# LEARNING SYMMETRIC REPRESENTATIONS FOR EQUIVARIANT WORLD MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Encoding known symmetries into world models can improve generalization. However, identifying how latent symmetries manifest in the input space can be difficult. As an example, rotations of objects are equivariant with respect to their orientation, but extracting this orientation from an image is difficult in absence of supervision. In this paper, we use equivariant transition models as an inductive bias to learn symmetric latent representations in a self-supervised manner. This allows us to train non-equivariant networks to encode input data, for which the underlying symmetry may be non-obvious, into a latent space where symmetries may be used to reason about outcomes of actions in a data-efficient manner. Our method is agnostic to the type of latent symmetry; we demonstrate its usefulness over $C_4 \times S_5$ using $G$-convolutions and GNNs, over $D_4 \ltimes (\mathbb{R}^2, +)$ using $E(2)$-steerable CNNs, and over $\mathrm{SO}(3)$ using tensor field networks. In all three cases, we demonstrate improvements relative to both fully-equivariant and non-equivariant baselines.

## 1 INTRODUCTION

Symmetry has proved to be a powerful inductive bias for improving generalization in supervised and unsupervised learning. The last several years have seen many proposed $G$-equivariant models which embed symmetry constraints within deep neural networks (Cohen & Welling, 2016a;b; Cohen et al., 2019; Weiler & Cesa, 2019; Weiler et al., 2018; Kondor & Trivedi, 2018; Bao & Song, 2019; Worrall et al., 2017). In addition to improved generalization, these models are more parameter efficient, sample efficient, and behave more consistently in new environments which often makes them safer to use in real-world applications.

However, a major impediment to the wider application of equivariant models is that the input data does not always have an obvious symmetry. As an example, let us consider the two pairs of images in Figure 1. For the top pair of MNIST digits, 2D rotations in pixel space should induce a corresponding rotation in feature space. Here it is possible to achieve state-of-the-art accuracy using an $E(2)$-equivariant network (Weiler & Cesa, 2019). By contrast, exploiting the underlying symmetry is much more challenging for the bottom pair of 3D images of a car. For these images, there is also an underlying symmetry group of rotations, but the action of this group on 3D images is nontrivial.

In this paper, we consider the task of learning symmetric representations of data in domains where the transformation law cannot be hard-coded. To do so, we propose to train a standard network to learn a mapping from an input space, for which the group action is difficult to characterize, into a latent space, for which the symmetry group is known. This mapping, which we refer to as a symmetric embedding network, can then be composed with any equivariant network for downstream predictions.

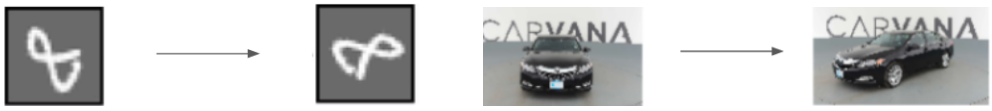



Figure 1: On MNIST, the rotation is easy compute allowing for equivariant models. The rotation is difficult to compute for the car, so equivariant methods do not apply (Carvana, 2017).

As a concrete instantiation of this idea, we focus on learning world models, i.e. models that encode the effects of actions in the state space of a planning domain. In planning domains we typically have

access to a training dataset of triples that comprise a state, an action, and a next state. This makes it possible to combine the symmetric embedding network with an equivariant transition network, which together can be trained end-to-end by minimizing a contrastive loss (Kipf et al., 2019). The symmetry group of the transition model hereby acts as an inductive bias guides the embedding network to a representation that is as equivariant as possible. At the same time, learning transition dynamics reflect the underlying symmetries of the domain has the potential to improve both data efficiency and out-of-distribution generalization. This is crucial in applications to robotics, where acquiring training data is often time-consuming and expensive, and where this data is typically not representative of the optimal policy for a test-time task.

While the idea of learning symmetric embedding networks has, to our knowledge, not previously been proposed or demonstrated, our application of this idea to learning equivariant world models builds on a number of existing lines of work. Note in particular that we are not proposing a new equivariant neural network design. In fact, our approach is useful precisely because it can be paired with any existing equivariant neural network to extend its usefulness to new domains with unspecified group actions. We apply our method to 3 different domains, 3 different symmetry groups, and 3 different equivariant architectures. (1) We use $G$-convolutions (Cohen & Welling, 2016a) and GNNs (Scarselli et al., 2008) to learn $C_4 \times S_5$-equivariant features for a moving blocks environment, (2) we use $E(2)$-steerable CNN (Weiler & Cesa, 2019) to learn $E(2)$-equivariant representations for the MuJoCo Reacher environment (Todorov et al., 2012), (3) we use tensor field networks (Thomas et al., 2018) to learn $SO(3)$-equivariant features for a 3D rotation environment.

We summarize our contributions as follows:

- We introduce a meta-architecture for equivariant world models with symmetric embedding networks, which can be trained end-to-end using triples from a replay buffer by minimizing a contrastive loss.
- We demonstrate that this meta-architecture can be use to learn world models with a variety of equivariances in a self-supervised manner.
- Moreover, we show that models that have been trained using only a subset of all input actions can generalize to unseen input actions at test time.

## 2 BACKGROUND

We highlight the difference between abstract symmetry groups and their concrete representations. This work occupies a middle ground between equivariant neural networks in which the group and its representations are known and symmetry discovery models such as (Zhou et al., 2020) in which neither the group nor its representation is known. Here we assume we know the abstract group, but only some of its relevant representations.

**Groups, Actions, and Representations** A symmetry *group* consists of a set $G$ together with a binary composition operation $\circ\colon G \times G \to G$. The group must be closed under composition, contain an identity $1 \in G$ and each element $g \in G$ must invertible with respect to composition. An action of the group $G$ on a set $S$ is a map $a\colon G \to \mathrm{Perm}(S)$ mapping each element of the group $g$ to a permutation $\pi_g \in \mathrm{Perm}(S)$ of the elements of $S$. Composition of group elements is compatible with the action such that $a(g_1 g_2, s) = a(g_1, a(g_2, s))$ for $g_1, g_2 \in G, s \in S$. A real *representation* of the group $G$ is a linear group action, given by a map $\rho\colon G \to \mathrm{GL}_n(\mathbb{R})$ which maps each element of $G$ to an invertible $n \times n$ matrix. The multiplication table of these matrices must match that of the abstract group elements under composition. That is, $\rho(g_1 \circ g_2) = \rho(g_1)\rho(g_2)$. See Hall (2003) for additional background on groups and their representations.

**Equivariant and Invariant Functions** Given a function $f\colon X \to Y$ between vector spaces $X$ and $Y$ and a group $G$ equipped with a representations $\rho_X$ and $\rho_Y$ acting on $X$ and $Y$ respectively, we say $f$ is *equivariant* if for all $x \in X, g \in G$ we have $f(\rho_X(g) \cdot x) = \rho_Y(g) \cdot f(x)$. This means that if the input is transformed by $g$ the output will be transformed correspondingly. The composition of equivariant functions is equivariant. Thus we can model equivariant functions using equivariant neural networks which alternate equivariant linear layers and equivariant non-linearities.

**World Models** World models learn state representations in a self-supervised fashion and ignore unnecessary information unrelated to predicting environment dynamics. Let $\mathcal{S}$ be the state space of the environment and $\mathcal{A}$ be action space of the agent. We consider a deterministic transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which outputs the next state $s' = T(s, a)$ resulting from taking action $a$ in state $s$. Our goal is to learn the transition model $T$ from tuples $(s, a, s')$ collected from offline data. As the space $\mathcal{S} \times \mathcal{A}$ is combinatorially large, it is useful to be able to learn a compact representation of the state and an accurate model of $T$ that can generalize to unseen transitions. This may be done by learning a state abstraction map $\mathcal{S} \to \mathcal{Z}$ and then learning transitions in latent space $T_{\mathcal{Z}} \colon \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$.

## 3 Related Work

**Equivariant Neural Networks** Different equivariant neural networks methods have been devised to impose symmetry with respect to various groups across different data types and in many different application areas. However, in all cases, equivariant neural networks have been limited by the requirement that the group $G$ is known and the group action on input, output, and hidden spaces is explicitly constructed. Our method, in contrast, extends to cases where the abstract symmetry group is known but the action on the input space is not. Equivariant neural networks vary considerably using different constructions for equivariance including $G$-convolution (Cohen & Welling, 2016a), $G$-steerable convolutions (Cohen & Welling, 2016b; Weiler & Cesa, 2019), tensor product and Clebsh-Gordon decomposition (Thomas et al., 2018), or convolution in the fourier domain (Esteves et al., 2017). They have been applied to many data types such as gridded data (Weiler & Cesa, 2019), spherical data (Cohen et al., 2018), point clouds (Dym & Maron, 2020), and sets (Maron et al., 2020). They have found applications in many domains including molecular dynamics (Anderson et al., 2019), particle physics (Bogatskiy et al., 2020), and trajectory prediction (Walters et al., 2020).

**Learning Symmetry** Since our method assumes knowledge of the group $G$ but not all group actions $\rho$, it sits between equivariant neural networks, which assume a known symmetry group $G$ and group actions $\rho$, and symmetry discovery methods, which attempt to learn both the symmetry group $G$ and actions $\rho$ from data. For example, Zhou et al. (2020) learn equivariance by learning a parameter sharing scheme using meta-learning. Dehmamy et al. (2021) similarly learn a basis for Lie algebra generating a symmetry group at the same time they learn parameters for the equivariant convolution over this symmetry group. Benton et al. (2020) propose an adaptive data augmentation scheme, in which they learn which group of spatial transformations best supports data augmentation.

**Contrastive World Models** For high-dimensional image inputs, latent world models are frequently used to encode abstract representations which in turn are used to predict environment dynamics. Such models usually employ auxiliary losses such as reconstruction or other pixel-level objectives (Ha & Schmidhuber, 2018; Watter et al., 2015; Hafner et al., 2019; 2020) which can be computationally costly. To avoid pixel-level losses, we use world models with contrastive learning as in (Kipf et al., 2019). These authors consider the inductive bias that the MDP action space and abstract state space are factored over objects as $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_k$ and $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_k$. Their model consists of a CNN, object shared MLP, and GNN. This may be considered as a special case of our method where the group $G$ is the symmetric group $S_n$, the CNN plays the role of symmetric embedding and the shared MLP and GNN are $S_n$ equivariant. We learn symmetric representations for groups $G$ besides $S_n$ and explicitly enforce $G$-equivariance constraints to latent transition networks.

## 4 Symmetric Embeddings for Equivariant World Models

Our method gives a general template that can be fit to different equivariant neural networks, symmetries, and data types. We describe the general template and the implementation in examples.

### 4.1 Model Overview

**Equivariant World Models** To learn world models using fewer samples, we exploit symmetries inherent to the MDP. We consider the case of *MDPs with symmetry* as in van der Pol et al. (2020). Let $G$ be a group of symmetries with group representations $\rho_{\mathcal{S}}$ and $\rho_{\mathcal{A}}$. The transition function $T$ is equivariant if $T(\rho_{\mathcal{S}}(g) \cdot s, \rho_{\mathcal{A}}(g) \cdot a) = \rho_{\mathcal{S}}(g) \cdot T(s, a)$. We would like to enforce this constraint on

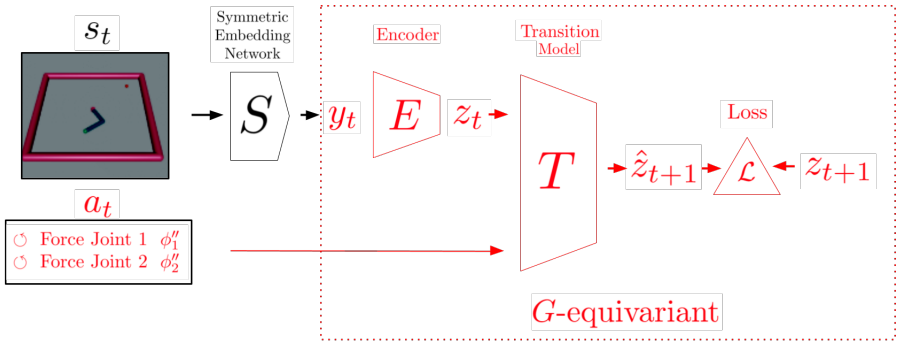



Figure 2: Diagram of model architecture of $G$-equivariant world model. The features in red have an explicit $G$-action $\rho$. The networks in red are $G$-equivariant. The example *Reacher* input has $G = D_4$ symmetry. The MDP actions have $G$-representation type $\rho_{\text{flip}}$ meaning they are reversed in sign by reflections and unaltered by rotations. The Symmetric Embedding is a CNN and the Encoder and Transition model are $E(2)$-CNNs with fiber group $D_4$.

a neural network model for $T$. If trained to predict $s' = T(s, a)$, the model will then automatically generalize to $gs' = T(gs, ga)$, thus enabling improved generalization and sample efficiency.

Current methods for constructing model classes of equivariant neural networks require that one have available $\rho_{\mathcal{S}}$ and $\rho_{\mathcal{A}}$ as explicit functions, so one can construct and compose equivariant linear and non-linear layers with respect to these group representations. In our case, however, although we assume the symmetry group $G$ and group action $\rho_{\mathcal{A}}$, ***we assume that $\rho_{\mathcal{S}}$ is not explicitly known.*** In many environments, $s$ is a pixel-level input and the transformation $\rho_{\mathcal{S}}$ may be very difficult to describe. Thus, we cannot directly enforce symmetry for $T$ using an equivariant neural network.

**Meta-Architecture** To learn an equivariant world model without access to $\rho_{\mathcal{S}}$, we learn a symmetric abstract state mapping from states $s$ to abstract states $z$ in a space $\mathcal{Z}$ with an explicit action $\rho_{\mathcal{Z}}$ of the symmetry group $G$. We then learn a transition model in latent space where can enforce symmetry using an equivariant neural network.

We learn the symmetric abstract state mapping in two parts. First we map the pixel-space state to an intermediate space $\mathcal{Y}$ using a *symmetric embedding* $S\colon \mathcal{S} \to \mathcal{Y}$. This is a non-equivariant neural network which maps observations into a space $\mathcal{Y}$ with an explicit symmetry group action $\rho_{\mathcal{Y}}$. We then map the intermediate space to the low-dimensional latent space using an *equivariant encoder* $E\colon \mathcal{Y} \to \mathcal{Z}$. Lastly, we compute the *transition* in latent space, $T\colon \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. We explicitly enforce that $E$ and $T$ are equivariant using equivariant neural networks.

We employ a contrastive loss as in Kipf et al. (2019). Let $(s, a, s')$ be a ground truth transition triplet and $s''$ be an incorrect next state $s'' \neq s'$. Let $z = E(S(s))$, $z' = E(S(s))$, and $z'' = E(S(s))$, then

$$\mathcal{L}(s, a, s', s'') = \|T(z, a) - z'\| + \alpha \max(\beta - \|T(z, a) - E(z'')\|, 0).$$

The loss pushes $T(z, a)$ towards the true next state $z'$ and away from the incorrect sample $z''$.

**Symmetric Embedding Network $S$** The symmetric embedding network takes the pixel-level input $s$ with unknown group action $\rho_{\mathcal{S}}$ and maps it to a vector $y$ with explicit representation $\rho_{\mathcal{Y}}$. We use CNNs for all environments but the specific architecture vary (see Appendix B.2).

In the case of object-centric environments with 5 objects and $\pi/2$-rotation symmetry, the symmetry group is $G = C_4 \times S_5$, the cyclic group of order 4 and the permutation group on 5 objects. The network $S$ is not equivariant, but the output $y$ has shape $[B, C, 5, 4]$ and caries an action $\rho_{\mathcal{Y}}$ of $S_5$ by permuting the 5-dimensional axis and of $C_4$ by cyclically permuting the 4-dimensional axis.

**Equivariant Encoder $E$ and Transition Model $T$** The equivariant encoder further maps the output $y$ to a much lower dimensional latent vector $z$. The action $a$ and latent state $z$ are input to an equivariant latent space transition model $T$ which encodes only necessary information required for the transition model to correctly predict the next latent state $z'$.

The architecture varies with group $G$. In the object-centric environments with $G = C_4 \times S_5$, the equivariant encoder $E$ is shared over all 5 objects and uses group convolution over the group $C_4$ (Cohen & Welling, 2016a), thus achieving $C_4 \times S_5$-equivariance. The transition function $T$ is implemented as a GNN with edge and node networks which use $C_4$-convolutions for their linear layers. Since GNNs are $S_5$ equivariant and the linear layers within the GNNs are $C_4$-equivariant, this is $C_4 \times S_5$-equivariant. For other implementations, see Table 1 and Appendix B.2.

### 4.2 SO(3)-Structured Symmetric Embeddings

**Symmetric Embedding** In the case of $G = \mathrm{SO}(3)$ symmetry, we expect the symmetric embedding network to detect the pose of object $z$ in 3D. We omit the equivariant encoder $E = \mathrm{id}$ and instead use a two-part Symmetric Embedding Network that directly encodes $y = z$ using a down-sampling CNN whose flattened output is passed to an MLP, and converted to an element of $\mathrm{SO}(3)$.

To force the output of the symmetric embedding network $y$ to be an element of $\mathrm{SO}(3)$, we have the last layer output 2 vectors in $u, v \in \mathbb{R}^3$ and perform Gram-Schmidt orthogonalization. This method is also used by Falorsi et al. (2018), who conclude it produces less topological distortion than alternatives. Only two vectors are necessary since orthogonality and orientation determine the third,

$$u' = u/\|u\|, \qquad v' = \frac{v - (u' \cdot v)u'}{\|v - (u' \cdot v)u'\|},$$
$$w' = u' \times v', \qquad y = [u'\ v'\ w'].$$

**Transition** We implement an equivariant transition model using Tensor Field Networks (Thomas et al., 2018; Geiger et al., 2020). This is an $\mathrm{SO}(3) \ltimes (\mathbb{R}^3, +)$-equivariant method which works over point clouds. Here $z \in \mathrm{SO}(3)$ and $a \in \mathrm{SO}(3)$. We consider $z$ as 3 points in $\mathbb{R}^3$ and add the origin to get a 4 point could. We embed the actions as features over these 4-points. The MDP action $a$ is then set as a feature over these 4 points, which has $\mathrm{SO}(3)$-representation $\rho_{\mathcal{A}}(g) \cdot a = gag^{-1}$.

The MDP action $a \in \mathrm{SO}(3)$ is a rotation matrix and the latent state $z \in \mathrm{SO}(3)$ is a positively-oriented orthogonal coordinate frame. Though $\mathcal{Z} = \mathcal{A} = \mathrm{SO}(3)$, these different semantics lead to differing $G = \mathrm{SO}(3)$ actions with $\rho_{\mathcal{Z}}(g)(z) = g \cdot z$ but $\rho_{\mathcal{Z}}(g)(a) = gag^{-1}$. If $z$ is correctly learned, then the ground truth latent transition function can be represented by a simple matrix multiplication $T_{\mathcal{Z}}(z, a) = az$ which is also equivariant,

$$T_{\mathcal{Z}}(\rho_{\mathcal{Z}}(g)(z), \rho_{\mathcal{A}}(g)(a)) = (gag^{-1})(gz) = gaz = \rho_{\mathcal{Z}}(g)T_{\mathcal{Z}}(z, a).$$

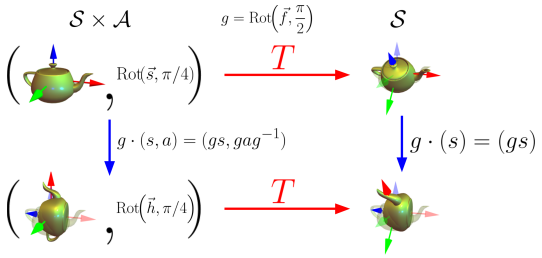



Figure 3: $\mathrm{SO}(3)$-equivariance of the transition function for $\mathrm{SO}(3)$ object manipulation.

### 4.3 Generalizing over the MDP Action Space

Although the state does not have a known group actions $\rho_{\mathcal{S}}$, the MDP action does have known $\rho_{\mathcal{A}}$. In the domains we consider, although the state is high-dimensional and has non explicit symmetry, the action is low dimensional and has clear symmetry. The MDP action is input directly to $T$ and thus bypasses the non-equivariant part $S$ of the neural network. Since the neural network is explicitly equivariant with respect to the MDP action, it is thus feasible to train the neural network using only a proper subset $\mathcal{A}' \subset \mathcal{A}$ of the action space, and then test on the entire $\mathcal{A}$. This may be useful in domains in which data collection is costly. Since the samples from $\mathcal{S}$ are still i.i.d., the non-equivariant neural network $S$ is still able to learn well.

We require that $\rho_{\mathcal{A}}(G) \cdot \mathcal{A}' = \mathcal{A}$. That is, every MDP action is $G$-transformed version of one in $\mathcal{A}'$. We assume after training $S$ is approximately equivariant, which since $T$ and $E$ are constrained to be equivariant implies $T_{\mathcal{S}}(s, a) = T(E(S(s)), a)$ is equivariant. Assume also error for $T$ is low for the restricted action set $(s_1, a', s_2) \in \mathcal{S} \times \mathcal{A}' \times \mathcal{S}$. Then given $(s_1, a, s_2) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, there exists $g \in G$ such that $a = \rho_{\mathcal{A}}(g) \cdot a'$. Let $s'_i = \rho_{\mathcal{S}}(g^{-1})s_i$. Then

$$T_{\mathcal{S}}(s, a) \approx T_{\mathcal{S}}(\rho_{\mathcal{S}}(g) \cdot s'_1, \rho_{\mathcal{A}}(g) \cdot a') \approx \rho_{\mathcal{S}}(g) \cdot T_{\mathcal{S}}(s'_1, a') \approx \rho_{\mathcal{S}}(g) \cdot s'_2 \approx s_2.$$

If performance is good for $\mathcal{A}'$ and $S$ has low equivariance error, performance will be good for $\mathcal{A}$.

# 5 EXPERIMENTS

We perform experiments on environments with different symmetries. For all environments, we compare our method against a non-equivariant model and a fully equivariant model, where all three components are equivariant, but an incorrect $\rho_{\mathcal{S}}$ is assumed. We show that our method can match the performance of a non-equivariant model, even in environments where the observations are skewed. More importantly, we show that our model is able to generalize better to unseen dynamics than non-equivariant and fully-equivariant models.

## 5.1 ENVIRONMENTS

We evaluate on the 2D shapes and 3D blocks grid world environments (Kipf et al., 2019), a variant of the 2D grid world, ***Rush Hour***, where objects can move relative to their orientation, a variant of the `Reacher-v2` MuJoCo environment where the goal position is fixed, and a 3D teapot rotation environment. All environments use pixels as observed states ($50$px color images for 2D shapes, 3D blocks, Rush Hour, $128$px for Reacher, and $64$px for 3D teapot). For Reacher, the previous and current frames are stacked as an observation to encode velocities. Additional details on the environments are given in the Appendix B.1.

## 5.2 MODEL ARCHITECTURES AND TRAINING

Each environment contains different symmetries and thus we customize the model architecture for every environment. Object-oriented structured models that factorize the latent state space and latent action space over objects were used for the grid world environments of 2D shapes, 3D blocks, and Rush Hour. Although the objects and actions are factorized, the world model must account for the pairwise interactions between objects (e.g. actions to move one object can be blocked by another object). These grid world environments admit rotational and also permutation symmetries of the objects. The Reacher and 3D Teapot environments do not consider objects. We use the dihedral group for Reacher and $\mathrm{SO}(3)$ for 3D Teapot. A summary of the environments, different symmetries, representation types, and model architectures are given in Table 1.

A random policy was used to create training and evaluation datasets of $(s, a, s')$ tuples. We do not consider the reward as our focus is on constructing accurate latent representations and their dynamics. For training, we use $1000$ episodes of length $100$ as training data for the grid world environments (2D shapes, 3D blocks, Rush Hour), $2000$ episodes of length $10$ for Reacher, and $100{,}000$ episodes of length $1$ for the 3D teapot. For Reacher, the starting state is restricted to a subset of the whole state space, so we perform warm starts with $50$ random actions in order to generate more diverse data. The evaluation datasets are generated with different seeds from the training data to ensure that transitions are different. For all environments, we have either a combinatorially large state space (with objects) or continuous states and thus overlap is highly unlikely.

As equivariant networks have more parameters than the non-equivariant counterparts, we reduce the number of hidden dimensions accordingly to keep the number of parameters approximately constant for all models. The Adam (Kingma & Ba, 2014) optimizer was used for all experiments. All other specific implementation details are provided in Appendix B.2.

## 5.3 METRICS

We use standard metrics from Kipf et al. (2019), modified versions of these metrics to adapt to continuous state spaces, and two metrics for evaluating the equivariance of the learned model.

| Environment | 2D Shapes & 3D Blocks | Rush Hour | Reacher | 3D Teapot |
|---|---|---|---|---|
| Observation $s$ | 50x50x3 | 50x50x3 | 128x128x3x2 | 64x64x1 |
| Action $a$ | { up,right,down,left } | { fwd,left,back,right } | $(\phi_1'', \phi_2'') \in \mathbb{R}^2$ (joint forces) | SO(3) |
| Symmetry $G$ | $C_4 \times S_5$ ($\frac{\pi}{2}$ rot.; obj. perm.) | $C_4 \times S_5$ ($\frac{\pi}{2}$ rot.; obj. perm.) | $D_4 \ltimes (\mathbb{R}^2, +)$ ($\frac{\pi}{2}$ rot; flip; trans.) | SO(3) |
| $\mathcal{Z}$-rep: $\rho_{\mathcal{Z}}$ | $(\rho_{\text{std}}, \mathbb{R}^2) \boxtimes (\rho_{\text{std}}, \mathbb{R}^5)$ | $(\rho_{\text{std}} \oplus \rho_{\text{reg}}, \mathbb{R}^6) \boxtimes (\rho_{\text{std}}, \mathbb{R}^5)$ | $(\rho_{\text{reg}}, \mathbb{R}^8)^4 \boxtimes \rho_{\text{triv}}$ | $gz$ (matrix mult.) |
| $\mathcal{A}$-rep: $\rho_{\mathcal{A}}$ | $(\rho_{\text{reg}}, \mathbb{R}^4) \boxtimes (\rho_{\text{std}}, \mathbb{R}^5)$ | $(\rho_{\text{triv}}, \mathbb{R})^4$ | $(\rho_{\text{flip}}, \mathbb{R})^2$ | $gag^{-1}$ (conjugation) |
| Non-Equ. Extractor | 2-layer CNN (2D) 4-layer CNN (3D) | 2-layer CNN | 7-layer CNN | 4 conv, 3 FC layers |
| Equ. Encoder | MLP + $C_4$-conv | MLP + $C_4$-conv | 3 $E(2)$-conv, 3 $D_4$-FC layers | Id. (None) |
| Equ. Transition | GNN + $C_4$-conv Cohen & Welling (2016a) Scarselli et al. (2008) | GNN + $C_4$-conv | MLP + $E(2)$-CNN Weiler & Cesa (2019) | MLP + Tensor Field Geiger et al. (2020) *or* matrix mult. |

Table 1: The symmetry and implementation for each domain. See Appendix C for the $\rho$ definitions.

**Hits, Hard Hits, and MRR.** In order to evaluate model performance in latent space, the ranking metrics from Kipf et al. (2019) are used. The evaluation samples are ranked according to the pairwise $L_2$ distance of the predicted next states and the true next states (both are encoded in latent space). Hits at Rank $k$ (H@k) measures the average percentage of time that the predicted next state is within $k$-nearest neighbors of the encoded true next state. The mean reciprocal rank (MRR) is the average inverse rank. We also consider a variant of Hits at Rank $k$ (HH@k) where we generate negative samples $s'_n$ of states that are close to the true next state $s'$. We compare the pairwise distance of the predicted next state and the positive sample (true distance) and the minimum pairwise distance of the predicted next state and the negative samples (false distance) and count the number of times that the true distance was lower than the false distance. This is a harder version of H@k as the model must distinguish between close negative samples and the true positive sample in latent space.

**Equivariance Error** In order to analyze the equivariance of each model, we generate a version of the evaluation dataset where one element of the symmetry group acts on the tuple $(s, a, s')$ and calculate the true equivariance error for the embedding network. Although by assumption $\rho_{\mathcal{S}}(g) \cdot s$ cannot be computed using $g$ and $s$, our synthetic datasets allow us to render both $s$ and $\rho_{\mathcal{S}}(g) \cdot s$ during generation. Specifically, the equivariance error of the symmetric embedding is calculated as

$$\text{EE} = \mathbb{E}_{s,g} \left[ |\rho_{\mathcal{Y}}(g) \cdot S(s) - S(\rho_{\mathcal{S}}(g) \cdot s)| \right].$$

**Distance Invariance Error** The above equivariance error can always be applied to the symmetric embedding network when its output space is spatial and we can manually perform group actions on the outputs. However it cannot be applied to the latent space $\mathcal{Z}$ in the case of non-equivariant models since the group action on the latent space $\rho_{\mathcal{Z}}$ cannot be meaningfully defined.

We therefore propose a proxy for the equivariance error using invariant distances. For a pair of input states $s, s'$, an equivariant model $f$ will have the same distances $\|f(s) - f(s')\|$ and $\|f(gs) - f(gs')\|$ assuming the action of $G$ is norm preserving as it is for all transformation considered in the paper. Due to the linearity of the action, $\|f(gs) - f(gs')\| = \|gf(s) - gf(s')\| = \|g(f(s) - f(s'))\| = \|(f(s) - f(s'))\|$. The distance invariance error is computed as

$$\text{DIE} = \mathbb{E}_{s,s',g} \left[ |\|f(s) - f(s')\| - \|f(gs) - f(gs')\|| \right].$$

We apply this metric for the symmetric embedding network and for the entire model.

| | Model | Hits@1 (10 step, %) | MRR (10 step, %) | EE($S$) | DIE($S$) ($\times 10^{-2}$) | DIE (model) (10 step, $\times 10^{-2}$) |
|---|---|---|---|---|---|---|
| 3D Blocks | None | $86.5_{\pm 3.0}$ | $50.6_{\pm 3.1}$ | $1.22_{\pm 0.1}$ | $6.54_{\pm 1.9}$ | $6.95_{\pm 1.5}$ |
| | Full | $89.4_{\pm 11}$ | $61.8_{\pm 13}$ | $1.18_{\pm 0.1}$ | $3.62_{\pm 0.8}$ | $4.87_{\pm 1.4}$ |
| | Ours | $90.8_{\pm 4.5}$ | $59.4_{\pm 4.6}$ | $1.28_{\pm 0.1}$ | $5.45_{\pm 0.9}$ | $4.95_{\pm 0.6}$ |
| Rush Hour | None | $95.8_{\pm 0.7}$ | $97.9_{\pm 0.3}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ |
| | Full | $92.3_{\pm 2.3}$ | $96.0_{\pm 1.2}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ |
| | Ours | $92.4_{\pm 5.6}$ | $96.0_{\pm 3.0}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ | $0.0_{\pm 0.0}$ |

Table 2: Model Performance on 3D Blocks and Rush Hour.

| | Model | H@10 (1 step, %) | MRR (1 step, %) | EE($S$) | DIE($S$) ($\times 10^{-2}$) | DIE (model) (1 step) |
|---|---|---|---|---|---|---|
| Reacher | None | $100_{\pm 0.0}$ | $88.3_{\pm 3.3}$ | $1.26_{\pm 0.1}$ | $4.53_{\pm 1.1}$ | $0.56_{\pm 0.2}$ |
| | Full | $100_{\pm 0.0}$ | $95.5_{\pm 1.9}$ | $1.19_{\pm 0.0}$ | $3.51_{\pm 0.7}$ | $0.39_{\pm 0.1}$ |
| | Ours | $100_{\pm 0.0}$ | $94.1_{\pm 2.8}$ | $1.29_{\pm 0.0}$ | $4.05_{\pm 0.7}$ | $0.52_{\pm 0.1}$ |

Reacher

| | HH@1 (1 step, %) | EE($S$) |
|---|---|---|
| None (S) | $0.1_{\pm 0.1}$ | $2.34_{\pm 0.1}$ |
| MatMul (S) | $31.6_{\pm 1.2}$ | $2.39_{\pm 0.0}$ |
| TFN (S) | $39.7_{\pm 1.2}$ | $2.22_{\pm 0.1}$ |
| None (L) | $7.4_{\pm 1.4}$ | $2.409_{\pm 0.0}$ |
| MatMul (L) | $100_{\pm 0.0}$ | $0.05_{\pm 0.0}$ |
| TFN (L) | $4.9_{\pm 0.7}$ | $2.41_{\pm 0.0}$ |

3D Teapot

Table 3: Model performance on Reacher (left) and 3D Teapot (right) environments. For the 3D Teapot models, None is the non-equivariant model, Matmul is the matrix multiplication model and Equiv is the version using Tensor Field Networks. (S) denotes the small action space with 6 discrete rotations of $\frac{2\pi}{30}$ in $SO(3)$ and (L) denotes continuous rotations.

### 5.4 Model performance comparison

We report results on all environments for our model and compare against baselines: the non-equivariant model (None) and the fully-equivariant (Full). In the 3D Blocks environment, the ranking metrics for 10 steps demonstrate that all models can create good latent representations without resorting to reconstruction losses. In particular, the fully equivariant model is still able to discover a good representation even when the group action on the input space $\rho_{\mathcal{S}}$ is not given/apparent. We see a similar pattern on the Reacher domain, where all models achieve $100\%$ H@10 for 1 step predictions. In Rush Hour, all models perform similarly and have low EE and DIE due to the environment simplicity and flat input space. Comparing the equivariance metrics, we can see that our model can outperform both the non-equivariant and fully-equivariant model for the DIE(model) metric for 3D Blocks, while the other metrics are similar across all models. In 3D Teapot, we observe contrasting training results for small (S, $2\pi/30$ rotations) and large (L, any rotation) action spaces. With small actions, Tensor Field Networks (TFN) and Matrix Multiplication (MatMul) perform similarly at 30 - 40% Hard Hits@1, whereas in large actions, MatMul finds the optimal solution and TFN fails. We hypothesize that the difference is caused by the contrastive loss function; small actions can lead to local minima where all states get mapped to close-by latent states.

**Visualization of latent embeddings** We visualize the latent embedding $z$ for our model to qualitatively analyze what kind of representations are learned. All figures are provided in the Appendix for space reasons. Figure 4 plots all the learned embeddings for Reacher for all observations in the evaluation set and shows a sample transition in both pixel and latent space. The encoded current state $z$ is highlighted in red and the encoded next state is highlighted $z'$ is highlighted in orange. which we factor into irreducible representations (irreps) before visualizing (see Hall (2003)). Some irreducible representations are 1-dimensional and are plotted as a line. The 2-dimensional irreps show a clear circular pattern, match the joint rotations of the environment.

Figure 5 shows the traversal of rotations in pixel and latent space for 3D Teapot. The latent space can choose its own base coordinate frame and thus is oriented downwards. We can clearly see that the effective rotations relative to the objects orientation perfectly align, demonstrating that the learned embeddings correctly encode 3D poses and rotations.

### 5.5 Generalization from limited actions

We now train on a limited subset of actions and evaluate on datasets generated with the full action space. These experiments aim to verify that our model, even where all components are not designed to be equivariant, can learn a good equivariant representation which can generalize to unseen actions.

| | H@10 (1 step, %) | MRR (1 step, %) | EE($S$) | DIE($S$) ($\times 10^4$) | DIE (model) (1 step) |
|---|---|---|---|---|---|
| CNN | $2.8_{\pm 0.6}$ | $5.3_{\pm 0.4}$ | $0_{\pm 0}$ | $0_{\pm 0}$ | $0.19_{\pm 0.05}$ |
| Ours/Full | $100_{\pm 0.0}$ | $99.9_{\pm 0.0}$ | $0_{\pm 0}$ | $0_{\pm 0}$ | $0_{\pm 0}$ |

Table 4: 2D Shapes, trained using only up action. Due to the simplicity of the environment, a simple CNN turns out to be equivariant, so both the baseline CNN and Ours/Full have equivariant $S$.

| | H@1 (10 step, %) | MRR (10 step, %) | EE($S$) | DIE($S$) ($\times 10^{-2}$) | DIE (model) (10 step, $\times 10^{-3}$) |
|---|---|---|---|---|---|
| None | $52.3_{\pm 14}$ | $61.8_{\pm 13}$ | $0.98_{\pm 0.2}$ | $3.64_{\pm 1.5}$ | $181_{\pm 79}$ |
| Full | $83.7_{\pm 36}$ | $86.0_{\pm 31}$ | $0.81_{\pm 0.5}$ | $3.32_{\pm 2.1}$ | $14.8_{\pm 9.1}$ |
| Ours | $99.9_{\pm 0.0}$ | $100_{\pm 0.0}$ | $0.96_{\pm 0.3}$ | $3.65_{\pm 1.6}$ | $5.0_{\pm 4.7}$ |

Table 5: 3D Blocks, trained using only up,right,down actions.

| | H@10 (1 step, %) | MRR (1 step, %) | EE($S$) | DIE($S$) ($\times 10^{-2}$) | DIE (model) (1 step, $\times 10^{-2}$) |
|---|---|---|---|---|---|
| None | $86.5_{\pm 3.0}$ | $50.6_{\pm 3.1}$ | $1.22_{\pm 0.1}$ | $6.54_{\pm 1.9}$ | $6.95_{\pm 1.5}$ |
| Full | $89.4_{\pm 11}$ | $61.8_{\pm 13}$ | $1.18_{\pm 0.1}$ | $3.62_{\pm 0.8}$ | $4.87_{\pm 1.4}$ |
| Ours | $90.8_{\pm 4.5}$ | $59.4_{\pm 4.6}$ | $1.28_{\pm 0.1}$ | $5.45_{\pm 0.9}$ | $4.95_{\pm 0.6}$ |

Table 6: Reacher with limited actions. The models were trained on data where the second joint is constrained to be positive and evaluated on unconstrained data.

We perform experiments on the 2D Shapes, 3D Blocks, and Reacher domains. For 2D Shapes, the training data only contains 'up' actions and we set the number of episodes to 100,000 with length 1 to avoid any distribution shifts in the data (e.g. performing up continuously will produce many transitions where all blocks are blocked by the boundaries). For 3D Blocks, we omit the left action in training and use similar modifications to the episode lengths. For Reacher, the action space represents joint actuation forces $\in [-1, 1]$ for each of the two joints. We restrict the range of the force for the second joint to be positive, meaning that the second arm rotates in only one direction.

Tables 4,5,6 show quantitative results for 2D Shapes, 3D Blocks, and Reacher respectively. We see that the baseline performs poorly, whereas our method is successfully able to learn equivariant representations. Ours either matches or slightly outperforms the performance of the fully equivariant model. The fully equivariant networks have a mismatched inductive bias in so far as they assume the correct transformation $\rho_{\mathcal{S}}$ is a simple transformation of the pixels. Nonetheless, they performed surprisingly well in our tests, suggesting that incorrect inductive biases can still lead to performance improvements in certain situations, a finding which may deserve further study in future work.

Figure 6 shows embeddings for all states in the evaluation dataset for our model and the non-equivariant model trained on only the up action. Our model shows a clear $5 \times 5$ grid, while the non-equivariant model learns a degenerate solution (possibly encoding only the row index $x$).

## 6 Conclusion and Future Work

We demonstrate a flexible method which can be used to extend equivariant neural networks to domains with known symmetry types, but transformation properties which cannot be easily explicitly described. We apply our method across a variety of domains and equivariant neural network architectures. Our methods confer some of the advantages of equivariant neural networks in situations where they did not previously apply, such as generalization to data outside the training distribution. Future work will include applying out method to tasks besides world models and using our method to develop disentangled and more interpretable features in domains with known but difficult to isolate symmetry.

## ETHICS STATEMENT

Our paper does directly address domains with privacy or safety concerns. However, our method can be used in robotics applications to train robots using fewer data samples. To the extent that robotics technology can be used for benefit or harm, our method enables both options.

## REPRODUCIBILITY STATEMENT

We will open-source our code, including all models and data generation scripts, thus allowing all experiments to be fully reproduced.

## REFERENCES

Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Erkao Bao and Linqi Song. Equivariant neural networks and equivarification. *arXiv preprint arXiv:1906.07172*, 2019.

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pp. 992–1002. PMLR, 2020.

Carvana. Carvana Image Masking Challenge, 2017. URL `https://kaggle.com/c/carvana-image-masking-challenge`.

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

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

Taco S. Cohen, Mario Geiger, Jonas K ohler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 1321–1330, 2019.

Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. *arXiv preprint arXiv:2109.07103*, 2021.

Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*, 2020.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

Luca Falorsi, Pim de Haan, Tim R Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Josh Rackers, and Michael Bailey. Euclidean neural networks: e3nn, 2020. URL `https://doi.org/10.5281/zenodo.5292912`.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Brian C Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 10. Springer, 2003.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pp. 2747–2755, 2018.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pp. 6734–6744. PMLR, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

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

Elise van der Pol, Daniel E Worrall, Herke van Hoof, Frans A Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *arXiv preprint arXiv:2006.16908*, 2020.

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

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf`.

Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 14334–14345, 2019.

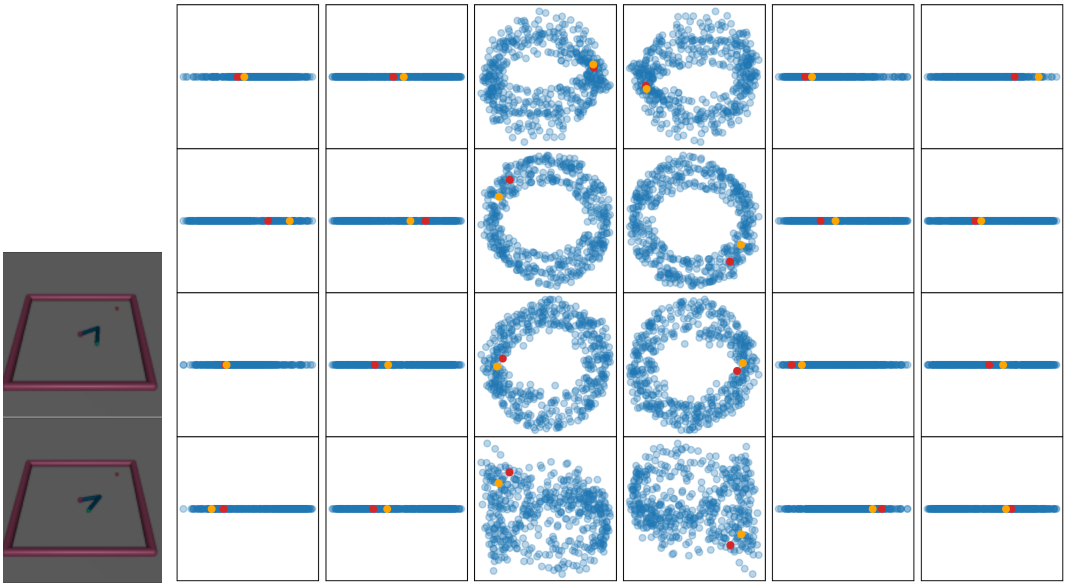
Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

# A LEARNED LATENT REPRESENTATIONS

The learned latent embedding $z$ for all states in the evaluation set for Reacher is shown in Figure 4 and the embeddings are factored into irreducible representations. The 2-dimensional representations show a circular pattern, mimicking the rotation of joints. Figure 5 shows a sample observation in pixel space (ground truth) in the top row and its encoded latent embedding in the bottom row. The latent space can have a different orientation than the ground truth. Applying rotations to the low-dimensional embedding yields a latent space traversal with smooth rotations, showing that the learned representations correctly encode the correct symmetries. Figure 6 show learned embeddings for all states in the evaluation dataset for our model (left) and the non-equivariant model (right) when trained on only the up action.



(a) $s, s'$ (b) $z \in \mathcal{Z}$.

Figure 4: Learned symmetric embeddings for Reacher: pixel observation $s$ (left top) and next observation $s'$ (left bottom), latent representation $z$ of the evaluation set (right). The representation type of $z$ is $\rho_{D_4,\text{reg}}$ which we factor into irreducible representations before visualizing (see Hall (2003)). All encoded samples in the evaluation set are shown and the encoded current observation is colored red and the encoded next observation is colored orange. There is a clear circular pattern that match joint rotations.

# B TRAINING DETAILS

## B.1 ENVIRONMENTS

**2D Shapes & 3D Blocks** We use the 2D Shapes and 3D Blocks environment in Kipf et al. (2019). There are five objects are arranged in a $5 \times 5$ grid and each object can occupy a single cell. Actions are the 4 cardinal directions for each object and an action moves one object at a time, unless it is blocked by the boundaries or by another object. Observations are $50 \times 50 \times 3$ RGB images for both 2D Shapes and 3D Blocks, with pixel values normalized to $[0, 1]$. The observations in 2D shapes are top down views of the grid and each object has a different color-shape combination. For 3D Blocks, the observations are rendered isometrically with a skewed perspective and each block has a $z$-height, introducing partial occlusions to the image.

**Rush Hour** We create a variant of 2D Shapes called Rush Hour. Each object has an orientation and the action is relative to the object's orientation: $\{$forward, backward, left, right$\}$. This increases the importance of rotational orientation in the environment increasing the significance of equivariance.

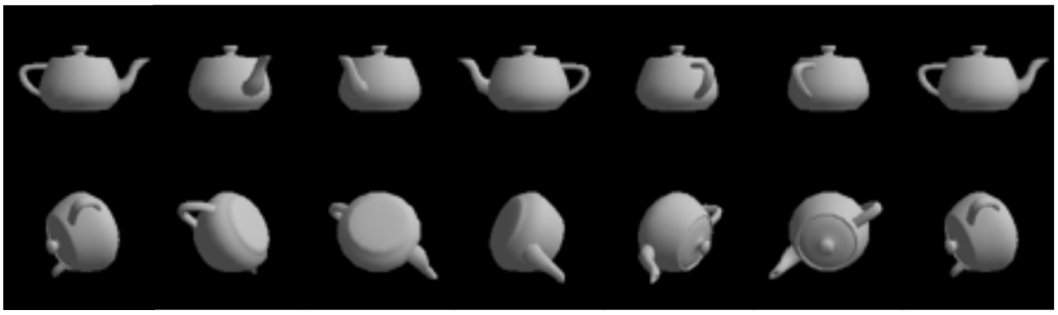
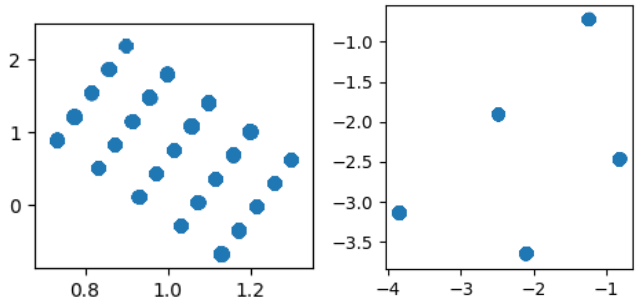



Figure 5: Latent space traversal for matrix multiplication (Matmul) transition model in Teapot. Top row: Ground truth rotation of a teapot. Bottom row: Projection in the latent space of the Matmul model, which is regularized to consist only of valid rotation matrices. The model, by construction, does not have a fixed reference frame; hence, the two sequences are offset by the learned latent reference frame.




(a) (b)

Figure 6: 2D Shapes: learned embeddings for all states in the evaluation set when trained on only the up action. Our model (left) is able to generalize well and learns the correct underlying $5 \times 5$ grid. The non-equivariant model (right) learns a degenerate solution

**Reacher** This environment makes a small modification to the original MuJoCo environment `Reacher-v2`. As we do not consider rewards, we fix the goal position to the position $[0.2, 0.2]$ so that features related to the goal are ignored. Instead of using the 11-dimensional state, we use pixel observations as images and preprocess them by cropping slightly and downsampling the original $500 \times 500$ RGB image to $128 \times 128$. The default camera position gives a slightly skewed perspective, see Table 1.

**3D Teapot** The 3D teapot enviroment contains images of the Utah teapot model rendered into the 64x64 greyscale images. The teapot varies in pose which can be described by a coordinate frame $z \in \mathrm{SO}(3)$. We consider both a small (S) and large (L) action space for this environment. In the small action space, 6 actions may be taken corresponding to multiplication of the pose $z \mapsto az$ by $a \in \mathrm{SO}(3)$ where $a$ is a rotation of $\pi 2\pi/30$ around the $x-$,$y-$,or $z$-axis. In the large action space, actions may be any element $a \in \mathrm{SO}(3)$.

### B.2 Model architectures

**Symmetric embedding network** $S$ For all models and environments except for 3D Teapot, we use CNNs with BatchNorm Ioffe & Szegedy (2015) and ReLU activations between each convolutional layer. For 3D Teapot, the symmetric embedding network maps directly to the latent $z$ space so we use 4 convolutional layers followed by 3 fully connected layers. The output is a $3 \times 3$ rotation matrix. The number of layers for each environment is given in Table 1. For the non-equivariant symmetric

embedding networks, we use 32 convolutional filters for every layer and use 8 filters for Reacher and 16 filters for all other environments.

**Encoder** $E$ The object-oriented environments use 3-layer MLPs with 512 hidden units for the non-equivariant networks and 256 for the equivariant counterparts. There is a ReLU activation after the first and second layers and a LayerNorm (Ba et al., 2016) after the second layer. For Reacher, we use 3 convolutional layers followed by 3 fully connected layers. The 3D Teapot does not have an explicit encoder, i.e. it is the identity function. The output of the non-equivariant encoder is a 2-dimensional vector for 2D Shapes, 3D Blocks, and Rush Hour and a 4-dimensional vector for Reacher. The output of the equivariant encoders for each environment is listed in Table 1.

**Transition** $T$ The object-oriented environments use GNN transition models where the edge and node networks have the same structure as the encoder (3-layer MLPs). For Reacher and 3D Teapot, the transition model $T$ is a MLP with 512 hidden units for the non-equivariant version and 256 for the equivariant version. Actions are concatenated to the latent $z$ and are input into the transition models which then outputs a $z'$ of the same dimension as the input $z$. We use one-hot encoding for discrete actions.

### B.3 Hyperparameters

For the object-oriented environments, we follow the hyperparameters used in (Kipf et al., 2019): a learning rate of $5 \times 10^{-4}$, batch size of 1024, 100 epochs, and the hinge margin $\gamma = 1$. We find that these hyperparameters work well for all other environments, except that Reacher uses a batch size of 256 and mixed precision training was used for both non-equivariant, fully-equivariant, and our method, in order to keep the batch size relatively high for stable contrastive learning. Most experiments were run on a single Nvidia RTX 2080Ti except for 3D Cubes which used a single Nvidia P100 12GB.

## C Group Representations

We explain the notation and definitions of the different representations of the groups considered in the paper and displayed in Table 1.

The $\rho_{\text{std}}$ representation of $C_4$ or $D_4$ on $\mathbb{R}^2$ is by 2-by-2 rotation and reflection matrices. The $\rho_{\text{std}}$ representation of $S_5$ permutes the standard basis of $\mathbb{R}^5$. The regular representation $\rho_{\text{reg}}$ of $G$ permutes the basis element of $\mathbb{R}^{|G|}$ according to the multiplication table of $G$. The trivial representation of $G$ fixes $\mathbb{R}$ as $\rho_{\text{triv}}(g) \cdot x = x$. For $D_4$, $\rho_{\text{flip}}(g) = \pm 1$ is a representation on $\mathbb{R}$ depending only on if $g$ contains a reflection. Given representations $(\rho_1, \mathbb{R}^{n_1})$ and $(\rho_2, \mathbb{R}^{n_2})$ of $G_1$ and $G_2$, $(\rho_1 \boxtimes \rho_2)(g_1, g_2)(v \otimes w) = g_1 v \otimes g_2 w$ gives a representation on $G_1 \times G_2$ on $\mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2}$.

(a) 3D Blocks (b) Reacher

Figure 7: Original observations and their $G$-transformed versions