Unsupervised Learning of Group Invariant and Equivariant Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Equivariant neural networks, whose hidden features transform according to representations of a group G acting on the data, exhibit training efficiency and an improved generalisation performance. In this work, we extend group invariant and equivariant representation learning to the field of unsupervised deep learning. We propose a general learning strategy based on an encoder-decoder framework in which the latent representation is separated in an invariant term and an equivariant group action component. The key idea is that the network learns to encode and decode data to and from a group-invariant representation by additionally learning to predict the appropriate group action to align input and output pose to solve the reconstruction task. We derive the necessary conditions on the equivariant encoder, and we present a construction valid for any G, both discrete and continuous. We describe explicitly our construction for rotations, translations and permutations. We test the validity and the robustness of our approach in a variety of experiments with diverse data types employing different network architectures.

1 Introduction

An increasing body of work has shown that incorporating knowledge about underlying symmetries in neural networks as inductive bias can drastically improve the performance and reduce the amount of data needed for training Cohen & Welling (2016a); Bronstein et al. (2021). For example, the equivariant design with respect to the translation symmetry of objects in images proper of convolutional neural networks (CNNs) has revolutionized the field of image analysis LeCun et al. (1995). Message Passing neural networks, respecting permutation symmetries in graphs, have enabled powerful predictive models on graph-structured data Gilmer et al. (2017); Defferrard et al. (2016). Recently, much work has been done utilizing 3D rotation and translation equivariant neural networks for point clouds and volumetric data, showing great success in predicting molecular ground state energy levels with high fidelity Miller et al. (2020); Anderson et al. (2019); Klicpera et al. (2020); Schütt et al. (2021). Invariant models take advantage of the fact that often properties of interest, such as the class label of an object in an image or the ground state energy of a molecule, are invariant to certain group actions (e.g., translations or rotations), while the data itself is not (e.g., pixel values, atom coordinates).

There are several approaches to incorporate invariance into the learned representation of a neural network. The most common approach consists of teaching invariance to the model by data augmentation: during training, the model must learn that a group transformation on its input does not affect its label. While this approach can lead to improved generalization performance, it reduces training efficiency and quickly becomes impractical for higher dimensional data Thomas et al. (2018). A second technique, known as feature averaging, consists of averaging model predictions over group transformations of the input Puny et al. (2021). While feasible with finite groups, this method requires, for instance, sampling for infinite groups Lyle et al. (2020). A third approach is to impose invariance

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

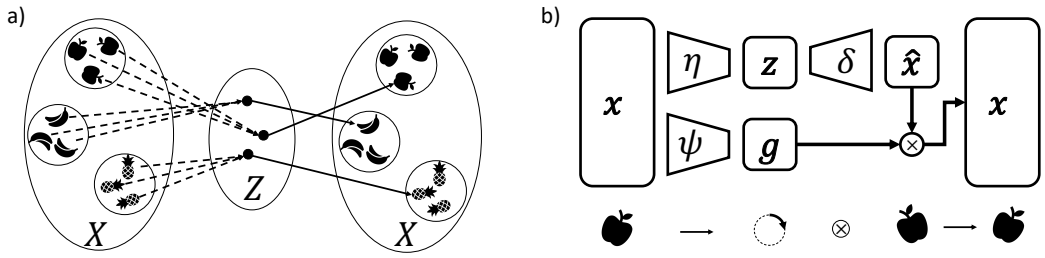

Figure 1: a) Schematic of the learning task this work is concerned with. Data points $x \in X$ are encoded to and decoded from latent space Z. Points in the same orbit in X are mapped to the same point (orbit) $z \in Z = X/G$. Latent points z are mapped to canonical elements $\hat{x} \in \{\rho_X(g)x | \forall g \in G\}$. b) Schematic of our proposed framework with data points x, encoding function η, decoding function δ, canonical elements $\hat{x}$, group function ψ and group action g

as a model architectural design. The simplest option is to restrict the function to be learned to be a composition of symmetric functions only Schütt et al. (2018). Such choice, however, can significantly restrict the functional form of the network. A more expressive variation of this approach consists of an equivariant neural network, followed by a symmetric function. This allows the network to leverage the benefits of invariance while having a larger capacity due to the less restrictive nature of equivariance. In fact, in many real-world application, equivariance is beneficial if not necessary Smidt (2020); Miller et al. (2020). For example, the interaction of a molecule (per se rotational invariant) with an external magnetic field is an intrinsically equivariant problem.

All aforementioned considerations require some sort of supervision to extract invariant representations from data. Unsupervised learning of group invariant representations, despite its potential in the field of representation learning, has been impaired by the fact that the representation of the data in general does not manifestly exhibit the group as a symmetry. For instance, in the case of an encoder-decoder framework in which the bottleneck layer is invariant, the reconstruction is only possible up to a group transformation. Nevertheless, the input data is typically parametrized in terms of coordinates in some vector space X, and the reconstruction task can only succeed by employing knowledge about the group action on X.

Following this line of thought, this work is concerned with the question: *Can we learn to extract both the invariant and equivariant representations of data in an unsupervised way?*

To this end, we introduce a group-invariant representation learning method that encodes data in a group-invariant latent code and a group action. By separating the embedding in a group-invariant and a group-equivariant part, we can learn expressive lower-dimensional group-invariant representations utilizing the power of autoencoders (AEs). In particular, all our results trivially extend to variational AEs (VAEs), thus advancing generative approaches of group-invariant data. We can summarize the main contributions of this work as follows:

- We introduce a novel framework for learning group equivariant representations. Our representations are *by construction* separated in an invariant and group action function (equivariant) component.
- We characterize the mathematical conditions of the group action function component and we propose an explicit construction suitable for *any* group G. To the best of our knowledge, this is the first method for unsupervised learning of separated invariant-equivariant representations valid for any group.
- We show in various experiments the validity and flexibility of our framework by learning representations of diverse data types with different network architectures.

2 Method

2.1 Background

We begin this section by introducing the basic concepts which will be central in our work.

A group G is a set equipped with an operation (here denoted $\cdot$) which is associative as well as having an identity element e and inverse elements. In the context of data, we are mainly interested in how groups represent geometric transformations by acting on spaces and, in particular, how they describe the symmetries of an object or of a set. In either case, we are interested in how groups act on spaces. This is represented by a **group action**: given a set X and a group G, a (left) action of G on X is a map $\rho : G \times X \to X$ such that it respects the group property of associativity and identity element. If X is a vector space, which we will assume for the remainder of the text, we refer to group actions of the form $\rho_X : G \to \mathrm{GL}(X)$ as **representations** of G, where the general linear group of degree n $\mathrm{GL}(X)$ is represented by the set of $n \times n$ invertible matrices. Given a group action, a concept which will play an important role in our discussion is given by the fixed points of such an action. Formally, given a point $x \in X$ and an action (representation) ρ_X of G on X, the **stabilizer** of G with respect to x is the subgroup $G_x = \{g \in G | \rho_X(g)x = x\} \subset G$.

In the context of representation learning, we assume our data to be defined as the space of representation-valued functions on some set V, i.e., $X = \{f | f : V \to W\}$. For instance, a point cloud in three dimensions can be represented as the set of functions $f : \mathbb{R}^3 \to \mathbb{Z}_2$, assigning to every point $\mathbf{r} \in \mathbb{R}^3$ the value $f(\mathbf{r}) = 0$ (the point is not included in the cloud) or $f(\mathbf{r}) = 1$ (the point is included in the cloud). Representations ρ_V of a group G on V can be extended to representations on f, and therefore on X, $\rho_X : G \to \mathrm{GL}(X)$, as follows

$$[\rho_X(g)f](x) \equiv \rho_W(g)f(\rho_V(g^{-1})x) \ . \tag{1}$$

In what follows, we will then only refer to representations for the space X, implicitly referring to equation (1) for mapping back to how the various components transform. A map $\varphi : V \to W$ is said to be G**-equivariant** with respect to the actions (representations) ρ_V, ρ_W if $\varphi(\rho_V(g)v) = \rho_W(g)\varphi(v)$ for every $g \in G$ and $v \in V$. Note that G-invariance is a particular case of the above, where we take ρ_V, ρ_W to be the trivial representations. An element $x \in X$ can be described in terms of a G-invariant component and a group element $g \in G$, as follows: let $\varphi_{\text{inv}} : X \to X/G$, be an invariant map mapping each element $x \in X$ to a corresponding canonical element $\hat{x}$ in the orbit in the quotient space X/G. Then for each $x \in X$ there exist a $g \in G$ such that $x = \rho_X(g)\varphi_{\text{inv}}(x)$.

2.2 Problem Definition

We consider a classical autoencoder framework with encoding function $\eta : X \to Z$ and decoding function $\delta : Z \to X$, mapping between the data domain X, and latent domain Z, minimizing the reconstruction objective $d(\delta(\eta(x)), x)$, with a difference measure d (e.g., L_p norm). As discussed above, we wish to learn the invariant map η (φ_{inv} in the previous paragraph), thus

Property 2.1. *The encoding function $\eta : X \to Z$ is G-invariant, i.e., $\eta(\rho_X(g)x) = \eta(x)\ \forall x \in X, \forall g \in G$.*

The decoding function δ maps the G-invariant representation $z \in Z$ back to the data domain X. However, as z is G-invariant, δ can at best map $\eta(x) \in Z$ back to an element $\hat{x} \in X$ such that $\hat{x} \in \{\rho_X(g)x | \forall g \in G\}$, i.e., an element in the orbit of x through G. This is depicted in Figure 1a. Thus, the task of the decoding function $\delta : Z \to X$ is to map encoded elements $z = \eta(x) \in Z$ to an element $\hat{x} \in X$ such that $\exists \hat{g}_x \in G$ such that

$$\delta(\eta(x)) = \hat{x} = \rho_X(\hat{g}_x)x \ . \tag{2}$$

We call $\hat{x}$ the **canonical** element of the decoder δ. We can rewrite the reconstruction objective with a G-invariant encoding function η as $d(\rho_X(\hat{g}^{-1})\delta(\eta(x)), x)$. One of the main results of this work consists in showing that $\hat{x}$ and $\hat{g}_x$ can be *simultaneously* learned by a suitable neural network. That is, we have the following property of our learning scheme:

Property 2.2. *There exists a **learnable** function $\psi : X \to G$ such that, given suitable η, δ as described above the relation $\rho_X(\psi(x))\delta(\eta(x)) = x$, holds for all $x \in X$.*

We call any function ψ satisfying (2.2) a **suitable** group function. Figure 1b describes schematically our proposed framework. In what follows, we will first characterize the defining properties of suitable group functions. Subsequently, we will describe our construction, valid for any group G.

2.3 Predicting Group Actions

In the following we further characterize the properties of ψ. We begin by stating two key results, while we refer to the Appendix A for the proofs.

Proposition 2.3. *Any suitable group function $\psi : X \to G$ is G-equivariant at a point $x \in X$ up the stabilizer G_x, i.e., $\psi(\rho_X(g)x) \subseteq g \cdot \psi(x)G_x$.*

Proposition 2.4. *The image of any suitable group function $\psi : X \to G$ is surjective into $\frac{G}{G_X}$, where G_X is the stabilizers of all the points of X.*

Let us briefly discuss an example. Suppose $X = \{x = (x_0, x_1, x_2, x_3) \in \mathbb{R}^{4\times 2} | x_i = \rho_{\mathbb{R}^2}(g_{\theta=\pi/2})^i x_0,\ x_0 \in \mathbb{R}^2\}$ and $G = \mathrm{SO}(2)$. X describes all collections of vertices of squares centered at the origin of $\mathbb{R}^2$, and it is easy to check that $g_X = \mathbb{Z}_4$, generated by a $\pi/2$ rotation around the origin. In this case, any such square can be brought to any other square (of the same radius) by a rotation of an angle $\theta < \pi/2$, thus $\mathrm{Im}\psi \supseteq \{g_\theta \in \mathrm{SO}(2) | 0 \leq \theta \leq \pi/2\} = \mathrm{SO}(2)/\mathbb{Z}_4$.

Combining the two propositions above we have the following

Lemma 2.5. *Any suitable group function ψ is an isomorphism $O_x \simeq G/G_x$ for any $x \in X$, where $O_x \subset X$ is the orbit of x with respect to G in X.*

2.4 Proposed Construction

Next, we turn to our proposed construction of a class of suitable group functions that satisfy Property 2.2 for any data space X and group G. As we described above, these functions must be learnable.

Property 2.6 (**Proposed construction**). *Without loss of generality, we write our target function $\psi = \xi \circ \mu$, where $\mu : X \to Y$ is a learnable map between the data space X and the embedding space Y, while $\xi : Y \to G$ is a deterministic map. Our construction is further determined by the following properties:*

- *We impose $\mu : X \to Y$ to be G-equivariant, that is, $\mu(\rho_X(g)x) = \rho_Y(g)\mu(x)$ for all $x \in X$ and $g \in G$.*
- *We ask that Y is an homogeneous space, that is, given any element $y_0 \in Y$, every element $y \in Y$ can be written as $y = \rho_Y(g)y_0$ for some $g \in G$.*
- *The map $\xi : Y \to G$ is defined as follows: $\xi(y) = g$ such that $y = \rho_Y(g)y_0$ for any chosen point $y_0 \in Y$.*

In what follows we will show that our construction satisfies the properties of the previous section. For proofs see Appendix. We begin with the following

Proposition 2.7. *Let $\psi = \xi \circ \mu$ be a suitable group function and let $\mu : X \to Y$ be G-equivariant. Then, $G_x = G_{\mu(x)}$ for all $x \in X$.*

The result of the above proposition is crucial for our desired decomposition of the learned embedding, as it ensures that no information about the group action on X is lost through the map μ: if a group element acts non-trivially in X, it will also act non-trivially in Y.

Proposition 2.8. *Given y, y_0, the element g such that $y \equiv \rho_Y(g)y_0$ is unique up to the stabilizer G_{y_0}.*

This proposition establishes the equivariant properties of the map ξ. Finally, we have

Proposition 2.9. *Let $\psi = \xi \circ \mu$ where μ and ξ are as described above. Then, ψ is a suitable group function.*

2.5 Intuition Behind the Proposed Framework

We conclude this rather technical section with a comment on the intuition behind our construction. Assuming for simplicity that the domain set V admits the structure of vector space, Y represents the space spanned by **all** basis vectors of V. The point y_0 represent a canonical orientation of such basis, and the element $\xi(y) = g$ is the group element corresponding to a basis transformation. As all elements can be expressed in terms of coordinates with respect to a given basis, it is natural to consider a canonical basis for all orbits, justifying the assumption of homonogeneity of the space Y.

Further, let us assume that the invariant autoencoder correctly solves its task, $x \sim \delta(\eta(x))$. Now let $\hat{x} \in O_x$ such that $\hat{x} = \delta(\eta(x))$, and by definition, $\hat{x} = \rho_V(g)x$ for some $g \in G$. Now, the

correct orbit element is identified when $\psi(\hat{x}) = e$, since $\psi(x) = g^{-1} \cdot \psi(\hat{x}) = g^{-1}$ and thus $\rho_X(g^{-1})\delta(\eta(x)) = \rho_X(g^{-1})x' = x$. Hence, during training ψ needs to learn which orbit elements are decoded as "canonical", i.e., without the need of an additional group transformation. To clarify, here "canonical" does not reflect any specific property of the element, but it simply refers to the orientation learned from the decoder during training. In fact, different decoder architectures or initializations will lead to different canonical elements.

Finally, note how the different parts of our proposed framework, as visualized in Figure 1b, can be jointly trained by minimizing the objective $d(\rho_X(\psi(x))\delta(\eta(x)), x)$, which is *by construction* group invariant, i.e., not susceptible to potential group-related bias in the data (e.g. data that only occurs in certain orientations).

3 Application to Common Groups

In this section we describe how our framework applies to a variety of common groups which we will then implement in our experiments. As discussed in Section 2.2 and visualized in Figure 1b, the main components of our proposed framework are the encoding function η, the decoding function δ and the group function ψ. As stated in Property 2.1, the only constraint for the encoding function η is that it has to be group invariant. This is in general straightforward to achieve for different groups as we will demonstrate in Section 5. Our proposed framework does not constrain the decoding function δ other than that it has to map elements from the latent space Z to the data domain X. Hence, δ can be designed independently of the group of interest. The main challenge is in defining the group function $\psi = \xi \circ \mu$ such that it satisfies Property 2.2. Following Property 2.6 we now turn to describing our construction of ξ, μ and Y for a variety of common groups.

Orthogonal group $\mathbf{SO}(2)$. The Lie group $\mathrm{SO}(2)$ is defined as the set of all rotations about the origin in $\mathbb{R}^2$. We take Y to be the circle $S^1 \subset \mathbb{R}^2$, that is, the space spanned by unit vectors in $\mathbb{R}^2$. Now, S^1 is a homogeneous space: any two points $s_0, s_1 \in S^1$ are related by a rotation. Without loss of generality, we take the reference vector y_0 to be the vector $(1, 0) \in S^1$. Then given a vector $y \in S^1$, we can write

$$y = \begin{pmatrix} y_x \\ y_y \end{pmatrix} = \begin{pmatrix} y_x & -y_y \\ y_y & y_x \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} . \tag{3}$$

thus, the function $\xi : S^1 \to \mathrm{SO}(2)$ is determined by $\xi(y) = g_\theta$ such that $\theta = \arccos(y_x) = \arcsin(y_y)$.

Orthogonal group $\mathbf{SO}(3)$. We assume that X has no fixed points, as this is usually the case for generic shapes (point clouds) in $\mathbb{R}^3$. It would be tempting to take Y to be the sphere $S^2 \subset \mathbb{R}^3$, that is, the space spanned by unit vectors in $\mathbb{R}^3$. While this space is homogeneous, it does not satisfy the condition that the stabilizers of G are trivial. In fact, given any vector $y_1 \in S^2$, we have $G_{y_1} = \{g \in \mathrm{SO}(3) | g \text{ is a rotation about } y_1\}$.

In order to construct a space with the desired property, consider a second vector $w_2 \in S^2$ orthogonal to y_1, $y_2 \subset y_1^\perp$. Taking Y to be the space spanned by $y_1, y_2 \in S^2$, it is easy to see that now all the stabilizers are trivial. Finally, let $y_3 = y_1 \times y_2 \in S^2$, then we construct the rotation matrix

$$R = \begin{pmatrix} y_{1,x} & y_{2,x} & y_{3,x} \\ y_{1,y} & y_{2,y} & y_{3,y} \\ y_{1,z} & y_{2,z} & y_{3,z} \end{pmatrix} \text{, which satisfies } \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = R \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}^\intercal = R y_0 .$$

Symmetric group S_n. A suitable space Y is the set of ordered collections of unique elements of the set $M = \{1, 2, \ldots, n\}$. For instance, for $n = 3$, we have $Y = \{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$. It is trivial to see that the action of the permutation group on the set Y is free, that is, all the stabilizers are trivial. Explicitely, given any permutation-equivariant vector $w \in \mathbb{R}^n$, we obtain an element $y = \mathrm{argsort}(w) \in Y$. Moreover, it is also obvious that any element in Y can be written as $P_\sigma(1, 2, \ldots, n) = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, that is, a group element acting on the canonical $y_0 = (1, 2, \ldots, n)$.

Translation group T_n. Here we take $Y = \mathbb{R}^n$, which is homogeneous with respect to the translation group. In fact, any vector $y \in Y$ can be trivially written as $y = y + \mathbf{0} = y + y_0$, where $\mathbf{0}$ is the origin of $\mathbb{R}^n$. Our group function takes therefore the form $\xi(y) = y$.

Euclidean group SE(n). A generic transformation of the Euclidean group on a n-dimensional representation $v \in V$ is

$$v \mapsto Av + b \,, \quad A \in \mathrm{SO}(n) \,, b \in T_n \,. \tag{4}$$

Let $\mu = (\mu_1, \mu_2, \ldots, \mu_{n+1})$ be a collection of $n+1$ n-dimensional SE(n)-equivariant vectors, that is, $\mu_i(\rho_X(g)x) = \rho_Y(g)\mu(x)$, $i = 1, \ldots, n$. We construct $\widehat{y}_a = (\mu_a - \mu_{n+1})/||\mu_a - \mu_{n+1}|| \in S^n$, $a = 1, \ldots, n$, where S^n is the unit n-dimensional sphere. These n ortho-normal vectors are **translation invariant** but rotation equivariant, and are suitable to construct the rotation matrix

$$R = (\widehat{y}_1 \quad \widehat{y}_2 \quad \cdots \quad \widehat{y}_n) \,, \tag{5}$$

while the extra vector $\widehat{y}_{n+1} = \mu_{n+1}$ can be used to predict the translation action. Putting all together, the space Y is described by n vectors $y_a = \widehat{y}_a + \widehat{y}_{n+1}$, and $y_0 = I_n$ is the $n \times n$ unit matrix, as

$$(R + \widehat{y}_{n+1})I_n = (\widehat{y}_1 \quad \cdots \quad \widehat{y}_n)^\intercal + \widehat{y}_{n+1}I_n = (y_1 \quad \cdots \quad y_n)^\intercal \,. \tag{6}$$

4 Related Work

Group equivariant neural networks. Group equivariant neural networks have shown great success for various groups and data types. There are two main approaches to implement equivariance in a layer and, hence, in a neural network. The first, and perhaps the most common, imposes equivariance on the space of functions and features learned by the network. Thus, the parameters of the model are constrained to satisfy equivariance Thomas et al. (2018); Weiler & Cesa (2019a); Weiler et al. (2018a); Esteves et al. (2020). The disadvantage of this approach consists in the difficulty of designing suitable architectures for all components of the model, transforming correctly under the group action Xu et al. (2021). The second approach to equivariance consists in lifting the map from the space of features to the group G, and equivariance is defined on functions on the group itself Romero & Hoogendoorn (2020); Romero et al. (2020); Hoogeboom et al. (2018). Although this strategy avoids the architectural constraints, applicability is limited to homogeneous spaces Hutchinson et al. (2021) and involves an increased dimensionality of the feature space, due to the lifting to G. Equivariance has been explored in a variety of architecture and data structures: Convolutional Neural Networks Cohen & Welling (2016a); Worrall et al. (2017); Weiler et al. (2018c); Bekkers et al. (2018); Thomas et al. (2018); Dieleman et al. (2016); Kondor & Trivedi (2018); Cohen & Welling (2016b); Cohen et al. (2018); Finzi et al. (2020), Transformers Vaswani et al. (2017); Fuchs et al. (2020); Hutchinson et al. (2021); Romero & Cordonnier (2020), Graph Neural Networks Defferrard et al. (2016); Bruna et al. (2013); Kipf & Welling (2016); Gilmer et al. (2017); Satorras et al. (2021) and Normalizing Flows Rezende & Mohamed (2015); Köhler et al. (2019, 2020); Boyda et al. (2021). These methods are usually trained in a supervised manner and combined with a symmetric function (e.g. pooling) to extract group-invariant representations.

Group equivariant autoencoders. Another line of related work is concerned with group equivariant autoencoders. Such models utilize specific network architectures to encode and decode data in an equivariant way, resulting into equivariant representations only Hinton et al. (2011); Sabour et al. (2017); Kosiorek et al. (2019); Guo et al. (2019). Feige (2019) use weak supervision in an AE to extract invariant and equivariant representations. Winter et al. (2021) implement a permutation-invariant AE to learn graph embeddings, in which the permutation matrix for graph matching is learned during training. In that sense, the present work can be seen as a generalization of their approach for a generic data type and any group.

5 Experiments

In this section we present a diverse set of experiments for the various symmetry groups discussed in Section 3. [1]

5.1 Rotated MNIST

[1] Source code for the different implementations is publicly available at `https://github.com/...` (also see Supplementary Information)

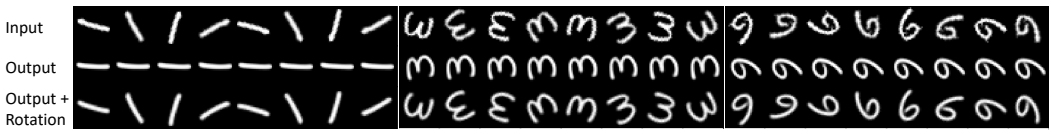

Figure 2: Input and predicted output for rotated versions of three MNIST images. Top row shows the input image successively rotated by $45°$. Middle row shows the decoded (canonical) image and bottom row shows the decoded image after applying the predicted rotation.

In the first experiment, we train an SO(2)-invariant autoencoder on the original (non-rotated) MNIST dataset and validate the trained model on the rotated MNIST dataset mni which consists of randomly rotated versions of the original MNIST dataset. For the functions η and ψ we utilize SO(2)-Steerable Convolutional Neural Networks Weiler & Cesa (2019b). For more details about the network architecture and training, we refer to Appendix B. In Figure 2 we show images in different rotations and the respective reconstructed images by the trained model. The model decodes the different rotated versions of the same image (i.e., elements from the same orbit) to the same canonical output orientation (second row in Figure 2). The trained model manages to predict the right rotation matrix (group action) to align the decoded image with the input image, resulting in an overall low reconstruction error. Note that the model never saw rotated images during training but still manages to encode and reconstruct them due to its inherent equivariant design. We find that the encoded latent representation is indeed rotation invariant (up to machine precision), but only for rotations of an angle $\theta = \frac{n \cdot \pi}{2},\ n \in \mathbb{N}$. For all other rotations, we see slight variations in the latent code, which, however, is to be expected due to interpolation artifacts for rotations on a discretized grid. Still, inspecting the 2d-projection of the latent code of our proposed model in Figure 3, we see distinct clusters for each digit class for the different images from the test dataset, independent of the orientation of the digits in the images. In contrast, the latent code of a classical autoencoder exhibits multiple clusters for different orientations of the same digit class.

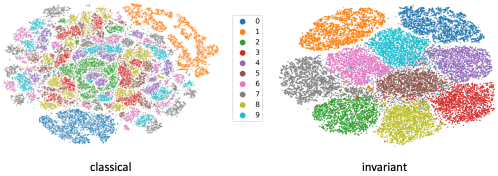

Figure 3: TSNE embedding of the encoded test dataset for a classical and our proposed SO(2) invariant autoencoder.

5.2 Set of Digits

Next, we train a permutation-invariant autoencoder on sets of digits. A set with N digits is represented by concatenating one-hot vectors of each digit in a $N \times D$-dimensional matrix, where we take $D = 10$. Notice that this matrix-representation of a set is *not* permutation invariant. We randomly sampled 1.000.000 different sets for training and 100.000 for the final evaluation with $N = 20, 30, 40, 100$, respectively, removing all permutation equivariant sets (i.e., there are no two sets that are the same up to a permutation). For comparison, we additionally trained a classical non-permutation-invariant autoencoder with the same number of parameters and layers as our permutation-invariant version. For more details on the network architecture and training we refer to Appendix C. Here, we demonstrate how the separation of the permutation-invariant information of the set (i.e., the composition of the set) from the (irrelevant) order-information results in a significant reduction of the space needed to encode the set. In Figure 4a, we plot the element-wise reconstruction accuracy of different sized sets for both models for varying embedding (bottleneck) sizes. As the classical autoencoder has to store both the composition of digits in the set (i.e., number of elements for each of the 10 digits classes) as well as their order in the permutation-dependent matrix representation, the reconstruction accuracy drops for increasing size of the set N for a fixed embedding size. For the same reason, perfect reconstruction accuracy is only achieved if the embedding dimension is at least as large as the number of digits in the set. On the contrary, our proposed permutation invariant autoencoder achieves perfect reconstruction accuracy with a significant lower embedding size. Crucially, as no order information has to be stored in the embedding, this embedding size for perfect reconstruction accuracy also stays the same for increasing size N of the set. In Figure 4b we show one example for a set x with $N = 100$ digits, with the predicted canonical orbit element $\hat{x}$ and the predicted permutation matrix. As perhaps expected, the canonical element clusters together digits with same value, while not using the commonly used

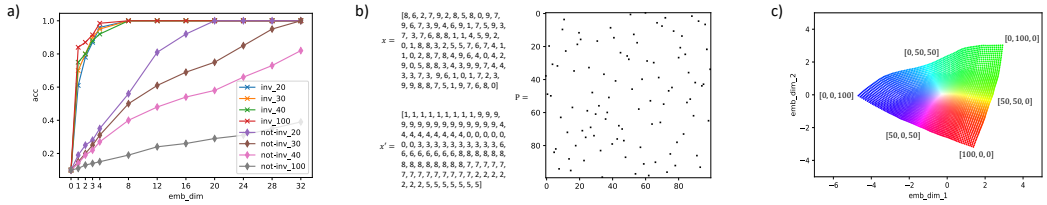

Figure 4: a) Element-wise reconstruction accuracy of our proposed permutation invariant autoencoder (cross) and a classical non-permutation invariant autoencoder (diamond) for different embedding and set sizes. b) Example set x with 100 elements with its canonical reconstruction $\hat{x}$ and the predicted permutation matrix P (resulting into a perfect reconstruction). One can confirm for oneself that, e.g., $x[38] = x'[0]$, matching $P[38, 0] = 1$. c) Best viewed in colour. Visualization of the two-dimensional embedding of a permutation-invariant autoencoder for all 5151 sets of 100 elements with 3 different element classes. Each point represents one set, colours represent set compositions (proportion of each element class, independent of the order). The three element classes are represented by the three colour channels (rgb). E.g., the higher the proportion of the third element class, the bluer the point, while same proportions results into white (point in the center).

order of Arabic numerals. This learned order (here [1,9,4,0,3,6,8,7,2,5]) stays fixed for the trained network for different inputs but changes upon re-initialization of the network.

In Figure 4c we show the two-dimensional embedding of a permutation invariant autoencoder trained on set of $N = 100$ elements chosen from $D = 3$ different classes (e.g. digits 0,1,2). As the sets only consists of 3 different elements (but in different compositions and order) we can visualize the $\binom{D+N-1}{N} = \binom{102}{100} = 5151$ elements in the two-dimensional embedding and colour them according to their composition. As our proposed auteoncoder only needs to store the information about the set composition and not the order, the embedding is perfectly structured with respect to the composition as can be seen by the colour gradients in the visualization of the embedding.

5.3 Point Cloud

Point clouds are a common way to describe objects in 3D space, such as the atom positions of a molecule or the surface of an object. As such, they usually adhere to 3D translation and rotation symmetries and are unordered, i.e., permutation invariant. Hence, we investigate in the next experiment a combined SE(3)- and S_N-invariant autoencoder for point cloud data. We use the Tetris Shape toy dataset Thomas et al. (2018) which consists of 8 shapes, where each shape includes $N = 4$ points in 3D space, representing the center of each Tetris block. To generate various shapes, we augment the 8 shapes by adding Gaussian noise with $\sigma = 0.01$ standard deviation on each node's position. Different orientations are obtained by rotating the point cloud with a random rotation matrix $R \in \mathrm{SO}(3)$ and further translating all node positions with the same random translation vector $t \in \mathbb{R}^3 \simeq T_3$. For additional details on the network architecture and training we refer to Appendix D. In Figure 5 we visualize the input points and output points before and after applying the predicted rotation. The model successfully reconstructs the input points with high fidelity (mean squared error of $\sim 4 \times 10^{-5}$) for all shapes and arbitrary translations and rotations. Figure 5b shows the two-dimensional embedding of the trained SE(3)- and S_N-invariant autoencoder. Augmenting the points with random noise results into slight variations in the embedding, while samples of the same Tetris shape class still cluster together. The embedding is invariant with respect to rotations, translation and permutations of the points. Notably, the SE(3)-invariant representations can distinguish the two chiral shapes (compare green and violet coloured shapes in the bottom right of Figure 5b). These two shapes are mirrored versions of themselves and should be distinguished in an SE(3) equivariant model. Models that achieve SE(3) invariant representations by restricting themselves to composition of symmetric functions only, such as working solely on distances (e.g. SchNet Schütt et al. (2018)) or angles (e.g. ANI-1 Smith et al. (2017)) between points fail to distinguish these two shapes Thomas et al. (2018).

Molecular Conformations. We showcase our learning framework on real-world data by autoencoding the atom types and geometries of small molecules from the `QM9` database Ramakrishnan et al. (2014). We achieved a reconstruction RMSE of 0.15 ± 0.07 Å for atom coordinates and perfect atom type accuracy on 5000 unseen test conformations (see Figure 5c for two examples and Appendix E for

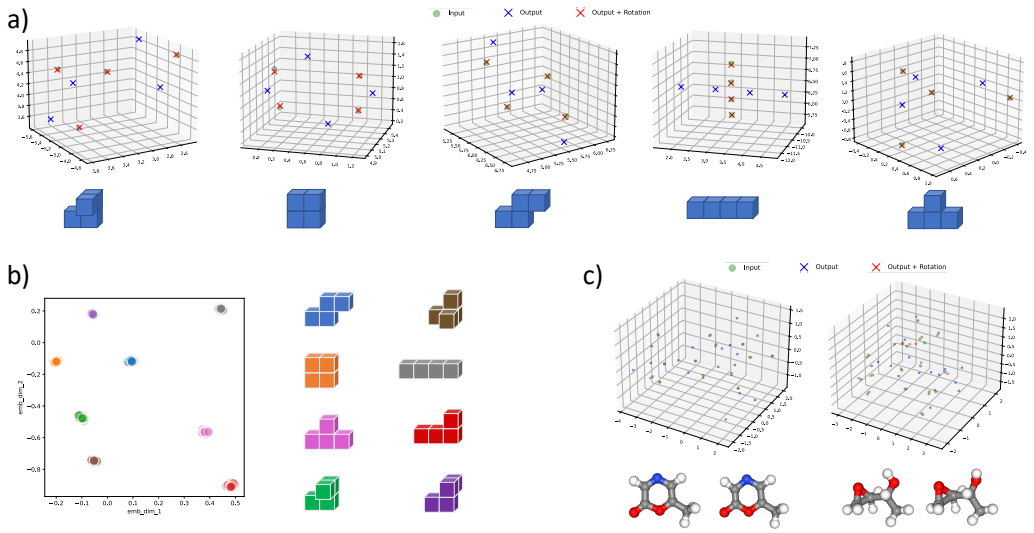

Figure 5: a) Five different Tetris shapes represented by points at the center of the four blocks respectively. Input points, output points and rotated (predicted group action) output points as reconstructed by our proposed SE(3)- and S(N)-invariant autoencoder are visualized. b) Two-dimensional latent space for all Tetris shapes augmented with Gaussian noise ($\sigma = 0.01$). Colors of points match colors of shapes on the right. c) Two molecular conformations and their reconstructions represented as point cloud and ball-and-stick model (left true, right predicted).
We only showcase the effect of aligning by applying the predicted rotation matrix, while the models are still trained in predicting the permutation matrix and translation vector to perform the alignment.

more reconstruction predictions). Given a point cloud of N nodes, notice that the G-inv. embedding z has to store information about the Cartesian coordinates $P \in \mathbb{R}^{3N}$ as well as the 5 distinct atom types $A \in \{0,1\}^{5N}$ represented as one-hot encodings. Given the largest molecule in the QM9 database with $N_{\text{max}} = 29$ atoms, the degrees of freedom of the data space[2] X can assumed to be $3 \cdot 29 \cdot 5 \cdot 29 = 12615$, making the input data high-dimensional, while we compress the molecular conformations into $z \in Z \subset \mathbb{R}^{256}$ dimensions.

6 Conclusion

In this work we proposed a novel unsupervised learning strategy to extract representations from data that are separated in a group invariant and equivariant part for any group G. We defined the sufficient conditions for the different parts of our proposed framework, namely the encoder, decoder and group function without further constraining the choice of a (G-) specific network architecture. In fact, we demonstrate the validity and flexibility of our proposed framework for diverse data types, groups and network architectures.

To the best of our knowledge, we propose the first general framework for unsupervised learning of separated invariant-equivariant representations valid for any group. Our learning strategy can be applied to any AE framework, including variational AEs. It would be compelling to extend our approach to a fully probabilistic approach, where the group action function samples from a probability distribution. Such formalism would be relevant in scenarios where some elements of a group orbit occur with different frequencies, enabling this to be reflected in the generation process. For instance, predicting protein-ligand binding sites depends on the molecule's orientation with respect to the protein pocket or cavity. Thus, in a generative approach, it would be highly compelling to generate a group action reflecting a candidate molecule's orientation in addition to a candidate ligand. We plan to return to these generalization and apply our learning strategy to non-trivial real-world applications in future work.

[2]Notice that the data space X can be described as the product space between $\mathbb{R}^{3N}$ and $\mathbb{N}^{5N}$.

References

Rotated MNIST. https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits. [Online; accessed 05-January-2021].

Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

Averkiou, M., Kim, V. G., and Mitra, N. J. Autocorrelation descriptor for efficient co-alignment of 3d shape collections. *Computer Graphics Forum*, 35, 2016.

Axelrod, S. and Gomez-Bombarelli, R. GEOM, 2021. URL https://doi.org/10.7910/DVN/JNGTDF.

Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A., Pluim, J. P., and Duits, R. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pp. 440–448. Springer, 2018.

Boyda, D., Kanwar, G., Racanière, S., Rezende, D. J., Albergo, M. S., Cranmer, K., Hackett, D. C., and Shanahan, P. E. Sampling using su (n) gauge equivariant flows. *Physical Review D*, 103(7): 074504, 2021.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Bunke, H. and Jiang, X. *Graph Matching and Similarity*, pp. 281–304. Springer US, Boston, MA, 2000. ISBN 978-1-4615-4401-2. doi: 10.1007/978-1-4615-4401-2_10. URL https://doi.org/10.1007/978-1-4615-4401-2_10.

Chaouch, M. and Verroust-Blondet, A. A novel method for alignment of 3d models. *2008 IEEE International Conference on Shape Modeling and Applications*, pp. 187–195, 2008.

Chaouch, M. and Verroust-Blondet, A. Alignment of 3d models. *Graph. Model.*, 71:63–76, 2009.

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

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

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

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, pp. 1889–1898. PMLR, 2016.

Ding, J., Ma, Z., Wu, Y., and Xu, J. Efficient random graph matching via degree profiles. *Probability Theory and Related Fields*, 179:29–115, 2020.

Esteves, C., Makadia, A., and Daniilidis, K. Spin-weighted spherical cnns. *ArXiv*, abs/2006.10731, 2020.

Feige, I. Invariant-equivariant representation learning for multi-class data, 2019.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3165–3176. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/finzi20a.html.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grover, A., Wang, E., Zweig, A., and Ermon, S. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

Guo, X., Zhu, E., Liu, X., and Yin, J. Affine equivariant autoencoder. In *IJCAI*, pp. 2413–2419, 2019.

Higgins, I., Amos, D., Pfau, D., Racanière, S., Matthey, L., Rezende, D. J., and Lerchner, A. Towards a definition of disentangled representations. *ArXiv*, abs/1812.02230, 2018.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *International conference on artificial neural networks*, pp. 44–51. Springer, 2011.

Hoogeboom, E., Peters, J. W. T., Cohen, T., and Welling, M. Hexaconv. *ArXiv*, abs/1803.02108, 2018.

Hosoya, H. Group-based learning of disentangled representations with generalizability for novel contents. In *IJCAI*, 2019.

Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pp. 4533–4543. PMLR, 2021.

Keurti, H., Pan, H.-R., Besserve, M., Grewe, B. F., and Scholkopf, B. Homomorphism autoencoder - learning group structured representations from observed transitions. *ArXiv*, abs/2207.12067, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pp. 5361–5370. PMLR, 2020.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pp. 2747–2755. PMLR, 2018.

Koneripalli, K., Lohit, S., Anirudh, R., and Turaga, P. K. Rate-invariant autoencoding of time-series. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3732–3736, 2020.

Kosiorek, A. R., Sabour, S., Teh, Y. W., and Hinton, G. E. Stacked capsule autoencoders. *arXiv preprint arXiv:1906.06818*, 2019.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. Graph matching networks for learning the similarity of graph structured objects. *ArXiv*, abs/1904.12787, 2019.

Lyle, C., van der Wilk, M., Kwiatkowska, M. Z., Gal, Y., and Bloem-Reddy, B. On the benefits of invariance in neural networks. *ArXiv*, abs/2005.00178, 2020.

Mehr, E., Lieutier, A., Bermudez, F. S., Guitteny, V., Thome, N., and Cord, M. Manifold learning in quotient spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9165–9174, 2018a.

Mehr, , Lieutier, A., Bermudez, F. S., Guitteny, V., Thome, N., and Cord, M. Manifold learning in quotient spaces. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9165–9174, 2018b. doi: 10.1109/CVPR.2018.00955.

Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*, 2020.

Prillo, S. and Eisenschlos, J. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pp. 7793–7802. PMLR, 2020.

Puny, O., Atzmon, M., Ben-Hamu, H., Misra, I., Grover, A., Smith, E. J., and Lipman, Y. Frame averaging for invariant and equivariant network design, 2021. URL `https://arxiv.org/abs/2110.03336`.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Romero, D. W. and Cordonnier, J.-B. Group equivariant stand-alone self-attention for vision. *arXiv preprint arXiv:2010.00977*, 2020.

Romero, D. W. and Hoogendoorn, M. Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. *ArXiv*, abs/1911.07849, 2020.

Romero, D. W., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. Attentive group equivariant convolutional networks. *ArXiv*, abs/2002.03830, 2020.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks, 2021.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24): 241722, 2018.

Schütt, K. T., Unke, O. T., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra, 2021.

Shu, Z., Sahasrabudhe, M., Güler, R. A., Samaras, D., Paragios, N., and Kokkinos, I. Deforming autoencoders: Unsupervised disentangling of shape and appearance. *ArXiv*, abs/1806.06503, 2018.

Smidt, T. Euclidean symmetry and equivariance in machine learning. *ChemRxiv*, 2020. doi: 10.26434/chemrxiv.12935198.v1.

Smith, J. S., Isayev, O., and Roitberg, A. E. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, Y., Asafi, S., van Kaick, O. M., Zhang, H., Cohen-Or, D., and Chen, B. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31:1 – 10, 2012.

Weiler, M. and Cesa, G. General e(2)-equivariant steerable cnns. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL `https://proceedings.neurips.cc/paper/2019/file/45d6637b718d0f24a237069fe41b0db4-Paper.pdf`.

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

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018a.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018b.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018c.

Winter, R., Noé, F., and Clevert, D.-A. Permutation-invariant variational autoencoder for graph-level representation learning. *arXiv preprint arXiv:2104.09856*, 2021.

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

Xu, J., Kim, H., Rainforth, T., and Teh, Y. W. Group equivariant subsampling, 2021.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

Checklist

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]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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]
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]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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)? [Yes]
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]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

A Proofs

Proposition A.1. *Any suitable group function $\psi : X \to G$ is G-equivariant at a point $x \in X$ up the stabilizer G_x, i.e., $\psi(\rho_X(g)x) \subseteq g \cdot \psi(x)G_x$.*

Proof: As the relation (see Property 2.2)

$$\rho_X(\psi(x))\delta(\eta(x)) = x \tag{7}$$

must hold for any $x \in X$, it must hold for any point $x' = \rho_X(g)x$ in the orbit of x, which then reads

$$\begin{aligned} x' &= \rho_X(\psi(x'))\delta(\eta(x')) \\ &= \rho_X(\psi(\rho_X(g)x))\delta(\eta(x)) \ , \end{aligned} \tag{8}$$

where we used the invariance of η. On the other hand, applying $\rho_X(g)$ to both sides of (7) we have

$$\begin{aligned} \rho_X(g)x &= \rho_X(g)\rho_X(\psi(x))\delta(\eta(x)) \\ &= \rho_X(g\psi(x))\delta(\eta(x)) \ , \end{aligned} \tag{9}$$

since $\eta(\rho_X(g')x) = \eta(x)$ and $\rho_X(g_1)\rho_X(g_2) = \rho_X(g_1g_2)$. Combining (8) and (9) it follows that

$$\rho_X(\psi(x)^{-1} \cdot g^{-1} \cdot \psi(\rho_X(g)x))\delta(\eta(x)) = \delta(\eta(x)) \ , \tag{10}$$

that is, $\psi(x)^{-1} \cdot g^{-1} \cdot \psi(\rho_X(g)x)) \in G_{\delta(\eta(x))}$. Now, since x and $\delta(\eta(x))$ by assumption belong to the same orbit of G, it follows that they have isomorphic stabilizers, $G_{\delta(\eta(x))} \simeq G_x$. Thus, we have shown that $\psi(\rho_X(g)x)) = g \cdot \psi(x) \cdot g'$, where $g' \in G_x$, which proves our claim. □

Proposition A.2. *The image of any suitable group function $\psi : X \to G$ is surjective into $\frac{G}{G_X}$, where G_X is the stabilizers of all the points of X.*

Proof: Let $x \in X$ be such that $x = \delta(\eta(x))$, that is, $\psi(x) = G_x$, the stabilizer of x. Note that each orbit cointains at least one such element. For any element $g \in G$ we have that, using Proposition A.1, $\psi(\rho_X(g)x) = g \cdot \psi(x) \cdot \tilde{g}$, where $\tilde{g} \in G_x$. Since $\psi(x) \cdot \tilde{g} \in G_x$ as well, it then follows that the image of ψ is G up to an action by an element of the stabilizer G_x. Applying the above reasoning to every points $x \in X$, we have that $\mathrm{Im}(\psi) = \cup_{x \in X} \frac{G}{G_x} = \frac{G}{\cap_{x \in X} G_x}$, where $\cap_{x \in X} G_x = G_X = \{g \in G | \rho_X(g)x = x, \ \forall x \in X\}$, proving our claim.□

Lemma A.3. *Any suitable group function ψ is an isomorphism $O_x \simeq G/G_x$ for any $x \in X$, where $O_x \subset X$ is the orbit of x with respect to G in X.*

Proof: Surjectivity follows directly from Proposition A.2. To show injectivity, consider $x, x' \in O_x$ such that $\psi(x') = \psi(x) \cdot \tilde{g}$, where $\tilde{g} \in G_x$. From Proposition 2.3 it follows that $x' = x$, which proves the claim. □

Proposition A.4. *Let $\psi = \xi \circ \mu$ be a suitable group function and let $\mu : X \to Y$ be G-equivariant. Then, $G_x = G_{\mu(x)}$ for all $x \in X$.*

Proof: Let $g \in G_x$, that is, $\rho_X(g)x = x$. Applying μ to both sides of this equation we obtain $\mu(x) = \mu(\rho_X(g)x) = \rho_Y(g)\mu(x)$, where we used the G-equivariance of μ. Hence, $G_x \subseteq G_{\mu(x)}$. To prove the opposite inclusion, let $g \in G_{\mu(x)}$ but $g \notin G_x$, and let $x' = \rho_X(g)x$. Now, $\mu(x') = \rho_Y(g)\mu(x) = \mu(x)$, thus μ, and therefore $\psi = \xi \circ \mu$, maps the distinct element x, x' to the same group element $\psi(x) = \psi(x')$, in contradiction with Proposition 2.3. □

Proposition A.5. *Given y, y_0, the element g such that $y \equiv \rho_Y(g)y_0$ is unique up to the stabilizer G_{y_0}.*

Proof: Suppose that there exist $g_1, g_2 \in G$ such that $\rho_Y(g_1)y_0 = \rho_Y(g_2)y_0$, then $\rho_Y(g_2^{-1}g_1)y_0 = y_0$, which implies $g_2^{-1}g_1 \in G_{y_0}$. □

Proposition A.6. *Let $\psi = \xi \circ \mu$ where μ and ξ are as described above. Then, ψ is a suitable group function.*

Proof: We show that our construction describes an isomorphism $O_x \simeq G/G_x$ for all $x \in X$. Given $x \in X$ and $g \in G$, Propositions A.4 and A.5 imply

$$\xi\left(\mu(\rho_X(g)x)\right) = \xi(\rho_Y(g)\mu(x)) \subseteq g \cdot \xi(\mu(g))G_x \ , \tag{11}$$

that is, ψ possesses the G-equivariant property as required in Proposition 2.3, which in turns imply injectivity, as in Lemma A.3. Surjectivity follows from the same argument as in Proposition A.2, since the proof only relies on the equivariant properties of ψ, which we showed in (11). □

B Model architecture Rotated MNIST

We follow Weiler & Cesa (2019b) and use steerable CNNs to parameterize functions η and μ. In contrast to classical CNNs, CNNs with O(2)-steerable kernels transform feature fields respecting the transformation law under actions of $O(2)$. We can define scalar fields $s : \mathbb{R}^2 \to \mathbb{R}$ and vector fields $v : \mathbb{R}^2 \to \mathbb{R}^2$ that transform under group actions (rotations) the following:

$$s(x) \mapsto s(g^{-1}x) \qquad v(x) \mapsto g \cdot v(g^{-1}x) \qquad \forall g \in O(2)\ . \tag{12}$$

Thus, scalar values are moved from one point on the plane $\mathbb{R}^2$ to another but are not changed, while vectors are moved and changed (rotated) equivalently. Hence, we can utilize steerable CNNs to encode samples in $\mathbb{R}^2$ in $O(2)$-invariant scalar features and $O(2)$-equivariant vector features. We can use the scalar features s as $\mathcal{G}$-invariant representation $z \in Z$ and following Section 3 (Orthogonal group SO(2)) utilizing a single vector features v to construct the rotation matrix R as:

$$R = \begin{bmatrix} \bar{v}_x & -\bar{v}_y \\ \bar{v}_y & \bar{v}_x \end{bmatrix}, \qquad \bar{v} = \frac{v}{\|v\|}. \tag{13}$$

In our experiments we used seven layers of steerable CNNs as implemented by Weiler & Cesa (2019b). We did not use pooling layers, as we found them to break rotation equivariance and only averaged over the two spatial dimensions after the final layer to extract the final invariant embedding and equivariant vector. In each layer we used 32 hidden scalar and 32 hidden vector fields. In the final layer we used 32 scalar fields (32 dimensional invariant embedding) and one vector feature field.

The Decoding function $\delta : Z \to \mathbb{R}^2$ can be parameterized by a regular CNN. In our experiments we used six layers of regular CNNs with 32 hidden channels, interleaved with bilinear upsampling layers starting from the embedding expanded to a $2 \times 2 \times 32$ tensor.

Training was done on one NVIDIA Tesla V100 GPU in approximately 6 hours.

We also implemented and trained the quotient autoencoder (QAE) approach proposed by Mehr et al. (2018a) on the MNIST dataset for the group SO(2), discretized in 36 rotations with the loss

$$\min_{\theta \in \{10i, i=0,\ldots,35\}} \{\text{MSE}(x - \rho_X(g(\theta))y)\}\ , \tag{14}$$

where x is a MNIST sample and y is the reconstructed sample. We evaluated the resulting embeddings on the rotated MNIST test set (in such a way that the evaluation is the same as for our model). In Figure 6 we plot TSNE embeddings for this approach, and we can observe that the embedding space shows a clearer structure, in comparison with the classical model. However, in comparison, our approach results in a better clustering of the different digits classes. That shows that the discretization step, while it helps in structuring the embedding space in "signal clusters", still does not capture the full continuous nature of the group. To further quantitatively compare the three methods (ours, QAE (a) and classical AE), we evaluated the reconstruction loss as well as the (digit class) classification accuracy of a KNN classifier trained on 1000 embeddings of each method. We present in the table below the results for the reconstruction loss and for the classification accuracy of a KNN classifier trained on the AE embeddings. To obtain a fair comparison, we kept the architecture and the training hyperparameters exactly identical for all the strategies. We note that our strategy outperforms both the classical AE as well as the strategy of (a) in both tasks.

We also performed an additional experiment, we trained a fully equivariant AE (that is, the embedding itself is fully equivariant, i.e. multiple 2-dimensional vectors) on MNIST with $G = \text{SO}(2)$, and we perform an invariant pooling afterwards to extract the invariant part. Specifically, we have trained KNN classifiers on (a) the invariant embedding corresponding to the norm of the 2-dimensional vectors forming the bottleneck representation, (b) the angles between the first and all other vectors and on (c) the full invariant embedding we obtained by combining the the norms and angles. We choose the number of vectors in the bottleneck in such a way that the dimensionality of the full invariant representation coincides with the one of our model. We visualized the resulting TSNE embeddings in Figure 6 and show the downstream performance of the KNN classifiers in Table 1. From the results we can see that, in comparison to the approximate invariant (QAE) and our invariant trained model, the invariant projected equivariant representations perform inferior. Although we extract a complete invariant representation (which performs better than a subset of this representation like the norm or angle part), the resulting representation is apparently not as expressive and e.g. useful in a downstream classification task. This aligns well with our hypothesis, that our proposed framework poses a sensible supervisory signal to extract expressive invariant representations that are superior to invariant projections of equivariant features.

Table 1: Comparison of our approach vs classical and quotient autoencoder (QAE) as well as an fully equivariant AE with invariant pooling after training.

Model	Rec. Loss	KNN Acc.
classical	0.0170	0.68
QAE	0.0227	0.82
invariant (ours)	0.0162	0.90
equiv AE (norm)	0.0189	0.56
equiv AE (angle)	0.0189	0.53
equiv AE (complete)	0.0189	0.67

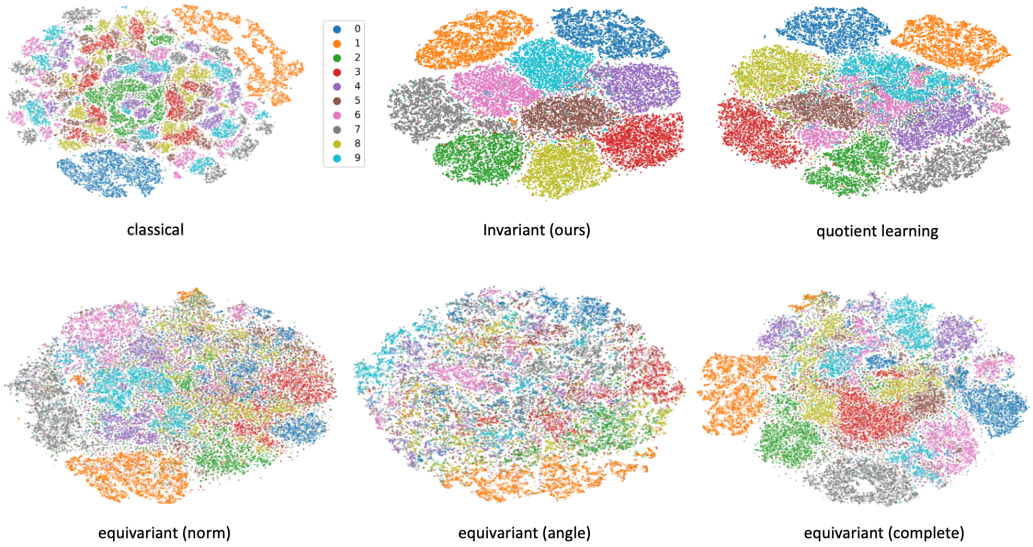

Figure 6: Top row: TSNE embedding of the encoded test dataset for a classical autoencoder, our proposed SO(2) invariant autoencoder, and for the quotient autoencoder of Mehr et al. (2018a). Bottom row: Fully equivariant trained autoencoder with invariant projection after training, either by taking the norm, angles between vectors or the combination (complete).

C Model architecture Set of Digits

We can rewrite the equation $P_\sigma(1, 2, \ldots, n) = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, in vector form by representing set elements by standard $n \times 1$ column vectors $\mathbf{e}_i$ (one-hot encoding) and σ by a permutation matrix P_σ whose (i,j) entry is 1 if $i = \sigma(j)$ and 0 otherwise, then:

$$P_\sigma \mathbf{e}_i = \mathbf{e}_{\sigma(i)} \tag{15}$$

Hence, encoding function η should encode a set of elements in a permutation invariant way and ψ should map a set M to a permutation matrix P_σ:

$$\psi : M \to P_\sigma \tag{16}$$

We follow Zaheer et al. (2017) and parameterize η by a neural network γ that is applied element-wise on the set followed by an invariant aggregation function Σ (e.g. sum or average) and a second neural network β:

$$\eta(X) = \beta(\Sigma_{x \in X} \gamma(x)) \ . \tag{17}$$

In our experiments we parameterized γ and β with regular feed-forward neural networks with three layers respectively, also using ReLU activations and Batchnorm.

The output of function γ is equivariant and can also be used to construct ψ. We follow Winter et al. (2021) and define a function $s : \mathbb{R}^d \to \mathbb{R}$ mapping the output of γ for every set element to a scalar

value. By sorting the resulting scalars, we construct the permutation matrix P_σ with entries p_{ij} that would sort the set of elements with respect to the output of s:

$$p_{ij} = \begin{cases} 1, & \text{if } j = \text{argsort}(s)_i \\ 0, & \text{else} \end{cases} \tag{18}$$

As the argsort operation is not differentiable, we utilizes a continuous relaxation of the argsort operator proposed in (Prillo & Eisenschlos, 2020; Grover et al., 2019):

$$\mathbf{P} \approx \hat{\mathbf{P}} = \text{softmax}(\frac{-d(\text{sort}(s)1^\top, 1s^\top)}{\tau}), \tag{19}$$

where the softmax operator is applied row-wise, $d(x, y)$ is the L_1-norm and $\tau \in \mathbb{R}_+$ a temperature-parameter.
Decoding function δ can be parameterized by a neural network that maps the permutation-invariant set representation back to either the whole set or single set elements. In the letter case, where the same function is used to map the same set representation to the different elements, additional fixed position embeddings can be fed into the function to decode individual elements for each position/index. For the reported results we choose this approach, using one-hot vectors as position embeddings and a 4-layer feed-forward neural network.

Training was done on one NVIDIA Tesla V100 GPU in approximately 1 hours.

D Model architecture Point Cloud - Tetris 3D

We implement a graph neural network (GNN) that transform equivariantly under rotations and translations in 3D space, respecting the invariance and equivariance constraints mentioned in Eq. (5) and (6) for $n = 3$.

Assume we have a point cloud of N particles each located at a certain position $x_i \in \mathbb{R}^3$ in Cartesian space. Now given some arbitrary ordering $\sigma(\cdot)$ for the points, we can store the positional coordinates in the matrix $P = [x_1, ..., x_N] \in \mathbb{R}^{N\times 3}$. Standard Graph Neural Networks (GNNs) perform message passing Gilmer et al. (2017) on a local neighbourhood for each node. Since we deal with a point cloud, common choice is to construct neighbourhoods through a distance cutoff $c > 0$. The edges of our graph are specified by *relative* positions

$$x_{ij} = x_j - x_i \in \mathbb{R}^3 \ ,$$

and the neighbourhood of node i is defined as $\mathcal{N}(i) = \{j : \ d_{ij} := ||x_{ij}|| \leq c\}$.

Now, our data (i.e., the point cloud) lives on a vector space X, where we want to learn an SE(3) invariant and equivariant embedding wrt. arbitrary rotations and translations in 3D space. Let the feature for node i consist of an invariant (type-0) embedding $h_i \in \mathbb{R}^{F_s}$, an equivariant (type-1) embedding $w_i \in \mathbb{R}^{3\times F_v}$ that transforms equivariantly wrt. arbitrary rotation **but** is invariant to translation. Such a property can be easily obtained, when operating with relative positions.
Optionally, we can model another equivariant (type-1) embedding $t_i \in \mathbb{R}^3$ which transforms equivariantly wrt. translation **and** rotation. As our model needs to learn to predict group actions in the SE(3) symmetry, we require to predict an equivariant translation vector ($b \in T_3$), as well as a rotation matrix ($A \in$ SO(3)), where we will dedicate the t vector to the translation and the w vector(s) to the rotation matrix.
As point clouds might not have initial features, we initialize the SE(3)-invariant embeddings as one-hot encoding $h_i = \mathbf{e}_i$ for each node $i = 1, \ldots, N$. The (vector) embedding dedicated for predicting the rotation matrix is initialized as zero-tensor for each particle, i.e., $w_i = \mathbf{0}$ and the translation vector is initialized as the absolute positional coordinate, i.e. to, $t_i = x_i$.

We implement following edge function $\phi_e : \mathbb{R}^{2F_s+1} \mapsto \mathbb{R}^{F_s+2F_v+k}$ with

$$m_{ij} = \phi_e(h_i, h_j, d_{ij}) = W_e[h_i, h_j, d_{ij}] + b_e, \tag{20}$$

and set $k = 1$ if the GNN should model the translation and $k = 0$ else. Notice that the message m_{ij} in Eq. (20) only depends on SE(3) invariant embeddings. Now, (assuming $k = 1$) we further split the message tensor into 4 tensors,

$$m_{ij} = [m_{h,ij}, m_{w_0,ij}, m_{w_1,ij}, m_{t,ij}] \ ,$$

which we require to compute the aggregated messages for the SE(3) invariant and equivariant node embeddings.
We include a row-wise transform $\phi_s : \mathbb{R}^{F_s} \mapsto \mathbb{R}^{F_s}$ for the invariant embeddings using a linear layer:

$$\tilde{h}_i = W_s h_i + b_s \ , \tag{21}$$

The aggregated messages for invariant (type-0) embedding h_i are calculated using:

$$m_{i,h} = \sum_{j \in \mathcal{N}(i)} m_{h,ij} \odot \tilde{h}_i \quad \in \mathbb{R}^{F_s}. \tag{22}$$

where $\odot$ is the (componentwise) scalar-product.
The aggregated equivariant features are computed using the tensor-product $\otimes$ and scalar-product $\odot$ from (invariant) type-0 representations with (equivariant) type-1 representations:

$$m_{i,w} = \sum_{j \in \mathcal{N}(i)} (x_{ij} \otimes m_{w_0,ij} + (w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \quad \in \mathbb{R}^{3 \times F_v} \ , \tag{23}$$

where $\mathbf{1} \in \mathbb{R}^3$ is the vector with 1's as components and $(a \times b)$ denotes the cross product between two vectors $a, b \in \mathbb{R}^3$.
The tensor in Eq. (23) is equivariant to arbitary rotations and invariant to translations. It is easy to prove the translation invariance, as any translation $t^* \in T_3$ acting on points x_i, x_j does not change the relative position $x_{ij} = (x_j + t^*) - (x_i + t^*) = x_j - x_i$.
To prove the rotation equivariance, we first observe that given any rotation matrix $A \in \mathrm{SO}(3)$ acting on the provided data, as a consequence relative positions rotate accordingly, since

$$Ax_j - Ax_i = A(x_j - x_i) = Ax_{ji} \in \mathbb{R}^3.$$

The tensor product $\otimes$ between two vectors $u \in \mathbb{R}^3$ and $v \in \mathbb{R}^{F_s}$, commonly also referred to as *outer product* is defined as

$$u \otimes v = uv^\top \in \mathbb{R}^{3 \times F_s} \ ,$$

and returns a matrix given two vectors. For the case that a group representation of SO(3), i.e. a rotation matrix R, acts on u, it is obvious to see with the associativity property

$$(Au) \otimes v = (Au)v^\top = Auv^\top = A(uv^\top) = A(u \otimes v) \ = Au \otimes v.$$

The cross product $(w_i \times w_j) \in \mathbb{R}^{3 \times F_v}$ used in equation (23) between type-1 features w_i and w_j is applied separately on the last axis. The cross product has the algebraic property of rotation invariance, i.e. given a rotation matrix A acting on two 3-dimensional vectors $a, b \in \mathbb{R}^3$ the following holds:

$$(Aa) \times (Ab) = A(a \times b) \ . \tag{24}$$

Now, notice that the quantities that "transform as a vector" which we call type-1 embeddings are in $S = \{x_{ij}, w_i, t_i\}_{i,j=1}^N$.
Given a rotation matrix A acting on elements of S, we can see that the result in (23)

$$\begin{aligned}
&\sum_{j \in \mathcal{N}(i)} (Ax_{ij} \otimes m_{w_0,ij} + (Aw_i) \times (Aw_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\
&= \sum_{j \in \mathcal{N}(i)} (Ax_{ij} \otimes m_{w_0,ij} + A(w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\
&= A \sum_{j \in \mathcal{N}(i)} (x_{ij} \otimes m_{w_0,ij} + (w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\
&= A m_{i,w}
\end{aligned}$$

is rotationally equivariant.
We update the hidden embedding with a residual connection

$$\begin{aligned}
h_i &\leftarrow h_i + m_{i,h} \ , \\
w_i &\leftarrow w_i + m_{i,w} \ ,
\end{aligned} \tag{25}$$

and use a Gated-Equivariant layer with equivariant non-linearities as proposed in the PaiNN architecture Schütt et al. (2021) to enable an information flow between type-0 and type-1 embeddings.
The type-1 embedding for the translation vector is updated in a residual fashion

$$\begin{aligned} t_i &\leftarrow t_i + \sum_{j \in \mathcal{N}(i)} x_{ij} \otimes m_{t,ij} \\ &= t_i + \sum_{j \in \mathcal{N}(i)} m_{t,ij} \odot x_{ij}, \end{aligned} \tag{26}$$

where we can replace the tensor-product with a scalar-product, as $m_{t,ij} \in \mathbb{R}$. The result in Eq. (26) is translation and rotation equivariant as the first summand t_i is rotation and translation equivariant, while the second summand is only rotation equivariant since we utilize relative positions.

For the SE(3) Tetris experiment, the encoding function $\eta : X \mapsto Z$ is a 5-layer GNN encoder with $F = F_s = F_v = 32$ scalar- and vector channels and implements the translation vector, i.e. $k = 1$.
The encoding network η outputs four quantities: two SE(3) invariant node embedding matrices $\widetilde{H}, M \in \mathbb{R}^{N \times F}$, one SO(3) equivariant order-3 tensor $\widetilde{W} \in \mathbb{R}^{N \times 3 \times F}$ as well as another SE(3) equivariant matrix $T \in \mathbb{R}^{N \times 3}$.
We use two linear layers[3] to obtain the SE(3) invariant embedding matrix $H \in \mathbb{R}^{N \times 2}$ as well as the SO(3) equivariant embedding tensor $W \in \mathbb{R}^{N \times 3 \times 2}$. Notice that the linear layer returning the W tensor can be regarded as the function ψ_{rot} that aims to predict the group action in the SO(3) symmetry, while we use the identity map for the translation vector, i.e. $\psi_{\text{transl}} = T$.
As point clouds can be regarded as sets, we obtain an permutation invariant embedding by averaging over the first dimension of the $\{H, W, T\}$ tensors,

$$h = \frac{1}{N} \sum_{i=1}^{N} H_i \ \in \mathbb{R}^2 \ , \tag{27}$$

$$t = \frac{1}{N} \sum_{i=1}^{N} T_i \ \in \mathbb{R}^3 \ , \tag{28}$$

$$w = \frac{1}{N} \sum_{i=1}^{N} W_i \ \in \mathbb{R}^{3 \times 2} \ , \tag{29}$$

while we use the M matrix to predict the permutation matrix P_σ with the ψ_{perm} function, in similar fashion as described in Eq. (16). To construct the rotation matrix R out of 2 vectors in $\mathbb{R}^3$ as described in Section 3, we utilize the SO(3) equivariant embedding w.
The decoding network $\delta : Z \mapsto X$ is similar to the encoder a 5-layer SE(3)-equivariant GNN but does not model the translation vector, i.e. $k = 0$. The decoder δ maps the SE(3) as well as S(N)-invariant embedding h back to a reconstructed point cloud $\hat{P} \in \mathbb{R}^{N \times 3}$. At the start of decoding, we utilize a linear layer to map the $G-$invariant embedding $h \in \mathbb{R}^2$ to a higher-dimension, i.e.

$$\tilde{h} = W_0 h + b_0 \ \in \mathbb{R}^{F_s}, \tag{30}$$

Next, to "break" the symmetry and provide the nodes with initial type-0 features, we utilize fixed (deterministic) positional encodings as suggested by Winter et al. (2021) for each node $i = 1, \ldots, N$ to be summed with $\tilde{h}$. Notice that this addition enables us to obtain distinct initial type-0 embeddings $\{\hat{h}_i\}_{i=1}^N$.
For the start positions, we implement a trainable parameter matrix P_θ of shape $(N \times 3)$ for the decoder.
Now, given an initial node embedding $\hat{H} \in \mathbb{R}^{N \times F_s}$, we apply the S($N$) group action, by multiplying the predicted permutation matrix P_σ with $\hat{H}$ from the left to obtain the canonical ordering as

$$\hat{H}_\sigma = P_\sigma \hat{H} \ . \tag{31}$$

To retrieve the correct orientation required for the pairwise-reconstruction loss, we multiply the constructed rotation matrix R with the initial start position matrix P_θ

$$\hat{P}_r = P_\theta R^\top . \tag{32}$$

[3]The transformation is always applied on the last (feature) axis.

Target	Fraction	Pretrained	From Scratch
H	0.05	0.7529	0.0970
	0.25	0.9908	0.9093
G	0.05	0.7703	0.4758
	0.25	0.9856	0.9751
U	0.05	0.6083	0.2574
	0.25	0.9962	0.9808
$\langle R^2 \rangle$	0.05	0.7806	0.1468
	0.25	0.9918	0.8546
μ	0.05	0.8698	0.8443
	0.25	0.9718	0.9718
α	0.05	0.9455	0.9237
	0.25	0.9937	0.9764

Table 2: Generalization performance in terms of the coefficient of determination R^2 of models on a held-out test set of 1000 samples. Higher R^2 indicates better performance.

With such construction, we can feed the two tensors to the decoder network δ to obtain the reconstructed point cloud as

$$\hat{P}_{\text{recon}} = \delta(\hat{H}_\sigma, \hat{P}_r) + t \ , \tag{33}$$

where $t \in \mathbb{R}^3$ is the predicted translation vector from the encoder network, added row-wise for each node position.

Architecture QM9. For the QM9 dataset, we use the same model components as described in the Tetris experiment, with the difference of including atom species as $SE(3)-$invariant features and setting $F_s = 256, F_v = 32$ and increasing the dimensionality of the latent space to 256.

Finetuning. We performed additional experiments on the pretrained group-invariant AE on the extended GEOM-QM9 dataset Axelrod & Gomez-Bombarelli (2021) which, as opposed to the standard QM9 dataset ($\approx 130k$ samples), contains multiple conformations of small molecules. We trained the autoencoder on a reduced set of GEOM-QM9 ($\approx 641k$), containing up to 10 conformations per molecule and utilized this pretrained encoder network to regress (invariant) energy targets, such as internal energy U or enthalpy H on the original QM9 dataset.

We observed that the pretrained encoder network learns faster and achieves better generalization performance than the architectural identical network trained from scratch. In Figure 7 we illustrates the learning curves for the two networks on different fraction on 5% and 25% labelled samples from original QM9 dataset to analzye the benefit of finetuning a pre-trained encoder network on a low-data regime, when regressing on the enthalpy H. On a held-out test dataset of 1000 samples, the pretrained encoder network achieves superior generalization performance in terms of R^2 with 0.7529 vs. 0.0970 in the 5% data regime, and 0.9908 vs. 0.9093 in the 25% data regime compared to the encoder that was trained from scratch. In Table2 we show additional comparisons of the pretrained network against a network that was trained from scratch for 50 epochs on the restricted dataset. As shown in Table 2, the pretrained encoder achieves improved generalization performance on the test dataset compared to its architectural identical model that was trained from scratch. We believe that training the group-invariant autoencoder on a larger diverse dataset of (high-quality) molecular conformations facilitates new opportunities in robust finetuning on different data-scarse datasets for molecular property prediction.

E Molecular Conformations: Further Examples

We show additional reconstructions of 12 randomly selected small molecules from the QM9 test dataset. Noticeably, our trained autoencoder is able to reconstruct molecular conformations with complex geometries as depicted in the third column (from the left). We notice that the AE is not able to perfectly reconstruct the conformation shown in the 4th column of the 2nd row. Although this molecule does not exhibit a complicated geometrical structure, its atomistic composition (of only containing nitrogen and carbon as heavy atoms) could be the reason why the encoding of the

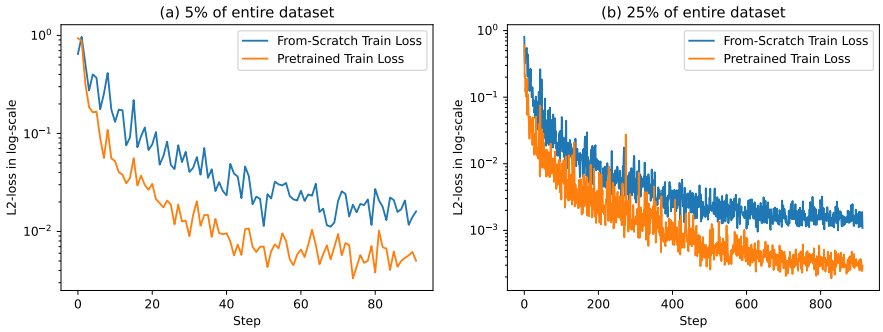

Figure 7: Learning curves of models trained on limited enthalpy H targets.

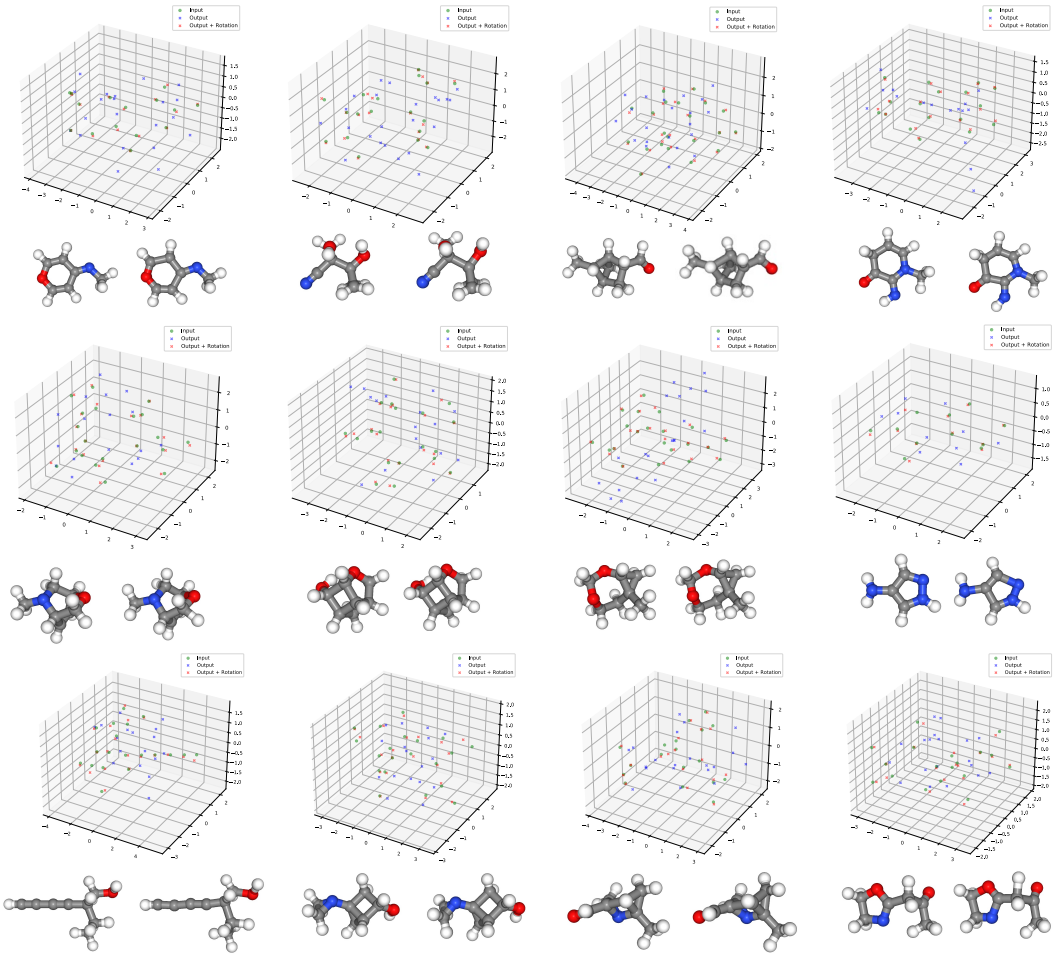

Figure 8: 12 molecular conformations and their reconstructions represented as point cloud and ball-and-stick model (left true, right predicted).

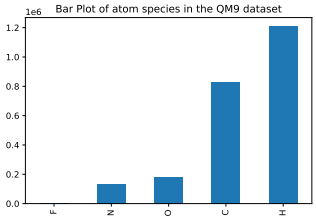

Figure 9: Atomistic species count on the QM9 dataset.

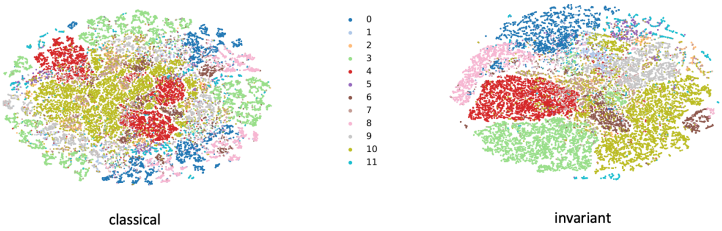

Figure 10: TSNE embedding of the encoded dataset for a classical autoencoder and our proposed SO(3) invariant autoencoder.

conformation is pointing into a non-densely populated region in the latent space, as nitrogen does not have a large count in the total QM9 database, see Figure 9.

Training was done on one NVIDIA Tesla V100 GPU in approximately 1 day.

F ShapeNet

We also run experiments on the ShapeNet dataset. As the dataset comes in an aligned form (e.g. cars are always aligned in the same orientation), we additionally applied random 90 degree rotations to remove this bias. We utilized 3D Steerable CNNs proposed by Weiler et al. (2018b) as equivariant encoder for the 3d voxel input space. We utilized the scalar outputs as rotation-invariant embedding (z) and predict (analogously to our experiments on 3d point clouds) 2 rotation-equivariant vectors to construct a rotation matrix (g). Similar to our MNIST experiment, we compared the resulting embedding space to the embeddings produced by a non-invariant autoencoder model. In Figure 10 we visualize a TSNE projection of the embeddings of both models. We can see a well structured embedding space for our model with distinct clusters for the different shape classes. On the other hand, the embeddings produced by the non-invariant autoencoder is less structured and one can make out different clusters for the same shape label but in different orientations. Moreover, we compared the downstream performance and generalizability of a KNN classifier on shape classification, trained on 1000 embeddings and tested on the rest. The classifier based on our rotation-invariant embeddings achieved an accuracy of 0.81 while the classifier based on the non-invariant embeddings achieved an accuracy of only 0.63.

G Additional Related Work Section

The field of unsupervised invariant representation learning can be roughly divided into two categories. The first consists in learning an approximate group action in order to match the input and the reconstructed data. For instance, Mehr et al. (2018b) propose (like us) to encode the input in quotient space, and train the model with a loss that is defined by taking the infimum over the group G. While this is feasible for (small) finite groups, for continuous groups they either have to approximately discretize them or perform a separate optimization of h at every back propagation step to find the best match. Other work Shu et al. (2018); Koneripalli et al. (2020) proposes to disentangle the

embedding in a shape-like and a deformation-like component. While this is in spirit with our work, their transformations are local (we focus on global transformations) and are approximative, that is, the components are not explicitly invariant and equivariant with respect to the transformation, respectively.

In the case of 2D/3D data, co-alignment of shapes can be used to match the input and the reconstructed shapes. Some approaches are unfeasible Wang et al. (2012) as they are not compatible with a purely unsupervised approach, while other Averkiou et al. (2016); Chaouch & Verroust-Blondet (2008, 2009) leverage symmetry properties of the data and PCA decomposition, exhibiting however limitation regarding scalability. For graphs, the problem of graph matching Bunke & Jiang (2000) has been tackled in several works and with different approaches, for instance algorithmically, e.g., Ding et al. (2020), or by means of a GNN Li et al. (2019).

On the topic of representation theory-based embedding disentanglement, the works Hosoya (2019); Keurti et al. (2022), based on the definition of Higgins et al. (2018) of a disentangled representations, design unsupervised generative VAEs approaches for learning representation corresponding to orthogonal symmetry actions on the data space. In our work, on the other hand, we disentangled different representations, which all can act on our data space.