Structuring Representations using Geometric Invariants

Anonymous Author(s)
Affiliation
Address
email

Abstract

Different geometries can be defined by their invariants. For example, distances and angles are preserved in Euclidean and Conformal geometries, respectively. We propose to learn equivariant representations using loss functions that enforce the corresponding geometric invariants. This approach requires no prior knowledge about the correspondence of potentially complex transformations in the input space and the symmetry group. We consider geometric invariants for different Lie groups such as the Euclidean, Orthogonal and Unitary groups and additionally show the feasibility of learning disentangled and decomposed representations. We provide favorable qualitative and quantitative results on downstream tasks, including world modeling and reinforcement learning.

1 Introduction

Sample efficient representation learning is a critical open challenge in deep learning for AI. When we have prior information about transformations that are relevant to a particular domain, building representations that are aware of these transformations can lead to better sample efficiency and generalization. One way to use such symmetry priors is to make the network invariant to the given transformations. A generalization of this idea is called equivariance, where transforming the input transforms the output in a specific way. An equivariant network that makes good predictions for a particular input also generalizes to all input transformations, making symmetry a useful prior.

While recent years have witnessed a range of exciting equivariant deep models, there are several limitations. First, most equivariant networks constrain the network architecture, which sometimes requires specialized implementations. Moreover, transformations considered by the existing methods are often assumed to be linear in both input and representation space. This is the case for architectures designed for finite permutation groups and continuous Lie groups. Approaches that go beyond linear transformations in the input space often assume access to group information – *i.e.*, the group member that transforms one input to another is known. This paper introduces a simple approach that addresses all of these limitations.

We are inspired by the connection between various geometries and group theory as characterized by Klein's Erlangen program [32]. According to this view, geometries are concerned with invariant quantities under certain transformations. For example, Euclidean geometry is concerned with the length, angle, and parallelism of lines, among others, because Euclidean transformations preserve these. However, moving to the more general and less structured Affine geometry, notions of distance and angle are no longer relevant, while parallelism remains an *invariant* of the geometry. The corresponding symmetry groups are examples of Lie groups that have a subgroup relation, $E(n) < Aff(n)$, thereby enabling the groups to characterize a hierarchy (or lattice) of different geometries. *We propose to induce a geometry on the embedding and make it equivariant to a given group by enforcing its defining invariant.* For example, distance is the defining invariant for Euclidean

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

geometry, which means all distance preserving transformations are Euclidean. Therefore, to enforce equivariance to the Euclidean group, it is sufficient to ensure that the embedding of any two data points has the same distance before and after the same transformation of the inputs; see Figure 1. While this approach uses the *defining action* of different Lie groups in the embedding space, the same group can have a non-linear and unknown action on the input space. In the pendulum example of Figure 1, the group $E(3)$ acts on the value of each pixel of the input image using an unknown and non-linear action. Moreover, this approach does not require the pairing of group members with transformations, a piece of information that is often unavailable.

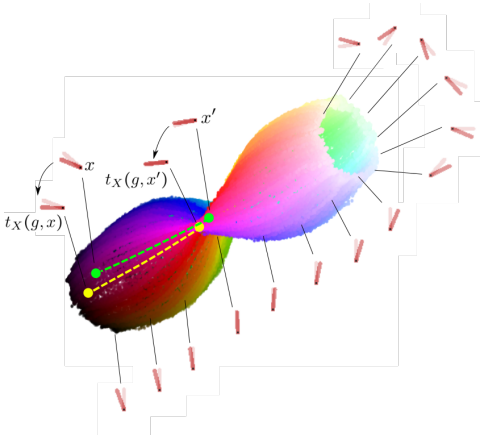

Figure 1: $E(3)$ **embedding for the pendulum.** The input x consists of a pair of images that identify both the angle and the angular velocity of a pendulum. The equivariant embedding learns to encode both: the true angle is shown by a change of color and angular velocity using a change of brightness. The two circular ends (black and white) correspond to states of maximum angular velocity in opposite directions. The SymReg objective for the *Euclidean group* learns this embedding by preserving the pairwise distance between the codes before $(f(x), f(x'))$ and after $(f(t_X(g, x)), f(t_X(g, x)))$ transformations of the input by t_X. Therefore dashed lines have equal lengths. For the pendulum, the transformations are in the form of applying positive or negative torque in some range.

In the rest of the paper, we arrive at the idea above from a different path: after reviewing related works in Section 2 and providing a background in Section 3, Section 4 observes that equivariance, in its general form, can be a weak inductive bias. This is because having an injective code is sufficient for equivariance to "any" transformation group. However, in this manifestation of equivariance, the group action on the embedding can be highly non-linear. Since the simplicity of the action on the embedding seems essential for equivariance to become a useful learning bias, Section 5 proposes to *regularize* the group action on the code to make it "simple". This *symmetry regularization* (SymReg) objective is group-dependent and the essence of our approach. Enforcing geometric invariants in the latent space is proposed as a symmetry regularization. While we focus on equivariant representation learning through self-supervision, in principle, supervised tasks can also benefit from the proposed SymReg. An important benefit of a symmetry-based representation is its ability to produce disentangled representations through group decomposition [28]. Section 6 studies disentanglement using SymReg. Section 7 presents a range of experiments to understand its behavior and puts it in the context of comparable baselines.

2 Related Works

Finding effective priors and objectives for deep representation learning is an integral part of the quest for AI [3]. Among these priors, learning equivariant deep representations has been the subject of many works over the past decade. Many recent efforts in this direction have focused on the design of equivariant maps [53, 14, 45, 34, 16, 24, 52, 20, 6] where the "linear" action of the group on the data is known. Due to this constraint, the application of these models has been focused on fixed geometric data such as images [36], sets [56, 43], graphs [39, 33], spherical data and the (special) orthogonal group [15, 1, 48, 23], the Euclidean group [50, 9, 25] or other physically motivated groups such as the Lorentz [4] or Poincare group [52], among others.

In the present work, the group action is unknown and possibly non-linear. Therefore, in contrast to many prior works, linear representation theory plays no significant role. Our setup is closer to the body of work on generative representation learning [7, 12, 40], in which the (linear) transformation is applied to the latent space [44, 54, 35, 37, 17, 22]. Among these generative coding methods, transforming autoencoder [29] is a closely related early work which, in addition to equivariance, seeks to represent the part-whole hierarchy in the data. What additionally contrasts our work with the follow-up works on capsule networks [46, 38] is that SymReg is agnostic to the choice of architecture and training. We only rely on our objective function to enforce equivariance.

Since we consider learning equivariant representations through self-supervision, exciting recent progress in this area is also quite relevant [26, 41, 10, 51, 27, 57, 21]. While the use of transformations is prominent in these works, in many settings, the objective encourages *invariance* to certain transformations, making such models useful for invariant downstream tasks such as classification. Similar to many of these methods, we also use transformed pairs to learn a representation, with the distinction of learning an *equivariant* representation. An exception is the recent work of Dangovski et al. [18], which learns an equivariant representation by separating the invariant embedding from the pose, where the relative pose is learned through supervision. Therefore, in that work, in contrast to ours, one needs to know the transformation that maps one input to another. When considering the Euclidean group, SymReg preserves distances in the embedding space under non-linear transformations of the input. This embedding should not be confused with isometric embedding [49], where the objective is to maintain the pairwise distances between points in the input and the embedding space.

3 Background on Symmetry Transformations

We can think of transformations as a set of bijective maps on a domain X. Since these maps are composable, we can identify their compositional structure using an abstract group G. For this reason, such transformations are called group actions. To formally define transformation groups, we first define an abstract group. A *group* G is a set equipped with a binary operation, such that the set is closed under the operation $gg' \in G \ \forall g, g' \in G$, every $g \in G$ has a unique inverse such that $gg^{-1} = e$, where e is the identity element of the group, and the group operations are associative $(gg')g'' = g(g'g'')$.

A G-*action* on a set X is defined by a function $t : G \times X \to X$, which can be thought of as a bijective transformation parameterized by $g \in G$. In order to maintain the group structure, the action should satisfy the following two properties: (1) action of the identity is the identity transformation $t(e, x) = x$; (2) composition of two actions is equal to the action of the composition of group elements $t(g, t(g'x)) = t(gg', x)$. The action t is *faithful* to G if transformations of X using each $g \in G$ are unique – *i.e.*, $\forall g, g' \ \exists x \in X$ s.t. $t(g, x) \neq t(g', x)$. If a G-action is defined on a set X, we call X a G-set. Many groups are defined using their defining action; for example, the $SO(3)$ is the group of rotations in 3D space. While this defining action is a linear transformation, the same group can act non-linearly on $\mathbb{R}^n$ using the action $t : SO(3) \times \mathbb{R}^n \to \mathbb{R}^n$.

4 Equivariance is Cheap, Actions Matter

A symmetry-based representation or embedding is a function $f : X \to Z$ such that both X and Z are G-sets, and furthermore f "knows about" G-actions, in the sense that transformations of the input using t_X have the same effect as transformations of the output using some action t_Z:

$$f(t_X(g, x)) = t_Z(g, f(x)) \quad \forall g, x \in G \times X \tag{1}$$

The following claim shows that despite many efforts in designing equivariant networks, simply asking for the representation to be equivariant is not a strong inductive bias, and we argue that the action matters. Put another way, the strong performance of existing equivariant networks should be attributed to the fact that the group action on the embedding space is simple (linear).

Proposition 4.1. *Given a transformation group $t_X : G \times X \to X$, the function $f : X \to Z$ is an equivariant representation if $\forall g \in G, x, x' \in X$*

$$f(x) = f(x') \Leftrightarrow f(t_X(g, x)) = f(t_X(g, x')). \tag{2}$$

That is, two embeddings are identical iff they are identical for all transformations.

The proof is in the appendix. The condition above is satisfied by all injective functions, indicating that many functions are equivariant to any group.

Corollary 4.2. *Any* injective *function $f : X \to Z$ is equivariant to any transformation group $t_X : G \times X \to X$, if we define G action on the embedding space as*

$$t_Z(g, z) \doteq f(t_X(g, f^{-1}(z))) \quad \forall g, z \in G \times Z \tag{3}$$

The ramification of the results above in what follows is two-fold:

1. While injectivity ensures equivariance, the group action on the embedding as shown in Equation (3) can become highly non-linear. Intuitively, this action recovers $x = f^{-1}(z)$, applies the group action $x' = t_X(x)$ in the input domains and maps back to the embedding space $f(x')$ to ensure equivariance. In the following, we push t_Z towards a simple linear G-action through optimization of f. This objective can be interpreted as a *symmetry regularization or a symmetry prior* (SymReg).

2. Although Corollary 4.2 uses injectivity of f for the entire X, we only need this for the data manifold. In practice, one could enforce injectivity on the training dataset D using a decoder, architectural choices such as momentum encoder [27], or loss functions defined on the training data, such as a hinge loss [26] $L_{\text{hinge}}(f, D) = \sum_{x,x' \neq x \in D} \max\left(\epsilon - \|f(x) - f(x')\|, 0\right)$ or other losses that monotonically decrease with distance, such as $\frac{1}{\|f(x)-f(x')\|}$, or its logarithm $-\log(\|f(x) - f(x')\|)$. In experiments we use the logarithmic barrier function.

5 Symmetry Regularization Objectives

In learning equivariant representations, we often do not know the abstract group G and how it transforms our data, t_X. We assume that one can pick a reasonable abstract group G that "contains" the ground truth abstract group acting on the data – *i.e.*, G action on the input may not be faithful. Our goal is to learn an $f : X \to Z$ that is equivariant w.r.t. the actions t_X, t_Z, where $t_X : X \times G \to X$ is unknown and t_Z is some (simple) G-action on Z of our choosing.

More Informed but Less Practical Setting. In the most informed case, the dataset also contains information about which group member $g \in G$ can be used to transform x to x' – that is the dataset consists of triples $(x, g, x_t = t_X(g, x))$. By having access to this information we can regularize the embedding using the following loss function: $L_G^{\text{informed}}(f, D) = \sum_{(x,g,x_t)\in D} \ell\big(f(x_t) - t_Z(g, f(x))\big)$, where ℓ is an appropriate loss function such as the square loss. At its minimum, we have $f(x_t) = t_Z(g, f(x))$ or $f(t_X(g, x)) = t_Z(g, f(x))$, enforcing equivariance condition of Equation (1). However, even if the optimal value is not reach, due to its injectivity, f is still G-equivariant, and the the objective above is regularizing the G action on the code. This informed setup is used in equivariant contrastive learning of [18]. The assumption of having access to g is realistic when we know the action t_X, so that we can generate (x, g, x_t) triplets. Fortunately, using group invariants, we may still learn an equivariant embedding, even if we do not have the group information tied to the dataset.

Euclidean Group. The defining action of the Euclidean group $E(n)$ is the set of transformations that preserve the Euclidean distance between any two points in $\mathbb{R}^n$, a.k.a. isometries. These transformations are compositions of translations, rotations, and reflections. Since for the real domain, all Euclidean isometries are linear and belong to $E(n)$, we can enforce the group structure on the embedding by ensuring that distances between the embeddings before and after any transformation match. For this, we need the dataset D to be a set of pairs of pairs $\big((x, x_t = t_X(g, x)), (x', x'_t = t_X(g, x'))\big)$, where x, x' are transformed using the same *unknown* group member g. Distance-preservation loss below combined with injection loss are sufficient to produce an $E(n)$-regularized embedding:

$$L_{E(n)}(f, D) = \sum_{\big((x,x_t),(x',x'_t)\big)\in D} \ell\Big(\overbrace{\|f(x) - f(x')\|}^{\substack{\text{distance before the}\\ \text{transformation}}} - \overbrace{\|f(x_t) - f(x'_t)\|}^{\substack{\text{distance after the}\\ \text{transformation}}} \Big) \tag{4}$$

For example, in the standard RL setup, where we have access to triplets (s, a, s'), we can easily form D by unrolling an episode and collecting two different state transitions corresponding to a particular action. In practice, with a finite actions, we can efficiently generate this dataset by keeping separate buffer for each action where we store state transitions for that action and sample from that buffer to train the embedding function f.

Orthogonal and Unitary Groups. The defining action of the orthogonal group $O(n)$ preserves the inner product between two vectors. The analogous group in the complex domain is the unitary group, which preserves the complex inner product. Our symmetry-regularization objective enforces this invariant: $L_{O(n)}(f, D) = \sum_{\big((x,x_t),(x',x'_t)\big)\in D} \ell\big(f(x)^\top f(x') - f(x_t)^\top f(x'_t) \big)$.

For the unitary group one additionally needs to embed to complex domain $Z = \mathbb{C}^n$, where the only difference is in the definition of the inner product.

Conformal Group. The invariant of conformal geometry is the angle. In a Euclidean embedding, conformal transformations include a combination of translation, rotation, dilation, and inversion with respect to an $n-1$-sphere. To enforce this group structure we need a triplets of inputs, before and after a transformation $\big((x, x_t), (x', x'_t), (x'', x''_t)\big)$, so that we can calculate the angle in the embedding. Conformal SymReg objective, which preserves angles, imposes a weaker constraint on the embedding than the distance preservation of the Euclidean group – since the latter implies the former. Moreover, it has an additional benefit that, compared to $L_{E(n)}$, the loss cannot be minimized by simply shrinking the embedding. Therefore in practice, the injection enforcing losses of Section 4 is not as crucial when using conformal symmetry regularization.

Other Groups and Geometries. Many other classical lie groups are equipped with defining invariants that are sometimes more expensive to compute. Special Linear group preserves volume and orientation; Lorentz and Poincare group are the analogs of the orthogonal and Euclidean group in the Minkowski space, therefore equipped with similar invariants; the Symplectic group preserves a bilinear form. Moreover, we can also use SymReg for finite groups; see Appendix C.

6 Decomposing the Representation

Higgins et al. [28] suggested a notion of disentangled representation based on decomposition of the abstract group into a direct product form $G = G_1 \times \ldots \times G_k$. There are two approaches to learning such decomposed representation using SymReg, depending on whether or not we can perform certain types of transformations in isolation. For example, an RL agent may transform its environment through actions like moving a single limb that can be performed in isolation. In this case, we call the decomposition *active* to contrast it with the *passive* case where the action of different subgroups is always mixed in our dataset.

Active Decomposition. Let $G = \{(g_1, \ldots, g_k) \in G_1 \times \ldots \times G_k\}$ be a product group, where $G_i \cong \{(e, \ldots, e, g_i, e, \ldots, e) \in G\}$ can be identified with a normal subgroup of G. In active decomposition, sub-groups can act in isolation and therefore we have k types of tuples in our dataset $D_1, \ldots, D_k \subset D$. Each subset D_i is associated with actions of a subgroup G_i using $t_X((e, \ldots, e, g_i, e \ldots, e), \cdot))$, $g_i \in G_i$. In this setting, the representation $f : X \to Z = Z_1 \times \ldots \times Z_k$ can be thought of as k separate functions where $f_i : X \to Z_i$ is equivariant to G_i-action and invariant to all $G_j, j \neq i$ actions. This gives the following objective

$$L_G^{\text{active}}(f, D) = \sum_{i=1}^{k} \underbrace{L_{G_i}(f_i, D_i)}_{\text{equivariance to } G_i} + \underbrace{L^{inv.}_{G/G_i}(f_i, D \backslash D_i)}_{\text{invariance to } G_j \text{ for } j \neq i}, \tag{5}$$

where $L^{inv.}_G(f, D)$ enforces invariance of f to G-transformations in D – e.g., by penalizing $\|f(x) - f(t_X(g, x))\|$.

Passive Decomposition. When we have no control over transformations, and we are simply given the data, it is still possible to use an abstract group that has a product form. Here again, $f : X \to Z = Z_1 \times \ldots \times Z_k$, but the loss function is simply enforced on each block separately – *i.e.*, $L_G^{\text{passive}}(f, D) = \sum_{i=1}^{k} L_{G_i}(f_i, D)$, where $L_{G_i}(f_i, D)$ is a SymReg objective from Section 5.

7 Experiments

We conducted many experiments to qualitatively study the representation learned by SymReg and its ability to produce a disentangled representation, and quantitatively compare it against simple baselines in representation learning and downstream RL tasks. For details of architecture and training, see Appendix G.

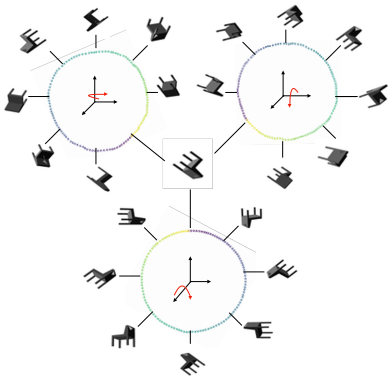

Figure 2: **Visualization of SymReg's latent projection for the rotating Chair dataset.** The chair is rotated in three orthogonal axes from 0 to 2π. The latent embedding for each chair pose is projected from 16D embedding space to a 2D space for visualization. The colors of the representations are mapped to the chair's angle of rotation. We notice that the mapping function f learned is continuous with respect to the transformations of the object, and it maps the rotations along an axis to a circular manifold. This is true for each orthogonal axis of rotation. We observe a similar result for any other initial pose for the chair.

7.1 Qualitative Analysis

In this section, we visualize the representation learned for two examples from the Gym environment [5], including the pendulum and the mountain car (see Appendix D), followed by an experiment involving a rotating object where we know the ideal embedding is the $SO(3)$ manifold. Finally, Figure 4 visualizes a conformal embedding for double-bump world. In most cases we also visualize a Variational AutoEncoder (VAE) [30] embedding for comparison. Our objective here is to visually demonstrate the behavior of SymReg and its remarkable ability to learn an embedding informed by the non-linear transformation of the input.

7.1.1 The Pendulum

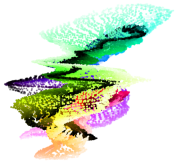

Figure 3: VAE embedding for the pendulum example; compare againts Figure 1

For this experiment, the input x is two consecutive frames of the pendulum that have been grayscaled and downsampled to 32×32 pixels. The action space is a range of torques that can be applied to the base of the pendulum. We use the action to transform the data. We use the objective of Equation (4) to learn an $E(3)$-equivariant representation. To efficiently estimate $L_{E(n)}$, we use a mini-batch that consists of 64 randomly sampled observations from the environment and their transformations via three randomly sampled actions (4×64 samples in total). Once the embedding $f(x)$ is produced, the pairwise distance between all 64^2 pairs is calculated, and the mini-batch loss penalizes the change in these distances between any pair of transformations in the mini-batch. Therefore, using this mini-batching procedure, the complexity of calculating the loss grows quadratically with both the number of independent samples and transformations.

The model learns to parameterize the embedding using the angle and the angular momentum of the pendulum from the input data; see Figure 1.For comparison, Figure 3 shows the embedding learned using a VAE from the same dataset. We perform a similar experiment in the Mountain Car Environment and report the results in Appendix D.

7.1.2 Rotating Chair

We consider a 3D chair from ModelNet40 [55] and transform it through the action of the group $SO(3)$. The group action on the input is the 2D projection into a 48×48 image after the 3D rotation of the chair. While the group of interest is $SO(3)$, we use SymReg loss of Equation (4). In choosing the abstract group, we often only need to ensure that the group is large enough to contain the ground truth as a subgroup. Although this results in a stronger symmetry regularization on the embedding, a G-equivariant embedding is equivariant to any $H \leq G$. This means that, for example, an $E(2)$ equivariant embedding can be useful for its finite subgroups such as dihedral or cyclic groups. We embed the chair in $\mathbb{R}^{16}$ using SymReg[1] and visualize the latent by rotating the chair along three orthogonal axes and projecting the latent codes into a 2D space. Figure 2 shows three circular latent traversals of SymReg embedding corresponding to rotation around each axis, which is consistent

[1]Note that while $SO(3)$ manifold is 3-dimensional, its isometric embedding requires a higher number of dimensions. Using a larger embedding dimension also often helps with the optimization.

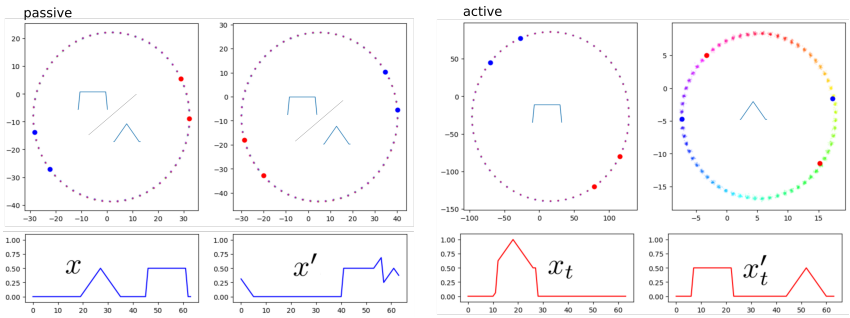

Figure 5: **Active versus passive decomposition for the double-bump world.** Four images at the bottom show a pair of inputs (x, x') and their transformations (x_t, x'_t). The figure shows the embedding for two inputs before, (x, x') in blue, and after, (x_t, x'_t) in red, the *same* transformation. This transformation cyclically shifts both the triangle and the square to the left, but the amount of translation is larger for the square. In both passive and active decomposition, the Euclidean distance is preserved by the transformation – the red points have the same distance with each other as the blue points on every manifold. In the active decomposition (right), one of the manifolds encodes the circular translation of the triangle bump, while the second one represents the location of the square bump. Various colors indicate the location of the triangle. In the case of passive decomposition (left), since the transformation of individual shapes does not guide the decomposition, the manifolds jointly encode the location of each bump type.

with the structure of $SO(3)$ manifold. The process of learning $SO(3)$ manifold is a challenging task, and previous works assumed that the group member corresponding to each transformation is given [44, 2]. In contrast, we only use the observations corresponding to similar actions during training and not the group members themselves. As we see later, this is critical in settings such as RL, where group information is unavailable. We could not produce a similar latent traversal for VAE due to collapse when rotating around some axes.

7.1.3 Conformal Embedding for Double-Bump World

The double-bump world consists of a rectangular bump signal and a triangular bump signal, both cyclically shifted and superimposed. These transformation are given by a pair (Δ_1, Δ_2) which cyclically shifts the rectangular bump by Δ_1 and the triangular bump by Δ_2. In our experiments, the signal length is 64, and the length of the bump is 16. SymReg embeds to a 4-dimensional conformal space in this example. Figure 4(right) shows the random projection of the embedding, where the colors change as the triangle bump moves. The figure suggests that SymReg can successfully learn to represent a data point using the location of two bumps. For comparison, we show the embedding found by VAE and SimCLR [10].

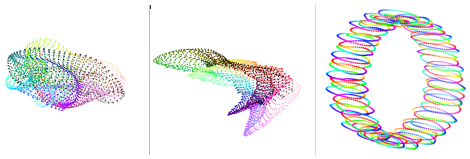

Figure 4: (left) SimCLR (middle) VAE (right) SymReg embeddings of double-bump world.

7.2 Experiments on Active and Passive Decomposition

In this section, we first contrast active and passive decomposition in their ability to disentangle the two bumps in the double bump world. We observe that while both can decompose the embedding into a product form $SO(2) \times SO(2)$, only active decomposition leads to disentanglement. Section 7.2.2 applies active decomposition to a more complex setting of ego-motion, where SymReg can decompose the representation of the agent's state into location and orientation.

7.2.1 Decomposition of the Double-Bump World

Here, we compare the active and passive decomposition for the same double-bump world. While the ground truth is $SO(2) \times SO(2)$, SymReg uses the larger group $E(2) \times E(2)$. In the active case, each subgroup moves one of the bumps, and the loss of Equation (5) is used to learn an embedding for each subgroup. In the passive case, both bumps move randomly. Figure 5 compares the decomposed

embedding found in each case. While in both cases, the $SO(2) \times SO(2)$ torus is decomposed into a product of circles, only the active case successfully disentangles the two bumps. Note that the color of each point is based on the location of the triangle bump. Our results agree with Caselles-Dupré et al. [8] who claim that learning a disentangled representation requires interaction with the environment; see also [42, 40]. However, we note that while the disentangling of the bump movements does not happen in the passive case, we can still successfully "decompose" the embedding.

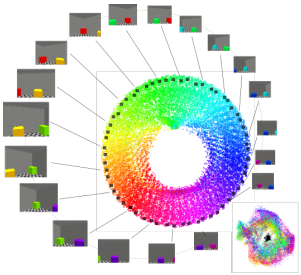

Figure 6: **Decomposition of the ego-motion.** The dataset contains a first-person view of a room. Transformations include right-left rotation and forward-backward movement. The equivariant embedding is produced by active decomposition using these two transformations, where the ring-structured manifold corresponds to the rotation action, and the smaller manifold corresponds to translations. Color-coding shows the ground truth angle of the image. The black square markers show the traversal of the embedding as the agent rotates while standing in the middle of the room. Note that black squares are concentrated in the center of the second manifold.

7.2.2 Active Decomposition for Ego-Motion

We used a modified version of the single-room environment of MiniWorld [13] for this experiment. The agent is standing in a 3D room containing eight differently colored boxes around the walls. A map of the room can be seen in Figure 7. An observation consists of a first-person view of the room, downsampled to 32×32 pixels. The agent can rotate left/right or move forward/backward. We learn an $E(2) \times E(2)$ equivariant embedding using the active decomposition objective of Equation (5). Each mini-batch consists of 64 random observations and the result of applying all four actions in those states (4×64 samples in total).

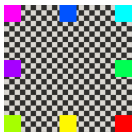

Figure 7: Top view of the room

Figure 6 visualizes the embedding of the input in two sub-figures, where the more prominent figure shows the embedding corresponding to the rotation action, and the more petite figure (bottom right) shows the embedding corresponding to forward-backward movement. The first figure also shows the first-person view when the agent rotates while standing in the middle of the room. The corresponding markers collapse around the center of the second embedding, demonstrating an intuitive embedding parameterized by rotation angle and location. Walking straight across the room also produces the expected behavior of traversing the second manifold while the rotation angle, for the most part, remains fixed (not shown).

7.3 Quantitave Evaluation in Downstream Tasks

7.3.1 World Modelling

We select the Atari games Pong and Space Invaders as our environments for the world modeling experiments. These environments were previously used by Kipf et al. [31] to evaluate the Contrastive Structured World Model (C-SWM). We train the encoder using Euclidean SymReg of Equation (4), freeze it, and then learn a Multi-Layer Perceptron (MLP) based transition function in the latent space. Following Kipf et al. [31], we report Hits at Rank 1 (H@1) and Mean Reciprocal Rank (MRR), which are invariant to the embedding scale. These evaluation metrics measure the relative closeness of the following state's representation predicted by the transition model and the representation of the observed next state. We use a set of reference state representations to measure the relative closeness (embedding random observations from the experience buffer). Table 1 reports these measures and shows that a simple transition model learned on top of our embedding outperforms C-SWM in both games.

Table 1: Hits at Rank 1 (H@1) and Mean Reciprocal Rank (MRR) of different method.

Environment	Method	H@1	MRR
Atari Pong	World Model(AE)	$23.8_{\pm 3.3}$	$44.7_{\pm 2.4}$
	World Model(VAE)	$1.0_{\pm 0.0}$	$5.1_{\pm 0.1}$
	C-SWM	$36.5_{\pm 5.6}$	$56.2_{\pm 6.2}$
	Ours	$\mathbf{45.2}_{\pm \mathbf{3.4}}$	$\mathbf{60.2}_{\pm \mathbf{3.9}}$
Space Invaders	World Model(AE)	$40.2_{\pm 3.3}$	$59.6_{\pm 3.5}$
	World Model(VAE)	$1.0_{\pm 5.3}$	$5.3_{\pm 0.1}$
	C-SWM	$48.5_{\pm 7.0}$	$66.1_{\pm 6.6}$
	Ours	$\mathbf{54.2}_{\pm \mathbf{6.3}}$	$\mathbf{68.7}_{\pm \mathbf{5.1}}$

Other reported baselines use an AutoEncodcer (AE) and a Variational AutoEncoder (VAE) to learn embeddings.

7.3.2 Reinforcement Learning

Next, we consider three Mujoco environments: InvertedPendulum, Reacher, and Swimmer from OpenAI Gym [5] and learn directly from the image observations. We compare our model with Auto-Encoder (AE) and Self-supervised Learning (SSL) based baselines. While AE learns to reconstruct the image observations of the states, SSL learns to inject invariance (IN-SSL) or equivariance (EQ-SSL) to agent actions. Given a triplet (s, a, s), IN-SSL maximizes the likelihood of $f(s)$ and $f(s')$ being similar (SimCLR [11]). EQ-SSL of Dangovski et al. [19], in this context, additionally predicts the action that lead to the state-transition. We introduce two variations of each model. In the first variation, the low-dimensional embedding is used as a substitute for the high-dimensional input data without further adjustment (-decoupled). The second variation allows for fine-tuning during the reinforcement learning stage (-fine-tuned). We use random policy to collect trajectories for the pre-training and use Proximal Policy Optimization (PPO) [47] algorithm for the downstream RL task. To evaluate the data-efficiency of these models, we report the average reward collected over 10 episodes in the first 100,000 steps for Reacher and Swimmer and 30,000 steps for Inverted Pendulum in Table 2 (since Inverted Pendulum generally learns faster, we took a fewer number of steps.)

Table 2: Average reward collected over 10 episodes for various models in Inverted Pendulum, Reacher and Swimmer. We provide the standard errors using 5 random seeds.

METHODS	INVERTEDPENDULUM	REACHER	SWIMMER
VANILLA	500$_{\pm 150}$	-11$_{\pm 2.5}$	25.6$_{\pm 3.4}$
AE-DECOUPLED	30$_{\pm 15}$	-13$_{\pm 3.0}$	16$_{\pm 3.9}$
AE-FINETUNED	580$_{\pm 130}$	-11.5$_{\pm 3.2}$	26$_{\pm 4.3}$
IN-SSL-DECOUPLED	100$_{\pm 17}$	-15$_{\pm 2.6}$	12 $_{\pm 2.5}$
IN-SSL-FINETUNED	550$_{\pm 21}$	-12$_{\pm 4.1}$	25.9 $_{\pm 4.8}$
EQ-SSL-DECOUPLED	456$_{\pm 190}$	-14.8$_{\pm 3.1}$	18 $_{\pm 4.5}$
EQ-SSL-FINETUNED	710 $_{\pm 120}$	**-10**$_{\pm \mathbf{2.6}}$	27 $_{\pm 3.5}$
SYMREG-DECOUPLED	800$_{\pm 180}$	-14.5$_{\pm 3.1}$	21$_{\pm 4.1}$
DEC-SYMREG-DECOUPLED	600$_{\pm 200}$	-12.8$_{\pm 2.7}$	19$_{\pm 5.6}$
SYMREG-FINETUNED	**950**$_{\pm \mathbf{50}}$	-10$_{\pm 3.4}$	**31.5**$_{\pm \mathbf{3.9}}$

We see that, out of all the representation learning methods, learned representations of SymReg most adequately capture the structure of the environment in Inverted Pendulum since the RL agent just trained on the fixed representation (SymReg-decoupled) outperforms all of them, including vanilla PPO. In Reacher, SymReg, along with other non-generative models, performs poorly compared to the AE. We believe that this is because the representation is focused on transformations caused by the agent's actions while details that can be valuable from the reward's perspective — in this case, the small object that the Reacher should reach - are ignored. This observation points to a limitation of all non-generative approaches that fine-tuning can resolve. To further verify this, we combined SymReg with a Decoder (Dec-SymReg) and noticed a significant improvement in the performance of the decoupled variation. In Swimmer, again we see that learning the agent's transformations is not enough to get all the reward information as the background movement decides how far the agent has swum. Indeed, allowing the encoder to fine-tune allows the representations to reflect the reward information and improve performance.

Conclusion

We proposed to learn equivariant representations by enforcing geometric invariants associated with the given group. We demonstrate this to be a simple, intuitive, and yet effective approach for representation learning. However, our approach also has several limitations that we hope to address in future work. Most importantly, a recipe for creating SymReg losses for general Lie groups is currently missing. This is because we rely on known invariants for important Lie groups or geometries. In the future, we would like to understand data characteristics that motivate the choice of one Lie group over others. Practical use of SymReg for finite groups is also a topic we would like to explore. We would also like to investigate further the best choice of objectives or mechanism for preventing collapse when enforcing geometric invariants.

References

[1] *Anderson Brandon, Hy Truong Son, Kondor Risi*. Cormorant: Covariant Molecular Neural Networks // Advances in Neural Information Processing Systems. 32. 2019. 14537–14546.

[2] *Anonymous* . Learning Symmetric Representations for Equivariant World Models // Submitted to The Tenth International Conference on Learning Representations. 2022. under review.

[3] *Bengio Yoshua, Courville Aaron, Vincent Pascal*. Representation learning: A review and new perspectives // IEEE transactions on pattern analysis and machine intelligence. 2013. 35, 8. 1798–1828.

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

[5] *Brockman Greg, Cheung Vicki, Pettersson Ludwig, Schneider Jonas, Schulman John, Tang Jie, Zaremba Wojciech*. Openai gym // arXiv preprint arXiv:1606.01540. 2016.

[6] *Bronstein Michael M, Bruna Joan, Cohen Taco, Veličković Petar*. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges // arXiv preprint arXiv:2104.13478. 2021.

[7] *Burgess Christopher P, Higgins Irina, Pal Arka, Matthey Loic, Watters Nick, Desjardins Guillaume, Lerchner Alexander*. Understanding disentangling in beta-VAE // arXiv preprint arXiv:1804.03599. 2018.

[8] *Caselles-Dupré Hugo, Garcia-Ortiz Michael, Filliat David*. Symmetry-based disentangled representation learning requires interaction with environments // arXiv preprint arXiv:1904.00243. 2019.

[9] General E (2)-Equivariant Steerable CNNs. // . 2019.

[10] *Chen Ting, Kornblith Simon, Norouzi Mohammad, Hinton Geoffrey*. A simple framework for contrastive learning of visual representations // International conference on machine learning. 2020. 1597–1607.

[11] *Chen Ting, Kornblith Simon, Norouzi Mohammad, Hinton Geoffrey E.* A Simple Framework for Contrastive Learning of Visual Representations // CoRR. 2020. abs/2002.05709.

[12] *Chen Xi, Duan Yan, Houthooft Rein, Schulman John, Sutskever Ilya, Abbeel Pieter*. Infogan: Interpretable representation learning by information maximizing generative adversarial nets // arXiv preprint arXiv:1606.03657. 2016.

[13] *Chevalier-Boisvert Maxime*. gym-miniworld environment for OpenAI Gym. 2018.

[14] *Cohen Taco, Welling Max*. Group equivariant convolutional networks // International conference on machine learning. 2016. 2990–2999.

[15] *Cohen Taco S., Geiger Mario, Köhler Jonas, Welling Max*. Spherical CNNs // International Conference on Learning Representations. 2018.

[16] *Cohen Taco S, Geiger Mario, Weiler Maurice*. A General Theory of Equivariant CNNs on Homogeneous Spaces // Advances in Neural Information Processing Systems. 32. 2019. 9145–9156.

[17] *Cohen Taco S, Welling Max*. Transformation properties of learned visual representations // arXiv preprint arXiv:1412.7659. 2014.

[18] *Dangovski Rumen, Jing Li, Loh Charlotte, Han Seungwook, Srivastava Akash, Cheung Brian, Agrawal Pulkit, Soljačić Marin*. Equivariant Contrastive Learning // arXiv preprint arXiv:2111.00899. 2021.

[19] *Dangovski Rumen, Jing Li, Loh Charlotte, Han Seungwook, Srivastava Akash, Cheung Brian, Agrawal Pulkit, Soljacic Marin*. Equivariant Self-Supervised Learning: Encouraging Equivariance in Representations // International Conference on Learning Representations. 2022.

[20] *Dehmamy Nima, Walters Robin, Liu Yanchen, Wang Dashun, Yu Rose*. Automatic Symmetry Discovery with Lie Algebra Convolutional Network // Advances in Neural Information Processing Systems. 2021. 34.

[21] *Ermolov Aleksandr, Siarohin Aliaksandr, Sangineto Enver, Sebe Nicu*. Whitening for self-supervised representation learning // International Conference on Machine Learning. 2021. 3015–3024.

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

[23] *Finkelshtein Ben, Baskin Chaim, Maron Haggai, Dym Nadav*. A simple and universal rotation equivariant point-cloud network // arXiv preprint arXiv:2203.01216. 2022.

[24] *Finzi Marc, Welling Max, Wilson Andrew Gordon*. A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups // arXiv preprint arXiv:2104.09459. 2021.

[25] *Fuchs Fabian, Worrall Daniel, Fischer Volker, Welling Max*. Se (3)-transformers: 3d roto-translation equivariant attention networks // Advances in Neural Information Processing Systems. 2020. 33. 1970–1981.

[26] *Hadsell Raia, Chopra Sumit, LeCun Yann*. Dimensionality reduction by learning an invariant mapping // 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). 2. 2006. 1735–1742.

[27] *He Kaiming, Fan Haoqi, Wu Yuxin, Xie Saining, Girshick Ross*. Momentum contrast for unsupervised visual representation learning // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. 9729–9738.

[28] *Higgins Irina, Amos David, Pfau David, Racaniere Sebastien, Matthey Loic, Rezende Danilo, Lerchner Alexander*. Towards a definition of disentangled representations // arXiv preprint arXiv:1812.02230. 2018.

[29] *Hinton Geoffrey E, Krizhevsky Alex, Wang Sida D*. Transforming auto-encoders // International conference on artificial neural networks. 2011. 44–51.

[30] *Kingma Diederik P, Welling Max*. Auto-encoding variational bayes // arXiv preprint arXiv:1312.6114. 2013.

[31] *Kipf Thomas, Pol Elise van der, Welling Max*. Contrastive Learning of Structured World Models // International Conference on Learning Representations. 2020.

[32] *Klein Felix*. A comparative review of recent researches in geometry // Bulletin of the American Mathematical Society. 1893. 2, 10. 215–249.

[33] *Kondor Risi, Son Hy Truong, Pan Horace, Anderson Brandon, Trivedi Shubhendu*. Covariant compositional networks for learning graphs // arXiv preprint arXiv:1801.02144. 2018.

[34] *Kondor Risi, Trivedi Shubhendu*. On the generalization of equivariance and convolution in neural networks to the action of compact groups // International Conference on Machine Learning. 2018. 2747–2755.

[35] *Kulkarni Tejas D, Whitney Will, Kohli Pushmeet, Tenenbaum Joshua B*. Deep convolutional inverse graphics network // arXiv preprint arXiv:1503.03167. 2015.

[36] *LeCun Yann, Bengio Yoshua, others* . Convolutional networks for images, speech, and time series // The handbook of brain theory and neural networks. 1995. 3361, 10. 1995.

[37] *Lenc Karel, Vedaldi Andrea*. Learning covariant feature detectors // European conference on computer vision. 2016. 100–117.

[38] *Lenssen Jan Eric, Fey Matthias, Libuschewski Pascal*. Group Equivariant Capsule Networks // Advances in Neural Information Processing Systems. 31. 2018. 8844–8853.

[39] *Maron Haggai, Ben-Hamu Heli, Shamir Nadav, Lipman Yaron*. Invariant and equivariant graph networks // arXiv preprint arXiv:1812.09902. 2018.

[40] *Mita Graziano, Filippone Maurizio, Michiardi Pietro*. An identifiable double VAE for disentangled representations // International Conference on Machine Learning. 2021. 7769–7779.

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

[42] *Painter Matthew, Hare Jonathon, Prugel-Bennett Adam*. Linear disentangled representations and unsupervised action estimation // arXiv preprint arXiv:2008.07922. 2020.

[43] *Qi Charles R, Su Hao, Mo Kaichun, Guibas Leonidas J*. Pointnet: Deep learning on point sets for 3D classification and segmentation // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. 652–660.

[44] *Quessard Robin, Barrett Thomas D, Clements William R*. Learning group structure and disentangled representations of dynamical environments // arXiv preprint arXiv:2002.06991. 2020.

[45] *Ravanbakhsh Siamak, Schneider Jeff, Poczos Barnabas*. Equivariance through parameter-sharing // International Conference on Machine Learning. 2017. 2892–2901.

[46] *Sabour Sara, Frosst Nicholas, Hinton Geoffrey E*. Dynamic Routing Between Capsules // Advances in Neural Information Processing Systems. 30. 2017. 3856–3866.

[47] *Schulman John, Wolski Filip, Dhariwal Prafulla, Radford Alec, Klimov Oleg*. Proximal Policy Optimization Algorithms. 2017.

[48] *Shakerinava Mehran, Ravanbakhsh Siamak*. Equivariant Networks for Pixelized Spheres // International Conference on Machine Learning. 2021. 9477–9488.

[49] *Tenenbaum Joshua B, De Silva Vin, Langford John C*. A global geometric framework for nonlinear dimensionality reduction // science. 2000. 290, 5500. 2319–2323.

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

[51] *Tian Yonglong, Krishnan Dilip, Isola Phillip*. Contrastive multiview coding // arXiv preprint arXiv:1906.05849. 2019.

[52] *Villar Soledad, Hogg David, Storey-Fisher Kate, Yao Weichi, Blum-Smith Ben*. Scalars are universal: Equivariant machine learning, structured like classical physics // Advances in Neural Information Processing Systems. 2021. 34.

[53] *Wood Jeffrey, Shawe-Taylor John*. Representation theory and invariant neural networks // Discrete applied mathematics. 1996. 69, 1-2. 33–60.

[54] *Worrall Daniel E, Garbin Stephan J, Turmukhambetov Daniyar, Brostow Gabriel J*. Interpretable transformations with encoder-decoder networks // Proceedings of the IEEE International Conference on Computer Vision. 2017. 5726–5735.

[55] *Wu Zhirong, Song Shuran, Khosla Aditya, Yu Fisher, Zhang Linguang, Tang Xiaoou, Xiao Jianxiong*. 3d shapenets: A deep representation for volumetric shapes // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. 1912–1920.

[56] *Zaheer Manzil, Kottur Satwik, Ravanbakhsh Siamak, Poczos Barnabas, Salakhutdinov Russ R, Smola Alexander J*. Deep Sets // Advances in Neural Information Processing Systems. 30. 2017. 3391–3401.

[57] *Zbontar Jure, Jing Li, Misra Ishan, LeCun Yann, Deny Stéphane*. Barlow twins: Self-supervised learning via redundancy reduction // arXiv preprint arXiv:2103.03230. 2021.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

(b) Did you describe the limitations of your work? [Yes] See the Conclusion section (Section 7.3.2).

(c) Did you discuss any potential negative societal impacts of your work? [No] We do not foresee any negative societal impact by our work.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] See the appendix.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the corresponding section for each experiment.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All of the quantitative experiments contain confidence intervals.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] This paper does not contain any large scale experiments and the specific resources that were used is not relevant to the work.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We have cited the environments that we used in the experiments.

(b) Did you mention the license of the assets? [No] The license is available on each environments GitHub page.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] We did not produce any new assets.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The environments that we used are open-source.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The data that we are using clearly does not contain such content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]