions using the trained HVAE encoder on their action observations.
5. *Train policy* on the task environment with RL. In each episode, an action set is randomly sampled from the train actions. The policy acts by using the list of inferred action representations as input.
6. **Evaluation**: In each episode, an action set is subsampled from the test (or validation) action set. The trained policy uses the inferred representations of these actions to act in the environment zero-shot. The performance metric (e.g. success rate) is averaged over multiple such episodes.
   (a) Perform hyperparameter tuning and model selection by evaluating on the *validation action set*.
   (b) Report final performance on the *test action set*.

[3]Complete code available at https://github.com/clvrai/new-actions-rl

5.3. Baselines

We validate the design choices of the proposed action encoder and policy architecture. For action encoder, we compare with a policy using action representations from a non-hierarchical encoder. For policy architecture, we consider alternatives that select actions using distances in the action representation space instead of learning a utility function.

- **Non-hierarchical VAE**: A flat VAE is trained over the individual action observations. An action's representation is taken as the mean of encodings of the constituent action observations.
- **Continuous-output**: The policy architecture outputs a continuous vector in the action representation space, following Dulac-Arnold et al. (2015). From any given action set, the action closest to this output is selected.
- **Nearest-Neighbor**: A standard discrete action policy is trained. The representation of this policy's output action is used to select the nearest neighbor from new actions.

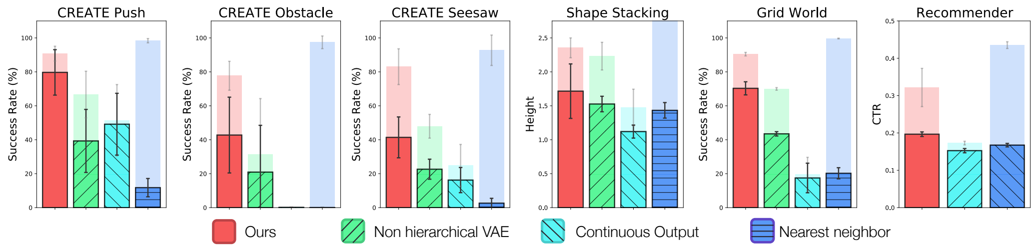


Figure 4. Comparison against baseline action representation and policy architectures on 6 environments, 3 of which are CREATE tasks. The solid bar denotes the test performance and the transparent bar the training performance, to observe the generalization gap. The results are averaged over 5000 episodes across 5 random seeds, and the error bars indicate the standard deviation (8 seeds for Grid World). All learning curves are present in Appendix C.5. Results on 9 additional CREATE tasks can be found in Appendix C.1.

5.4. Ablations

We individually ablate the two proposed regularizations:

- **Ours without subsampling**: Trained over the entire set of training actions without any action space sampling.
- **Ours without entropy**: Trained without entropy regularization, by setting the entropy coefficient to zero.

6. Results and Analysis

Our experiments aim to answer the following questions about the proposed problem and framework: (1) Can the HVAE extract meaningful action characteristics from the action observations? (2) What are the contributions of the proposed action encoder, policy architecture, and regularizations for generalization to new actions? (3) How well does our framework generalize to varying difficulties of test actions and types of action observations? (4) How inefficient is finetuning to a new action space as compared to zero-shot generalization?

6.1. Visualization of Inferred Action Representations

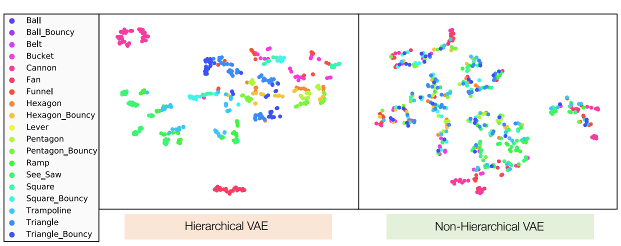


Figure 5. t-SNE visualization of action representations for held-out tools in CREATE inferred using a trained HVAE (left) and a VAE (right). The color indicates the tool class (e.g. cannons, buckets). The HVAE encoder learns to organize semantically similar tools together, in contrast to the flat VAE, which shows less structure.

To investigate if the HVAE can extract important characteristics from observations of new actions, we visualize the inferred action representations for unseen CREATE tools. In Figure 5, we observe that tools from the same class are clustered together in the HVAE representations. Whereas in the absence of hierarchy, the action representations are less organized. This shows that encoding action observations independently, and averaging them to obtain a representation can result in the loss of semantic information, such as the tool's class. In contrast, hierarchical conditioning on action representation enforces various constituent observations to be encoded together. This helps to model the diverse statistics of the action's observations into its representation.

6.2. Results and Comparisons

6.2.1. Baselines

Figure 4 shows that our framework outperforms the baselines (Section 5.3) in zero-shot generalization to new actions on six tasks. The non-hierarchical VAE baseline has lower policy performance in both training and testing. This shows that HVAE extracts better representations from action observations that facilitate easier policy learning.

The continuous-output baseline suffers in training as well as testing performance. This is likely due to the complex task of indirect action selection. The distance metric used to find the closest action does not directly correspond to the task relevance. Therefore the policy network must learn to adjust its continuous output, such that the desired discrete action ends up closest to it. Our method alleviates this through the utility function, which first extracts task-relevant features to enable an appropriate action decision. The nearest-neighbor baseline achieves high training performance since it is merely discrete-action RL with a fixed action set. However, at test time, the simple nearest-neighbor in action representation space does not correspond to the actions' task-relevance. This results in poor generalization performance.

6.2.2. Ablations

Figure 6 assesses the contribution of the proposed regularizations to avoid overfitting to training actions. Entropy regularization usually leads to better training and test performance due to better exploration. In the recommender envi-

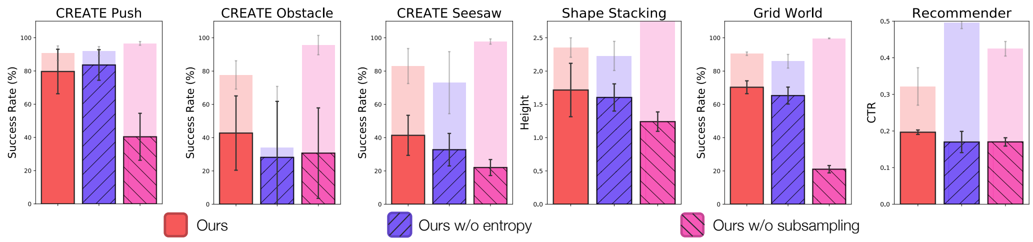
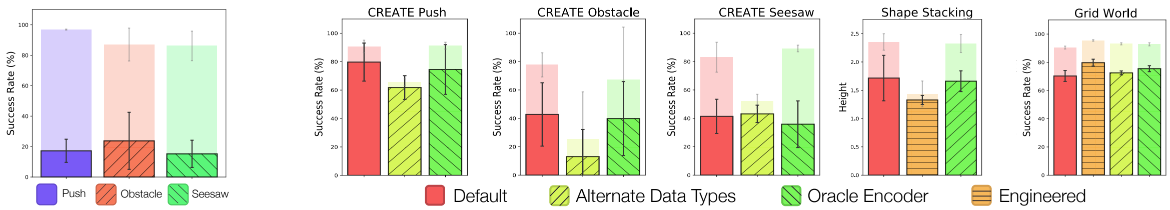


Figure 6. Analyzing the importance of the proposed action space subsampling and entropy regularization in our method. The training and evaluation details are the same as Figure 4.



(a) Unseen tool classes in CREATE

(b) Alternate action representations

Figure 7. Additional analyses. (a) Our method achieves decent performance on out-of-distribution tools in 3 CREATE tasks, but the generalization gap is more pronounced. (b) Various action representations can be successfully used with our policy architecture.

ronment, the generalization gap is more substantial without entropy regularization. Without any incentive to diversify, the policy achieved high training performance by overfitting to certain products. We observe a similar effect in the absence of action subsampling across all tasks. It achieves a higher training performance, due to the ease of training in non-varying action space. However, its generalization performance is weak because it is easy to overfit when the policy has access to all the actions during training.

6.3. Analyzing the Limits of Generalization

6.3.1. Generalization to Unseen Action Classes

Our method is expected to generalize when new actions are within the distribution of those seen during training. However, what happens when we test our approach on completely unseen action classes? Generalization is still expected because the characteristic action observations enable the representation of actions in the same space. Figure 7a evaluates our approach on held-out tool classes in the CREATE environment. Some tool classes like trampolines and cannons are only seen during training, whereas others like fans and conveyor belts are only used during testing. While the generalization gap is more substantial than before, we still observe reasonable task success across the 3 CREATE tasks. The performance can be further improved by increasing the size and diversity of training actions. Appendix C.6 shows a similar experiment on Shape Stacking.

6.3.2. Alternate Action Representations

In Figure 7b, we study policy performance for various action representations. See Appendix B for t-SNE visualizations.

- **Alternate Data Types** of action observations are used to learn representations. For CREATE, we use video data instead of the state trajectory of the test ball (see Figure 3). For Grid World, we test with a one-hot vector of agent location instead of (x, y) coordinates. The policy performance using these representations is comparable to the default. This shows that HVAE is suitable for high-dimensional action observations, such as videos.
- **Oracle** HVAE is used to get representations by training on the test actions. The performance difference between default and oracle HVAE is negligible. This shows that HVAE generalizes well to unseen action observations.
- **Hand-Engineered** action representations are used for Stacking and Grid World, by exploiting ground-truth information about the actions. In Stacking, HVAE outperforms these representations, since it is hard to specify the information about shape geometry manually. In contrast, it is easy to specify the complete skill in Grid World. Nevertheless, HVAE representations perform comparably.

6.3.3. Varying the Difficulty of Generalization

Figure 9 shows a detailed study of generalization on various degrees of differences between the train and test actions in 3 CREATE tasks. We vary the following parameters:

a) **Tool Angle**: Each sampled test tool is at least θ degrees different from the most similar tool seen during training.

b) **Tool Embedding**: Each test tool's representation is at least d Euclidean distance away from each training tool.

c) **Unseen Ratio**: The test action set is a mixture of seen and unseen tools, with $x\%$ unseen.

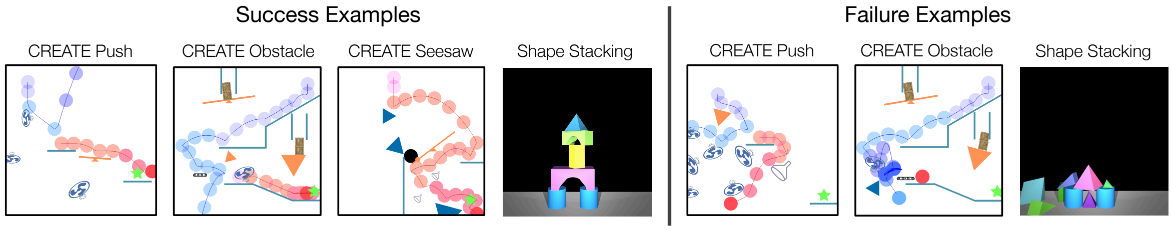
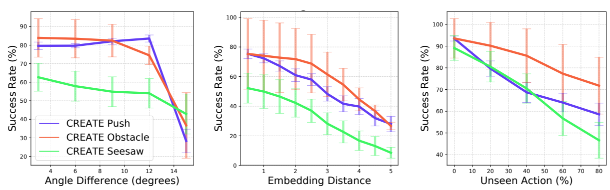


Figure 8. Evaluation results showing the trajectories of objects in CREATE and the final tower in Shape Stacking. Our framework is generally able to infer the dynamic properties of tools and geometry of shapes and subsequently use them to make the right decisions.



(a) Tool Angle (b) Tool Embedding (c) Unseen Ratio

Figure 9. Varying the test action space. An increasing x-axis corresponds to more difficult generalization conditions. Each value plotted is the average test performance over 5 random seeds with the error bar corresponding to the standard deviation.

The results suggest a gradual decrease in generalization performance as the test actions become more different from training actions. We chose the hardest settings for the main experiments: 15° angle difference and 100% unseen actions.

6.3.4. Qualitative Analysis

Figure 8 shows success and failure examples when using unseen actions in the CREATE and Stacking environments. In CREATE, our framework correctly infers the directional pushing properties of unseen tools like conveyor belts and fans from their action observations and can utilize them to solve the task. Failure examples include placements being off and misrepresenting the direction of a belt. Collecting more action observations can improve the representations.

In Shape Stacking, the geometric properties of 3D shapes are correctly inferred from image action observations. The policy can act in the environment by selecting the appropriate shapes to drop based on the current tower height. Failures include greedily selecting a tall but unstable shape in the beginning, like a pyramid.

6.4. The Inefficiency of Finetuning on New Actions

In Figure 10, we examine various approaches to continue training on a particular set of new actions in CREATE Push. First, we train a policy from scratch on the new actions either with our adaptable policy architecture (Ours Scratch) or a regular discrete policy (Discrete Scratch). These take around 3 million environment steps to achieve our pretrained method's zero-shot performance (Ours Zero-Shot). Next, we consider ways to transfer knowledge from training actions. We train a regular discrete policy and finetune on new

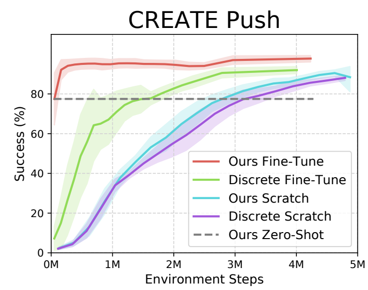


Figure 10. Finetuning or training policies from scratch on the new action space. The horizontal line is the zero-shot performance of our method. Each line is the average test performance over 5 random seeds, while the shaded region is the standard deviation.

actions by re-initializing the final layer (Discrete Fine-Tune). While this approach transfers some task knowledge, it disregards any relationship between the old and new actions. It still takes over 1 million steps to reach our zero-shot performance. This shows how expensive retraining is on a single action set. Clearly, this retraining process is prohibitive in scenarios where the action space frequently changes. This demonstrates the significance of addressing the problem of zero-shot generalization to new actions. Finally, we continue training our pretrained policy on the new action set with RL (Ours Fine-Tune). We observe fast convergence to optimal performance, because of its ability to utilize action representations to transfer knowledge from the training actions to the new actions. Finetuning results for all other environments are in Appendix C.2.

7. Conclusion

Generalization to novel circumstances is vital for robust agents. We propose the problem of enabling RL policies to generalize to new action spaces. Our two-stage framework learns action representations from acquired action observations and utilizes them to make the downstream RL policy flexible. We propose four challenging benchmark environments and demonstrate the efficacy of hierarchical representation learning, policy architecture, and regularizations. Exciting directions for future research include building general problem-solving agents that can adapt to new tasks with new action spaces, and autonomously acquiring informative action observations in the physical world.

Acknowledgements

This project was funded by SKT. The authors are grateful to Youngwoon Lee and Jincheng Zhou for help with RL experiments and writing. The authors would like to thank Shao-Hua Sun, Karl Pertsch, Dweep Trivedi and many members of the USC CLVR lab for fruitful discussions. The authors appreciate the feedback from anonymous reviewers who helped improve the paper.

References

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175, 2017. 2

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013. 2

Bousquet, O., Boucheron, S., and Lugosi, G. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pp. 169–207. Springer, 2003. 5

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. S. Learning action representations for reinforcement learning. *arXiv preprint arXiv:1902.00183*, 2019. 2

Chandak, Y., Theocharous, G., Nota, C., and Thomas, P. Lifelong learning with a changing action set. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3373–3380, Apr 2020. ISSN 2159-5399. doi: 10.1609/aaai.v34i04.5739. URL http://dx.doi.org/10.1609/aaai.v34i04.5739. 2

Chen, Y., Chen, Y., Yang, Y., Li, Y., Yin, J., and Fan, C. Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291*, 2019. 2

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018. 5

Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pp. 1008–1017, 2018. 4

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018. 1, 2

Denton, E. L. and Birodkar, v. Unsupervised learning of disentangled representations from video. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4414–4423. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7028-unsupervised-learning-of-disentangled-representations-from-video.pdf. 2

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015. 2, 6

Ebert, F., Finn, C., Lee, A. X., and Levine, S. Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning*, pp. 344–356, 2017. 4

Edwards, H. and Storkey, A. Towards a neural statistician. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HJDBUF5le. 2, 3, 4

Fang, K., Zhu, Y., Garg, A., Kurenkov, A., Mehta, V., Fei-Fei, L., and Savarese, S. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *arXiv preprint arXiv:1806.09266*, 2018. 2

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017. 1

Gershman, S. J. and Niv, Y. Novelty and inductive generalization in human reinforcement learning. *Topics in cognitive science*, 7(3):391–415, 2015. 3

Hausknecht, M. and Stone, P. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015. 4

Hawkins, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004. 2

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*, 2015. 2

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017a. 4

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1480–1490. JMLR. org, 2017b. 2

Kim, H., Kim, J., Jeong, Y., Levine, S., and Song, H. O. EMI: Exploration with mutual information. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3360–3369, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/kim19a.html`. 2

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 5

Laversanne-Finot, A., Pere, A., and Oudeyer, P.-Y. Curiosity driven exploration of learned disentangled goal spaces. In Billard, A., Dragan, A., Peters, J., and Morimoto, J. (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 487–504. PMLR, 29–31 Oct 2018. URL `http://proceedings.mlr.press/v87/laversanne-finot18a.html`. 2

Legg, S. and Hutter, M. Universal intelligence: A definition of machine intelligence. *Minds and machines*, 17(4): 391–444, 2007.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 5

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018. 2

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018. 1, 2

Oh, J., Singh, S., Lee, H., and Kohli, P. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pp. 2661–2670, 2017. 2

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018. 2

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *arXiv preprint arXiv:1802.07569*, 2018. 1

Pathak, D., Lu, C., Darrell, T., Isola, P., and Efros, A. A. Learning to control self-assembling morphologies: A study of generalization via modularity. In *arXiv preprint arXiv:1902.05546*, 2019. 1, 2

Rohde, D., Bonner, S., Dunlop, T., Vasile, F., and Karatzoglou, A. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018. 5

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, 2018. 2

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 4

Steenbrugge, X., Leroux, S., Verbelen, T., and Dhoedt, B. Improving generalization for abstract reasoning tasks using disentangled feature representations. *arXiv preprint arXiv:1811.04784*, 2018. 2, 4

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000. 4

Tennenholtz, G. and Mannor, S. The natural language of actions. *arXiv preprint arXiv:1902.01119*, 2019. 2

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. 6

Vapnik, V. Statistical learning theory, 1998. 5

Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 2013. 5

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1sqHMZCb`. 1, 2

Wang, Z., Merel, J. S., Reed, S. E., de Freitas, N., Wayne, G., and Heess, N. Robust imitation of diverse behaviors. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp.

5320–5329. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7116-robust-imitation-of-diverse-behaviors.pdf. 4

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 4

Xie, A., Ebert, F., Levine, S., and Finn, C. Improvisation through physical understanding: Using novel objects as tools with visual foresight, 2019. 2

Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., and Savarese, S. Neural task programming: Learning to generalize across hierarchical tasks. In *International Conference on Robotics and Automation*, 2017. 2

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pp. 1433–1438. AAAI Press, 2008. 5