Equivariant Q Learning in Spatial Action Spaces

Anonymous Author(s)
Affiliation
Address
email

Abstract: Recently, a variety of new equivariant neural network model architectures have been proposed that generalize better over rotational and reflectional symmetries than standard models. These models are relevant to robotics because many robotics problems can be expressed in a way that makes them rotationally symmetric. This paper focuses on equivariance over a visual state space and a spatial action space – the setting where the robot action space includes a subset of $\mathrm{SE}(2)$. In this situation, we know a priori that rotations and translations in the state image should result in the same rotations and translations in the spatial action dimensions of the optimal policy. Therefore, it should be possible to use equivariant model architectures to make Q learning more sample efficient. This paper identifies conditions under which the optimal Q function is indeed equivariant and proposes Q function model architectures for this setting. We show experimentally that this approach is more sample efficient than standard methods and that we can solve challenging robotics problems more quickly and with higher performance.

Keywords: Reinforcement Learning, Equivariance, Manipulation

1 Introduction

A key question in policy learning for robotics is how to leverage structure present in the robot and the world to improve learning. This paper focuses on a fundamental type of structure present in visuo-motor policy learning for most robotics problems: translational and rotational invariance with respect to camera viewpoint. Specifically, the reward and transition dynamics of most robotics problems can be expressed in a way that is invariant with respect to the camera viewpoint from which the agent observes the scene. In spite of the above, most visuo-motor policy learning agents do not leverage this invariance in camera viewpoint. The agent's value function or policy typically considers different perspectives on the same scene to be different world states. A popular way to combat this problem is through visual data augmentation, i.e., to create additional samples or experiences by randomly translating and rotating observed images [1] but keeping the same labels. This can be used in conjunction with a contrastive term in the loss function which helps the system learn an invariant latent representation [2, 3]. While these methods can improve generalization, they involve creating large datasets and they generally make policy learning harder rather than simpler.

Our key idea in this paper is to model rotational and translation invariance in policy learning using neural network model architectures that are equivariant over finite subgroups of $\mathrm{SE}(2)$. These equivariant model architectures reduce the number of free parameters that must be learned using steerable convolutional layers [4]. Instead of slowing learning by enlarging the dataset or adding contrastive loss terms, this approach can significantly improve sample efficiency. Moreover, it enables us to generalize in a very precise way: everything learned with respect to one camera viewpoint is automatically also represented in other camera perspectives via selectively tied parameters in the model architecture. We focus our work on Q learning in spatial action spaces, where the agent's action space spans $\mathrm{SE}(2)$ or $\mathrm{SE}(3)$. We make the following contributions. First, we identify the conditions under which the optimal Q function is $\mathrm{SE}(2)$ equivariant, and equivariant Q learning is therefore applicable. Second, we propose neural network model architectures that encode $\mathrm{SE}(2)$ equivariance in the Q function. Third, since most policy learning problems are only equivariant in *some* of the state variables, we propose partially equivariant model architectures that can accommodate this. Finally, we compare equivariant models against non-equivariant counterparts in the context of several

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

robotic manipulation problems. The results show that equivariant models are more sample efficient than non-equivariant models, often by a significant margin.

2 Related Work

Data Augmentation: Data augmentation techniques have long been employed in computer vision to encode the invariance property of translation and reflection into neural networks [5, 6]. Recent work demonstrates the use of data augmentation improves the data efficiency and the policy's performance in reinforcement learning [7, 8, 9]. In the context of robotics, data augmentation is often used to generate additional samples [1, 10, 11]. Data augmentation improves the data efficiency of a system, but not the sample efficiency. In contrast to learning the equivariance property using data augmentation, our work utilizes the equivariant network to hard code the symmetries in the structure of the network to achieve better sample efficiency.

Contrastive Learning: Another approach to learning a representation that is invariant to translation and rotation is to add a contrastive learning term to the loss function [2]. This idea has been applied to reinforcement learning in general [3] and robotic manipulation in particular [12]. While this approach can help the agent learn an invariant encoding of the data, it does not necessarily improve the sample efficiency of policy learning.

Equivariant Learning: Equivariant model architectures hard-code $\mathrm{E}(2)$ symmetries into the structure of the neural network and have been shown to be useful in computer vision [13, 4, 14]. In reinforcement learning, some recent work applies equivariant model architectures to structure-finding problems involving MDP homomorphisms [15, 16]. In addition, Mondal et al. [17] recently applied an $\mathrm{E}(2)$-equivariant model to Q learning in an Atari game domain, but showed limited improvement. To our knowledge, equivariant model architectures have not been explored in the context of robotics applications.

Spatial Action Representations: Several researchers have applied policy learning in spatial action spaces to robotic manipulation. A popular approach is to do Q learning with a dense pixel action space using a fully convolutional neural network (this is the FCN approach we describe and extend in Section 4.2) [18, 19, 20, 21]. Variations on this approach have been explored in [22, 23]. The FCN approach has been adapted to a variety of different manipulation tasks with different action primitives [24, 25, 26, 27, 28, 1, 29, 30, 31]. In this paper, we extend the work above by proposing new equivariant architectures for the spatial action space setting.

3 Problem Statement

We are interested in solving complex robotic manipulation problems such as the packing and construction problems shown in Figure 3. We focus on problems expressed in a spatial action space.

Manipulation as an MDP in over a visual state space and a spatial action space: We assume that the manipulation problem is formulated as a Markov decision process (MDP): $\mathcal{M} = (S, A, T, R, \gamma)$. We focus on MDPs in visual state spaces and spatial action spaces [29, 20, 31]. The state space is factored into the state of the objects in the world, expressed as an n-channel $h \times w$ image $I \in S_{\text{world}} = \mathbb{R}^{n \times h \times w}$, and the state of the robot (including objects held by the robot) $s_{\text{rbt}} \in S_{\text{rbt}}$, expressed arbitrarily. The total state space is $S = S_{\text{world}} \times S_{\text{rbt}}$. The action space is expressed as a cross product of $\mathrm{SE}(2)$ (hence it is spatial) and a set of additional arbitrary action variables: $A = \mathrm{SE}(2) \times A_{\text{arb}}$. The spatial component of action expresses where the robot hand is to move and the additional action variables express how it should move or what it should do. For example, in the pick/place domains shown in Figure 3, $A_{\text{arb}} = \{\text{PICK}, \text{PLACE}\}$, giving the agent the ability to move to a pose and close the fingers (pick) or move and open the fingers (place). We will sometimes decompose the spatial component of action $a_{\text{sp}} \in \mathrm{SE}(2)$ into its translation and rotation components, $a_{\text{sp}} = (x, \theta)$. The goal of manipulation is to achieve a desired configuration of objects in the world, as expressed by a reward function $R : S \times A \to \mathbb{R}$.

Translation and Rotation in $\mathrm{SE}(2)$: We are interested in learning policies that are invariant to translation and rotation. To do that, we define rotation and translation of state and action as follows. Let $g \in \mathrm{SE}(2)$ be an arbitrary rotation and translation in the plane and let $s = (I, s_{\text{rbt}}) \in S_{\text{world}} \times S_{\text{rbt}}$ be a state. g operates on s by rotating and translating the image I, but leaving s_{rbt} unchanged: $gs =$

(gI, s_{rbt}), where gI denotes the image I translated and rotated by g. For action $a = (a_{\text{sp}}, a_{\text{arb}})$, g rotates and translates a_{sp} but not a_{arb}: $ga = (ga_{\text{sp}}, a_{\text{arb}})$. Notice that both S and A are closed under $g \in \text{SE}(2)$, i.e. that $\forall g \in \text{SE}(2)$, $a \in A \implies ga \in A$ and $s \in S \implies gs \in S$.

Assumptions: We assume the reward and transition dynamics of the system are invariant with respect to translation and rotation of state and action as defined above.

Assumption 3.1 (Goal Invariance)**.** *The manipulation objective is to achieve a desired configuration of objects in the world without regard to the position and orientation of the scene. That is,* $R(s, a) = R(gs, ga)$ *for all* $g \in \text{SE}(2)$.

Assumption 3.2 (Transition Invariance)**.** *The outcome of robot actions is invariant to translations and rotations of both the scene and the action. Specifically,* $T(s, a, s') = T(gs, ga, gs')$ *for all* $g \in \text{SE}(2)$.

We also assume that translation and rotation operations on state and action are invertible, i.e. for any $g \in \text{SE}(2)$ applied to a state or action, we can always obtain the original state or action by transforming it back again with g^{-1}.

Assumption 3.3 (Invertibility)**.** *Translations and rotations in state and action are invertible. That is,* $\forall g \in \text{SE}(2)$, $g^{-1}(gs) = s$ *and* $g^{-1}(ga) = a$.

Assumptions 3.1 and 3.2 are satisfied in problem settings where the objective and the transition dynamics can be expressed intrinsically to the world without reference to an external coordinate frame imposed by the system designer. These assumptions are satisfied in many manipulation domains including all those shown in Figure 3. In House Building, for example, the reward and transition dynamics of the system are independent of the coordinate frame of the image or the action space. Assumption 3.3 is needed to guarantee the Q function invariance described in the next section.

4 Approach

Assumptions 3.1, 3.2, and 3.3 imply that the optimal Q function is invariant to translations and rotations in $\text{SE}(2)$.

Proposition 4.1. *Given an MDP* $\mathcal{M} = (S, A, T, R, \gamma)$ *for which Assumptions 3.1, 3.2, and 3.3 are satisfied, the optimal Q function is invariant to translation and rotation, i.e.* $Q^*(s, a) = Q^*(gs, ga)$, *for all* $g \in \text{SE}(2)$. *(Proof in Appendix A.)*

Our key idea is to use the invariance property of Proposition 4.1 to structure Q learning (and make it more sample efficient) by defining a neural network that is hard-wired to encode only invariant Q functions. However, in order to accomplish this in the context of DQN, we must allow for the fact that state is an input to the neural network while action values are an *output*. This neural network is therefore a function $q : S \to \mathbb{R}^A$, where $\mathbb{R}^A$ denotes the space of functions $\{A \to \mathbb{R}\}$. The invariance property of Proposition 4.1 now becomes an *equivariance* property,

$$q(gs)(a) = q(s)(g^{-1}a), \tag{1}$$

where $q(s)(a)$ denotes the Q value of action a in state s. We implement this constraint using equivariant convolutional layers as described below.

4.1 Equivariant Convolutions

Equivariance over a finite group: In order to implement the equivariance constraint, it is standard in the literature to approximate $\text{SE}(2)$ by a finite subgroup [4, 14]. Recall that the spatial component of an action is $a_{\text{sp}} = (x, \theta) \in \text{SE}(2)$. We constrain position to be a discrete pair of positive integers $x \in \{1 \dots h\} \times \{1 \dots w\} \subset \mathbb{Z}^2$, corresponding to a pixel in the input image I. We constrain orientation to be a member of a finite cyclic group $\theta \in C_u$, i.e. one of u discrete orientations. For example, if $u = 8$, then $C_8 = \{0, \frac{\pi}{4}, \frac{2\pi}{4}, \frac{3\pi}{4}, \frac{4\pi}{4}, \frac{5\pi}{4}, \frac{6\pi}{4}, \frac{7\pi}{4}\}$. Our finite approximation of $a_{\text{sp}} \in \text{SE}(2)$ is $\hat{a}_{\text{sp}}$ in the subgroup $\hat{\text{SE}}(2)$ generated by translations $\mathbb{Z}^2$ and rotations C_u.

Input and output of an equivariant convolutional layer: A standard convolutional layer h takes as input an n-channel feature map and produces an m-channel map as output, $h_{\text{standard}} : \mathbb{R}^{n \times h \times w} \to \mathbb{R}^{m \times h \times w}$. We can construct an equivariant convolutional layer by adding an additional dimension

to the feature map that encodes the values for each element of a group (C_u in our case).[1] The equivariant mapping therefore becomes $h_{\text{equiv}} : \mathbb{R}^{u\times n\times h\times w} \rightarrow \mathbb{R}^{u\times m\times h\times w}$ for all layers except the first. The first layer of the network generally takes a "flat" image as input: $h^{\text{in}}_{\text{equiv}} : \mathbb{R}^{1\times n\times h\times w} \rightarrow \mathbb{R}^{u\times m\times h\times w}$.[2]

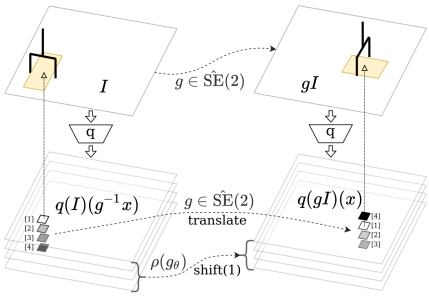


Figure 1: Illustration of Q-map equivariance when $C_u = C_4$. The output Q-map rotates and translates with the input image. The 4-vector at each pixel does a circular shift, i.e., the optimal rotation changes from 0 (the 1st element of C_4) to $\frac{\pi}{2}$ (the 2nd element of C_4)

Equivariance constraint: Let $h_i(I)(x)$ denote the output of convolutional layer h at channel i and pixel x given input I. For an equivariant layer, $h_i(I)(x) \in \mathbb{R}^{C_u}$ describes feature values for each element of C_u. For an element $g \in \hat{\text{SE}}(2)$, denote the rotational part by $g_\theta \in C_u$. If we identify functions $\mathbb{R}^{C_u}$ with vectors $\mathbb{R}^u$, then the group action of $g_\theta \in C_u$ on $\mathbb{R}^{C_u}$ becomes left multiplication by a permutation matrix $\rho(g_\theta)$ that performs a circular shift on the vector in $\mathbb{R}^u$. Then the group action of $\hat{\text{SE}}(2)$ on a feature map $h(I) \in \mathbb{R}^{u\times m\times h\times w}$ can be expressed as $g(h_i(I))(x) = \rho(g_\theta)h_i(I)(g^{-1}x)$. The mapping h is equivariant if and only if

$$h_i(gI)(x) = g(h_i(I))(x) = \rho(g_\theta)h_i(I)(g^{-1}x), \tag{2}$$

for each $i \in \{1 \dots m\}$. This is illustrated in Figure 1. We can calculate the output feature map in the lower right corner by transforming the input by g and then doing the convolution (left side of Equation 2) or by doing the convolution first and then taking the value of $g^{-1}x$ and circular-shifting the output vector (right side of Equation 2). In order to create a network that enforces the constraint of Equation 1, we can simply stack equivariant convolutions layers that each satisfy Equation 2.

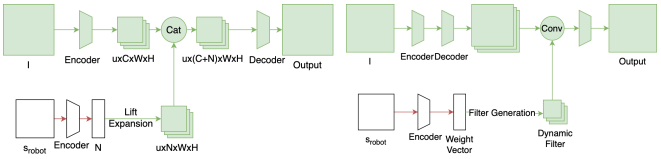
Kernel constraint: The equivariance constraint of Equation 2 can be implemented by strategically tying weights together in the convolutional kernel [4]. Since the standard convolutional kernel is already translation equivariant [13], we must only enforce rotational (C_u) equivariance [32]:

$$K(g_\theta y) = \rho_{\text{out}}(g_\theta)K(y)\rho_{\text{in}}(g_\theta)^{-1}, \tag{3}$$

where $\rho_{\text{in}}(g_\theta)$ and $\rho_{\text{out}}(g_\theta)$ are the permutation matrix of the group element g_θ (note that for the first layer, $K(y)$ will be a $1 \times u$ matrix, and $\rho_{\text{in}}(g_\theta)$ will be 1). More details are in Appendix B.

4.2 Equivariant Fully Convolutional Q Functions in $\text{SE}(2)$

A baseline approach to encoding the Q function over a spatial action space is to use a fully convolutional network (FCN) that stacks convolutional layers to produce an output Q map with the same resolution as the input image. If we ignore the non-image state variables s_{rbt} and the non-spatial action variables a_{arb}, then we have all the tools we need – we simply replace all convolutional layers with equivariant convolutions and the Q network becomes fully equivariant.



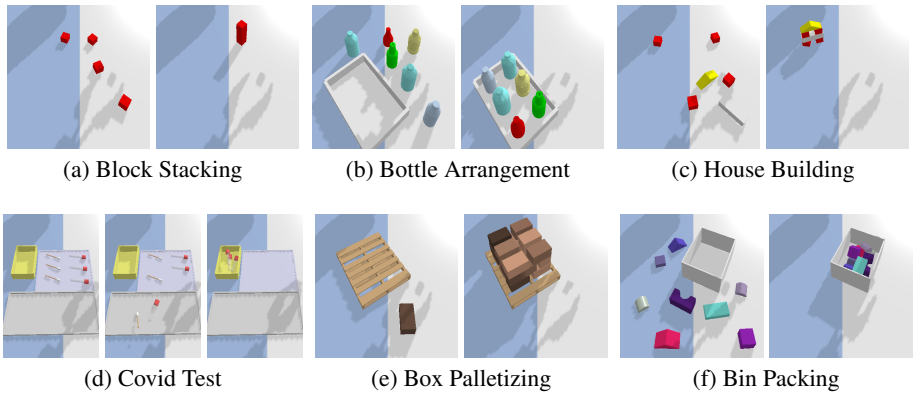
(a) Lift Expansion (b) Dynamic Filter

Figure 2: Illustration of approaches to partial equivariance. Green blocks are equivariant and white blocks are not.

Partial Equivariance: Unfortunately, in realistic robotics problems, Q function is generally not equivariant with respect to all state and action variables. For example, the non-equivariant parts of state and action in Section 3 are s_{rbt} and a_{arb}. We encode a_{arb} by simply having a separate output head for each. However, to encode s_{rbt}, we need a mechanism for inserting the non-equivariant information into the neural network model without "breaking" the equivariance property. We explored two approaches: the lift expansion approach and the dynamic filter approach. In the lift expansion, we

[1] In the language of [14], this is a steerable convolution between regular representations of C_u.

[2] This is a steerable convolution between the trivial representation and regular representation of C_u.



(a) Block Stacking (b) Bottle Arrangement (c) House Building

(d) Covid Test (e) Box Palletizing (f) Bin Packing

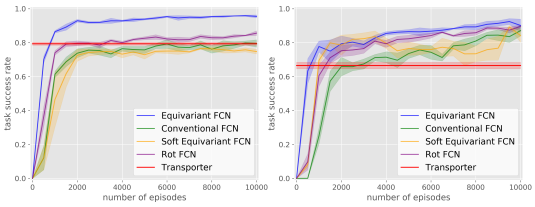
Figure 3: The experimental environments. The left image in each sub figure shows an initial state of the environment; the right image in each sub figure shows the goal state.

tile the non-equivariant information across the equivariant dimensions of the feature map as additional channels (Figure 2a). In the dynamic filter approach [33], the non-equivariant data is passed through a separate pathway that outputs the weights of an equivariant kernel that is convolved into the main equivariant backbone. We constrain this filter to be equivariant by enforcing the kernel constraint of Equation 3 (Figure 2b). We empirically find that both methods have similar performance (Appendix H.1). In the remainder of this paper, we use the dynamic filter approach because it is more memory efficient.

Encoding Gripper Symmetry Using Quotient Groups: Another symmetry that we want to leverage is the bilateral symmetry of the gripper. The outcome of a pick action performed using a two-finger gripper in orientation θ is the same as for the gripper in orientation $\theta + k\pi$ for any integer k. Similarly, it is often valid to assume that the outcome of place actions is invariant [3]. We model this invariance using the quotient group C_u/C_2. The $C_2 = \{0, \pi\}$ action equates rotations which differ by multiples of π in C_u/C_2. The steerable layer defined under the quotient group is applied with the same constraint as in Equation 3, except that the output space will be in C_u/C_2.

Experimental Domains: We evaluate the equivariant FCN approach in the Block Stacking and Bottle Arrangement tasks shown in Figure 3. In the Block Stacking task, the robot needs to stack four blocks. In the Bottle Arrangement task, the agent needs to gather six bottles in a tray. Both environments have sparse rewards (+1 at goal and 0 otherwise). The world state is encoded by a 1-channel heightmap $I \in \mathbb{R}^{1\times h\times w}$ and robot state is encoded by an image patch H that describes the contents of the robotic hand. The non-spatial action variable, $a_{\text{arb}} \in \{\text{PICK}, \text{PLACE}\}$, describes whether the robot is performing a pick or place action. a_{arb} is selected by the gripper state, i.e., $a_{\text{arb}} = \text{PLACE}$ if the gripper is holding an object, and PICK otherwise. The equivariant layers of the FCN are defined over group C_{12} where the output is with respect to the quotient group C_{12}/C_2 to encode the gripper symmetry. See Appendix C and D for more detail on the experimental domains and the FCN architecture respectively.

Experimental Comparison With Baselines: We evaluate against the following baselines: 1) Conventional FCN: FCN with 1-channel input and 6-channel output where each output channel corresponds to the Q map for a different orientation (similar to Satish et al. [19] but without the z dimension); 2) Soft Equivariant FCN: same architecture as 1), while at each training step, we augment the minibatch with three rotations every 90 degrees; 3) Rot FCN: FCN with 1-channel input and 1-channel output, the rotation is encoded by rotating the input and output



(a) (b)

Figure 4: Comparison of Equivariant FCN (blue) with baselines. Results averaged over four runs. Shading denotes standard error.

[3] Strictly speaking, this is true only when the grasped object is also symmetric.

for each θ [21]; 4) Transporter Network [1], an FCN-based architecture with the last layer being a dynamic kernel generated by a separate FCN with an input of an image crop at the pick location. More detail on the baselines is in Appendix E.1. All methods except the Transporter Network use SDQfD, an approach to imitation learning in spatial action spaces that combines a TD loss term with penalties on non-expert actions [29]. (Transporter Network is a behavior cloning method.) We use 50 steps ($\approx$ 8 episodes) of expert demonstrations in Block Stacking and 240 steps (20 episodes) in Bottle Arrangement. Those expert transitions are augmented by 9 random $\mathrm{SE}(2)$ transformations. See Appendix F.2 for more detail. Figure 4 shows the results. Our equivariant FCN outperforms all baselines in both tasks. Notice that in the Bottle Arrangement task, the equivariant network outperforms the baselines less than it does with Block Stacking. This is because the domain itself is already partially rotationally equivariant because the bottles are cylindrical and therefore our network has less of an advantage.

4.3 Equivariant Augmented State Q Functions in $\mathrm{SE}(2)$

The FCN approach does not scale well to challenging manipulation problems. Therefore, we apply the equivariance idea to the augmented state representation (ASR) method of [29] which has been shown to do better. The ASR method transforms the original MDP with a high dimensional action space into a new MDP with an augmented state space but a lower dimensional action space. Instead of encoding the value of all dimensions of action in a single neural network, this model encodes the value of different dimensions of action as separate neural networks conditioned on prior action choices.

ASR in $\mathrm{SE}(2)$: Fig 5 illustrates the idea for an $\mathrm{SE}(2)$ action space. As before, the action space is $\hat{a}_{\mathrm{sp}} = (x, \theta) \in \hat{\mathrm{SE}}(2)$. However, there are now two networks, q_1 and q_2, that are evaluated sequentially. $q_1 : \mathbb{R}^{n\times h\times w} \to \mathbb{R}^{1\times h\times w}$ is a fully convolutional network that takes an n-channel image I as input and produces a 1-channel Q map that describes the Q values for all values of x (the position component of $\hat{a}_{\mathrm{sp}}$). Given a choice of x encoded as "augmented" state (represented as an image patch $P = \textsc{crop}(I, x) \in \mathbb{R}^{n\times h'\times w'}$ cropped from I and centered at x), the $q_2 : \mathbb{R}^{n\times h'\times w'} \to \mathbb{R}^{u}$ network outputs Q values for the u different values of the orientation component of action. These two networks are used together for both action selection and evaluation of target values during learning. We evaluate $x^* = \arg\max_x q_1(I)$, calculate $P = \textsc{crop}(I, x^*)$, and then evaluate $\theta^* = \arg\max_\theta q_2(P)$.

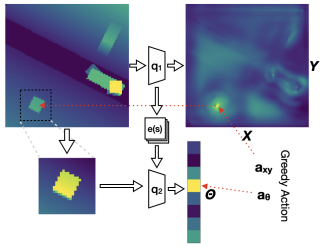
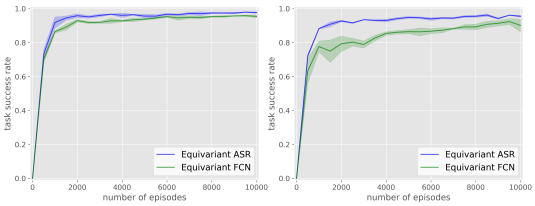


Figure 5: Illustration of the ASR approach in $\mathrm{SE}(2)$.

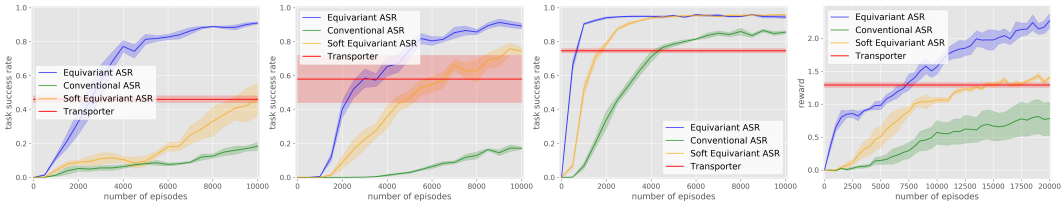
Equivariant architecture for ASR in $\mathrm{SE}(2)$: We decompose the $\mathrm{SE}(2)$ equivariance property of Equation 1 into two equivariance properties for q_1 and q_2, respectively: $q_1(gI)(x) = q_1(I)(g^{-1}x)$ where $g \in \hat{\mathrm{SE}}(2)$, and $q_2(g_\theta P) = \rho(g_\theta) q_2(P)$ where $g_\theta \in C_u$. The equivariance property of q_1 is similar to that of Equation 2 except that q_1 only has one channel output that represents the value of the optimal rotation for each pixel (modeled using the trivial representation). This means we can rewrite the q_1 equivariance property as $q_1(gI) = g q_1(I)$, where g on the RHS of this equation translates and rotates the output Q map. In practice, we obtained the best performance in q_1 using the Dihedral group D_4 that encodes 4 rotations every 90 degrees and reflection. For q_2, we use a equivariant feature map that outputs a single u-vector of Q values corresponding the finite cyclic group used (we use C_{12}/C_2 and C_{32}/C_2 in our experiments below). We handle the partial equivariance of our input and output space using the same strategies as earlier. Appendix D describes the full model details.



(a) Block Stacking (b) Bottle Arrangement

Figure 6: Comparison between Equivariant ASR (blue) and Equivariant FCN (green). Results averaged over four runs. Shading denotes standard error.

Experimental Comparison with Equivariant FCN on Block Stacking and Bottle Arrangement: Figure 6 shows a comparison between equivariant ASR (this section) and equivariant FCN (the method of Section 4.2) for the Block Stacking and Bottle Arrangement tasks. q_2 is defined under C_{12} and



(a) House Building (b) Covid Test (c) Box Palletizing (d) Bin Packing

Figure 7: Comparison of Equivariant ASR (blue) with baselines. Results averaged over four runs. Shading denotes standard error.

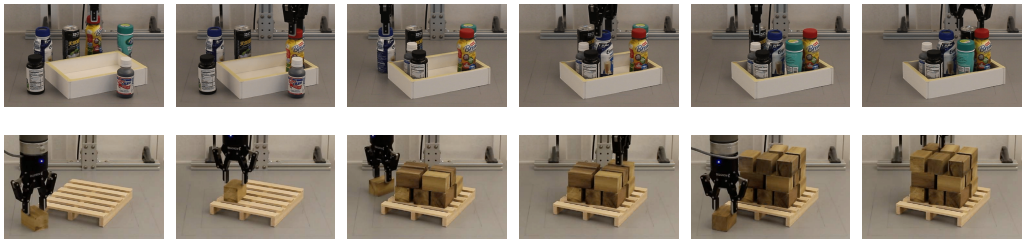


Figure 8: Top row: the robot finishing the Bottle Arrangement task. Bottom row: the robot finishing the Box Palletizing task. Full episodes are in Appendix I.

its quotient group C_{12}/C_2 to match Section 4.2. Both architectures do well, but the ASR method learns better, surpassing the FCN curve by a small amount.

More Challenging Experimental Domains: The equivariant ASR method is able to solve more challenging manipulation tasks than equivariant FCN can (we could not run FCN on these domains due to memory limitations). We evaluate on the following four additional domains: House Building, Covid Test, Box Palletizing, and Bin Packing (Figure 3(c-f)). In House Building, the robot must build the house-like structure shown in the figure using blocks. In Covid Test, the robot needs to administer three COVID-19 tests by transporting three swabs and test tubes to the test area (in gray), and then collecting the test tubes (with swabs inside) when the test is complete. Box Palletizing requires the robot to stack 18 boxes in three layers on top of a pallet (this domain was introduced in [1]). In the Bin Packing task, the agent must compactly pack eight blocks into a bin while minimizing the highest point of all objects. All domains except Bin Packing have binary sparse rewards ($+1$ at goal and 0 otherwise). In Bin Packing, the agent obtains a positive reward with magnitude inversely proportional to the highest point in the pile after packing all objects. See Appendix C for more detail. All experimental parameters mirror those of Section 4.2, except that we now define q_2 under the cyclic group C_{32} and its quotient group C_{32}/C_2, i.e., we now encode 16 orientations rather than just 6. As in Section 4.2, we use the SDQfD loss term to incorporate expert demonstrations (except for Transporter Net which uses standard behavior cloning exclusively). The number of expert transitions provided the agent for each domain is: 200 steps ($\approx$20 episodes) in House Building; 1000 steps ($\approx$28 episodes) in Box Palletizing; 2000 steps ($\approx$111 episodes) in Covid Test; 2000 steps ($\approx$125 episodes) in Bin Packing. 9 random $\mathrm{SE}(2)$ augmentations are applied to the expert transitions.

Experimental Comparison with Non-Equivariant Baselines: We compare equivariant ASR against the following non-equivariant baselines: 1) Conventional ASR: ASR in $\mathrm{SE}(2)$ with conventional CNNs rather than equivariant layers; 2) Soft Equivariant ASR: same architecture as (1) above but each minibatch is augmented three fold for rotations every 90 degrees; 3) the Transporter Network architecture of [1]. See Appendix E.2 for the network architecture for the baselines. The results in Figure 7 show that equivariant ASR outperforms the other methods on all tasks, followed by Soft Equivariant ASR and Transporter Net, followed by Conventional ASR.

Robot Experiment: We evaluate the equivariant ASR models for Bottle Arrangement, House Building, and Box Palletizing trained in PyBullet on a Universal Robots UR5 arm equipped with a Robotiq 2F-85 gripper. The observation is provided by an Occipital Structure sensor mounted on top of the

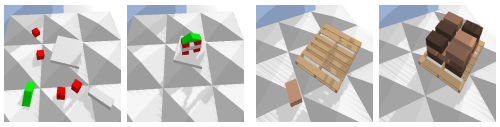
workspace. Table 1 shows the results. In Bottle Arrangement, the robot successfully completes 18 of 20 episodes, showing a 90% success rate. In one of the two failures, the arrangement is not compact enough, leaving no enough space left for the last bottle. In the other failure, the robot arranges the bottles outside of the tray. In the House Building task, the robot succeeds all 20 episodes. In the Box Palletizing task, the robot finishes 19 of 20 episodes, demonstrating a 95% success rate. In the failure, the robot correctly stacks 16 boxes, but the 17th box's placement position is offset slightly from the the rest of the stack and there is no room to place the last box. The same problem happens in another episode, where the fingers squeeze the boxes and make room for the last box. Figure 8 shows an example episode of the Bottle Arrangement and the Box Palletizing tasks.

Environment	SR
Bottle Arrangement	90%(18/20)
House Building	100%(20/20)
Box Palletizing	95%(19/20)

Table 1: Robot experiment results

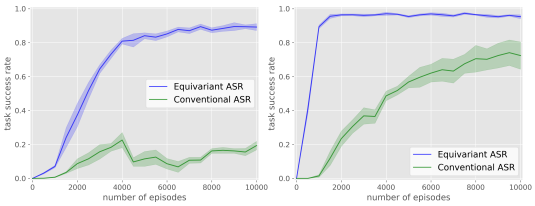
4.4 Equivariant Augmented State Q Functions in $\mathrm{SE}(3)$

A strength of the ASR method is that it can be extended into $\mathrm{SE}(3)$ by adding networks similar to q_2 that encode Q values for additional dimensions of action [29]. Specifically, we add three networks, q_3, q_4, and q_5 that encode Q values for Z (height above the plane), Φ, and Ψ (out of plane orientation) dimensions of $\mathrm{SE}(3)$. Each of these networks takes as input a multi-channel image patch $P \in \mathbb{R}^{n' \times h' \times w'}$ that encodes information about previous action choices, similar to how we encoded the choice of x to q_2 (see [29] for details). q_1 and q_2 can use the same equivariant network we proposed so far. To approximate the equivariance in the additional dimensions outside of $\mathrm{SE}(2)$, we create an encoding that explicitly transforms the input to these networks in a way that corresponds to a set of candidate robot positions or orientations (called a "deictic" encoding in [22]). See Appendix G for more details.



(a) House Building (b) Box Palletizing

Figure 9: The 6DOF experimental domains.



(a) House Building (b) Box Palletizing 18

Figure 10: Comparison of the equivariant network with the baseline in 6DOF tasks. Results averaged over four runs. Shading denotes standard error.

Comparison to Non-Equivariant Approaches: We evaluate the ASR method in $\mathrm{SE}(3)$ in the House Building and Box Palletizing domains. We modified those environments so that objects are presented randomly on a bumpy surface with a maximum out of plane orientation of 15 degrees (Figure 9). In order to succeed, the agent must correctly perform pick and place actions with the needed height and out of plane orientation. We evaluated the Equivariant ASR in comparison with a baseline Conventional ASR (same as [29]). Both methods use SDQfD with 2000 expert demonstration steps. The results are shown in Figure 10. Our proposed approach outperforms the baseline by a significant margin. Note that we cannot run the Transporter Net baseline [1] here because it does not allow for an $\mathrm{SE}(3)$ action space.

5 Discussion

In this paper, we show that equivariant neural network architectures can be used to improve Q learning in spatial action spaces. We propose multiple approaches and model architectures that can be used to accomplish this and demonstrate improved sample efficiency and performance on several robotic manipulation applications both in simulation and on a physical system. This work has several limitations and directions for future research. First, our methods apply directly only to problems in spatial action spaces. While many robotics problems can be expressed this way, it would clearly be useful to develop equivariant models for policy learning that can be used in other settings. Second, although we extend our ASR approach from $\mathrm{SE}(2)$ to $\mathrm{SE}(3)$ in the last section of this paper, this solution is not fully equivariant in $\mathrm{SE}(3)$ and it may be possible to do better by exploiting methods that are directly equivariant in $\mathrm{SE}(3)$.

References

[1] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

[2] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[3] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[4] T. S. Cohen and M. Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[7] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[8] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[9] N. Hansen and X. Wang. Generalization in reinforcement learning by soft data augmentation. *arXiv preprint arXiv:2011.13389*, 2020.

[10] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[11] Y. Lin, J. Huang, M. Zimmer, Y. Guan, J. Rojas, and P. Weng. Invariant transform experience replay: Data augmentation for deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6615–6622, 2020.

[12] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[13] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[14] M. Weiler and G. Cesa. General $e(2)$-equivariant steerable cnns. *arXiv preprint arXiv:1911.08251*, 2019.

[15] E. van der Pol, T. Kipf, F. A. Oliehoek, and M. Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963*, 2020.

[16] E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[17] A. K. Mondal, P. Nair, and K. Siddiqi. Group equivariant deep reinforcement learning. *arXiv preprint arXiv:2007.03437*, 2020.

[18] D. Morrison, P. Corke, and J. Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*, 2018.

[19] V. Satish, J. Mahler, and K. Goldberg. On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters*, 4 (2):1357–1364, 2019.

[20] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3750–3757. IEEE, 2018.

[21] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[22] R. Platt, C. Kohler, and M. Gualtieri. Deictic image mapping: An abstraction for learning pose invariant manipulation policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8042–8049, 2019.

[23] M. Gualtieri and R. Platt. Learning manipulation skills via hierarchical spatial attention. *IEEE Transactions on Robotics*, 36(4):1067–1078, 2020.

[24] L. Berscheid, P. Meißner, and T. Kröger. Robot learning of shifting objects for grasping in cluttered environments. *arXiv preprint arXiv:1907.11035*, 2019.

[25] H. Liang, X. Lou, and C. Choi. Knowledge induced deep q-network for a slide-to-wall object grasping. *arXiv preprint arXiv:1910.03781*, 2019.

[26] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv preprint arXiv:1903.11239*, 2019.

[27] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. *arXiv preprint arXiv:1910.13675*, 2019.

[28] L. Berscheid, P. Meißner, and T. Kröger. Self-supervised learning for precise pick-and-place without object model. *IEEE Robotics and Automation Letters*, 5(3):4828–4835, 2020.

[29] D. Wang, C. Kohler, and R. Platt. Policy learning in se (3) action spaces. *arXiv preprint arXiv:2010.02798*, 2020.

[30] O. Biza, D. Wang, R. Platt, J.-W. van de Meent, and L. L. Wong. Action priors for large action spaces in robotics. *arXiv preprint arXiv:2101.04178*, 2021.

[31] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser. Spatial action maps for mobile manipulation. *arXiv preprint arXiv:2004.09141*, 2020.

[32] T. Cohen, M. Geiger, and M. Weiler. A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*, 2018.

[33] X. Jia, B. De Brabandere, T. Tuytelaars, and L. Van Gool. Dynamic filter networks. In *NIPS*, 2016.

[34] W. Wohlkinger, A. Aldoma Buchaca, R. Rusu, and M. Vincze. 3DNet: Large-Scale Object Class Recognition from CAD Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[36] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[37] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[38] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[40] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. ISSN 00034851. URL http://www.jstor.org/stable/2238020.

[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[42] K. Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.