EQUIVARIANT GRASP LEARNING IN REAL TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual grasp detection is a key problem in robotics where the agent must learn to model the grasp function, a mapping from an image of a scene onto a set of feasible grasp poses. In this paper, we recognize that the grasp function is $\mathrm{SE}(2)$-equivariant and that it can be modeled using an equivariant convolutional neural network. As a result, we are able to significantly improve the sample efficiency of grasp learning to the point where we can learn a good approximation of the grasp function within only 500 grasp experiences. This is fast enough that we can learn to grasp completely on a physical robot in about an hour.

1 INTRODUCTION

The low sample efficiency of policy learning is a key challenge in robotics. For example, in robotics, it is typical for an agent to require tens if not hundreds of thousands of interactions with the environment to learn even relatively simple manipulation policies, making it virtually impossible to learn directly in the physical world. Researchers typically deal with this problem in one of two ways: 1) by learning in simulation rather than on the real robot; 2) by learning state representations that simplify the on-line learning problem Laskin et al. (2020b); Watter et al. (2015). Unfortunately, neither of these solutions is easy: bridging the sim2real domain gap remains a challenge and representation learning is itself a major research area. Ideally, we would simplify the robotic learning problem in a more direct way, by incorporating a strong inductive bias into the model that nevertheless generalizes well in its application domain. This paper attempts to accomplish this using geometric deep learning.

We explore the application of $\mathrm{SO}(2)$-equivariant model architectures to $\mathrm{SE}(2)$ visual grasp detection, a key problem in robotics. In $\mathrm{SE}(2)$ visual grasp detection, the robot observes a scene as an image and must detect good grasp points in $\mathrm{SE}(2)$. Our key observation is that the grasp function we want to learn is $\mathrm{SE}(2)$-equivariant. That is, rotations and translations of the input image should correspond to the same rotations and translations of the detected grasp poses at the output of the function. This suggests that we can model the grasp function as an $\mathrm{SO}(2)$-equivariant neural network. Framing grasp learning as a contextual bandit, we introduce an appropriate $\mathrm{SO}(2)$-equivariant model and demonstrate that it can learn a good grasp function in approximately 500 grasp trials. This is much faster than most competing grasp learning approaches: Zeng et al. (2018a) takes 2k grasp trials to learn to grasp simple objects; Pinto & Gupta (2015) takes roughly 50k grasp trials; QT-Opt is trained using 580k grasp examples (Kalashnikov et al., 2018); GPD is trained using 200k examples (ten Pas et al., 2017); GraspNet is trained using 7m examples (Mousavian et al., 2019); and DexNet is trained with 6.7m examples (Mahler et al., 2017). The fact that we can learn a good grasp function so quickly means that we can learn directly on a real robot in approximately one hour without any additional pretraining. This kind of sample efficiency could be critical in robotics applications because it has the potential to enable robotic learning systems to adapt to the idiosyncrasies of the real world through direct interaction, thereby making them much more reliable.

2 RELATED WORK

Equivariant networks: Equivariant neural networks inject group symmetry into the architecture of the neural network, allowing it to automatically generalize to the transformation of the input. This concept is first introduced as G-Convolution (Cohen & Welling, 2016a) and Steerable CNN (Cohen & Welling, 2016b). The E2CNN framework proposes a generic approach for implementing $\mathrm{E}(2)$

Steerable CNN (Weiler & Cesa, 2019). The applications of equivariant learning in computer vision (Walters et al., 2020; Wang et al., 2020b) and reinforcement learning (van der Pol et al., 2020; Mondal et al., 2020; Wang et al., 2021) demonstrate improvements over traditional approaches. This paper applies the equivariant learning in robotic grasping to substantially decreases the amount of sample required for learning a grasping policy.

Sample efficient reinforcement learning: Recent work has shown that using translational data augmentation (e.g., random crop or random shift) can improve the sample efficiency of conventional reinforcement learning algorithms (Laskin et al., 2020a; Kostrikov et al., 2020). The combination of data augmentation and contrastive learning (Oord et al., 2018), as is explored in CURL (Laskin et al., 2020b), encourages the encoder to learn an invariant encoding thus improving sample efficiency. Zhan et al. (2020) extends the idea of CURL in the context of robotic manipulation. However, compared with equivariant networks, data augmentation requires the model to learn the invariance/equivariance in addition to learning the task itself, which necessitates additional training time and greater model capacity.

Grasp learning: Robotic grasping learning often involves massive data collection. One typical way of supervised grasping learning is to train using a labeled dataset (e.g., 200k data points in ten Pas et al. (2017); 7m data points in Mousavian et al. (2019); 6.7m data points in Mahler et al. (2017); 1.8k images (with multiple grasps identified in each image) in Zeng et al. (2018b)). An alternative is to collect real grasp data using robots. Pinto & Gupta (2015) collect 50k data points using a Baxter robot over 700 hours. Levine et al. (2018) collect 800k grasp attempts using between 6 and 16 robots over 2 months. QT-Opt (Kalashnikov et al., 2018) collect 580k grasp attempts using 7 robots over 800 hours. James et al. (2019) extend QT-Opt and uses 28k of additional online grasp samples. Berscheid et al. (2021) collect 27k grasp attempts within 120 hours. Song et al. (2020) collect 8k grasp demonstrations from human using a low-cost hand-hold gripper. Zeng et al. (2018a) learns a online pushing-grasping policy using 2.5k steps. Compared with prior works, our method learns a good grasp policy in just 500 grasp trials, significantly fewer than in prior work.

3 Background

3.1 Equivariant Neural Network Models

The cyclic group $C_n \leq \mathrm{SO}(2)$: In this paper, we are primarily interested in equivariance with respect to the group of planar rotations, $\mathrm{SO}(2)$. However, in practice, in order to make our models computationally tractable, we will use the cyclic subgroup C_n of $\mathrm{SO}(2)$, $C_n = \{\frac{i \times 2\pi}{n} | 0 \leq i < n\}$. C_n is the group of discrete rotations every $\frac{2\pi}{n}$ radians.

Representation of a group: Members of the cyclic group $g \in C_n$ represent rotations. We are often interested in applying this rotation to data – this happens via a *representation* of the group element. However, the type of representation needed depends upon the type of data to be rotated. There are two main representations relevant to this paper. The *regular representation* acts on an m-vector $(x_1, x_2, \ldots, x_m) \in \mathbb{R}^m$ by permuting its elements: $\rho_{reg}(g)x = (x_m, x_1, x_2, \ldots, x_{m-1})$. The *trivial representation* acts on a scalar $x \in \mathbb{R}$ and makes no change at all: $\rho_0(g)x = x$.

Feature maps of equivariant convolutional layers: An equivariant convolutional layer is associated with a finite group and a group representation and it adds an extra channel to the input and output feature maps which encodes the elements of that group. So, whereas the feature map used by a standard convolutional layer is a tensor $\mathcal{F} \in \mathbb{R}^{m \times h \times w}$, an equivariant convolutional layer adds an extra dimension: $\mathcal{F} \in \mathbb{R}^{k \times m \times h \times w}$, where k denotes the dimension of the group representation. This tensor associates each pixel $(x, y) \in \mathbb{R}^{h \times w}$ with a matrix $\mathcal{F}(x, y) \in \mathbb{R}^{k \times m}$.

Action of the group operator on the feature map: Given a feature map $\mathcal{F} \in \mathbb{R}^{k \times m \times h \times w}$ associated with group G and representation ρ, a group element $g \in G$ acts on $\mathcal{F}$ via:

$$g\mathcal{F}(x) = \rho(g)\mathcal{F}(\rho_1(g)^{-1}x), \tag{1}$$

where $x \in \mathbb{R}^2$ denotes pixel position. In the above, $\rho_1(g)^{-1}x$ is the coordinates of pixel rotated by g^{-1} and $\mathcal{F}(\rho_1(g)^{-1}x) \in \mathbb{R}^{k \times m}$ is the matrix associated with pixel $\rho_1(g)^{-1}x$. g operates on $\mathcal{F}(\rho_1(g)^{-1}x)$ via the representation ρ associated with the feature map. For example, if $\rho = \rho_0$ (the trivial representation), then $k = 1$ and g acts on $\mathcal{F}$ by rotating the image but leaving the m-vector associated with each pixel unchanged. In contrast, if $\rho = \rho_{reg}$ (the regular representation), then $k = |G|$ (the number of elements in G) and g acts on $\mathcal{F}$ by performing a circular shift on the group dimension of $\mathcal{F}(\rho_1(g)^{-1}x)$. This last action (by the regular representation) is the one primarily associated with the equivariant convolutional layers used in this paper.

The equivariant convolutional layer: An equivariant convolutional layer is a function h from $\mathcal{F}_{in}$ to $\mathcal{F}_{out}$ that is constrained to represent only equivariant functions with respect to a chosen group G. Each of $\mathcal{F}_{in}$ and $\mathcal{F}_{out}$ is associated with either with the trivial representation or the regular representation: $\mathcal{F}_{in}$ is associated with $\rho_{in} \in \{\rho_0, \rho_{reg}\}$ and $\mathcal{F}_{out}$ is associated with $\rho_{out} \in \{\rho_0, \rho_{reg}\}$. Then the equivariant constraint for h is:

$$h(\rho_{in}(g)\mathcal{F}_{in}) = \rho_{out}(g)h(\mathcal{F}_{in}) = \rho_{out}(g)\mathcal{F}_{out}. \quad (2)$$

This constraint can be implemented by tying kernel weights in such a way as to satisfy the following kernel constraint (Cohen et al., 2018):

$$K(gy) = \rho_{out}(g)K(y)\rho_{in}(g)^{-1}. \quad (3)$$

3.2 Augmented State Representation (ASR)

We formulate robotic grasping as the problem of learning a function from an image, $s \in \mathbb{R}^{m\times h\times w}$, to a gripper pose $a \in \mathrm{SE}(2)$ from which an object may be grasped. In this case, since the desired gripper pose is considered to be the action, we must learn a function onto a region of SE(2) – something that is challenging to do using a single neural network. Instead, this paper will leverage the Augmented State Representation (ASR), an approach that models a function onto SE(2) as a pair of two Q functions, Q_1 and Q_2 (Wang et al., 2020a). Let $Q : \mathbb{R}^{m\times h\times w} \times \mathrm{SE}(2) \to \mathbb{R}$ denote the action-value function over the full action space. We factor $\mathrm{SE}(2) = \mathbb{R}^2 \times \mathrm{SO}(2)$ into a translational component $X \subseteq \mathbb{R}^2$ and a rotational component $\Theta \subset \mathrm{SO}(2)$. The first function is a mapping $Q_1 : \mathbb{R}^{m\times h\times w} \times X \to \mathbb{R}$ which maps from the image s and the translational component of action X onto value. This function is defined to be: $Q_1(s,x) = \max_{\theta\in\Theta} Q(s,(x,\theta))$. The second function is a mapping $Q_2 : \mathbb{R}^{m\times h'\times w'} \times \Theta \to \mathbb{R}$ with $h' \le h$ and $w' \le w$ which maps from an image patch and an orientation onto value. This function takes as input a cropped version of s centered on a position x, $\mathrm{crop}(s,x)$, and an orientation, θ, and outputs the corresponding Q value: $Q_2(\mathrm{crop}(s,x),\theta) = Q(s,(x,\theta))$. Inference is performed on the model by evaluating $x^* = \arg\max_{x\in X} Q_1(s,x)$ first and then evaluating $Q_2(\mathrm{crop}(s,x^*),\theta)$. Since each of these two models, Q_1 and Q_2, are significantly smaller than Q would be, inference is much faster. Figure 1 shows an illustration of this process. The top of the figure shows the action of Q_1 while the bottom shows Q_2. Notice that the semantics of Q_2 imply that the θ depends only on $\mathrm{crop}(s,x)$, a local neighborhood of x, rather than on the entire scene. This assumption is generally true for grasping because grasp orientation typically depends only on the object geometry nearly the target grasp point.

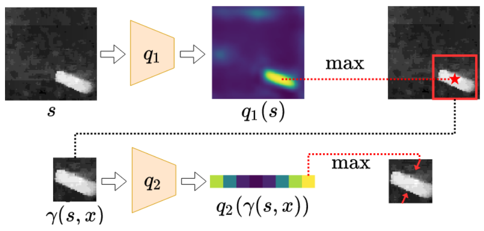


Figure 1: Illustration of the ASR representation. Q_1 selects the translational component of an action, Q_2 selects the rotational component.

4 Problem Statement

In *visual grasp detection*, we must estimate a *grasp function* $\Gamma : \mathbb{R}^{m\times h\times w} \to \mathrm{SE}(2)$ that maps from an image of a scene containing graspable objects, $s \in \mathbb{R}^{m\times h\times w}$, to a planar gripper pose $a \in \mathrm{SE}(2)$ from which an object can be grasped. We make the following assumptions:

Assumption 4.1 (Aligned reference frames)**.** *The reference frame in which the gripper pose a is expressed is aligned with the image plane of s.*

Assumption 4.2 (Equivariance of the grasp function)**.** *The grasp function Γ is equivariant with respect to* SE(2)*. That is:*

$$ga = g\Gamma(s) = \Gamma(gs), \quad (4)$$

where $g \in \mathrm{SE}(2)$ is a transformation that rotates and translates, gs denotes the image s rotated and translated by g, and ga denotes the gripper pose a transformed by g.

Assumption 4.1 is needed in order for the grasp function Γ to be equivariant. We need rotations of the image and rotations of the gripper pose to be aligned such that they rotate about the same axis. Essentially, this assumption is a constraint on the robot setup. Assumption 4.2 says that Γ is equivariant with respect to $g \in \mathrm{SE}(2)$. This is satisfied in most grasping scenarios because the Newtonian dynamics of the grasp interaction are independent of the reference frame from which the system is viewed.

5 METHOD

We formulate visual grasp detection as a contextual bandit problem. The agent begins training with no prior knowledge. On each time step, it perceives the state expressed as an image $s \in \mathbb{R}^{m\times h\times w}$ and must select an action expressed as a grasp pose $a \in \mathrm{SE}(2)$ which is executed by the robot. The agent receives a positive unit reward for each successful grasp and zero reward otherwise. As the agent gains experience, it updates its Q value function. Our key innovation is to express the Q function using $\mathrm{SE}(2)$-equivariant neural networks.

5.1 EQUIVARIANT LEARNING

Invariance properties of Q_1 and Q_2: Since our agent is only rewarded for achieving a successful grasp, the grasp equivariance assumption (Assumption 4.2) ensures that both the expected reward function and the optimal Q function are invariant with the group operator g: $r(s,a) = r(gs, ga)$ and $Q^*(s,a) = Q^*(gs, ga)$. In the context of ASR (the augmented state representation of Section 3.2), this translates into separate invariance properties for Q_1^* and Q_2^*:

$$Q_1^*(gs, gx) = Q_1^*(s, x) \tag{5}$$

and

$$Q_2^*(g_\theta(\mathrm{crop}(s,x)), g_\theta + \theta) = Q_2^*(\mathrm{crop}(s,x), \theta), \tag{6}$$

where $g_\theta \in \mathrm{SO}(2)$ denotes the rotational component of $g \in \mathrm{SE}(2)$. In the equations above, gs denotes the rotated and translated image s, gx denotes the rotated and translated vector $x \in \mathbb{R}^2$, and $g_\theta(\mathrm{crop}(s,x))$ denotes the cropped image rotated by g_θ.

Finite Approximation of $\mathrm{SE}(2)$: In order to implement the invariance constraints of Equation 5 and 6 using neural networks, we first need to discretize $\mathrm{SE}(2)$ into a finite approximation. We constrain the positional component of the action 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 s, and constrain the rotational component of the action to be a member of a finite cyclic group $C_n = \{\frac{i\times 2\pi}{n} | 0 \leq i < n, i \in \mathbb{Z}\}$. This discretized action space will be written $\hat{\mathrm{SE}}(2) = \mathbb{Z}^2 \times C_n$.

Equivariant Q-Learning: In order to do learning, we need to define Q_1^* and Q_2^* as neural networks. We model Q_1 as a fully convolutional UNet (Ronneberger et al., 2015) $q_1 : \mathbb{R}^{m\times h\times w} \to \mathbb{R}^{1\times h\times w}$ that takes as input the state image and outputs a Q-map that assigns each pixel in the input a Q value. We model Q_2 as a standard convolutional network $q_2 : \mathbb{R}^{m\times h'\times w'} \to \mathbb{R}^n$ that takes the image patch as input and outputs an n-vector of Q values over C_n. Then the invariant properties of Equation 5 and 6 become equivariant properties:

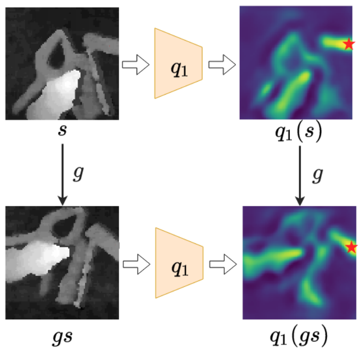
$$q_1(gs) = gq_1(s) \tag{7}$$

$$q_2(g_\theta \mathrm{crop}(s,x)) = \rho_{reg}(g_\theta) q_2(\mathrm{crop}(s,x)) \tag{8}$$

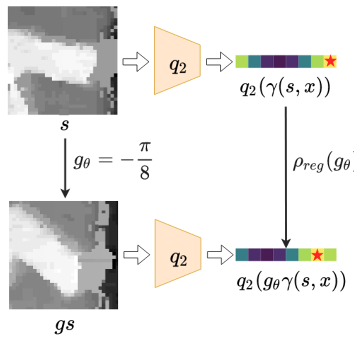
where $g \in \hat{\mathrm{SE}}(2)$ acts on the output of q_1 through rotating the Q-map, and $g_\theta \in C_n$ acts on the output of q_2 by performing a circular shift of the output Q values via the regular representation ρ_{reg}.

This is illustrated in Figure 2. In Figure 2a we are given the depth image s in the upper left corner. If we rotate this image by g (lower left of Figure 2a) and then evaluate q_1, we arrive at $q_1(gs)$. This corresponds to the LHS of Equation 7. However, because q_1 is an equivariant function, we can calculate the same result by first evaluating $q_1(s)$ and *then* applying the rotatation g (RHS of Equation 7). Figure 2b illustrates the same concept for Equation 8. Here, the network takes the image patch $\text{crop}(s, x)$ as input. If we rotate the image patch by g_θ and then evaluate q_2, we obtain the LHS of Equation 8, $q_2(g_\theta \text{crop}(s, x))$. However, because q_2 is equivariant, we can obtain the same result by evaluating $q_2(\text{crop}(s, x))$ and circular shifting the resulting vector to denote the change in orientation by one group element.

Equivariant q_1: As a fully convolutional network, q_1 inherits the translational equivariance property of standard convolutional layers. The challenge is to encode rotational equivariance so as to satisfy Equation 7. We accomplish this using equivariant convolutional layers that satisfy the equivariance constraint of Equation 2 where we assign $\mathcal{F}_{in} = s \in \mathbb{R}^{1\times m\times h\times w}$ to encode the input state s and $\mathcal{F}_{out} \in \mathbb{R}^{1\times 1\times h\times w}$ to encode the output Q map. Both feature maps are associated with the trivial representation ρ_0 such that the rotation g operates on these feature maps by rotating pixels without changing their values. We use the regular representation ρ_{reg} for the hidden layers of the network to encode more comprehensive information in the intermediate layers. We empirically achieve the best performance when defining q_1 in the Dihedral group D_4 that encodes 4 rotations every 90 degrees and reflection.



(a) Illustration of Equation 7



(b) Illustration of Equation 8

Figure 2: Equivariance relations expressed by Equation 7 and Equation 8.

Equivariant q_2: Whereas the equivariance constraint in Equation 7 is over $\hat{\text{SE}}(2)$, the constraint in Equation 8 is over C_n only. We implement Equation 8 using Equation 2 with an input of $\mathcal{F}_{in} = \text{crop}(s, x) \in \mathbb{R}^{1\times m\times h'\times w'}$ as a trivial representation, and an output of $\mathcal{F}_{out} \in \mathbb{R}^{n\times 1\times 1\times 1}$ as a regular representation. q_2 is defined in the group C_n, where the number of rotations in C_n must match the number of rotations in the action space. Since the parallel jaw gripper is symmetric when rotated by $\frac{\pi}{2}$, we introduce additional structure by replacing C_n with the quotient group $C_n/C_2 = \{\frac{i\times 2\pi}{n} | 0 \le i < n/2, i \in \mathbb{Z}\}$.

5.2 Other Optimizations

While our use of equivariant models to encode the Q function is responsible for most of our gains in sample efficiency, there are several additional algorithmic details that have a small positive impact on performance.

Loss Function: In the standard ASR loss function, both q_1 and q_2 have a Monte Carlo target, i.e. the target set equal to the transition reward (Wang et al., 2020a):

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2; \tag{9}$$

$$\mathcal{L}_1 = \tfrac{1}{2}(Q_1(I, x) - r)^2;\ \mathcal{L}_2 = \tfrac{1}{2}(Q_2(\gamma(I, x), \theta) - r)^2. \tag{10}$$

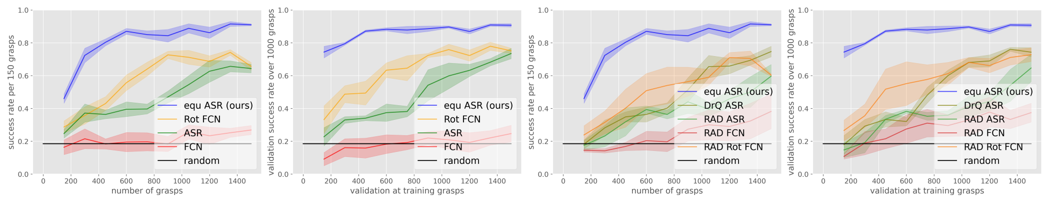
However, in order to reduce variance in the binary rewards scenario ($r \in \{0, 1\}$), we modify $\mathcal{L}_1$:

$$\mathcal{L}_1' = \frac{1}{2}(Q_1(I, x) - (r + (1 - r)\max_{\theta\in\bar{\Theta}} Q_2(\gamma(I, x), \theta)))^2, \tag{11}$$

where $\bar{\Theta} = \{\bar{\theta} \neq \theta | \forall\bar{\theta} \in C_n/C_2\}$. For a positive sample ($r = 1$), the target will simply be 1, as it was in Equation 9. However, for a negative sample ($r = 0$), we use a TD target calculated by maximizing over the Q_2 action component (but not including the failed θ action component). In addition to the above, we add an off-policy loss term $\bar{\mathcal{L}}_1$ that is evaluated with respect to an additional k grasp positions $\bar{X} \subset X$ sampled using a Boltzmann distribution from X:

$$\mathcal{L}_1'' = \frac{1}{k}\sum_{x_i\in\bar{X}} \frac{1}{2}(Q_1(I, x_i) - \max_{\theta\in\Theta} Q_2(\gamma(I, x_i), \theta))^2. \tag{12}$$

Our combined loss function is therefore $\mathcal{L} = \mathcal{L}_1' + \mathcal{L}_1'' + \mathcal{L}_2$.



(a) Training (b) Validation (c) Training (d) Validation

Figure 3: (a), (b) Baselines that do not use image augmentation. (c) (d) Image augmentation baselines. (a) and (c) show learning curves as a running average over the last 150 training grasps. (b) and (d) show average near-greedy performance of 1000 validation grasps performed every 150 training steps.

Softmax at the output of q_1 and q_2: Since the reward function of the contextual bandit is either zero or one, the optimal Q function at a given state can be viewed as the probability of a success at that state. We encoded this prior knowledge using a softmax layer at the output of the q_1 and q_2 networks.

Prioritizing failure experiences in minibatch sampling: In the contextual bandit setting, we want to avoid the situation where the agent selects the same incorrect action several times in a row. This can happen because when a grasp fails, the depth image of the scene does not change and therefore the Q map will not change. One way to address this problem is to ensure that following a failed grasp experience, that the failed grasp is included in the sampled minibatch on the next SGD step Zeng et al. (2018a), thereby changing the Q function prior to reevaluating it on the next time step. This reduces the chance that the same (bad) action will be selected.

Boltzmann exploration: We compared Boltzmann with ϵ-greedy exploration and found Boltzmann to be better in our grasp setting. We use a temperature of τ_{training} during training and a lower temperature of τ_{test} during testing. Using a non-zero temperature at test time helped reduce the chances of repeat sampling of a bad action.

Data augmentation: Even though we are using equivariant neural networks to encode the Q function, it can still be helpful to do data augmentation as well. This is because the granularity of the rotation group encoded in q_1 (D_4) is smaller than that of the action space (C_n/C_2). We address this problem by augmenting the data with translations and rotations sampled from $\hat{\mathrm{SE}}(2)$. For each experienced transition, we add eight additional images to the replay buffer that have been transformed in this way.

Selection of the z coordinate: Since our model only infers grasp pose a in $\mathrm{SE}(2)$, we are limited to detecting top down grasps, e.g. grasps where the gripper is pointed directly down at the table. However, we must still somehow calculate the z coordinate based on action a. Here, we calculate z by taking the maximum (highest) depth in a small neighborhood of the grasp point a in the depth image and then offsetting the height of the gripper.

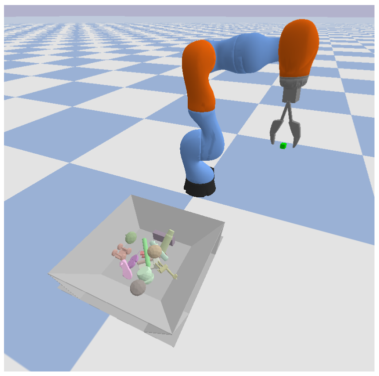
6 Experiments

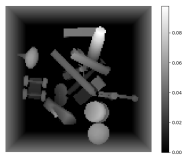
6.1 Experiments in Simulation

Simulation environment: The simulation experiments are performed in Pybullet (Coumans & Bai, 2016). The environment includes a Kuka robot arm and a 0.3m $\times$ 0.3m tray with inclined walls (Figure 4a). At the beginning of each episode, the environment is initialized with 15 objects drawn uniformly at random from a Dataset of 76 mesh models (Figure 4c) and dropped arbitrarily into the tray. State is a depth image captured from a top-down camera (Figure 4b). On each time step, the agent perceives state and selects an action to execute which specifies the planar pose to which to move the gripper. A grasp is considered to have been successful if the robot is able to lift the object more than 0.1m above the table. The episode continues until all objects have been removed from the tray or until 30 grasp attempts have been made at which point the episode terminates and the environment is reinitialized.

Model details: The q_1 network is defined with respect to the Dihedral group, D_4, that encodes 4 rotations every 90 degrees as well as reflections. q_2 is defined with respect to the group C_{16}/C_2 – eight rotations ranging from 0 to π radians. We use Boltzmann action sampling with a temperature of $\tau_{\text{training}} = 0.01$ during training and a temperature of $\tau_{\text{test}} = 0.002$ during validation.

Comparison with baselines that do not use image augmentation: First, we compare our method against a set of baselines that do not leverage image augmentation: 1) *Rot-FCN* (Zeng et al., 2018a): an FCN with single-channel output that estimates the Q-map for each rotation in the action space by rotating the input image; 2) *FCN* (Satish et al., 2019): an FCN with 8-channel output that associates each grasp rotation to a channel of the output; 3) *ASR* (Wang et al., 2020a): a sequential approach that selects the translational component and rotational component of the action using two networks. Figure 3a and b show the results. Figure 3a shows the performance of the methods during training when the Boltzmann exploration constant is set to $\tau_{\text{training}} = 0.01$. Figure 3b evaluates the methods by temporarily halting training and executing 1000 validation grasps every 150 time steps of training using a near-greedy exploration ($\tau_{\text{test}} = 0.002$) policy. Notice that our our method learns faster and converges to a higher success rate than the other baselines. In fact, the validation success rate of our method at grasp 150 is as high as that of the best baseline, *Rot FCN*, at grasp 1500.


(a) Simulation environment



(b) Observation

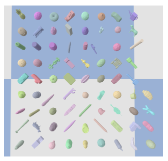

(c) Object set

Figure 4: The Pybullet simulation environment.

Comparison with image augmentation baselines: Here, we compare our method against two recent baselines that use image augmentation to improve sample efficiency during learning, RAD (Laskin et al., 2020a) and DrQ (Kostrikov et al., 2020). RAD augments each transition in the mini-batch during training. In DrQ, the Q-target is calculated by averaging over two augmented versions of the sampled transition; the loss is also calculated by averaging over two augmentations. In each of these methods, we augment the data by drawing random transformations from $\hat{\text{SE}}(2)$, the same transformation group we use in our equivariant method. We adjust RAD and DrQ to fit our application by transforming both state and action (rather than just state alone) during image augmentation. We evaluate RAD with three different model architectures: ASR, Rot FCN and FCN. We evaluate DrQ only with ASR. Altogether, these baselines are: 1) *RAD + ASR*; 2) *DrQ + ASR*; 3) *RAD + Rot FCN*; 4) *RAD + FCN*. Figure 3a and b show the results. Our equivariant method outperforms the baselines convincingly.

Ablation study: We ablate the following components of our method as described in Section 5: 1) *no equ*: uses conventional FCN instead of the equivariant network; 2) *no FCB*: no modification of the loss function, no Softmax at the output, and no prioritizing failure experiences sampling; 3) *egreedy*: ϵ-greedy exploration instead of Boltzmann exploration (linear anneal ϵ from 0.5 to 0.1 over 500 time steps); 4) *no aug buff*: no data augmentation in the replay buffer. 5) *equ ASR*: combination of the three ablations above, i.e. no loss function modification, no Softmax at the output, no prioritizing failure experiences sampling, ϵ-greedy exploration, and no data augmentation. Figure 5 shows the results. Notice that the *no equ* version dramatically underperforms, demonstrating the importance of using the equivariant model. Also, notice that *equ ASR* underperforms slightly, demonstrating that the methods of Section 5.2 are helpful.

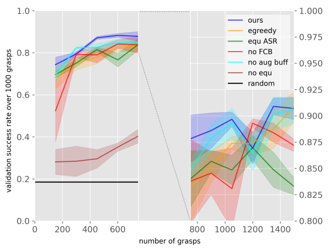
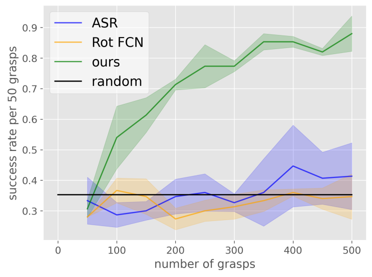
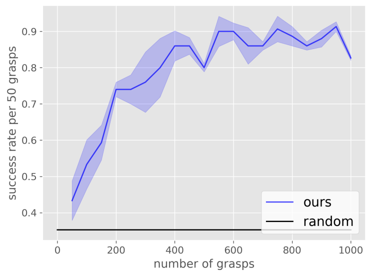


Figure 5: Ablation study of our method. The figure shows validation results. We zoomed in the plot after the 750th grasp.

6.2 Experiments on a physical robot



(a)



(b)

Figure 7: (a) Learning curves from 500-grasp experiment. (b) Learning curves from 1000-grasp experiment. All curves are averaged over 3 runs.

Robot environment: Our experimental platform is comprised of a Universal Robots UR5 manipulator equipped with a Robotiq 2F-85 parallel-jaw gripper, an Occipital Structure Sensor, and the dual-tray grasping environment shown in Figure 7a.

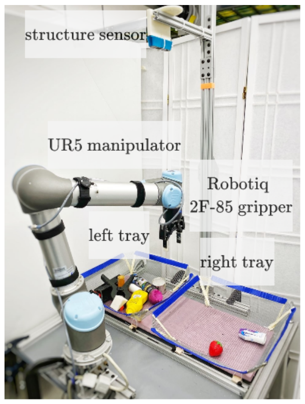


Figure 6: Robot setup.

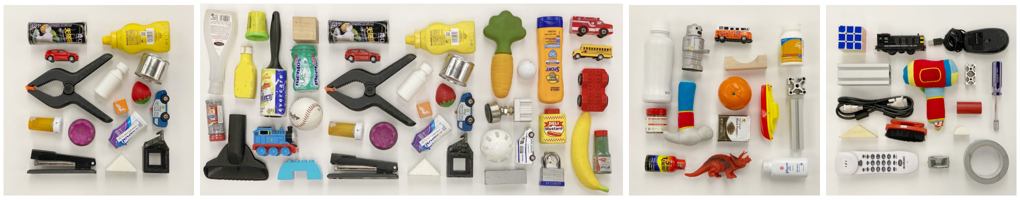
Training system: Grasp execution during training is automated. Initially, 15 training objects are randomly placed into one of the two trays (the object sets used during training are shown in Figure 8a and b.) Then, the robot grasps each of the objects and transports it to the other tray. This process repeats with all 15 objects being transported between one tray and the other until sufficient training has occurred. To avoid systematic bias in the way objects fall into a tray, we sample the drop position randomly from a Gaussian distribution centered in the middle of the receiving tray.

Optimizations for training on a physical robot: We were able to significantly accelerate real-robot training by processing camera data and executing SGD steps while the robotic system was in motion. This was implemented as a producer-consumer process using a mutex. With this improvement, the robot is constantly in motion during training. As a result, training speed completely determined by the speed of robot motion. In our setup, this approach enabled us to increase robot training speed from approximately 230 grasps per hour to roughly 500 grasps per hour.

Table 1: Evaluation for 500-Grasp experiment. Results are an average from 100 grasps performed on the 15 novel test objects of Figure 8c after training is complete.

Baseline	avg SR	std
Random	35.3%	4.11%
Rot FCN	35.0%	3.56%
ASR	42.0%	10.6%
Ours 500	92.0%	3.56%

500-Grasp Experiment: In this experiment, we train each of three different methods over a period of 500 grasp trials for the set of 15 objects shown in Figure 8a on the UR5 system. At the beginning of each grasp run, we deposit these same 15 objects into one of the two trays and run 500 grasp trials during which the learning algorithm trains, actively selecting each successive grasp according to its model and its action selection mechanism. At the end of training, the model is frozen and evaluated for the 15 test objects shown in Figure 8c. As in the simulation experiments, our q_1 network is defined for the group D_4 and our q_2 network is defined for the group C_{16}/C_2. During training, training the Boltzmann temperature is 0.01. After training, during evaluation, it is 0.002. We compare against two baselines: 1) *Rot FCN* (Zeng et al., 2018a) where we have an FCN with a single channel output that estimates the Q map for each rotation in the action space by rotating the input image; 2) *ASR* (Wang et al., 2020a) where we use the two-stage model described in Section 3.2 without the contributions proposed in this paper. Figure 7a shows the learning curves for the three methods during learning. Each curve is an average of three runs starting with different random seeds. Table 1 shows the performance of the model frozen after training averaged over 100 test grasps on the 15 novel test objects shown in Figure 8c. Both results show that our method significantly outperforms the baselines.



(a) Training object set 15 (b) Training object set 40 (c) Test object set, easy, 15 objects (d) Test object set, hard, 15 objects

Figure 8: The train/test object sets. Objects need to be graspable of the gripper at any configuration (smaller than the gripper open width and has enough height), and need be able been captured by the depth camera (no transparent nor has thing wall).

1000 Grasp Training: In this experiment, we evaluate training our method on the physical system with a larger set of objects (40 objects) and over a longer period of time (1000 grasps). We train using objects sampled from the 40-object set shown in Figure 8b. During training, after each 100 grasps, 15 new objects are sampled from the training set and used for the next 100 grasps. (Therefore, we resample the object set 10 times over the 1000 grasps.) Whereas in the 500-grasp experiment, we defined q_2 over the group C_{16}/C_2, we now define q_2 over C_{32}/C_2, i.e. 16 rotations ranging from 0 to π. We also augment each experience with 16 additional transformed images instead of just 8. We use the same Boltzmann action sampling parameters as in the 500 grasp experiment. After training is complete, the learned policy is frozen and evaluated on both held-out "easy" test set of Figure 8c and the held out "hard" test set of Figure 8d. Figure 7b shows the learning curve from training and Table 2 shows the results on the novel object test sets. The results indicate that our method learns a grasp function that generalizes well to novel objects.

Table 2: Evaluation for 1000-Grasp experiment, compared with 500-Grasp experiment. Results are an average from 100 grasps performed on the novel objects from Figure 8c and d.

	test set easy		test set hard	
Baseline	avg SR	std	avg SR	std
Random	35.3%	4.11%	31.0%	2.16%
Ours 500	92.0%	3.56%	89.3%	0.94%
Ours 1000	93.7%	1.25%	90.0%	4.08%

7 Conclusion and Future Work

This paper recognises that the grasp function that is learned in the visual grasp detection problem is SE(2)-equivariant. We propose using an SO(2)-equivariant model architecture to encode this structure. The resulting method is much more sample efficient than other grasp learning approaches and can learn a good grasp function in only 500 grasp samples. A key advantage of this increase in sample efficiency is that we are able to learn to grasp completely on the physical robotic system and without any pretraining in simulation. This increase in sample efficiency could be important in robotics for a couple of reasons. First, it obviates the need for training in simulation (at least for some problems like grasping), thereby making the sim2real gap less of a concern. Second, it opens up the possibility for our system to adapt to idiosyncrasies of the robot hardware or the physical environment that are hard to simulate. One limitation of these results, both in simulation and on the physical robot, is that despite the fast learning rate, grasp success rates (after training) still seems to be limited to the low/mid 90% range. This is the same success rate seen in with other grasp detection methods Mahler et al. (2017); ten Pas et al. (2017); Mousavian et al. (2019), but it is disappointment here because one might expect faster adaptation to lead ultimately to better grasp performance. This could simply be an indication of the complexity of the grasp function to be learned or it could be a result of stochasticity in the simulator and on the real robot. However, further exploration of ways to get closer to a perfect grasp success rate seems to be an important direction for future work.

8 Reproducibility Statement

Our code is accessible at `https://anonymous.4open.science/r/equivariant_grasp_in_real_time/README.md`. We will make our code publicly available in the final submission.

References

Lars Berscheid, Christian Friedrich, and Torsten Kröger. Robot learning of 6 dof grasping using model-based adaptive primitives. *arXiv preprint arXiv:2103.12810*, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016a.

Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*, 2018.

Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016b.

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

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018. URL `http://arxiv.org/abs/1806.10293`.

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

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *CoRR*, abs/2004.13649, 2020. URL `https://arxiv.org/abs/2004.13649`.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *CoRR*, abs/2004.14990, 2020a. URL `https://arxiv.org/abs/2004.14990`.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020b.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL `http://arxiv.org/abs/1411.4038`.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

Arnab Kumar Mondal, Pratheeksha Nair, and Kaleem Siddiqi. Group equivariant deep reinforcement learning. *arXiv preprint arXiv:2007.03437*, 2020.

Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, abs/1509.06825, 2015. URL `http://arxiv.org/abs/1509.06825`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

Vishal Satish, Jeffrey Mahler, and Ken 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. doi: 10.1109/LRA.2019.2895878.

Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *Robotics and Automation Letters*, 2020.

Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Robin Walters, Jinxi Li, and Rose Yu. Trajectory prediction using equivariant continuous convolution. *arXiv preprint arXiv:2010.11344*, 2020.

Dian Wang, Colin Kohler, and Robert Platt Jr. Policy learning in SE(3) action spaces. *CoRR*, abs/2010.02798, 2020a. URL `https://arxiv.org/abs/2010.02798`.

Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant \$q\$ learning in spatial action spaces. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=IScz42A3iCI`.

Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint arXiv:2002.03061*, 2020b.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

Maurice Weiler and Gabriele Cesa. General e(2)-equivariant steerable cnns. *CoRR*, abs/1911.08251, 2019. URL `http://arxiv.org/abs/1911.08251`.

Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas A. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *CoRR*, abs/1803.09956, 2018a. URL `http://arxiv.org/abs/1803.09956`.

Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald 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)*, pp. 3750–3757. IEEE, 2018b.

Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *CoRR*, abs/2012.07975, 2020. URL `https://arxiv.org/abs/2012.07975`.

Table 3: The baseline variation comparison. The variations in bold font are used in simulation experiment in section 6.1.

	SR at the 450th grasps		SR at the 1500th grasps	
baseline	avg SR	std	avg SR	std
ASR	34	3.9	66.8	3.1
ASR aug buff	**34**	**1.5**	**73.5**	**3.2**
ASR FCB aug buff	29.6	4.4	43.5	7.8
FCN	**15.8**	**6.1**	**24.6**	**5.1**
FCN aug buff	9.6	2.3	23.1	8.4
FCN FCB aug buff	14.1	0.8	21.3	7.1
Rot FCN	**49.4**	**7.3**	**75.1**	**1.6**
Rot FCN aug buff	42.9	3.7	69.1	1.6
Rot FCN FCB aug buff	35.5	7.3	78.1	1.8
DrQ ASR	30.9	4.5	66.9	4.1
DrQ ASR aug buff	**33.3**	**4.6**	**74.3**	**2.8**
DrQ ASR FCB	24.9	7	47.7	14
DrQ ASR FCB aug buff	25.3	6.3	37.3	11
RAD ASR	24.2	6	60.9	7.3
RAD ASR aug buff	**33.4**	**2.4**	**64.6**	**8.3**
RAD ASR FCB	29	5.2	39.8	2.8
RAD ASR FCB aug buff	30.5	8	34.9	9.5
RAD FCN	20.5	4.4	34.8	1.4
RAD FCN aug buff	21.4	8.2	37.6	0.5
RAD FCN FCB	21.5	8	32.4	10.5
RAD FCN FCB aug buff	**21.5**	**6**	**37.4**	**5.7**
RAD Rot FCN	28.2	10.1	70	4.6
RAD Rot FCN aug buff	**51.8**	**12.3**	**72.8**	**4.2**
RAD Rot FCN FCB	32.8	14.1	59.5	14.9
RAD Rot FCN FCB aug buff	48.5	11.3	68.7	1.7
equ ASR	81.6	1.2	83	0.8
ours	87.1	0.7	90.7	1.1

A Baseline variations comparison

The baseline could have several variations depend on whether applying 1) *aug buff* data augmentation in the replay buffer; 2) *FCB*: modification of the loss function, Softmax at the output, and prioritizing failure experiences sampling. In the simulation experiments (section 6.1), all the baseline we used are the best variation of that baseline, see table 3. The best variation of a baseline is in bold font.

B Neural network architecture

The network architectures for q_1 and q_2 networks are shown in figure 9. The $q1$ network is a fully convolutional U net Ronneberger et al. (2015). The $q2$ network is a fully convolutional network Long et al. (2014). These networks are implemented by Paszke et al. (2019), and the equivariant networks are implemented by Weiler & Cesa (2019). Adam optimizer is used for the SGD step Kingma & Ba (2015).

C Parameter choices

The parameters we choose in simulation experiment (section 6.1), physical robot 500-grasp experiment (section 6.2, 500-grasp), and physical robot 1000-grasp experiment (section 6.2, 1000-grasp) are listed in table 4.

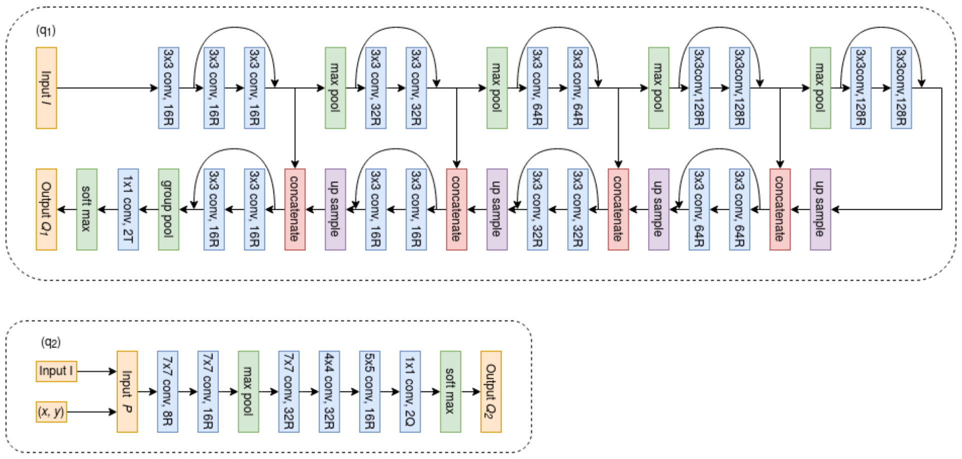


Figure 9: The neural network architecture of q_1 and q_2. R means regular representation, T mean trivial representation, Q means quotient representation. q_1 network is in D_4 group . q_2 network in 500-grasp training is in C_{16}/C_2 group, in 1000-grasp training is in C_{32}/C_2 group.

Table 4: Parameter choices.

environment	parameter	value
simulation, physical robot 500-grasp, physical robot 1000-grasp	learning rate	1e-4
	weight decay	1e-5
	augment buffer	random $SE(2)$, flip
	patch size	32
	obs size	128^2 pixel
	action range	96^2 pixel
	k in L_1''	10
	τ in L_1''	1
	SGD step per grasps	1
	train SGD step after	the 20th grasps
simulation	batch size	8
	num rotations	8
	onpolicy data aug n	8
	$s_{\text{threshold}}$	0.5cm
	workspace size	0.3^2m
physical robot 500-grasp	batch size	8
	num rotations	8
	onpolicy data aug n	8
	$s_{\text{threshold}}$	1.5cm
	workspace size	0.25^2m
physical robot 1000-grasp	batch size	16
	num rotations	16
	onpolicy data aug n	16
	$s_{\text{threshold}}$	1.5cm
	workspace size	0.25^2m

Table 5: Ablation study

	SR at the 450th grasps		SR at the 1500th grasps	
baseline	avg SR	std	avg SR	std
ours	87.1	0.7	90.7	1.1
no aug buff	82.6	1.9	90.5	1.2
equ ASR	81.6	1.2	83.3	0.8
no FCB	79.0	3.7	87.2	0.6
no prioritizing failure	78.5	1.9	89.3	1.4
egreedy	78.0	3.2	90.9	1.4
no $L_1^{'}$	77.3	4.0	91.0	1.0
no $L_1^{''}$	75.8	0.3	87.4	0.7
no equivariant network	29.6	4.4	43.5	7.8

D HEURISTIC Z AXIS VALUE FOR GRASPING

The Z axis heuristic value is calculated to maximize the probability of success grasp of a object, at the same time protect the gripper from collision. The first goal, maximize the probability of success grasp is implemented by $a_z^{'} = \max(rec) - d_{\text{gripper}}$. Where rec is the image crop corresponding to the rectangle that the gripper pad will sweep when close in the depth image. The d_{gripper} is the depth of the gripper pad. The second goal, protecting the gripper, is implemented by $a_z^{''} = \max(rec_{edge}) - \frac{1}{2}d_{\text{gripper}}$, where rec_{edge} is the edge of rec image. Finally we have $a_z = \max(a_z^{'}, a_z^{''}, 0)$. 0 is the tray button height.

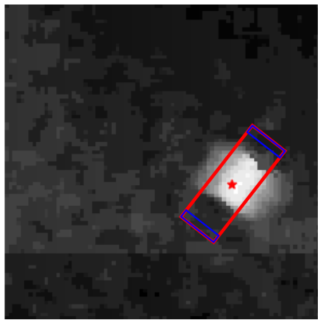


Figure 10: The red box is rec, the two blue boxes are rec_{edge}.

E ABLATION STUDY

In this experiment, each component of our method is ablated to shown the necessity of them. Beside ablation runs, we also included the performance of equivariant ASR Wang et al. (2021) to show our improvement over this method. Table 5 shows the ablation results. At the end of the training (grasps 1500), our method outperformed the equivariant ASR by a margin. Moreover, all the ablated methods converges to the similar success rate. At the one third of the training (grasps 450), our method achieved the success rate of the equivariant ASR at grasps 1500, leading to $\times 3$ sample efficient. At this training step, $L^{''}$ contributes to the learning the most, from this we can see the importance of minimizing the gap between the two networks. The secondary component is $L^{'}$, the correct formulation of the task affects performance when there is limited training data. The third important factor is *Boltzmann exploration*, which generates data close to the target policy.