A Network Architecture

Our equivariant neural networks are implemented using the E2CNN [7] library in PyTorch [33]. The network architecture of the discrete group variations (C_4, C_8, D_4, D_8) is illustrated in Figure 7. The actor network π (Figure 7 top) takes in $\mathcal{F}_s$ as a trivial representation feature map. The hidden layers are all implemented using the regular representation. The output of π is a mixed representation feature map with 1×1 spatial dimension (so that it can also be viewed as a vector). This vector consists of 1 ρ_1 feature, representing the means of the action component (x, y), and 8 trivial representation features, representing the means of the action component (z, θ, λ) as well as the standard deviations of all action dimensions. The critic network q (Figure 7 bottom) takes in $\mathcal{F}_s$ as a trivial representation feature map. The upper path of q generates a 64-channel regular representation feature map with 1×1 spatial dimensions. This regular representation feature map is concatenated with the action a, which is encoded as 1 ρ_1 feature (representing (x, y)) and 3 ρ_0 features (representing (z, θ, λ)). The concatenated feature map is sent to two separate blocks to generate two Q estimates in the form of 1×1 trivial representation feature map. In both the actor and critic network, we use ReLU as the activation function.

For the continuous group variations ($\mathrm{SO}(2), \mathrm{O}_2$), we define the networks with a maximum frequency of 3 (since the continuous group $\mathrm{SO}(2), \mathrm{O}_2$ have infinite number of irreducible representations, the maximum frequency specifies the maximum frequency of the irreducible representations to build [7]). The forms of the input and output feature types are the same as the discrete group variations. However, for the hidden layers, we use $\rho_0^k \oplus \rho_1^k \oplus \rho_2^k \oplus \rho_3^k$ as the representation type for the $\mathrm{SO}(2)$ network. In principle, $\rho_m(g) = \rho_1(mg)$ describes the rotation symmetries in different frequencies. We use $\rho_0^k \oplus \rho_{(1,0)}^k \oplus \rho_{(1,1)}^k \oplus \rho_{(1,2)}^k \oplus \rho_{(1,3)}^k$ for the hidden layers in the $\mathrm{O}(2)$ network. The group $\mathrm{O}(2)$ is generated by rotations $\mathrm{Rot}(\theta)$ and a reflection f over the x-axis. For $k > 0$, $\rho_{(1,k)}$ is the irreducible representation of $\mathrm{O}(2)$ on $\mathbb{R}^2$ in which rotations $\mathrm{Rot}(\theta)$ rotate vectors about the origin by $k\theta$ and f reflects vectors over the x-axis. We use the gated nonlinearity as the activation function.

B Simulation Environment Details

In all simulation environments, the workspace has a size of $0.3m \times 0.3m \times 0.24m$. All workspace is inside a bin with a bounding box size of $0.45m \times 0.45m \times 0.1m$. The bottom size of the bin is $0.3m \times 0.3m$. We use a simulated Franka Panda arm in all simulations. All environments raise a sparse reward, i.e., +1 when the agent finishes the task, and 0 otherwise. The maximum time steps for all tasks is 50. At the beginning of each episode, the gripper is moved to the center of the workspace.

B.1 Block Picking

In the Block Picking environment, the robot needs to pick up a block with a size of $5cm \times 5cm \times 5cm$. At the start of each episode, the block is initialized with random position and orientation. The goal is to grasp the block and lift it s.t. the gripper is at least $0.15m$ above the ground. The observation in this environment covers an area of $0.3m \times 0.3m$.

B.2 Clutter Grasping

In the Clutter Grasping environment, the robot needs to pick up an object from a clutter of at most five objects. At the start of training, five random objects are initialized with random position and orientation. The shapes of the objects are randomly sampled from the object set shown in Figure 9a. The object set contains 76 objects derived from the 3DNet [34] dataset. Every time the agent successfully grasps all five objects, the environment will re-generate five random objects with random positions and orientations. If an episode terminates with any remaining objects in the bin, the object will not be re-initialized. The goal of this task is to grasp any object and lift it s.t. the gripper is at least $0.15m$ above the ground. The observation in this environment covers an area of $0.3m \times 0.3m$.

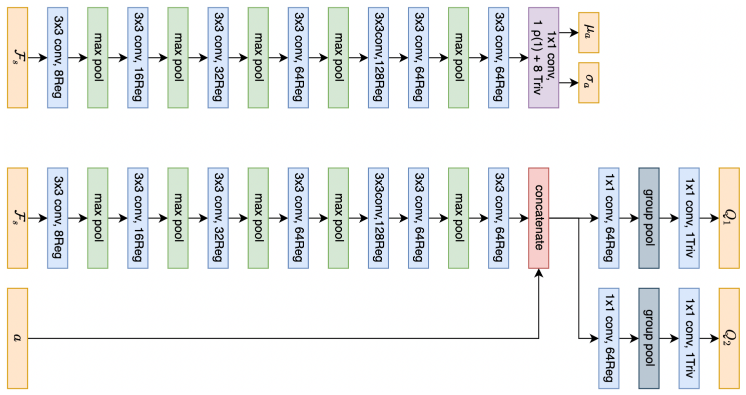


Figure 7: The network architecture of Equivariant SAC. Top: the architecture of the actor network. Bottom: the architecture of the critic network. ReLU non-linearity is omitted in the figure.

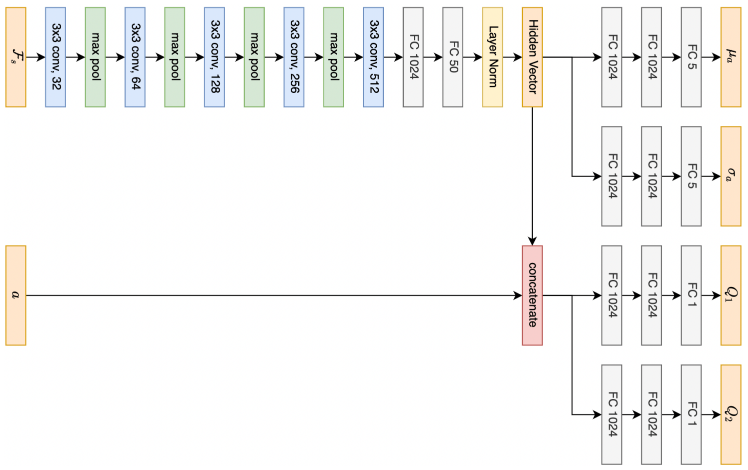


Figure 8: The network architecture of the FERM baseline. ReLU non-linearity is omitted in the figure.

B.3 Block Pushing

In the Block Pushing environment, the robot needs to push a block with a size of $5cm \times 5cm \times 5cm$ to the goal area with a size of $9cm \times 9cm$. At the start of each episode, the block is initialized with a random position and orientation, and the goal is initialized with a random position at least $9cm$ away from the block. The goal is to push the block s.t. the distance between the block's center and the goal's center is within $5cm$. The observation in this environment covers an area of $0.45m \times 0.45m$. The goal area is drawn on the observation by adding $2cm$ to the height values.

B.4 Block in Bowl

In the Block in Bowl environment, the robot needs to pick up a block with a size of $5cm \times 5cm \times 5cm$ and place it inside a bowl with a bounding box size of $16cm \times 16cm \times 7cm$. At the start of each episode, the block and the bowl are initialized with a random position and orientation. The observation in this environment covers an area of $0.45m \times 0.45m$.

C On-Robot Learning Details

In the on-robot learning environment, the two bins have a bottom size of $0.25m \times 0.25m$. The bounding box of each bin has a size of $0.4m \times 0.4m \times 0.11m$. The workspace of the arm is inside one of the two bins, with a size of $0.23m \times 0.23m \times 0.2m$. Similar to simulation environments, all

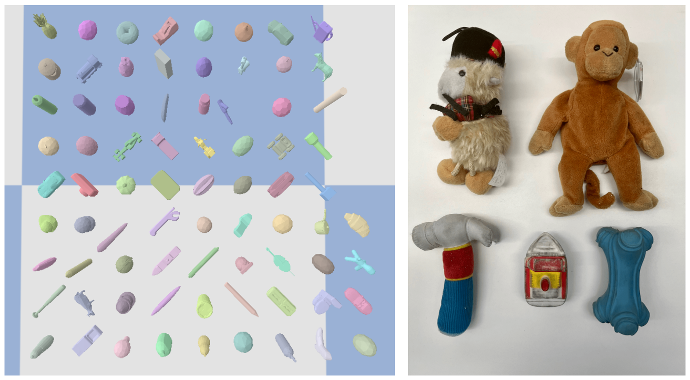

(a) (b)

Figure 9: (a) The object set for Clutter Grasping in simulation contains 76 objects from the 3DNet [34] dataset. (b) The object set for on-robot Clutter Grasping

real-world environments raise a sparse reward, +1 when success and 0 otherwise. We implement a collision protection algorithm to prevent the arm from colliding with the environment by reading the force applied to the end-effector. A -0.1 reward is given when a protective stop is triggered on the UR5. The maximum number of time steps per episode for all environments is 50. Reset and reward functions are implemented specifically for different tasks as follows.

C.1 Block picking

In the block picking environment, only one bin will be used throughout the training. The block has a size of $5cm \times 5cm \times 5cm$. A reward will be given when the block is grasped and lifted for 0.1m above the ground by reading the signals from the gripper. At the start of each episode, the block is reset to a random position and orientation in the current bin. The observation covers an area of $0.3m \times 0.3m$.

C.1.1 Clutter Grasping

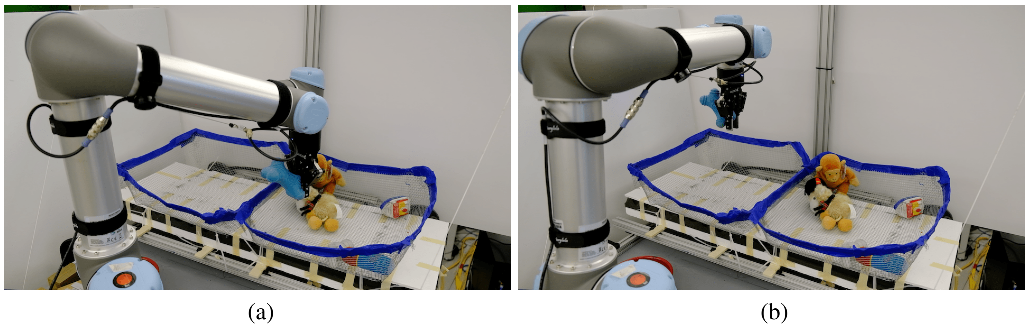
The Clutter Grasping environment contains a clutter of five objects, as is shown in Figure 9b. In this experiment, one bin is used as the active workspace, and the other bin is used as the reset bin. The goal is to grasp an object and lift it to 0.1m above the ground. Once grasped, the object will be reset with random positions and orientations into the reset bin. After all objects in the active workspace are grasped, the active workspace and the reset bin is swapped. This process is shown in Figure 10. If the robot fails to grasp any object for five consecutive episodes, all remaining objects in the active workspace will be moved to the reset bin. The observation covers an area of $0.3m \times 0.3m$.

C.2 Block Pushing

In the Block Pushing environment, only one bin will be used throughout the training. The goal is to move the block s.t. more than 77 pixels of the block are within the goal area. The block has a size of $5cm \times 5cm \times 5cm$. At the beginning of each episode, a goal area of $0.77m \times 0.77m$ is virtually generated at least $9cm$ away from the block. The goal area is drawn to the observation by adding the height value by $0.02m$. When resetting, the arm will move the block to a new random position. The observation covers an area of $0.45m \times 0.45m$, in order to get a larger field of view that includes both object and goal area.

C.3 Block in Bowl

In the Block in Bowl environment, we use one bin as the active workspace and the other bin as the reset bin. The goal is to grasp the block and then put it into the bowl. The bowl has a radius of $8cm$ and the block has a size of $5cm \times 5cm \times 5cm$. At the beginning of each episode, both the block and the bowl will be reset into the reset bin with random positions and orientations, keeping


(a) (b)

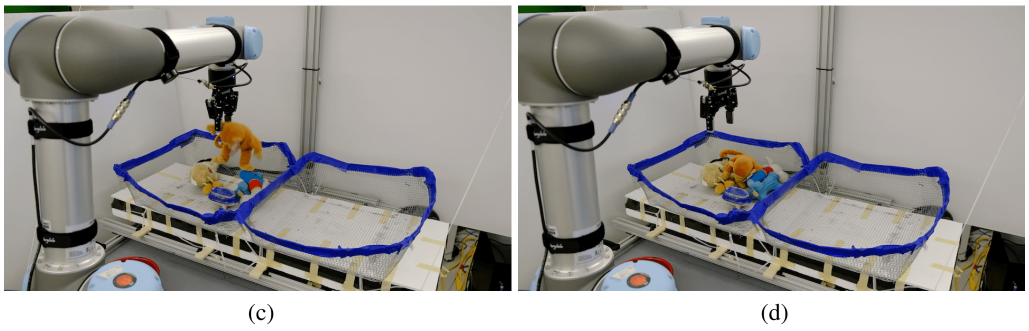

(c) (d)

Figure 10: The reset process of Clutter Grasping. (a) The robot successfully grasps an object in the active workspace (right bin). (b) The robot places the grasped object in the reset bin (left bin). (c) The robot grasps and places the last object from the active workspace (right bin) to the reset bin (left bin). (d) The robot switches the active workspace to the left bin.

at least $14cm$ away from each other. The active workspace and the reset bin are then swapped. This process is shown in Figure 11. The reset function utilizes circle detection for grasping the bowl. The observation covers an area of $0.45m \times 0.45m$ to provide a larger view to cover both the block and the bowl.

D Training Details

The pixel size of the observation is 128×128 for all methods except for the FERM baseline and RAD crop baseline, where the observation's pixel size is 142×142 and will be cropped to 128×128.

We train the networks using the Adam [35] optimizer with a learning rate of 10^{-3}. We perform one SGD step per environmental step. The entropy temperature α of SAC is initialized at 10^{-2}. The target entropy is -5. The discount factor is $\gamma = 0.99$. We use a target network for the critic network and soft target update with $\tau = 10^{-2}$. The replay buffer has a capacity of 100,000 transitions. The mini-batch size is 64.

E Baseline Details

The FERM [27] baselines use random crop for data augmentation. The random crop crops a 142×142 state image to the size of 128×128. As in [27], the contrastive encoder has an output size of 50, and the contrastive encoder is pre-trained for 1.6k steps using the expert data. Figure 8 shows the network architecture for our baseline FERM.

F Buffer Augmentation

In this ablation study, we consider five different numbers of augmentations for each transition: 0, 2, 4, 8, and 16 (0 means no extra augmented transition is added). Figure 12 shows the result. First,

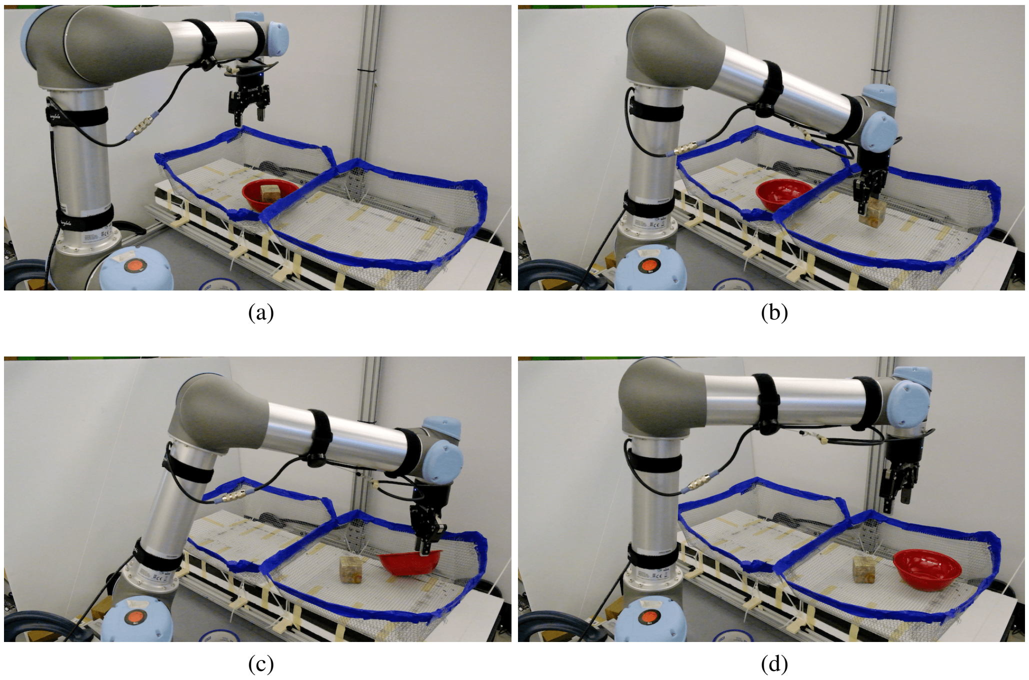


Figure 11: The reset process of Block in Bowl. (a) The robot finishes an episode in the active workspace (left bin). (b) The robot picks up the block and places it in the reset bin (right bin). (c) The robot picks up the bowl and places it in the reset bin (right bin). (d) The robot switches the active workspace to the right bin.

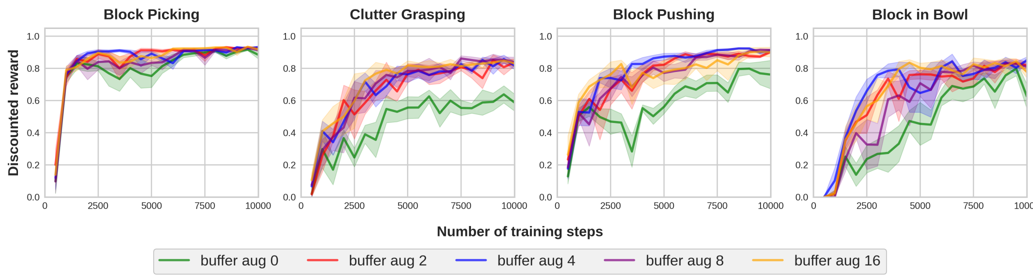


Figure 12: Comparison of Equivariant SAC equipped with rotational data augmentation in the replay buffer. The plots show the performance of the behavior policy in terms of the discounted reward. Each point is the average discounted reward in the previous 500 steps. Results are averaged over four runs. Shading denotes standard error.

note that no augmentation at all (green) is always the worst-performing variation, suggesting that providing extra augmented samples to the agent is beneficial. Second, note that more augmentation does not necessarily mean better performance (e.g., buffer aug 8 (purple) and buffer aug 16 (orange) underperforms buffer aug 4 (blue) in Block in Bowl). Four augmentations (blue) shows the best performance overall.

G Effect of Expert Demonstration

Expert demonstrations are critical when solving challenging sparse rewards tasks. Without it, the agent must search randomly for a goal state and this can take a long time. This section evaluates

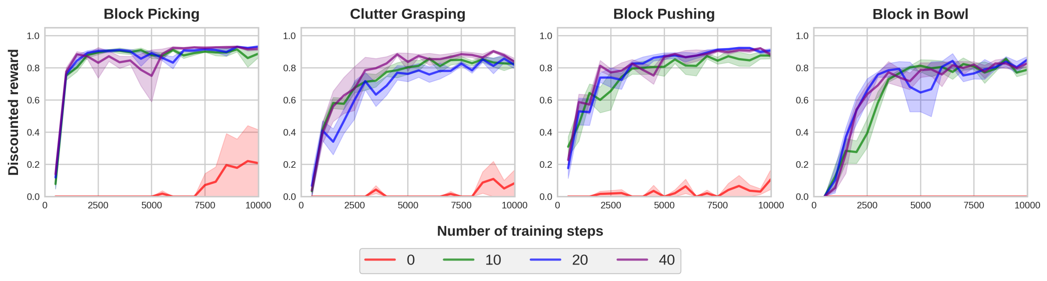


Figure 13: Comparison of Equivariant SAC with different amount of expert demonstration episodes. The plots show the performance of the behavior policy in terms of the discounted reward. Each point is the average discounted reward in the previous 500 steps. Results are averaged over four runs. Shading denotes standard error.

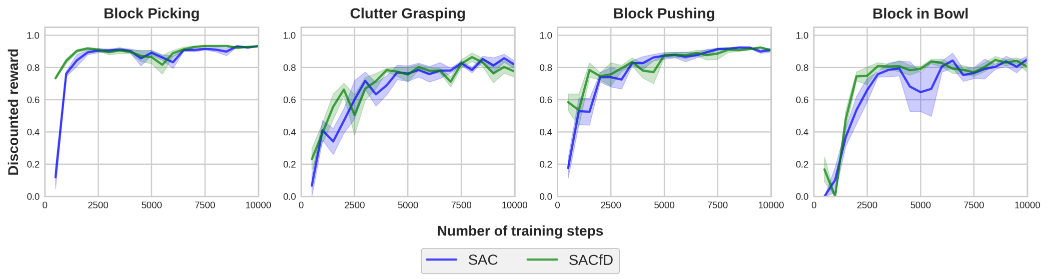


Figure 14: Comparison of Equivariant SAC with Equivariant SAC from Demonstration (SACfD). The plots show the performance of the behavior policy in terms of the discounted reward. Each point is the average discounted reward in the previous 500 steps. Results are averaged over four runs. Shading denotes standard error.

two essential factors in injecting expert demonstration to Equivariant SAC: the number of expert demonstrations needed and if a behavior cloning loss will be beneficial.

G.1 Number of Expert Demonstration

This experiment studies two questions: 1) if expert demonstration is necessary when using Equivariant SAC to solve our tasks; 2) if it is necessary, how many demonstrations are needed. We consider four different amount of expert demonstration episodes provided to the agent: 0, 10, 20, and 40. Figure 13 shows the comparison result. First, note that expert demonstration is always required since the variation without any demonstration (red) struggles to learn a good policy in all four environments. Second, we found that the Equivariant SAC can do well with just 10 or 20 expert demonstrations. The fact that we require so few demonstrations is especially important in situations where it is a human who must provide the demonstrations.

G.2 SACfD

In this experiment, we evaluate if an auxiliary demonstration loss will be beneficial for Equivariant SAC. We use SACfD [5] as the baseline for incorporating a demonstration loss to the actor:

$$\mathcal{L}^{\text{actor}}_{\text{SACfD}} = \mathbb{1}_e \left[\frac{1}{2}((a \sim \pi(s)) - a_e)^2 \right], \tag{1}$$

where $\mathbb{1}_e = 1$ if the sampled transition is an expert demonstration and 0 otherwise, $a \sim \pi(s)$ is an action sampled from the output Gaussian distribution of $\pi(s)$, and a_e is the expert action. This demonstration loss is used in addition to the loss of SAC to incline its policy to the expert policy. Figure 14 shows the comparison between Equivariant SACfD (green) and Equivariant SAC (blue).

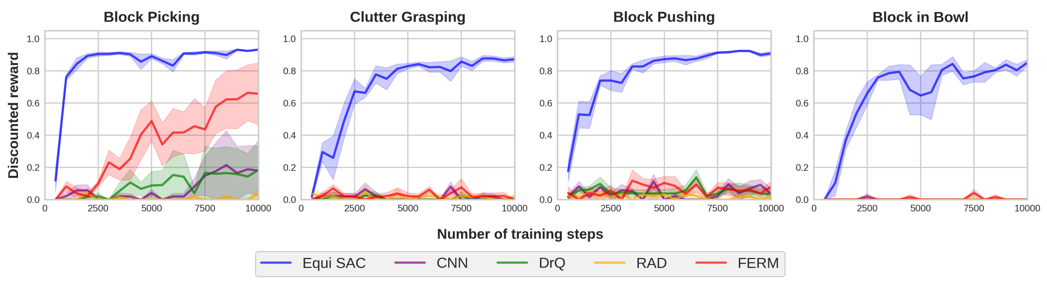


Figure 15: Comparison of Equivariant SAC (blue) with baselines using conventional CNN. The plots show the performance of the behavior policy in terms of the discounted reward. Each point is the average discounted reward in the previous 500 steps. Results are averaged over four runs. Shading denotes standard error.

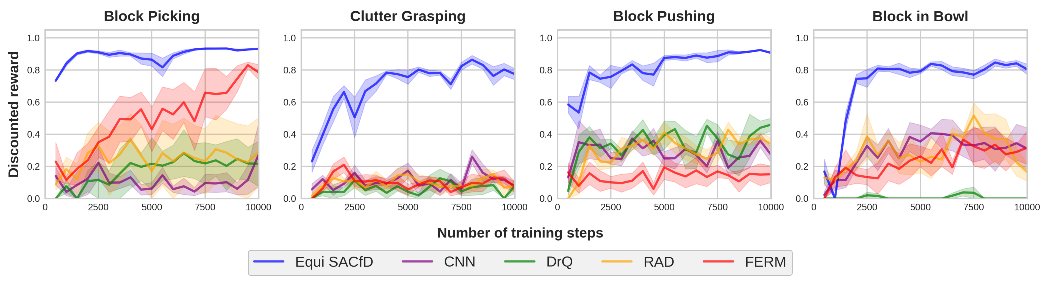


Figure 16: Comparison of Equivariant SACfD (blue) with baselines using conventional CNN. The plots show the performance of the behavior policy in terms of the discounted reward. Each point is the average discounted reward in the previous 500 steps. Results are averaged over four runs. Shading denotes standard error.

Interestingly, Equivariant SACfD outperforms Equivariant SAC in three of the environments, but underperforms in Clutter Grasping. This is because the expert policy for Clutter Grasping is sub-optimal (since the planner does not have access to the optimal grasping point of each random object), while the expert for the other three tasks are nearly optimal. The performance of SACfD indicates that when the expert demonstration is sub-optimal, an auxiliary demonstration loss will harm the performance of Equivariant SAC. This is an important finding because the planner that generates the expert policy in on-robot learning is often sub-optimal (since it does not have access to the pose of the objects as in the simulation).

H Comparing Equivariant SAC with CNN baselines

In this experiment, we compare the Equivariant SAC (Equi SAC) with some sample efficient baselines using standard convolutional neural networks and data augmentation. We consider five baselines: 1) CNN: standard SAC [28] implemented using conventional CNN instead of equivariant networks. 2) DrQ [36]: 1) with random shift data augmentation. At each training step, both the Q-targets and the TD losses are calculated by averaging over two random-shift augmented transitions. 3) RAD [32]: 1) with random crop data augmentation. At each training step, each transition in the minibatch is applied with a random crop data augmentation. 4): FERM [27] similar architecture as 1) with an extra contrastive loss term that learns an invariant encoder from random crop augmentation. As is shown in Figure 15, Equivariant SAC (blue) significantly outperforms all baselines, where non of the baselines can solve Clutter Grasping, Block Pushing and Block in Bowl.

We further perform the same experiment using SACfD (Appendix G.2) instead of SAC. As is shown in Figure 16. SACfD improves the performance of the CNN baselines, however, they still underperform Equivariant SACfD (blue) dramatically.