HOLOGRAPHIC-(V)AE: AN END-TO-END SO(3)-EQUIVARIANT (VARIATIONAL) AUTOENCODER IN FOURIER SPACE

Gian Marco Visani
University of Washington
gvisan01@cs.washington.edu

Michael Neal Pun
University of Washington
mpun@uw.edu

Arman Angaji
University of Cologne
aangaji1@uni-koeln.de

Armita Nourmohammad
University of Washington
Fred Hutchinson Cancer Research Center
armita@uw.edu

ABSTRACT

Group-equivariant neural networks have emerged as a data-efficient approach to solve classification and regression tasks, while respecting the relevant symmetries of the data. However, little work has been done to extend this paradigm to the unsupervised and generative domains. Here, we present *Holographic*-(V)AE (H-(V)AE), a fully end-to-end SO(3)-equivariant (variational) autoencoder in Fourier space, suitable for unsupervised learning and generation of data distributed around a specified origin. H-(V)AE is trained to reconstruct the spherical Fourier encoding of data, learning in the process a latent space with a maximally informative invariant embedding alongside an equivariant frame describing the orientation of the data. We show the potential utility of H-(V)AE on structural biology tasks. Specifically, we train H-(V)AE on protein structure microenvironments, and show that its latent space can be used to extract compact embeddings of local structural features which, paired with a Random Forest Regressor, enable state-of-the-art predictions of protein-ligand binding affinity.

1 INTRODUCTION

In supervised learning, the success of state-of-the-art algorithms is often attributed to respecting known inductive biases of the function they are trying to approximate. One such bias is the invariance of the function to certain transformations of the input. For example, image classification is translationally invariant. To achieve such invariance, conventional techniques use data augmentation to train an algorithm on many transformed forms of the data. However, this solution is only approximate and increases training time significantly, up to prohibitive scales for high-dimensional and continuous transformations ($\sim$500 augmentations are required to learn 3D rotation-invariant patterns (Geiger & Smidt, 2022)). Alternatively, one could use invariant features of the data (e.g. pairwise distance between different features) as input to train any machine learning algorithm (Capecchi et al., 2020). However, the choice of these invariants is arbitrary and the resulting network could lack in expressiveness.

Recent advances have developed neural network architectures that are equivariant under actions of different symmetry groups. These networks can systematically treat and interpret various transformation in data, and learn models that are agnostic to these transformations. For example, models equivariant to euclidean transformations have recently advanced the state-of-the-art on tasks over 3D point cloud data (Liao & Smidt, 2022; Musaelian et al., 2022; Brandstetter et al., 2022). These models are more flexible and expressive compared to their purely invariant counterparts (Geiger & Smidt, 2022), and exhibit high data efficiency.

Extending such group invariant and equivariant paradigms to unsupervised learning could map out compact representations of data that are agnostic to a specified symmetry transformation (e.g. the global orientation of an object). In recent work Winter et al. (2022) proposed a general mathematical framework for autoencoders that can be applied to data with arbitrary symmetry structures by learning an invariant latent space and an equivariant factor, related to the elements of the underlying symmetry group.

Here, we focus on unsupervised learning that is equivariant to rotations around a specified origin in 3D, denoted by the group SO(3). We encode the data in spherical Fourier space and construct holograms of the data that are conveniently structured for equivariant operations. These data holograms are inputs to our end-to-end SO(3)-equivariant (variational) autoencoder in spherical Fourier space, with a fully equivariant encoder-decoder architecture trained to reconstruct the Fourier coefficients of the input; we term this approach *Holographic*-(V)AE (H-(V)AE). Our network learns an SO(3)-equivariant latent space composed of a maximally informative set of invariants and an equivariant frame describing the orientation of the data.

We extensively test the performance of H-(V)AE and demonstrate high accuracy in unsupervised classification and clustering tasks for spherical images. Furthermore, we leverage H-(V)AE trained on a large corpus of protein structure micro-environments to construct SE(3)-invariant representations of protein-ligand binding pockets. When combined with a Random Forest Regressor, we achieve state-of-the-art prediction on the Ligand Binding Affinity (LBA) task.

2 Background

2.1 Spherical harmonics and irreps of SO(3)

We are interested in modeling 3D data (i.e., functions in $\mathbb{R}^3$), for which the global orientation of the data should not impact the inferred model (Einstein, 1916). We consider functions distributed around a specified origin, which we express by the resulting spherical coordinates (r, θ, ϕ) around the origin. In this case, the set of rotations about the origin define the 3D rotation group SO(3), and we will consider models that are rotationally equivariant under SO(3).

It is convenient to project data to spherical Fourier space to define equivariant transformations for rotations.

To map a radially distributed function $\rho(r, \theta, \phi)$ to a spherical Fourier space, we use the Zernike Fourier Transform (ZFT),

$$\hat{Z}^n_{\ell m} = \int \rho(r, \theta, \phi)\, Y_{\ell m}(\theta, \phi) R^n_\ell(r)\, \mathrm{d}V \tag{1}$$

where $Y_{\ell m}(\theta, \phi)$ is the spherical harmonics of degree ℓ and order m, where ℓ is a non-negative integer ($\ell \geq 0$) and m is an integer within the interval $-\ell \leq m \leq \ell$. $R^n_\ell(r)$ is the radial Zernicke polynomial in 3D (Eq. A.7) with radial frequency $n \geq 0$ and degree ℓ. $R^n_\ell(r)$ is non-zero only for even values of $n - \ell \geq 0$. Zernike polynomials - defined as the product $Y_{\ell m}(\theta, \phi) R^n_\ell(r)$ - form a complete orthonormal basis in 3D, and therefore can be used to expand and retrieve 3D shapes, if large enough ℓ and n values are used; approximations that restrict the series to finite n and ℓ are often sufficient for shape retrieval, and hence, desirable algorithmically. Thus, in practice, we cap the resolution of the ZFT to a maximum degree L and a maximum radial frequency N.

The operators that describe how spherical harmonics transform under rotations are called the Wigner D-matrices. Notably, Wigner-D matrices are the irreducible representations (irreps) of SO(3). Spherical harmonics form a basis for the irreps of SO(3), and thus, the SO(3) group acts on spherical Fourier space via a direct sum of irreps. Specifically, the ZFT encodes a data point into a *tensor* composed of a direct sum of *features*, each associated with a degree ℓ indicating the irrep that it transforms with under the action of SO(3). We refer to these tensors as SO(3)-*steerable tensors* and to the vector spaces they occupy as SO(3)-*steerable vector spaces*, or simply *steerable* for short since we only deal with the SO(3) group in this work. We note that a tensor may contain multiple features of the same degree ℓ, which we generically refer to as distinct *channels* c. Throughout the paper, we refer to generic steerable tensors as $\boldsymbol{h}$ and index them by ℓ, m and c. We adopt the "hat" notation for individual entries to remind ourselves of the analogy with Fourier coefficients. See Figure 1A for a graphical illustration of a tensor.

2.2 Mapping between SO(3)-steerable vector spaces

Constructing equivariant operations equates to constructing maps between steerable vector spaces. There are precise rules constraining the kinds of operations that guarantee a valid SO(3)-steerable output, the most important one being the Clebsch-Gordan (CG) tensor product $\otimes_{cg}$. The CG tensor product combines two features of degrees ℓ_1 and ℓ_2 to produce another feature of degree $|\ell_2 - \ell_1| \leq \ell_3 \leq |\ell_1 + \ell_2|$. Let $\boldsymbol{h}_\ell \in \mathbb{R}^{2\ell+1}$ be a generic degree ℓ feature, with individual components $\hat{h}_{\ell m}$ for $-\ell \leq m \leq \ell$. Then, the CG tensor product is given by:

$$\hat{h}_{\ell_3 m_3} = (\boldsymbol{h}_{\ell_1} \otimes_{cg} \boldsymbol{h}_{\ell_2})_{\ell_3 m_3} = \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C_{(\ell_1 m_1)(\ell_2 m_2)}^{(\ell_3 m_3)} \hat{h}_{\ell_1 m_1} \hat{h}_{\ell_2 m_2} \tag{2}$$

where $C_{(\ell_1 m_1)(\ell_2 m_2)}^{(\ell_3 m_3)}$ are the Clebsch-Gordan coefficients (Tung, 1985).

3 Holographic-(V)AE

3.1 SO(3) equivariant layers

Linearity (Lin). We construct linear layers acting on steerable tensors by learning degree-specific linear operations. Specifically, we learn weight matrices specific to each degree ℓ, and use them to map across degree-ℓ feature spaces by learning linear combinations of degree-ℓ features in the input tensor (Section A.2.1).

Efficient Tensor Product nonlinearity (ETP). We utilize the CG tensor product to inject nonlinearity in the network in an equivariant way, as was originally prescribed by Kondor et al. (2018) for SO(3)-equivariant convolutional neural networks (CNNs). This type of nonlinearity enables information flow between features of different degrees, which is necessary for constructing expressive models, and injects square nonlinearity. To significantly reduce the computational and memory costs of the tensor products, we leverage some of the modifications proposed by Cobb et al. (2021). Specifically, we compute tensor products channel-wise, i.e., only between features belonging to the same channel, and we limit the connections between features of different degrees to what Cobb et al. (2021) calls the MST subset. Notably, the channel-wise computation constrains the input tensors to have the same number of channels for each feature. We found these modifications to be necessary to efficiently work with data encoded in large number of channels C and with large maximum degree L. See Section A.2.2 for details, and Table A.9 for an ablation study showing the improvement in parameter efficiency provided by the ETP.

Batch Norm (BN). We normalize intermediate tensor representations degree-wise and channel-wise by the batch-averaged norms of the features, as initially proposed by Kondor et al. (2018), and do it before the ETP; see Figure 1B and Section A.2.3 for details. We found the use of this layer to speed up model convergence (Fig A.3).

Signal Norm (SN). It is necessary to normalize activations computed by the CG tensor product to avoid their explosion. We found using batch norm alone often caused activations to explode in the decoder during evaluation. Thus, we introduce Signal Norm, whereby we divide each steerable tensor by its *total* norm, defined as the sum of the norms of each of the tensor's features, and apply a degree-specific affine transformation (w_ℓ) for added flexibility. Formally, the total norm for an individual tensor h is computed as:

$$N_{tot} = \sum_{\ell} \frac{\sum_c \sum_{m=-\ell}^{\ell} (\hat{h}_{\ell m}^c)^2}{2\ell + 1} \tag{3}$$

and the features are updated as $\overline{\hat{h}_{\ell m}^c} = \hat{h}_{\ell m}^c w_\ell / \sqrt{N_{tot}}$. We note that each pre-affine normalized tensor has a total norm of 1, thus constraining the values of the individual features. Signal Norm can be seen as a form of Layer Norm that respects SO(3) equivariance (Ba et al., 2016).

Clebsch-Gordan block (CG bl.) We construct equivariant blocks, which we term Clebsch-Gordan blocks, by stacking together the equivariant layers introduced above, as shown in Figure 1B. Each block can take a steerable tensor of arbitrary size composed of multiple features of arbitrary degrees, and can output a steerable tensor of arbitrary size with multiple features of arbitrary degrees, by following the sparsity of the CG tensor product ($|\ell_2 - \ell_1| \leq \ell_3 \leq |\ell_1 + \ell_2|$). Crucially, if the maximum feature degree in the input tensor is ℓ_{max}, then the maximum feature degree that can be generated is $2\ell_{\text{max}}$, achieved by combining two features of degree ℓ_{max}. We employ additive skip connections, zero-padded when appropriate, to favor better gradient flow.

3.2 Model architecture

H-(V)AE has a fully rotationally equivariant architecture, and learns a *disentangled* latent space consisting of a maximally informative invariant ($\ell = 0$) component $\mathbf{z}$ of arbitrary size, as well as three orthonormal vectors ($\ell = 1$), which represent the global 3D orientation of the object and reflect the *coordinate frame* of the input tensor. Crucially, the disentangled nature of the latent space is respected at all stages of training, and is guaranteed by the model's rotational equivariance. The architecture is shown in Figure 1C. The encoder takes as input a steerable tensor with maximum degree $\ell_{\text{max}} = L$ and, via a stack of Clebsch-Gordan blocks, iteratively and equivariantly transfers information from higher degrees to lower ones, down to the final encoder layer with $\ell_{\text{max}} = 1$. The frame is constructed by learning two vectors and using Gram-Schmidt to find the corresponding orthonormal basis (Schmidt, 1907). The third orthonomal basis vector is then calculated as the cross product of the first two. The decoder learns to reconstruct the input from $\mathbf{z}$ and the frame, iteratively increasing the maximum degree ℓ_{max} of the intermediate representations by leveraging the CG Tensor Product within the Clebsch-Gordan blocks.

To understand the properties of H-(V)AE, we ask ourselves what the output of the decoder looks like if the frame is held constant. We experimentally find that the reconstructed elements tend to be aligned with each other and hypothesize that the model is implicitly learning to maximize the overlap between training elements, providing empirical evidence in the Appendix (Fig. A.4). We call this frame the **canonical** frame following the analogy with the canonical elements in Winter et al. (2022). We note that it is possible to rotate original elements to the canonical frame thanks to the equivalence between the frame we learn and the rotation matrix within our implementation.

Within the decoder, the maximum degree $\ell_{\text{max},b}$ that can be outputted by each block b is constrained by the sparsity of the CG tensor product. Specifically, $\ell_{\text{max},b} \leq 2^b$ where b ranges from 1 (first block) to B (last block). Since we need to reconstruct features up to degree L in the decoder, we arrive at a lower bound for the number of blocks in the decoder set by $\ell_{\text{max},B} \geq L$, or $B \geq \log_2 L$. In our experiments, we set $\ell_{\text{max},b} = \min\{2^b, L\}$ and do not let $\ell_{\text{max},b}$ exceed the input's maximum degree L. Relaxing this condition might increase the expressive power of the network but at a significant increase in runtime and memory. We leave the analysis of this trade-off to future work. For the encoder and the decoder to have similar expressive power, we construct them to be symmetric with respect to the latent space (Fig. 1C). Optionally, we apply a linearity at the beginning of the encoder and at the end of the decoder; this is required for input data that does not have the same number of channels per degree since the ETP operates channel-wise. We empirically verify the equivariance of our model up to numerical errors in Table A.10.

For H-VAE, we parameterize the *invariant* part of the latent space by an isotropic Gaussian, i.e., we learn two sets of size z invariants, corresponding to means and standard deviations.

3.3 Training objective

We train H-(V)AE to minimize the reconstruction loss $\mathcal{L}_{rec}$ between the original and the reconstructed tensors, and, for H-VAE only, to minimize the KL-divergence of the posterior invariant latent space distribution $q(\boldsymbol{z}|\boldsymbol{x})$ from the selected prior $p(\boldsymbol{z})$ (Kingma & Welling, 2013):

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = \alpha \mathcal{L}_{\text{rec}}(\boldsymbol{x}, \boldsymbol{x}') + \beta D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{4}$$

We use mean square error (MSE) for $\mathcal{L}_{\text{rec}}$, which as we show in Sec. A.2.5, respects the necessary property of SO(3) pairwise invariance, ensuring that the model remains rotationally equivariant.

We cast an isotropic normal prior to the invariant latent space: $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Hyperparameters α and β control the trade-off between reconstruction accuracy and latent space

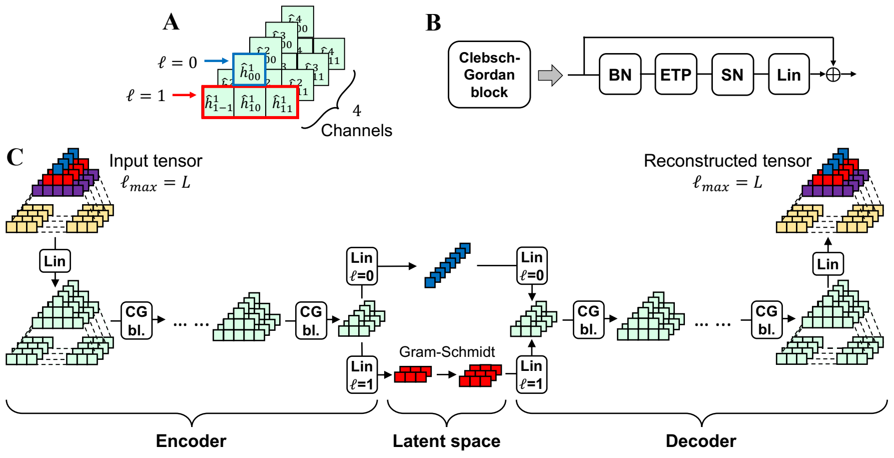

Figure 1: **Schematic of the Network architecture.** **A:** Schematic of a steerable tensor with $\ell_{\text{max}} = 1$ and 4 channels per feature degree. We choose a pyramidal representation that naturally follows the expansion in size of features of higher degree. **B:** Schematic of a Clebsch-Gordan Block (CG bl.), with batch norm (BN), efficient tensor product (ETP), and signal norm (SN), and Linear (Lin) operations. **C:** Schematic of the H-AE architecture. We color-code features of different degrees in the input and in the latent space for clarity. The H-VAE schematic differs only in the latent space, where two sets of invariants are learned (means and standard deviations of an isotropic Gaussian distribution).

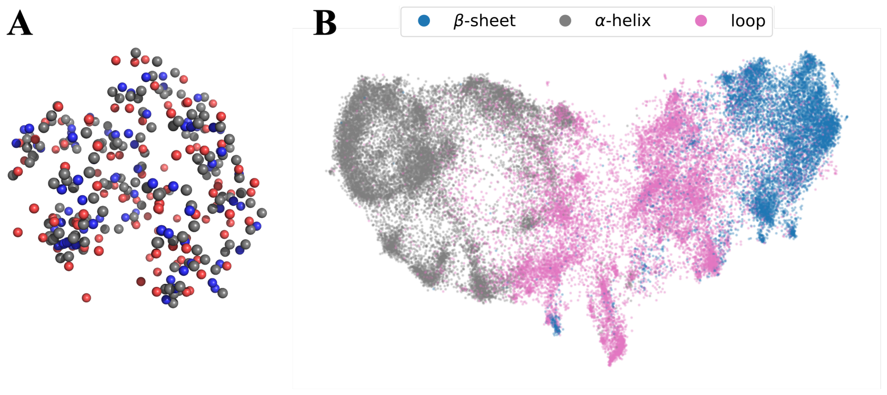

Figure 2: **A:** Example protein neighorhood (point cloud of atoms) used to train our models. **B:** 2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained with $L = 6$. Each point represents a neighborhood.

regularization Higgins et al. (2022). We find it practical to scale the reconstruction loss by a dataset-specific scalar α since the MSE loss varies in average magnitude across datasets. When training H-VAE, we find it beneficial to keep $\beta = 0$ for a few epochs (E_{rec}) so that the model can learn to perform meaningful reconstructions, and then linearly increasing it to the desired value for E_{warmup} epochs to add structure to the latent space, an approach first used by Bowman et al. (2016).

4 Experiments

We test our method on several diverse datasets. We exclude from the main text experiments made on spherical images, and show them in the Appendix (Sec. A.8).

4.1 Protein Neighborhoods

We test H-(V)AE on a challenging point cloud dataset comprised of spherical atomic environments surrounding a residue within a protein structure, which we term protein *neighborhoods*. Specifically, we define neighborhoods as point clouds of atoms within 10Å of a residue's Cα. We extract protein neighborhoods from all proteins in ProteinNet (AlQuraishi, 2019), and then construct each neighborhood's Fourier representation by computing the ZFT (Eq. 1) over atom-type-specific clouds (C, N, O and S) and concatenating atom-type-specific features of the same degree. We use $N = 20$ and vary the maximum spherical degree L. We train several H-AE models with varying architectures and latent space sizes z (Sec. A.6.5). H-AE shows strong reconstruction ability, but its accuracy worsens with smaller latent space sizes and higher maximum spherical degree L (Table A.8). Notably, the learned latent space is structured according to key local structural features (Fig. 2).

4.2 Protein-Ligand Binding Affinity (LBA)

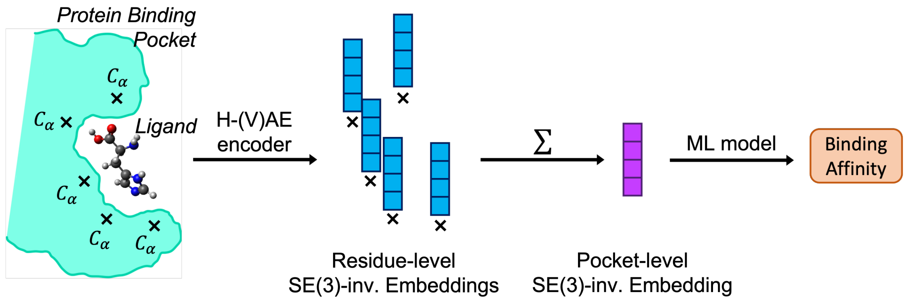

Figure 3: **Ligand Binding Affinity prediction pipeline using H-(V)AE.** We use H-(V)AE to extract residue-level embeddings in the binding pocket of a protein-ligand structure complex. We then sum over these embeddings to form an SE(3)-invariant pocket embedding that is used as an input to a standard Machine Learning model to predict the binding affinity between the protein and the ligand.

Here we provide a strong use case for H-(V)AE in structural biology. Specifically, we show that H-(V)AE can be leveraged to predict the affinity of protein-ligand interactions.

We follow the pipeline in Fig. 3 and leverage the inferred protein structure embeddings from H-AE (Sec. 4.1) to predict protein-ligand binding affinities. Given a protein-ligand structure complex, we identify residues in the binding pocket (i.e., residues with Cα within 10Å of the ligand) and extract their structure neighborhoods, which include atoms from both the protein and the ligand. We then pass the residue-centered neighborhoods of the binding pocket through a trained H-AE's encoder to extract their SO(3)-invariant embeddings. Since protein-ligand binding affinity is an extensive quantity in the number of interacting residues, we construct a pocket embedding by summing over residue-level embeddings, rather than averaging over them; the resulting pocket embedding is SE(3)-invariant, reflecting the natural symmetry of the LBA task. We use these pocket embeddings as feature vectors to train simple Machine Learning models to predict protein-ligand binding affinities. We use the LBA dataset in ATOM3D (Townshend et al., 2021) to benchmark our method, which contains $\sim 5,000$ protein-ligand complexes in total split into sets by ensuring a maximum of 30% sequence similarity between training and validation/testing sets. We note that 139 of the complexes in ATOM3D were also present in the training and validation data of our H-AE models (130 in training, 2 in validation and 7 in the test sets of ATOM3D). We don't consider this to be a limitation of our method since the objective of H-AE is unsupervised and can be seen as a form of pre-training. See Sec. A.7 for further details.

Table 1 shows the test performance of our method using simple Linear and Random Forest Regressors (we choose the H-AE model with best RMSD on validation split, which is the model with $L = 6$ and $z = 128$ for both Linear Regression and Random Forest). For each set of predictions, we use an ensemble of 10 Regressors as we noted a small but consistent improvement in performance. The Linear model achieves competitive results, whereas the Random Forest Regressor achieves state-of-the-art. We further compare against a baseline constructed using the SO(3)-invariant ($\ell = 0$)

Table 1: **Comprehensive benchmarking results on the Ligand Binding Affinity task with Atom3D's 30% similarity split.** Models are sorted by date of release. In addition to the H-AE infomed models, we also report the performance of baseline models that only use the SO(3)-invariant ($\ell = 0$) component of each neighborhood's Zernike transform (Zernike Inv.). H-AE consistently outperforms this baseline, indicating that the SO(3)-invariant information from the higher-degree features extracted by H-AE are informative for this regression task. Best scores are in **bold** and second-best scores are underlined. For our methods, we report scores ensembled across 10 Machine Learning models, with standard deviation as error.

	Ligand Binding Affinity 30% Similarity			
Method	RMSD $\downarrow$	Pearson's r $\uparrow$	Spearman's r $\uparrow$	Kendall's τ $\uparrow$
DeepDTA (Öztürk et al., 2018)	1.565	0.573	0.574	-
DeepAffinity (Karimi et al., 2019)	1.893 ± 0.650	0.415	0.426	-
Cormorant (Anderson et al., 2019)	1.568 ± 0.012	0.389	0.408	-
ProtTrans (Elnaggar et al., 2022)	1.544 ± 0.015	0.438 ± 0.053	0.434 ± 0.058	-
3DCNN (Townshend et al., 2021)	1.414 ± 0.021	0.550	0.553	-
GNN (Townshend et al., 2021)	1.570 ± 0.025	0.545	0.533	-
MaSIF (Gainza et al., 2020)	1.484 ± 0.018	0.467 ± 0.020	0.455 ± 0.014	-
DGAT (Nguyen et al., 2021)	1.719 ± 0.047	0.464	0.472	-
DGIN (Nguyen et al., 2021)	1.765 ± 0.076	0.426	0.432	-
DGAT-GCN (Nguyen et al., 2021)	1.550 ± 0.017	0.498	0.496	-
GVP-GNN (Jing et al., 2021)	1.648 ± 0.014	0.213 ± 0.013	0.164 ± 0.009	0.110 ± 0.012
EGNN (Satorras et al., 2022)	1.492 ± 0.012	0.489 ± 0.017	0.472 ± 0.008	0.329 ± 0.014
HoloProt (Somnath et al., 2022)	1.464 ± 0.006	0.509 ± 0.002	0.500 ± 0.005	-
GBPNet (Aykent & Xia, 2022)	1.405 ± 0.009	0.561	0.557	-
EGNN + PLM (Wu et al., 2022b)	1.403 ± 0.013	0.565 ± 0.016	0.544 ± 0.005	0.379 ± 0.007
ProtMD (Wu et al., 2022a)	1.367 ± 0.014	0.601 ± 0.036	0.587 ± 0.042	-
Zernike Inv. + Linear Regression	1.455 ± 0.005	0.513 ± 0.005	0.516 ± 0.006	0.357 ± 0.005
Zernike Inv. + Random Forest	1.361 ± 0.011	0.587 ± 0.009	0.584 ± 0.010	0.408 ± 0.008
H-AE + Linear Regression	1.397 ± 0.019	0.560 ± 0.017	0.568 ± 0.018	0.397 ± 0.016
H-AE + Random Forest	$\mathbf{1.332 \pm 0.012}$	$\mathbf{0.612 \pm 0.009}$	$\mathbf{0.619 \pm 0.009}$	$\mathbf{0.436 \pm 0.006}$

component of each neighborhood's Zernike transform (our Zernike inv. baseline), instead of the complete embeddings learned by H-AE. Using H-AE consistently outperforms this baseline, implying that invariant information encoded in higher spherical degrees, and extracted by H-AE, can lead to more expressive models for downstream regression tasks. Overall, these results show the utility of unsupervised learning for residue-level protein structure representations in predicting complex protein functions. Most of the competing structure-based methods for LBA (Table 1) learn complex graph-based functions on top of simple atomic representation, whereas our method uses simpler machine learning models over rich residue-level representations. Given the computational cost of training complex atom-level graph-based models, our residue-based approach can offer a more viable alternative for modeling large protein interfaces.

5 Conclusion

In this work, we develop the first end-to-end SO(3)-equivariant (V)AE, suitable for data distributed around a center. The model learns an invariant embedding describing the data in a "canonical" orientation alongside an equivariant frame describing the data's original orientation relative to the canonical one.

We show that H-(V)AE can be used to extract rich residue-level representations of protein structures, which can be used as embeddings for downstream structure-based tasks such as Ligand Binding Affinity prediction. Indeed, H-(V)AE representations paired with a simple Random Forest Regressor achieve state-of-the-art results on the Atom3D benchmark for LBA. More broadly, we expect our method to be useful for coarse-graining full-atom representations of protein structures to facilitate structure-based predictions of protein function. For example, a large protein graph can be coarse-grained by substituting its full-atom representation with rich embeddings of local structural neighborhoods learned from an unsupervised model. With an added supervised step, these coarse-grained embedding can be leveraged to predict complex protein functions. This approach is akin to using protein embeddings for sequence data, learned by language models, to inform (few-shot) predictions for protein function (Swanson et al., 2022).

Overall, H-(V)AE can be used to extract rich, symmetry-aware features from spherical images and complex 3D objects, to be used in downstream tasks that benefit from the symmetry constraints.

REFERENCES

SHREC 2017: Large-scale 3D Shape Retrieval from ShapeNet Core55. URL `https://shapenet.cs.stanford.edu/shrec17/`.

Mark Aldenderfer and Roger Blashfield. *Cluster Analysis*. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America, 1984. ISBN 978-0-8039-2376-8 978-1-4129-8364-8. doi: 10.4135/9781412983648. URL `https://methods.sagepub.com/book/cluster-analysis`.

Mohammed AlQuraishi. ProteinNet: a standardized data set for machine learning of protein structure. *BMC Bioinformatics*, 20(1):311, June 2019. ISSN 1471-2105. doi: 10.1186/s12859-019-2932-0. URL `https://doi.org/10.1186/s12859-019-2932-0`.

Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant Molecular Neural Networks, November 2019. URL `http://arxiv.org/abs/1906.04015`. arXiv:1906.04015 [physics, stat].

Sarp Aykent and Tian Xia. GBPNet | Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 2022. URL `https://dl.acm.org/doi/abs/10.1145/3534678.3539441`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL `http://arxiv.org/abs/1607.06450`. arXiv:1607.06450 [cs, stat].

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13 (1):2453, May 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29939-5. URL `https://www.nature.com/articles/s41467-022-29939-5`. Number: 1 Publisher: Nature Publishing Group.

Erik J. Bekkers, Maxime W. Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-Translation Covariant Convolutional Networks for Medical Image Analysis, June 2018. URL `http://arxiv.org/abs/1804.03393`. arXiv:1804.03393 [cs, math].

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.235. URL `rcsb.org`.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space, May 2016. URL `http://arxiv.org/abs/1511.06349`. arXiv:1511.06349 [cs].

John P. Boyd and Fu Yu. Comparing seven spectral methods for interpolation and for solving the Poisson equation in a disk: Zernike polynomials, Logan–Shepp ridge polynomials, Chebyshev–Fourier Series, cylindrical Robert functions, Bessel–Fourier expansions, square-to-disk conformal mapping and radial basis functions. *Journal of Computational Physics*, 230(4): 1408–1438, February 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.11.011. URL `https://www.sciencedirect.com/science/article/pii/S0021999110006133`.

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and Physical Quantities Improve E(3) Equivariant Message Passing, March 2022. URL `http://arxiv.org/abs/2110.02905`. arXiv:2110.02905 [cs, stat].

Alice Capecchi, Daniel Probst, and Jean-Louis Reymond. One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *Journal of Cheminformatics*, 12(1):43, June 2020. ISSN 1758-2946. doi: 10.1186/s13321-020-00445-4. URL https://doi.org/10.1186/s13321-020-00445-4.

Oliver J. Cobb, Christopher G. R. Wallis, Augustine N. Mavor-Parker, Augustin Marignier, Matthew A. Price, Mayeul d'Avezac, and Jason D. McEwen. Efficient Generalized Spherical CNNs, March 2021. URL http://arxiv.org/abs/2010.11661. arXiv:2010.11661 [astro-ph].

Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical CNNs, February 2018. URL http://arxiv.org/abs/1801.10130. arXiv:1801.10130 [cs, stat].

Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012. ISSN 1558-0792. doi: 10.1109/MSP.2012.2211477. Conference Name: IEEE Signal Processing Magazine.

Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1):014104, January 2019. doi: 10.1103/PhysRevB.99.014104. URL https://link.aps.org/doi/10.1103/PhysRevB.99.014104. Publisher: American Physical Society.

Albert Einstein. Die Grundlage der allgemeinen Relativitätstheorie. *Annalen der Physik*, (49):770–822, 1916.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, October 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3095381.

Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. Spin-Weighted Spherical CNNs, October 2020. URL http://arxiv.org/abs/2006.10731. arXiv:2006.10731 [cs].

Ilya Feige. Invariant-equivariant representation learning for multi-class data. February 2022. URL https://openreview.net/forum?id=B1e4wo09K7.

Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks, November 2020. URL http://arxiv.org/abs/2006.10503. arXiv:2006.10503 [cs, stat].

P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, February 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0666-6. URL https://www.nature.com/articles/s41592-019-0666-6. Number: 2 Publisher: Nature Publishing Group.

Mario Geiger and Tess Smidt. e3nn: Euclidean Neural Networks, July 2022. URL http://arxiv.org/abs/2207.09453. arXiv:2207.09453 [cs].

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. July 2022. URL https://openreview.net/forum?id=Sy2fzU9gl.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming Auto-Encoders. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2011*, Lecture Notes in Computer Science, pp. 44–51, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-21735-7. doi: 10.1007/978-3-642-21735-7_6.

Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. LieTransformer: Equivariant self-attention for Lie Groups, June 2021. URL http://arxiv.org/abs/2012.10885. arXiv:2012.10885 [cs, stat].

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from Protein Structure with Geometric Vector Perceptrons, May 2021. URL http://arxiv.org/abs/2009.01411. arXiv:2009.01411 [cs, q-bio, stat].

Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen. DeepAffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*, 35(18):3329–3338, September 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz111. URL https://doi.org/10.1093/bioinformatics/btz111.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. December 2013. doi: 10.48550/arXiv.1312.6114. URL https://arxiv.org/abs/1312.6114v10.

Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network, November 2018. URL http://arxiv.org/abs/1806.09231. arXiv:1806.09231 [cs, stat].

Kaushik Koneripalli, Suhas Lohit, Rushil Anirudh, and Pavan Turaga. Rate-Invariant Autoencoding of Time-Series. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3732–3736, May 2020. doi: 10.1109/ICASSP40776.2020.9053983. ISSN: 2379-190X.

Adam R. Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E. Hinton. Stacked Capsule Autoencoders, December 2019. URL http://arxiv.org/abs/1906.06818. arXiv:1906.06818 [cs, stat].

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs, June 2022. URL http://arxiv.org/abs/2206.11990. arXiv:2206.11990 [physics].

Suhas Lohit and Shubhendu Trivedi. Rotation-Invariant Autoencoders for Signals on Spheres, December 2020. URL http://arxiv.org/abs/2012.04474. arXiv:2012.04474 [cs].

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. URL http://arxiv.org/abs/1802.03426. arXiv:1802.03426 [cs, stat].

Éloi Mehr, André Lieutier, Fernando Sanchez Bermudez, Vincent Guitteny, Nicolas Thome, and Matthieu Cord. Manifold Learning in Quotient Spaces. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9165–9174, June 2018. doi: 10.1109/CVPR.2018.00955. ISSN: 2575-7075.

Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics, April 2022. URL http://arxiv.org/abs/2204.05249. arXiv:2204.05249 [cond-mat, physics:physics].

Felix Musil, Andrea Grisafi, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Physics-Inspired Structural Representations for Molecules and Materials. *Chemical Reviews*, 121 (16):9759–9815, August 2021. ISSN 0009-2665. doi: 10.1021/acs.chemrev.1c00021. URL https://doi.org/10.1021/acs.chemrev.1c00021. Publisher: American Chemical Society.

Thin Nguyen, Hang Le, Thomas P Quinn, Tri Nguyen, Thuc Duy Le, and Svetha Venkatesh. GraphDTA: predicting drug–target binding affinity with graph neural networks. *Bioinformatics*, 37(8):1140–1147, April 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa921. URL https://doi.org/10.1093/bioinformatics/btaa921.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. ISSN 1533-7928. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

David W. Romero and Jean-Baptiste Cordonnier. Group Equivariant Stand-Alone Self-Attention For Vision, March 2021. URL `http://arxiv.org/abs/2010.00977`. arXiv:2010.00977 [cs, stat].

Andrew Rosenberg and Julia Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/D07-1043`.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks, February 2022. URL `http://arxiv.org/abs/2102.09844`. arXiv:2102.09844 [cs, stat].

Erhard Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. *Mathematische Annalen*, 63(4):433–476, December 1907. ISSN 1432-1807. doi: 10.1007/BF01449770. URL `https://doi.org/10.1007/BF01449770`.

M. Shanker, M. Y. Hu, and M. S. Hung. Effect of data standardization on neural network training. *Omega*, 24(4):385–397, August 1996. ISSN 0305-0483. doi: 10.1016/0305-0483(96)00010-2. URL `https://www.sciencedirect.com/science/article/pii/0305048396000102`.

Zhixin Shu, Mihir Sahasrabudhe, Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance, June 2018. URL `http://arxiv.org/abs/1806.06503`. arXiv:1806.06503 [cs].

Vignesh Ram Somnath, Charlotte Bunne, and Andreas Krause. Multi-Scale Representation Learning on Proteins, April 2022. URL `http://arxiv.org/abs/2204.02337`. arXiv:2204.02337 [cs, q-bio].

Minyi Su, Qifan Yang, Yu Du, Guoqin Feng, Zhihai Liu, Yan Li, and Renxiao Wang. Comparative Assessment of Scoring Functions: The CASF-2016 Update. *Journal of Chemical Information and Modeling*, 59(2):895–913, February 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00545. URL `https://doi.org/10.1021/acs.jcim.8b00545`. Publisher: American Chemical Society.

Kyle Swanson, Howard Chang, and James Zou. Predicting Immune Escape with Pretrained Protein Language Model Embeddings. In *Proceedings of the 17th Machine Learning in Computational Biology meeting*, pp. 110–130. PMLR, December 2022. URL `https://proceedings.mlr.press/v200/swanson22a.html`. ISSN: 2640-3498.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds, May 2018. URL `http://arxiv.org/abs/1802.08219`. arXiv:1802.08219 [cs].

Raphael Townshend, Martin Vögele, Patricia Suriana, Alex Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Bowen Jing, Brandon Anderson, Stephan Eismann, Risi Kondor, Russ Altman, and Ron Dror. ATOM3D: Tasks on Molecules in Three Dimensions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, December 2021. URL `https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c45147dee729311ef5b5c3003946c48f-Abstract-round1.html`.

Wu-Ki Tung. *Group Theory in Physics*. 1985. ISBN 978-9971-966-57-7.

Martin Uhrin. Through the eyes of a descriptor: Constructing complete, invertible descriptions of atomic environments. *Physical Review B*, 104(14):144110, October 2021. ISSN 2469-9950, 2469-9969. doi: 10.1103/PhysRevB.104.144110. URL `http://arxiv.org/abs/2104.09319`. arXiv:2104.09319 [cond-mat].

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 1096–1103, New York, NY, USA, July 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL `https://doi.org/10.1145/1390156.1390294`.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data, October 2018. URL `http://arxiv.org/abs/1807.02547`. arXiv:1807.02547 [cs, stat].

Robin Winter, Marco Bertolini, Tuan Le, Frank Noé, and Djork-Arné Clevert. Unsupervised Learning of Group Invariant and Equivariant Representations, February 2022. URL `http://arxiv.org/abs/2202.07559`. arXiv:2202.07559 [cs].

Fang Wu, Shuting Jin, Yinghui Jiang, Xurui Jin, Bowen Tang, Zhangming Niu, Xiangrong Liu, Qiang Zhang, Xiangxiang Zeng, and Stan Z. Li. Pre-training of Equivariant Graph Matching Networks with Conformation Flexibility for Drug Binding. *Advanced Science*, 9(33):2203796, November 2022a. ISSN 2198-3844, 2198-3844. doi: 10.1002/advs.202203796. URL `http://arxiv.org/abs/2204.08663`. arXiv:2204.08663 [cs].

Fang Wu, Yu Tao, Dragomir Radev, and Jinbo Xu. When Geometric Deep Learning Meets Pre-trained Protein Language Models, December 2022b. URL `http://arxiv.org/abs/2212.03447`. arXiv:2212.03447 [cs, q-bio].

Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, September 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty593. URL `https://doi.org/10.1093/bioinformatics/bty593`.

A Appendix

A.1 Expanded background on SO(3)-equivariance

A.1.1 Invariance and Equivariance

Let $f : X \to Y$ be a function between two vector spaces and $\mathfrak{G}$ a group, where $\mathfrak{G}$ acts on X and via representation $\boldsymbol{D}_X$ and on Y via representation $\boldsymbol{D}_Y$. Then, f is said to be $\mathfrak{G}$-equivariant iff $f(\boldsymbol{D}_X(\mathfrak{g})\boldsymbol{x}) = \boldsymbol{D}_Y(\mathfrak{g})f(\boldsymbol{x})\,, \forall \boldsymbol{x} \in X \wedge \forall \mathfrak{g} \in \mathfrak{G}$. We note that invariance is a special case of equivariance where $\boldsymbol{D}_Y(\mathfrak{g}) = \boldsymbol{I}\,, \forall \mathfrak{g} \in \mathfrak{G}$.

A.1.2 Group representations and the irreps of SO(3)

Groups can concretely act on distinct vector spaces via distinct group representations. Formally, a group representation defines a set of invertible matrices $\boldsymbol{D}_X(\mathfrak{g})$ parameterized by group elements $\mathfrak{g} \in \mathfrak{G}$, which act on vector space X. As an example, two vector spaces that transform differently under the 3D rotation group SO(3)- and thus have different group representations - are scalars, which do not change under the action of SO(3), and 3D vectors, which rotate according to the familiar 3D rotation matrices.
A special kind of representation for any group are the irreducible representations (irreps) which are provably the "smallest" nontrivial (i.e., they have no nontrivial group-invariant subspaces) representations. The irreps of a group are special because it can be proven that any finite-dimensional unitary group representation can be decomposed into a direct sum of irreps (Tung, 1985). This applies to SO(3) as well, whose irreps are the Wigner-D matrices, which are $(2\ell + 1 \times 2\ell + 1)$-dimensional matrices, each acting on a $(2\ell + 1)$-dimensional vector space:

$$\boldsymbol{D}_\ell(\mathfrak{g}) \quad \text{for } \ell = 0, 1, 2, ... \tag{A.1}$$

Therefore, every element of the SO(3) group acting on any vector space can be represented as a direct sum of Wigner-D matrices.

A.1.3 Steerable features

A G-steerable vector is a vector $\boldsymbol{x} \in X$ that under some transformation group $\mathfrak{G}$, transforms via matrix-vector multiplication $\boldsymbol{D}_X(\mathfrak{g})\boldsymbol{x}$; here, $\boldsymbol{D}_X(\mathfrak{g})$ is the group representation of $\mathfrak{g} \in \mathfrak{G}$. For example, a vector in 3D Euclidean space is SO(3)-steerable since it rotates via matrix-vector multiplication using a rotation matrix.

However, we can generalize 3D rotations to arbitrary vector spaces by employing the irreps of SO(3). We start by defining a degree-ℓ feature as a vector that is SO(3)-steerable by the ℓ^{th} Wigner-D matrix $\boldsymbol{D}_\ell$. Given the properties of irreps, we can represent any SO(3)-steerable vector as the direct sum of two or more independent degree-ℓ features, e.g. $\boldsymbol{x} = \boldsymbol{x}_{\ell_1} \oplus \boldsymbol{x}_{\ell_2} \oplus ... \oplus \boldsymbol{x}_{\ell_n}$. The resulting vector, which we refer to as a *tensor* to indicate that it is composed of multiple individually-steerable vectors, is SO(3)-steerable via the direct sum of Wigner-D matrices of corresponding degrees. This tensor is a block-diagonal matrix with the Wigner-D matrices along the diagonal: $\boldsymbol{D}(\mathfrak{g}) = \boldsymbol{D}_{\ell_1}(\mathfrak{g}) \oplus \boldsymbol{D}_{\ell_2}(\mathfrak{g}) \oplus ... \oplus \boldsymbol{D}_{\ell_n}(\mathfrak{g})$.

A.1.4 Spherical harmonics and the Spherical Fourier Transform

Spherical harmonics are a class of functions that form a complete and orthonormal basis for functions $f(\theta, \phi)$ defined on a unit sphere; θ and ϕ are the azimuthal and the polar angles in the spherical coordinate system. In their complex form, spherical harmonics are defined as,

$$Y_{\ell m}(\theta, \phi) = \sqrt{\frac{2n+1}{4\pi}\frac{(n-m)!}{(n+m)!}} e^{im\phi} P_\ell^m(\cos\theta) \tag{A.2}$$

where ℓ is a non-negative integer ($0 \leq \ell$) and m is an integer within the interval $-\ell \leq m \leq \ell$. $P_\ell^m(\cos\theta)$ is the Legendre polynomial of degree ℓ and order m, which together with the complex exponential $e^{im\phi}$ define sinusoidal functions over the angles θ and ϕ in the spherical coordinate system. Spherical harmonics are used to describe angular momentum in quantum mechanics.

Notably, spherical harmonics also form a basis for the irreps of SO(3), i.e., the Wigner-D matrices. Specifically, the SO(3) group acts on the ℓ^{th} spherical harmonic via the ℓ^{th} Wigner-D matrix:

$$Y_{\ell m}(\theta, \phi) \xrightarrow{\mathfrak{g} \in SO(3)} \sum_{m'=-\ell}^{\ell} D_{\ell}^{m'm}(\mathfrak{g}) Y_{\ell m'}(\theta, \phi) \tag{A.3}$$

Therefore, any data encoded in a spherical harmonics basis is acted upon by the SO(3) group via a direct sum of the Wigner-D matrices corresponding to the basis functions being used. Using our nomenclature, any such data encoding constitutes a steerable tensor. We can thus map any function $f(\theta, \phi)$ defined on a sphere into a steerable tensor using the Spherical Fourier Transform (SFT):

$$\hat{f}_{\ell m} = \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi) Y_{\ell m}(\theta, \phi) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \tag{A.4}$$

The signal can be reconstructed in the real space using the corresponding inverse Fourier transform. For computational purposes, we truncate Fourier expansions at a maximum angular frequency L, which results in an approximate reconstruction of the signal $\tilde{f}(\theta, \varphi)$ up to the angular resolution allowed by L,

$$\tilde{f}(\theta, \varphi) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \hat{f}_{\ell m} Y_{\ell m}(\theta, \phi) \tag{A.5}$$

Here, $\hat{f}_{\ell m}$ are the functions' Spherical Fourier coefficients.

A.1.5 Zernike polynomials and the Zernike Fourier transform

To encode a function $\rho(r, \theta, \phi)$ with both radial and angular components, we use Zernike Fourier transform,

$$\hat{Z}_{\ell m}^{n} = \int \rho(r, \theta, \phi) \, Y_{\ell m}(\theta, \phi) R_{\ell}^{n}(r) \, \mathrm{d}V \tag{A.6}$$

where $R_{\ell}^{n}(r)$ is the radial Zernike polynomial in 3D defined as,

$$R_{\ell}^{n}(r) = (-1)^{\frac{n-\ell}{2}} \sqrt{2n+3} \binom{\frac{n+\ell+3}{2} - 1}{\frac{n-\ell}{2}} |r|^{\ell} \, {}_2F_1\left(-\frac{n-1}{2}, \frac{n+\ell+3}{2}; \ell + \frac{3}{2}; |r|^2\right) \tag{A.7}$$

Here, ${}_2F_1(\cdot)$ is an ordinary hypergeometric function, and n is a non-negative integer representing a radial frequency, controlling the radial resolution of the coefficients. $R_{\ell}^{n}(r)$ is non-zero only for even values of $n - \ell \geq 0$. Zernike polynomials form a complete orthonormal basis in 3D, and therefore, can be used within a Fourier transform to expand and retrieve any 3D shape, if large enough ℓ and n coefficient are used. We refer to the Fourier transform of Eq. A.6 as the Zernike Fourier Trasform (ZFT).

To represent point clouds, a common choice for the function $\rho(\boldsymbol{r}) \equiv \rho(r, \theta, \phi)$ is the sum of Dirac-δ functions centered at each point:

$$\rho(\boldsymbol{r}) = \sum_{i \in \text{points}} \delta(\rho(\boldsymbol{r}_i) - \rho(\boldsymbol{r})) \tag{A.8}$$

This choice is powerful because the forward transform has a closed-form solution that does not require a discretization of 3D space for numerical computation. Specifically, the ZFT of a point cloud follows:

$$\hat{Z}_{\ell m}^{n} = \sum_{i \in \text{points}} R_{n}^{\ell}(r_i) Y_{\ell m}(\theta_i, \varphi_i) \tag{A.9}$$

Similar to SFT, we can reconstruct the data using inverse ZFT and define approximations by truncating the angular and radial frequencies at L and N, respectively,

$$\tilde{\rho}(r, \theta, \varphi) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} \sum_{n}^{N} \hat{Z}_{\ell m}^{n} R_{\ell}^{n}(r) Y_{\ell m}(\theta, \varphi) \tag{A.10}$$

The use of other radial bases is possible within our framework, as long as they are complete. Orthonormality is also desirable as it ensures that each basis encodes different information, resulting in a more efficient encoding of the coefficients. We use Zernike polynomials following Boyd & Yu (2011), which demonstrates that encoding with Zernike polynomials result in a faster convergence compared to the radial basis functions localized at different radii, as well as most other orthogonal harmonic bases, with the exception of Logan-Shepp. "Faster convergence" indicates that fewer frequencies are required to encode the same information. Uhrin (2021) also uses Zernike to construct invariant descriptors of atomic environments. Other equivariant methods use Bessel functions (Musaelian et al., 2022), though, according to Boyd & Yu (2011), Zernike encoding results in faster convergence.

A.2 DETAILS OF H-(V)AE COMPONENTS

A.2.1 LINEARITY

Let us consider a feature $\boldsymbol{h}_\ell$ containing C features of the same degree ℓ. $\boldsymbol{h}_\ell$ can be represented as a $C \times (2\ell+1)$ matrix where each row constitutes an individual feature. Then, we learn weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{C \times K}$ that linearly maps h_ℓ to $\overline{h}_\ell \in \mathbb{R}^{K \times (2\ell+1)}$:

$$\overline{\boldsymbol{h}}_\ell = \boldsymbol{W}_\ell^T \boldsymbol{h}_\ell \tag{A.11}$$

A.2.2 EFFICIENT TENSOR PRODUCT (ETP)

Channel-wise tensor product nonlinearity. We effectively compute C tensor products, each between features belonging to the same channel c, and concatenate all output features of the same degree. In other words, features belonging to different channels are not mixed in the nonlinearity; the across-channel mixing is instead done in the linear layer. This procedure reduces the computational time and the output size of the nonlinear layers with respect to the number of channels C, from $\mathcal{O}(C^2)$ for a "fully-connected" tensor product down to $\mathcal{O}(C)$. The number of learnable parameters in a linear layer are proportional to the size of the output space in the preceding nonlinear layer. Therefore, reducing the size of the nonlinear output substantially reduces the complexity of the model and the number of model parameters. This procedure also forces the input tensor to have the same number of channels for all degrees. We refer the reader to Cobb et al. (2021) for further details and for a nice visualization of this procedure.

Minimum Spanning Tree (MST) subset for degree mixing. To compute features of the same degree ℓ_3 using the CG Tensor Product, pairs of features of varying degrees may be used, up to the rules of the CG Tensor Product. Specifically, pairs of features with any degree pair (ℓ_1, ℓ_2) may be used to produce a feature of degree ℓ_3 as long as $|\ell_1 - \ell_2| \leq \ell_3 \leq \ell_1 + \ell_2$. Features of the same degree are then concatenated to produce the final equivariant (steerable) output tensor.
Since each produced feature (often referred to as a "fragment" in the literature Kondor et al. (2018); Cobb et al. (2021)) is independently equivariant, computing only a subset of them still results in an equivariant output, albeit with lower representational power. Reducing the number of computed fragments is desirable since their computation cannot be easily parallelized. In other words, to reduce complexity we should identify a small subset of fragments that can still offer sufficient representational power. In this work we adopt the "MST subset" solution proposed by Cobb et al. (2021), which adopts the following strategy: when computing features of the same degree ℓ_3, exclude the degree pair (ℓ_0, ℓ_2) if the (ℓ_0, ℓ_1) and the (ℓ_1, ℓ_2) pairs have already been computed. The underlying assumption behind this solution is that the last two pairs already contain some information about the first pair, thus making its computation redundant.
The resulting subset of pairs can be efficiently computed via the Minimum Spanning Tree of the graph describing the possible pairs used to generate features of a single degree ℓ_3, given the maximum desired degree ℓ_{max}. As multiple such trees exist, we choose the one minimizing the computational complexity by weighting each edge (i.e. each pair) in the graph accordingly (edge (ℓ_1, ℓ_2) gets weight $(2\ell_1 + 1)(2\ell_2 + 1)$). The subset is also augmented to contain all the pairs with same degree to inject more nonlinearity. This procedure reduces the complexity in number of pairs with respect to ℓ_{max} from $\mathcal{O}(\ell_{\text{max}}^2)$ - when all possible pairs are used - down to $\mathcal{O}(\ell_{\text{max}})$. We refer the reader to Cobb et al. (2021) for more details and for a nice visualization.

A.2.3 Batch norm

Let us consider a batch of steerable tensors h which we index by batch b, degree ℓ, order m and channel c. During training, we compute a batch-averaged norm for each degree ℓ and each channel c as,

$$N_\ell^c = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{2\ell+1}\sum_{m=-\ell}^{\ell}(\hat{h}_{\ell m}^{cb})^2 \tag{A.12}$$

Similar to standard batch normalization, we also keep a running estimate of the training norms $N_\ell^{c,tr(i)}$ using momentum ξ, set to 0.1 in all our experiments:

$$N_\ell^{c,tr(i)} = \xi N_\ell^c + (1-\xi)N_\ell^{c,tr(i-1)} \tag{A.13}$$

We then update the features of the steerable tensor using the real batch-averaged norms during training, and the running batch-averaged norms during testing, together with a learned affine transformation:

$$\overline{\hat{h}_{\ell m}^{cb}} = \frac{\hat{h}_{\ell m}^{cb}}{\sqrt{N_\ell^c}}\,w_\ell^c \qquad \text{training} \tag{A.14}$$

$$\overline{\hat{h}_{\ell m}^{cb}} = \frac{\hat{h}_{\ell m}^{cb}}{\sqrt{N_\ell^{c,tr(i)}}}\,w_\ell^c \qquad \text{evaluation} \tag{A.15}$$

A.2.4 Data normalization

As per standard machine learning practice (Shanker et al., 1996), we normalize the data. We do this by dividing each tensor by the average square-root total norm of the training tensors, analogously to the Signal Norm. This strategy puts the target values on a similar scale as the normalized activations learned by the network, which we speculate to favor gradient flow.

A.2.5 Pairwise invariant reconstruction loss

To reconstruct a signal within an equivariant model it is desirable to have a *pairwise invariant* reconstruction loss, i.e., a loss $\mathcal{L}_{rec}$ such that $\mathcal{L}_{rec}(\boldsymbol{x},\boldsymbol{y}) = \mathcal{L}_{rec}(\boldsymbol{D}(\mathfrak{g})\boldsymbol{x},\boldsymbol{D}(\mathfrak{g})\boldsymbol{y})$ where $\boldsymbol{D}$ is the representation of the group element $\mathfrak{g}$ acting on the space that x and y inhabit (e.g. a rotation matrix if $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors in Euclidean 3D space, or a degree-ℓ wigner-D matrix if $\boldsymbol{x}$ and $\boldsymbol{y}$ are degree-ℓ vectors). This property is necessary for the model to remain equivariant, i.e., given that the network is agnostic to the transformation of the input under group operation $\boldsymbol{x}\to\boldsymbol{D}(\mathfrak{g})\boldsymbol{x}$ by producing a similarly transformed output $\boldsymbol{y}\to\boldsymbol{D}(\mathfrak{g})\boldsymbol{y}$, we want the reconstruction loss to be agnostic to the same kind of transformation as well

The MSE loss is pairwise invariant for any degree-ℓ feature on which SO(3) acts via the ℓ's Wigner-D matrix. Consider two degree-ℓ features $\boldsymbol{x}_\ell$ and $\boldsymbol{y}_\ell$ acted upon by a Wigner-D matrix $D_\ell(\mathfrak{g})$ parameterized by rotation $\mathfrak{g}$ (we drop the $\mathfrak{g}$ and ℓ indexing for clarity):

$$\begin{aligned}\text{MSE}(\boldsymbol{Dx},\boldsymbol{Dy}) &= (\boldsymbol{Dx}-\boldsymbol{Dy})^T(\boldsymbol{Dx}-\boldsymbol{Dy})\\ &= (\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{y}))^T(\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{y}))\\ &= (\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{D}^T\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{y})\\ &= (\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) \qquad \text{since Wigner-D matrices are unitary}\\ &= \text{MSE}(\boldsymbol{x},\boldsymbol{y})\end{aligned} \tag{A.16}$$

Since the MSE loss is pairwise invariant for every pair of degree-ℓ features, it is thus pairwise invariant for pairs of steerable tensors composed via direct products of steerable features.

A.2.6 Reconstruction assessment via cosine loss

As the scale of MSE depends on the characteristics of the data, e.g. the size of the tensors representing the data and their irreps (Fig. A.5), it is difficult to contextualize MSE values across datasets.

It would be desirable to have a dimensionless metric that measures absolute "goodness" of reconstructions that is comparable across datasets. For this purpose, we propose the metric *Cosine loss* which is a normalized dot product generalized to operate on pairs of steerable tensors (akin to cosine similarity), and modified to be interpreted as a loss:

$$\text{Cosine}(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x} \odot \boldsymbol{y}}{\sqrt{(\boldsymbol{x} \odot \boldsymbol{x})(\boldsymbol{y} \odot \boldsymbol{y})}}, \qquad \text{with} \quad \boldsymbol{x} \odot \boldsymbol{y} = \sum_{\ell'} (\boldsymbol{x}_{\ell'} \otimes_{cg} \boldsymbol{y}_{\ell'})_{\ell=0} \tag{A.17}$$

The Cosine loss is interpretable across different datasets: it is pairwise invariant, has a minimum of zero for perfect reconstructions, and has an average value of 1.0 for a pair of random tensors of any size, with a smaller variance for larger tensors (Sec. A.5). The interpretability of Cosine loss comes at the price of ignoring the relative norms of the features that are being compared, making the measure unable to reconstruct norms and thus not suitable as a training objective. However, as norms are easier to reconstruct than directions, we still find the Cosine loss useful as a noisy estimate of the model's reconstruction ability. Furthermore, Cosine loss correlates almost perfectly with MSE, especially in the mid-to-low reconstruction quality regime (SpearmanR $= 0.99$, Fig. A.5 and Table A.8).

Proof that Cosine loss is pairwise invariant. The generalized dot product $\odot$ from Eq. A.17 is pairwise invariant in the same way that the dot product between two 3D vectors depends only on their relative orientations but not the global orientation of the whole two-vector system. Therefore, the whole Cosine loss expression is pairwise invariant, since all of its components are pairwise invariant.

On the use case of Cosine loss. We introduce the Cosine loss as a measure of reconstruction that is both interpretable and comparable across datasets– the two characteristics that MSE does not have. A measure with these characteristics is practically useful for training of a network because it provides an estimate for how much better the reconstructions can get if the network's hyperparameters were to be further optimized. For example, looking at the Cosine loss in Table A.8, we see that our model trained on Shrec17 (best Cosine = 0.130) is not as well optimized as our model trained on MNIST (best Cosine = 0.017). Using MSE, the trend is reversed (1.8×10^{-3} vs. 6.7×10^{-3}), since the scale of MSE depends on the size of the irreps of the data (Fig. A.5).

A.3 Related Work

Group-equivariant neural networks. Group-equivariant neural networks have improved the state-of-the-art on many supervised tasks, thanks to their data efficiency (Bekkers et al., 2018; Hutchinson et al., 2021; Romero & Cordonnier, 2021). Related to our work are 3D Euclidean neural networks, which are equivariant to (subsets of) the 3D Euclidean group: (Weiler et al., 2018; Thomas et al., 2018; Fuchs et al., 2020; Brandstetter et al., 2022; Batzner et al., 2022; Musaelian et al., 2022; Satorras et al., 2022; Liao & Smidt, 2022) and often use spherical harmonics and tensor products to construct SO(3) equivariant layers. Seminal work on SO(3) equivariance has been conducted for spherical images (Cohen et al., 2018; Esteves et al., 2020); we were inspired by the fully Fourier approach of Kondor et al. (2018), and leveraged operations proposed by Cobb et al. (2021).

Equivariant representations of atomic systems. There is a diverse body of literature on constructing representations of atomic systems that are invariant/equivariant to euclidean symmetries, leveraging Fourier transforms and CG tensor products (Drautz, 2019; Musil et al., 2021). Notably, Uhrin (2021) constructs SO(3)-invariant representations of point clouds using the ZFT and CG tensor product iterations. Our work can be seen as a data-driven instance of this framework, where we learn a compact invariant and equivariant latent space from data.

Invariant autoencoders. Several works attempt to learn representations that are invariant to certain classes of transformations. Shu et al. (2018) and Koneripalli et al. (2020) learn general "shape" embeddings by learning a separate "deformation" embedding. However, their networks are not explicitly equivariant to the transformations of interest. Other work proposes to learn an exactly invariant embedding alongside an approximate (but not equivariant) group action to align the input and the reconstructed data. For example, Mehr et al. (2018) learns in quotient space by sampling the group's orbit and computing the reconstruction loss by taking the infimum over the group. This approach is best suited for discrete and finite groups, and it is computationally expensive as it is akin to data augmentation. Lohit & Trivedi (2020) construct an SO(3)-invariant autoencoder for spherical signals by learning an invariant latent space and minimizing a loss which first finds the rotation that best aligns the true and reconstructed signals, introducing an added optimization step - potentially very expensive for 3D data - and reconstructing the signals only up to a global rotation. To our knowledge, the method proposed by Lohit & Trivedi (2020) is the only approach to date for unsupervised learning of non-discretized SO(3)-invariant representations. However, the rotational invariance is manually imposed by Lohit & Trivedi (2020), which is distinct from our approach that is fully equivariant and only requires simple MSE for reconstruction of data in its original orientation.

Group-equivariant autoencoders. A small body of work focuses on developing equivariant autoencoders. Several methods construct data and group-specific architectures to auto-encode data equivariantly, learning an equivariant representation in the process (Hinton et al., 2011; Kosiorek et al., 2019). Others use supervision to extract class-invariant and class-equivariant representations (Feige, 2022). A recent theoretical work proposes to train an encoder that encodes elements into an invariant embedding and an equivariant group action, then using a standard decoder that uses the invariants to reconstruct the elements in a canonical form, and finally applying the learned group action to recover the data's original form (Winter et al., 2022). Our method in SO(3) is closely related to this work, with the crucial difference that we use an equivariant decoder and that we learn to reconstruct the Fourier encoding of the data. A more detailed comparison of the two approaches and the benefits of our fully equivariant approach can be found in the Appendix (Sec. A.4) and in Table A.1.

A.4 Using a non-equivariant decoder

Winter et al. (2022) propose to construct group-equivariant autoencoders by using an equivariant encoder that learns an invariant embedding and a group element, and an unconstrained decoder which uses the invariants alone to reconstruct each datapoint in the "canonical" form, before applying the learned group action in the output space. By contrast, for SO(3) we propose to use an equivariant decoder, whereby the learned group element is fed as input to the decoder. Such "unconstrained decoder" procedure can in principle be merged with our equivariant encoder and Fourier-space approach in two ways. For each, we argue in favor of using our equivariant decoder.

1) Reconstructing the Fourier coefficients of the data. To apply the learned group element on the decoder's output, the Wigner-D matrices for the data's irreps need to be computed from the group element. Then, the Wigner-D matrices can be used to "rotate" the tensor. This has to be done on-the-fly, and it can be done quickly using functions provided in the e3nn package (Geiger & Smidt, 2022) and by smartly vectorizing operations. We implemented this procedure by using a simple Multi-Layer Perceptron with SiLU non-linearities as a decoder. By using e3nn to compute Wigner-D matrices in batches, and by clever construction of tensor multiplications such that runtime scales linearly with ℓ_{max} and is constant with regards to number of channels and batch size, we achieve models that run with comparable speed to those using our equivariant decoder, and have comparable performance on MNIST (Table A.1). Given the empirical similarities we observe, though on a limited use case, we favor the simplicity and elegance of our equivariant decoder. "Simplicity" because we construct the decoder to be symmetric to the encoder, thus endowing it automatically with similar representational power and without the need to tune an architecture made with different base components. Furthermore, we highlight that our method generates intermediate equivariant representations in the decoder, rather than intermediate invariant representations. These intermediate equivariant representations may be of interest to study in and of themselves.

2) Reconstructing the data in real space. In this case, we do not have to compute Wigner-D matrices on-the-fly, since the learned frame can be used directly in the output space as a rotation matrix. However, since the encoder only sees a truncated Fourier representation of the data, which is by construction lossy, while the loss is computed over fine-grained real-space, this model might be too difficult to train. We suspect this would make the model akin to a denoising autoencoder (Vincent et al., 2008) and it might be interesting to analyze, but that would be beyond the scope of this paper. To avoid the denoising effect, we could learn to reconstruct data in real space after an Inverse Fourier Transform (IFT). However, computing the IFT on-the-fly is very expensive and requires a discretization of the input space, to the point of being prohibitive for point clouds. This is not a bottleneck for the Forward Fourier Transform if the cloud is parameterized by Dirac-Delta distributions, i.e., for point clouds, as the integral can be computed exactly (Eq. A.9).

Table A.1: **Performance comparison between our H-(V)AE and H-(V)AE with Winter et al. (2022)'s non-equivariant decoder formulation, on the MNIST-on-the-sphere dataset.** The non-equivariant decoders are constructed as simple MLPs with SiLU non-linearities, with the following hidden layer sizes: [32,64,128,160,256]. We keep the number of parameters approximately the same to make model comparison fair, but we do not tune the architecture of the invariant decoders. All other training details are kept the same (Sec. A.6).

Method	**z**	**Speed**	**MSE**	**Cosine**	**Purity**	**V-meas.**	**LC Class. Acc.**	**KNN Class. Acc.**
H-AE NR/R	16	**1.0**	$\mathbf{9.3 \times 10^{-4}}$	**0.025**	**0.62**	**0.51**	**0.820**	**0.862**
H-AE unconst. decoder NR/R	16	1.3	1.3×10^{-3}	0.037	0.61	0.48	0.802	0.856
H-VAE NR/R	16	**1.0**	$\mathbf{1.4 \times 10^{-3}}$	**0.057**	**0.67**	**0.54**	**0.812**	0.848
H-VAE unconst. decoder NR/R	16	1.3	2.1×10^{-3}	**0.057**	0.62	0.51	0.781	**0.853**

A.5 Implementation details

Without loss of generality, we use real spherical harmonics for implementation of H-(V)AE. We leverage `e3nn` Geiger & Smidt (2022), using their computation of the real spherical harmonics and their Clebsch-Gordan coefficients.

In our code, we offer the option to use the Full Tensor Product instead of the ETP. Specifically, at each block we allow the users to specify whether to compute the Tensor Product channel-wise or fully-connected across channels, and whether to compute using efficient or fully connected degree mixing.

A.6 Experimental details

A.6.1 Architecture specification

We describe model architectures as follows. We specify the number of blocks B, which is the same for the encoder and the decoder. We specify two lists, (i) DegreesList which contains the maximum degree $\ell_{max,b}$ of the output of each block b, and (ii) ChannelsList, containing the channel sizes C_b, of each block b. These lists are in the order as they appear in the encoder, and are reversed for the decoder. When it applies, we specify the number of output channels of the initial linear projection C_{init}. As noted in the main text, we use a fixed formula to determine $\ell_{max,b}$, but we specify it for clarity.

A.6.2 MNIST on the sphere

Model architectures. For models with invariant latent space size $z = 16$, we use 6 blocks, DegreesList $= [10, 10, 8, 4, 2, 1]$ and ChannelsList $= [16, 16, 16, 16, 16, 16]$, with a total of 227k parameters.
For models with invariant latent space size $z = 120$, we use 6 blocks, DegreesList $= [10, 10, 8, 4, 2, 1]$ and ChannelsList $= [16, 16, 16, 32, 64, 120]$, with a total of 453k parameters.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 50$. We train all models for 80 epochs using the Adam optimizer (Kingma & Ba, 2017) with default parameters, a batch size of 100, and an initial learning rate of 0.001, which we decrease exponentially by one order of magnitude over 25 epochs. For VAE models, we use $\beta = 0.2$, $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 35$. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models. Training took ~ 4.5 hours on a single NVIDIA A40 GPU for each model.

A.6.3 SHREC17

Model architectures. Both AE and VAE models have $z = 40$, 7 blocks, DegreesList $= [14, 14, 14, 8, 4, 2, 1]$, ChannelsList $= [12, 12, 12, 20, 24, 32, 40]$, $C_{\text{init}} = 12$, with a total of 518k parameters.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 1000$. We train all models for 120 epochs using the Adam optimizer with default parameters, a batch size of 100, and an initial learning rate of 0.0025, which we decrease exponentially by two orders of magnitude over the entire 120 epochs. For VAE models, we use $\beta = 0.2$, $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 10$. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models. Training took ~ 11 hours on a single NVIDIA A40 GPU for each model.

A.6.4 TOY AMINO ACIDS

Pre-processing of protein structures. We sample residues from the set of training structures pre-processed as described in Sec. A.6.5.

Fourier projection. We set the maximum radial frequency to $N = 20$ as it corresponds to a radial resolution matching the minimum inter-atomic distances after rescaling the atomic neighborhoods of radius $10.0\AA$ to fit within a sphere of radius 1.0, necessary for Zernike transform.
The channel composition of the data tensors can be described in a notation - analogous to that used by e3nn but without parity specifications - which specifies the number of channels C for each feature of degree ℓ in single units Cxℓ: 44x0 + 40x1 + 40x2 + 36x3 + 36x4.

Model architectures. All models have $z = 2$, 6 blocks, DegreesList $= [4, 4, 4, 4, 2, 1]$, ChannelsList $= [60, 40, 24, 16, 16, 8]$, $C_{\text{init}} = 48$, with a total of 495k parameters. We note that the initial projection is necessary since the number of channels differs across feature degrees in the data tensors.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 400$. We train all models for 80 epochs using the Adam optimizer with default parameters and an initial learning rate of 0.005, which we decrease exponentially by by one order of magnitude over 25 epochs. For VAE models, we use $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 10$. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models.
We vary the batch size according to the size of the training and the validation datasets. We use the following (dataset_size-batch_size) pairs: (400-4), (1,000-10), (2,000-20), (5,000-50), (20,000-20). Training took ~ 45 minutes on a single NVIDIA A40 GPU for each model.

Evaluation. We perform our data ablations by considering training and validation datasets of the following sizes: 400, 1,000, 2,000, 5,000 and 20,000. We keep relative proportions of residue types even in all datasets. We perform the data ablation with H-AE as well as H-VAE models with $\beta = 0.025$ and 0.1.
We further perform a β ablation using the full (20,000) dataset, over the following choices of β: $[0(\text{AE}), 0.025, 0.05, 0.1, 0.25, 0.5]$.

For robust results, we train 3 versions of each model and compute averages of quantitative metrics of reconstruction loss and classification accuracy.

For a fair comparison across models with varying amounts of training and validation data, we perform a 5-fold cross-validation-like procedure over the 10k test residues, where the classifier is trained over 4 folds of the test data and evaluated on the fifth one. If validation data is needed for model selection (e.g. for LC), we use 10% of the training data.

A.6.5 PROTEIN NEIGHBORHOODS

Pre-processing of protein structures. We model protein neighborhoods extracted from tertiary protein structures from the Protein Data Bank (PDB) (Berman et al., 2000). We use ProteinNet's splittings for training and validation sets to avoid redundancy, e.g. due to similarities in homologous protein domains (AlQuraishi, 2019). Since PDB ids were only provided for the training and validation sets, we used ProteinNet's training set as both our training and validation set and ProteinNet's validation set as our testing set. Specifically, we make a $[80\%, 20\%]$ split in the ProteinNet's training data to define our training and validation sets. This splitting resulted in 10,957 training structures, 2,730 validation structures, and 212 testing structures.

Projection details. We set the maximum radial frequency to $N = 20$ as it corresponds to a radial resolution matching the minimum inter-atomic distances after rescaling the atomic neighborhoods of radius 10.0 Å to fit within a sphere of radius 1.0, necessary for Zernike transform. We vary L and construct models tailored to each one (Table A.2).

Model architectures. See Table A.2. We note that the initial projection (C_{init}) is necessary since the number of channels differs across feature degrees in the data tensors, and the ETP necessitates equal number of channels for all degrees (Sec. A.2.2).

Table A.2: **Model architecture hyperparameters and training time for each H-AE trained on protein neighborhoods.**

z	L	**Tensor size**	$\mathbf{C_{init}}$	**ChannelsList**	**DegreesList**	**# Params**	**Training Speed (hours)**
64	3	616	44	[50,50,50,64,64,64]	[3,3,3,3,2,1]	631k	3.0
64	4	940	44	[50,50,50,64,64,64]	[4,4,4,4,2,1]	950k	3.8
64	6	1708	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	1.6M	5.9
64	8	2604	44	[50,50,50,64,64,64]	[8,8,8,4,2,1]	2.4M	9.3
128	3	616	44	[60,60,60,90,128,128]	[3,3,3,3,2,1]	1.2M	3.8
128	4	940	44	[60,60,60,90,128,128]	[4,4,4,4,2,1]	1.7M	4.4
128	6	1708	44	[60,60,60,90,128,128]	[6,6,6,4,2,1]	2.6M	6.0
128	8	2604	44	[60,60,60,90,128,128]	[8,8,8,4,2,1]	3.8M	10.1
256	3	616	44	[90,90,90,120,200,256]	[3,3,3,3,2,1]	2.8M	4.9
256	4	940	44	[90,90,90,120,200,256]	[4,4,4,4,2,1]	3.8M	5.6
256	6	1708	44	[90,90,90,120,200,256]	[6,6,6,4,2,1]	4.4M	6.0
256	8	2604	44	[90,90,90,120,200,256]	[8,8,8,4,2,1]	7.9M	14.5

Training details. We keep the learning schedule the same for all models. We use $\alpha = 1000$. We train all models for 8 epochs using the Adam optimizer with default parameters, a batch size of 512, and a constant learning rate of 0.001. We utilize the model with the lowest loss on validation data.

A.6.6 Latent space classification

Linear classifier. We implement the linear classifier as a one-layer fully connected neural network with input size equal to the invariant embedding of size z, and output size equal to the number of desired classes. We use cross entropy loss with logits as training objective, which we minimize for 250 epochs using the Adam optimizer with batch size 100, and initial learning rate of 0.01. We reduce the learning rate by one order of magnitude every time the loss on validation data stops improving for 10 epochs (if validation data is not provided, the training data is used). At evaluation time, we select the class with the highest probability value. We use PyTorch for our implementation.

KNN Classifier We use the `sklearn` (Pedregosa et al., 2011) implementation with default parameters. At evaluation time, we select the class with the highest probability value.

A.6.7 Clustering metrics

Purity (Aldenderfer & Blashfield, 1984). We first assign a class to each cluster based on the most prevalent class in it. Purity is computed as the sum of correctly classified items divided by the total number of items. Purity measures classification accuracy, and ranges between 0 (worst) and 1 (best).

V-measure (Rosenberg & Hirschberg, 2007). This common clustering metric strikes a balance between favoring homogeneous (high homogeneity score) and complete (high completeness score) clusters. Clusters are defined as homogeneous when all elements in the same cluster belong to the same class (akin to a precision). Clusters are defined as complete when all elements belonging to the same class are put in the same cluster (akin to a recall). The V-measure is computed as the harmonic mean of homogeneity and completeness in a given clustering.

A.6.8 On the complementary nature of classification accuracy and clustering metrics

The clustering metrics "purity" and "V-measure" and the supervised metric "classification accuracy" characterize different qualities of the latent space, and, while partly correlated, they are complementary to each other.

Both classes of metrics are computed by comparing the ground truth labels to the predicted labels, and they mainly differ by how the predicted labels are assigned; the clustering metrics use an unsupervised clustering algorithm, while the classification metric uses a supervised classification algorithm to do so. As a result, these metrics focus on different features of the latent space. For example, the clustering metrics are largest when the test data naturally forms clusters with all data points of the same label. While this case can result in a high supervised classification accuracy, clustering is not a necessary condition for high classification accuracy. Indeed, the supervised signal could make the predicted labels depend more heavily on a subset of the latent space features, instead of relying on all of them equally, which is what the clustering algorithm naturally does. Therefore, it is reasonable to conclude that having higher clustering metrics and a lower classification accuracy is a sign that class-related information is more evenly distributed across the latent space dimensions. Overall, the complementary aspect of these metrics makes it necessary to use all of them when comparing the performance of different models in each task.

A.7 Protein-Ligand Binding Affinity Prediction

A.7.1 Data Preprocessing Details

We leverage the H-AE models trained on Protein Neighborhoods (Sec. A.6.5) to predict the binding affinity between a protein and a ligand, given their structure complex– an important task in structural biology. For this task, we use the data from the "refined-set" of PDBBind (Su et al., 2019), containing $\sim 5,000$ structures. We use the dataset splits provided by ATOM3D (Townshend et al., 2021) to benchmark our predictions. ATOM3D provides two splits based on the maximum sequence similarity between proteins in the training set and the validation / test sets. We use the most challenging split for our benchmarking in which the sequences of the two sets have at most 30% similarity.

For each complex, we first identify residues in the binding pocket, which we define as residues for which the Cα is within 10Å of any of the ligand's atoms. We then extract the 10Å neighborhood for each of the pocket residues that can contain atoms from both the protein and the ligand. We compute the ZFT (Eq. 1) for each neighborhood with maximum degree L matching the degree used by the H-AE of interest. We then compute residue-level SO(3)-invariant embeddings by running the pre-trained H-AE encoder on the neighborhoods surrounding each of the residues in the pocket. We sum over the residue-level embeddings to compute a single pocket-level embedding, which is SE(3)-invariant by construction. Each pocket embedding is used as a feature vector within a standard Machine Learning Regression model to predict protein-ligand binding affinity, provided by PDBBind either in the form of a *dissociation constant* K_d, or an *inhibition constant* K_i. Due to data limitations, and consistent with other studies on LBA, we do not distinguish between K_d and K_i and regress over the negative log of either of the constants that is provided by PDBBind; the transformed quantity is closely related to the binding free energy of protein-ligand interactions.

The protein-ligand binding free energy is an extensive quantity, meaning that its magnitude depends on the number of residues in the binding pocket. In other words, a larger protein-ligand complex can establish a stronger binding. The pocket embedding, which we define as the sum of the residue-level embeddings, is a simple quantity that is extensive and SE(3)-invariant, and therefore, suitable for regressing over protein-ligand binding free energy. Nonetheless, this map is not unique and other extensive transformations to pool together residue-level embeddings into a pocket-level embedding can be used for this purpose.

It should be noted that in the protein-ligand structural neighborhoods we only include the ligand atoms that are found in proteins and were used in the training of the H-AE models (i.e., C, N, O,

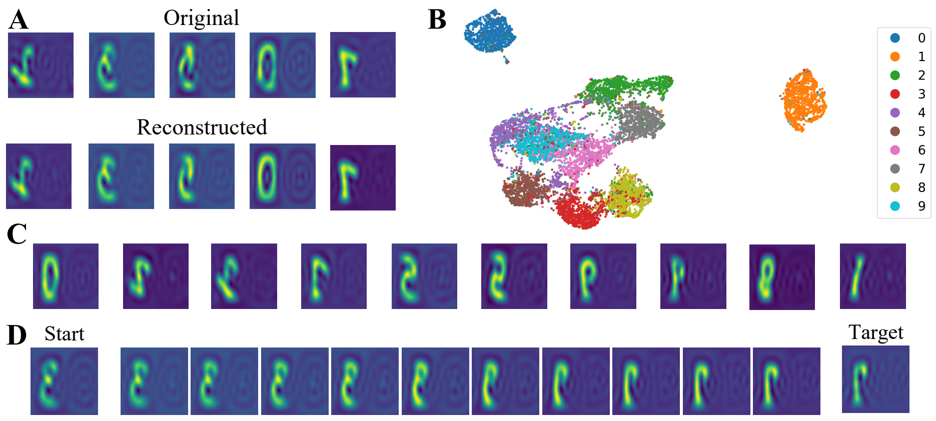

Figure A.1: **H-VAE on MINST-on-the-sphere.** Evaluation on rotated digits for an H-VAE trained on non-rotated digits with $z = 16$. **A:** Original and reconstructed images in the canonical frame after inverse transform from Fourier space. The images are projected onto a plane. Distortions at the edges and flipping are side-effects of the projection. **B:** visualization of the latent space via 2D UMAP (McInnes et al., 2020). Data points are colored by digit identity. **C:** Shown are cherry-picked images generated by feeding the decoder invariant embeddings sampled from the prior distribution and the canonical frame. **D:** Example image trajectory by linearly interpolating through the learned invariant latent space. We interpolate between the learned invariant embeddings of the Start and the Target images. Then, we feed each embedding to the decoder alongside the canonical frame.

and S). About 31% of the ligands contain other kinds of atoms, and since our model does not "see" these atoms, we hypothesize that our predictions are worse in these cases. In fact, for H-AE+R.F. (Table 1), the Spearman's correlation over the ligands containing only protein atoms is 0.605 ± 0.016 against 0.577 ± 0.040 for the ligands containing other kinds of atoms. Therefore, we expect training H-(V)AE to recognize more atom types - or at least making it aware that some other unspecified non-protein atom is present - would yield even better results; as protein structures are often found in complex with other non-protein entities, e.g. ligands and ions, there is training data available for constructing such models.

A.7.2 Machine Learning models used for prediction

Linear Regression. We implement a linear regression model as a one-layer fully connected neural network with one output layer. We use MSE loss as training objective, which we minimize for 250 epochs using the Adam optimizer with batch size 32, and initial learning rate of 0.01. We reduce the learning rate by one order of magnitude every time the loss on validation data stops improving for 10 epochs. We use PyTorch for our implementation.

Random Forest Regression. We use the `sklearn` (Pedregosa et al., 2011) implementation. We tune hyperparameters via grid search, choosing the combination minimizing RMSD on validation data. We consider the following grid of hyperparameter values:

- max_features: [1.0, 0.333, sqrt]
- min_samples_leaf: [2, 5, 10, 15]
- n_estimators: [32, 64, 100]

A.8 Additional experiments

A.8.1 Rotated MNIST on the sphere

We extensively test the performance of H-(V)AE on the MNIST-on-the-sphere dataset (Deng, 2012). Following Cohen et al. (2018) we generate a discrete unit sphere using the Driscoll-Healey (DH) method with a bandwidth (bw) of 30, and project the MNIST dataset onto the lower hemisphere. We consider two variants, NR/R and R/R, differing in whether the training/test images have been

Table A.3: **Performance metrics on MNIST-on-the-sphere and Shrec17**. Reconstruction loss, clustering metrics, classification accuracy in the latent space using a linear classifier, and retrieval metrics (Shrec17 only) are shown. We only report scores presented in the corresponding papers of origin.

Dataset	Type	Method	z	bw	Cosine	Purity	V-meas.	Class. Acc.	P@N	R@N	F1@N	mAP	NDCG
MNIST	Supervised	(Cobb et al., 2021) NR/R	-	30	-	-	-	0.993	-	-	-	-	-
	Unsupervised	(Lohit & Trivedi, 2020) NR/R	120	30	-	0.40	0.35	**0.894**	-	-	-	-	-
		H-AE NR/R (Ours)	120	30	0.017	0.62	0.48	0.877	-	-	-	-	-
		H-AE R/R (Ours)	120	30	0.018	0.51	0.41	0.881	-	-	-	-	-
		H-AE NR/R (Ours)	16	30	0.025	0.62	0.51	0.820	-	-	-	-	-
		H-AE R/R (Ours)	16	30	0.024	0.65	0.52	0.833	-	-	-	-	-
		H-VAE NR/R (Ours)	120	30	0.037	0.70	**0.59**	0.883	-	-	-	-	-
		H-VAE R/R (Ours)	120	30	0.037	0.65	0.53	0.884	-	-	-	-	-
		H-VAE NR/R (Ours)	16	30	0.057	0.67	0.54	0.812	-	-	-	-	-
		H-VAE R/R (Ours)	16	30	0.055	**0.72**	0.57	0.830	-	-	-	-	-
Shrec17	Supervised	(Esteves et al., 2020)	-	128	-	-	-	-	0.717	0.737	-	0.685	-
		(Cobb et al., 2021)	-	128	-	-	-	-	0.719	0.710	0.708	0.679	0.758
	Unsupervised	(Lohit & Trivedi, 2020)	120	30	-	0.41	0.34	0.578	0.351	0.361	0.335	0.215	0.345
		H-AE (Ours)	40	90	0.130	0.50	0.41	**0.654**	**0.548**	**0.569**	**0.545**	**0.500**	**0.597**
		H-VAE (Ours)	40	90	0.151	**0.52**	**0.42**	0.631	0.512	0.537	0.512	0.463	0.568

randomly rotated (R) or not (NR). For each dataset, we map the images to steerable tensors via the Zernike Fourier Transform (ZFT) with $L = 10$, and a constant radial function $R_\ell^n = 1$, resulting in tensors with 121 coefficients.

We train 8 models with combinations of the following features: training mode (NR vs. R), invariant latent space size z (16 vs. 120), and model type (AE vs. VAE). In all cases the model architecture follows from Fig. 1C; see Section A.6.2 for details. All 8 models achieve very low reconstruction loss (Table A.3) with no significant difference between training modes, indicating that the models successfully leverage SO(3)-equivariance to generalize to unseen orientations. Predictably, AE models have lower reconstruction loss than VAE models, and so do models with a larger latent space. Nonetheless, H-VAE achieves reliable reconstructions, as shown in Fig. A.1A and Table A.3. All 8 models produce an invariant latent space that naturally clusters by digit identity. We show this qualitatively for one of the models in Fig. A.1B, and quantitatively by clustering the data via K-Means with 10 centroids and computing standard clustering metrics of Purity (Aldenderfer & Blashfield, 1984) and V-measure (Rosenberg & Hirschberg, 2007) in Table A.3.

All 8 models achieve much better clustering metrics than Rot-Inv AE (Lohit & Trivedi, 2020), with VAE models consistently outperforming AE models. We also train a linear classifier (LC) to predict digit identity from invariant latent space descriptors, achieving comparable accuracy to Rot-Inv AE with the same latent space size. We do not observe any difference between VAE and AE models in terms of classification accuracy. Using a KNN classifier instead of LC further improves performance (Table A.4).
As H-VAE is a generative model, we generate random spherical images by sampling invariant latent embeddings from the prior distribution, and observing diversity in digit type and style (Fig. A.1C and Fig. A.6). We further assess the quality of the invariant latent space by generating images via linear interpolation of the invariant embeddings associated with two test images. The interpolated images present spatially consistent transitions (Fig. A.1D and Fig. A.7), which is a sign of a semantically well-structured latent space.
To understand the meaning of the learned frames, we visualize the sum of images in the canonical frame (i.e. the identity matrix): these images are well aligned within the same digit type and with varying degrees across different digit types depending on the content of training data (Fig. A.4). This indicates that H-(V)AE learns to optimally overlap the training images when in the same frame.

A.8.2 SHREC17

The Shrec17 dataset consists of 51k colorless 3D models belonging to 55 object classes, with a 70/10/20 train/valid/test split (noa). We use the variant of the dataset where each model is perturbed by random rotations. Converting 3D shapes into spherical images preserves topological surface information, while significantly simplifying the representation. We follow Cohen et al. (2018) and project surface information from each model onto an enclosing DH spherical grid with a bandwidth of 90 via a ray-casting scheme, generating spherical images with 6 channels. We then apply the ZFT with $L = 14$ and a constant radial function $R_\ell^n = 1$ to each channel individually, resulting in a tensor with 1350 coefficients. We train an AE and a VAE model (Sec. A.6.3) and evaluate them similarly to the MNIST dataset and compute the Shrec17 retrieval metrics via the latent space linear classifier's predictions. H-AE achieves the best classification and retrieval results for autoencoder-based models, and is competitive with supervised models despite the lower grid bandwidth and the small latent space (Table A.3). Using KNN classification instead of a linear classifier further improves performance (Table A.5). H-VAE achieves slightly worse classification results but better clustering metrics compared to H-AE. While reconstruction loss is low, there is still significant margin of improvement. We partially attribute Lohit & Trivedi (2020)'s low scores to the low grid bandwidth. However, we note that the size and runtime of our method does not scale with grid bandwidth, since the size of the reconstructed tensor learned by our method does not depend on it.

A.8.3 Toy Amino-Acids

We train H-(V)AE on single amino acids represented as atomic point clouds, extracted from structures in the Protein Data Bank (PDB) (Berman et al., 2000). We collect atomic point clouds of 50k residues from PDB evenly distributed across residue types and apply a 40/40/20 train/valid/test split. Residues of the same type have different conformations and naturally have noisy coordinates, making this problem a natural benchmark for our method.

We consider atom-type-specific clouds (C, O, N and S; we exclude H) centered at the residue's Cα and compute the ZFT (Eq. 1) with $L = 4$ and $N = 20$ within a radius of 10Å from the residue's Cα, and concatenate features of the same degree, resulting in a tensor with 940 coefficients. We train several H-AE and H-VAE models, all with $z = 2$; see Sec. A.6.4 for details.

We consistently find that the latent space clusters by amino acid conformations (Fig. A.2), with sharper cluster separations as more training data is added (Fig. A.9 and A.10). We find that test reconstruction loss decreases with more training data but the reconstruction is accurate even with little training data (from 0.153 Cosine loss with 400 training residues to 0.034 with 20,000); Table A.6. A similar trend is observed for KNN-based classification accuracy of residues (from 0.842 with 400 training residues to 0.972 with 20,000); (Table A.6). Notably, an untrained model, while achieving random reconstruction loss, still produces an informative invariant latent space (0.629 residue type accuracy), suggesting that the forced SO(3)-invariance grants a "warm start" to the encoder. We do not find significant improvements in latent space classification by training with a variational objective, and present ablation results in Table A.7.

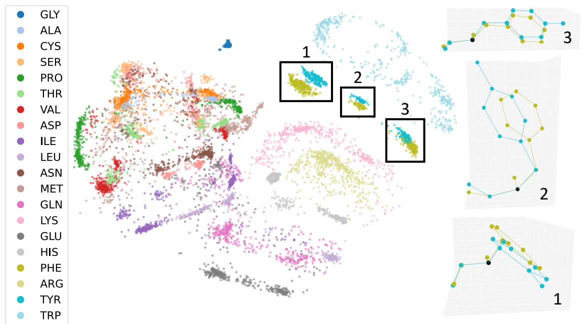

Figure A.2: **H-VAE on amino acids.** H-VAE was trained on 1,000 residues with $\beta = 0.025$ and $z = 2$. The invariant latent space clusters by amino acid conformations. The highlighted clusters for PHE and TYR contain residue pairs with similar conformations; TYR and PHE differ by one oxygen at the end of their benzene rings. We compare conformations by plotting each residue in the standard backbone frame (right); x and y axes are set by the orthonormalized Cα-N and Cα-C vectors, and z axis is their cross product.

A.9 Limitations

We believe the main limitation of the model is its higher inaccuracy in reconstructing features of higher degrees (Fig. A.11), resulting in a potential failure to capture the data's fine-grained details in the latent space. In future work, we plan to quantify the amount of information loss associated with this.

Table A.4: Evaluation of network performances for MNIST-on-the-sphere using a KNN classifier instead of linear classifier in the latent space. Results are significantly better than when using a linear classifier for models with smaller ($z = 16$) latent space, comparable for the other models.

Type	**Method**	**z**	**bw**	**LC Acc.**	**KNN Acc.**
Unsupervised	H-AE NR/R	120	30	0.877	0.875
	H-AE R/R	120	30	0.881	0.886
	H-AE NR/R	16	30	0.820	0.862
	H-AE R/R	16	30	0.833	0.876
	H-VAE NR/R	120	30	0.883	0.879
	H-VAE R/R	120	30	0.884	**0.895**
	H-VAE NR/R	16	30	0.812	0.848
	H-VAE R/R	16	30	0.830	0.874

Table A.5: QEvaluation of network performances for Shrec17 using a KNN classifier instead of linear classifier in the latent space. Results are better than when using a linear classifier.

Type	**Method**	**z**	**bw**	**Class. Acc.**	**P@N**	**R@N**	**F1@N**	**mAP**	**NDCG**
Unsupervised + LC	H-AE	40	90	0.654	0.548	0.569	0.545	0.500	0.597
	H-VAE	40	90	0.631	0.512	0.537	0.512	0.463	0.568
Unsupervised + KNN	H-AE	40	90	**0.672**	**0.560**	**0.572**	**0.555**	**0.501**	**0.599**
	H-VAE	40	90	0.658	0.541	0.558	0.539	0.487	0.591

Table A.6: Quantitative data ablation results on the Toy amino acids dataset. A random-guessing classifier has an expected accuracy of 0.050.

	H-AE				**H-VAE** ($\beta = 0.025$)				**H-VAE** ($\beta = 0.1$)			
# train	**MSE**	**Cosine loss**	**LC Acc.**	**KNN Acc.**	**MSE**	**Cosine loss**	**LC Acc.**	**KNN Acc.**	**MSE**	**Cosine loss**	**LC Acc.**	**KNN Acc.**
0	1.3×10^{-2}	1.015	0.409	0.629	1.5×10^{-2}	0.981	0.424	0.656	1.5×10^{-2}	0.981	0.424	0.656
400	9.4×10^{-4}	0.153	0.586	0.842	9.8×10^{-4}	0.160	0.616	0.848	1.0×10^{-3}	0.163	0.558	0.780
1,000	5.9×10^{-4}	0.099	0.583	0.856	6.3×10^{-4}	0.101	0.569	0.854	6.9×10^{-4}	0.113	0.564	0.844
2,000	4.5×10^{-4}	0.073	0.560	0.900	4.9×10^{-4}	0.081	0.554	0.905	5.5×10^{-4}	0.092	0.593	0.890
5,000	3.3×10^{-4}	0.053	0.629	0.940	3.3×10^{-4}	0.053	0.638	0.961	4.3×10^{-4}	0.072	0.588	0.921
20,000	2.2×10^{-4}	0.034	0.578	0.972	2.4×10^{-4}	0.037	0.667	0.971	2.9×10^{-4}	0.047	0.662	0.966

Table A.7: Quantitative data ablation results on the Toy amino acids dataset. Models were trained on the full dataset (# train = 20,000).

β	**MSE**	**Cosine**	**LC Acc.**	**KNN Acc.**
0 (AE)	2.2×10^{-4}	0.034	0.580	0.972
0.025	2.4×10^{-4}	0.037	0.666	0.971
0.05	2.5×10^{-4}	0.039	0.669	0.968
0.1	2.9×10^{-4}	0.047	0.661	0.966
0.25	6.8×10^{-4}	0.132	0.597	0.854
0.5	1.1×10^{-3}	0.203	0.467	0.722

Table A.8: **Test MSE and Cosine loss for H-(V)AE models trained on MNIST, Shrec17 and Protein Neighborhoods.** MSE and Cosine values are strongly correlated within datasets but not across datasets.

Dataset	**Method**	L	z	**bw**	**# Params**	**MSE**	**Cosine**
MNIST	H-AE NR/R	10	120	30	453k	6.2×10^{-4}	0.017
	H-AE R/R	10	120	30	453k	6.8×10^{-4}	0.018
	H-AE NR/R	10	16	30	227k	9.3×10^{-4}	0.025
	H-AE R/R	10	16	30	227k	8.9×10^{-4}	0.024
	H-VAE NR/R	10	120	30	453k	1.4×10^{-3}	0.037
	H-VAE R/R	10	120	30	453k	1.4×10^{-3}	0.037
	H-VAE NR/R	10	16	30	227k	2.2×10^{-3}	0.057
	H-VAE R/R	10	16	30	227k	2.1×10^{-3}	0.055
Shrec17	H-AE	14	40	90	518k	1.8×10^{-4}	0.130
	H-VAE	14	40	90	518k	2.2×10^{-4}	0.151
Protein NBs	H-AE	3	64	-	631k	3.3×10^{-4}	0.084
	H-AE	4	64	-	950k	3.9×10^{-4}	0.125
	H-AE	6	64	-	1.6M	5.4×10^{-4}	0.219
	H-AE	8	64	-	2.4M	6.3×10^{-4}	0.287
	H-AE	3	128	-	1.2M	2.1×10^{-4}	0.053
	H-AE	4	128	-	1.7M	2.8×10^{-4}	0.087
	H-AE	6	128	-	2.6M	4.0×10^{-4}	0.156
	H-AE	8	128	-	3.8M	4.9×10^{-4}	0.213
	H-AE	3	256	-	2.8M	1.0×10^{-4}	0.025
	H-AE	4	256	-	3.8M	1.9×10^{-4}	0.060
	H-AE	6	256	-	4.4M	2.7×10^{-4}	0.102
	H-AE	8	256	-	7.9M	3.5×10^{-4}	0.148

Table A.9: **Training speed and reconstruction ablations of H-(V)AE models with different Tensor Product rules.** To make comparison fair, models were trained using the same training hyperparameters as described in A.6, and all models were constructed to have comparable number of parameters. Speed was computed as training time and divided by the time of the model using ETP within each dataset. Models with the ETP consistently generates better reconstructions and are usually the fastest. The speed and performance gains of the ETP are most apparent on the Protein Neighborhoods task, where we also note that, as the angular resolution of the data (L) is increased from 6 to 8, the relative speed gain of the ETP over the Full-TP is significantly accentuated (from 2.1x to 3.3x).

Dataset	**Method**	L	z	**TP-type**	$\mathbf{C_{init}}$	**ChannelsList**	**DegreesList**	**# Params**	**Speed**	**MSE**	**Cosine**
MNIST	H-AE	10	16	ETP	None	[16,16,16,16,16,16]	[10,10,8,4,2,1]	227k	**1.0**	$\mathbf{9.3 \times 10^{-4}}$	**0.025**
	H-AE	10	16	Full-TP	None	[7,6,6,6,16]	[10,8,4,2,1]	229k	1.1	1.6×10^{-3}	0.044
	H-AE	10	16	Full-TP	None	[5,5,5,5,7,16]	[10,10,8,4,2,1]	227k	1.7	1.5×10^{-3}	0.041
Shrec17	H-AE	14	40	ETP	12	[12,12,12,20,24,32,40]	[14,14,14,8,4,2,1]	518k	1.0	$\mathbf{1.8 \times 10^{-4}}$	**0.130**
	H-AE	14	40	Full-TP	None	[5,5,5,8,40]	[14,8,4,2,1]	518k	**0.9**	1.9×10^{-4}	0.137
	H-AE	14	40	Full-TP	None	[4,3,3,6,6,6,40]	[14,14,14,8,4,2,1]	513k	1.9	2.0×10^{-4}	0.142
Protein NBs	H-AE	6	64	ETP	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	1.6M	**1.0**	$\mathbf{5.4 \times 10^{-4}}$	**0.219**
	H-AE	6	64	Full-TP	None	[8,8,12,36]	[6,4,2,1]	1.6M	2.1	6.7×10^{-4}	0.286
	H-AE	8	64	ETP	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	2.4M	**1.0**	$\mathbf{6.3 \times 10^{-4}}$	**0.287**
	H-AE	8	64	Full-TP	None	[8,8,12,36]	[6,4,2,1]	2.6M	3.3	7.6×10^{-4}	0.371

Table A.10: **Mean equivariance error for some of our trained H-(V)AE models.** Errors were computed over 2,000 randomly sampled spherical tensors, each with a randomly sampled rotation. Standard deviation is shown alongside the mean. We also show the mean and standard deviation of the absolute value of the output coefficients, to enable contextualization of the measured equivariance error. The equivariance error due to numerical error (absolute difference in coefficients by rotating input vs. output tensor) is consistently three orders of magnitude lower than the typical absolute value of the coefficients, indicating that equivariance is preserved. The same trend occurs for *untrained* models (not shown here for simplicity).

Dataset	**Method**	L	**z**	**# train**	**Equiv. Error**	**Abs. Value**
MNIST	H-AE NR/R	10	16	-	$(2.3 \pm 2.7)\times10^{-4}$	$(1.0 \pm 0.9)\times10^{-1}$
	H-VAE NR/R	10	16	-	$(2.5 \pm 1.9)\times10^{-4}$	$(1.3 \pm 1.8)\times10^{-1}$
Shrec17	H-AE	14	40	-	$(1.4 \pm 2.4)\times10^{-5}$	$(0.4 \pm 1.1)\times10^{-2}$
	H-VAE	14	40	-	$(1.4 \pm 3.5)\times10^{-5}$	$(0.4 \pm 2.2)\times10^{-2}$
Toy Aminoacids	H-AE	4	2	1,000	$(4.7 \pm 3.7)\times10^{-5}$	$(2.6 \pm 4.1)\times10^{-2}$
	H-VAE $\beta = 0.025$	4	2	1,000	$(5.4 \pm 3.9)\times10^{-5}$	$(2.8 \pm 4.0)\times10^{-2}$
	H-AE	4	2	20,000	$(9.3 \pm 6.4)\times10^{-5}$	$(3.5 \pm 4.7)\times10^{-2}$
	H-VAE $\beta = 0.025$	4	2	20,000	$(9.0 \pm 4.7)\times10^{-5}$	$(3.0 \pm 4.3)\times10^{-2}$
Protein NBs	H-AE	3	64	-	$(3.3 \pm 4.2)\times10^{-5}$	$(1.6 \pm 2.1)\times10^{-2}$
	H-AE	4	64	-	$(0.6 \pm 0.6)\times10^{-5}$	$(0.6 \pm 2.1)\times10^{-2}$
	H-AE	6	64	-	$(0.9 \pm 1.0)\times10^{-5}$	$(0.4 \pm 1.3)\times10^{-2}$
	H-AE	8	64	-	$(0.7 \pm 1.0)\times10^{-5}$	$(0.2 \pm 0.9)\times10^{-2}$
	H-AE	3	128	-	$(1.1 \pm 1.3)\times10^{-5}$	$(1.0 \pm 3.0)\times10^{-2}$
	H-AE	4	128	-	$(0.6 \pm 0.7)\times10^{-5}$	$(0.6 \pm 2.5)\times10^{-2}$
	H-AE	6	128	-	$(0.1 \pm 0.1)\times10^{-5}$	$(0.2 \pm 1.7)\times10^{-2}$
	H-AE	8	128	-	$(0.1 \pm 0.1)\times10^{-5}$	$(0.1 \pm 1.0)\times10^{-2}$
	H-AE	3	256	-	$(0.6 \pm 0.4)\times10^{-5}$	$(0.6 \pm 2.0)\times10^{-2}$
	H-AE	4	256	-	$(0.4 \pm 0.6)\times10^{-5}$	$(0.6 \pm 2.2)\times10^{-2}$
	H-AE	6	256	-	$(0.1 \pm 0.1)\times10^{-5}$	$(0.2 \pm 1.6)\times10^{-2}$
	H-AE	8	256	-	$(0.09 \pm 0.07)\times10^{-5}$	$(0.1 \pm 1.3)\times10^{-2}$

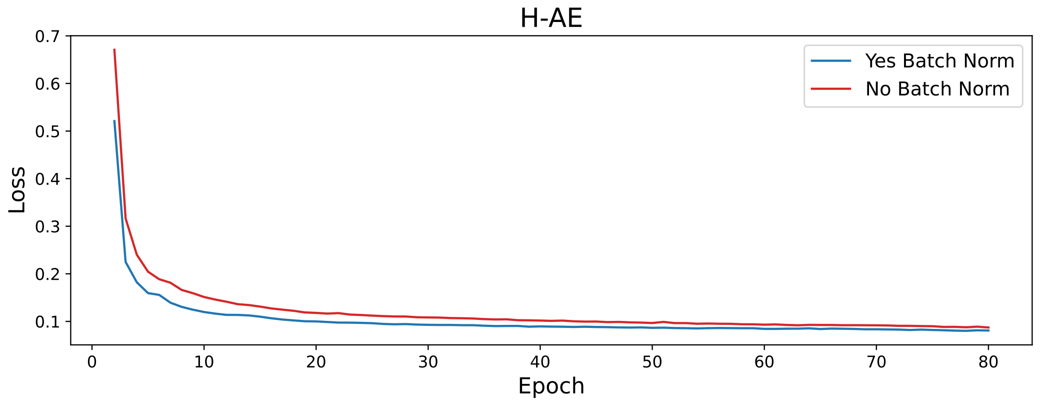

Figure A.3: **Training loss trace of H-AE, with and without Batch Norm, on MNIST-on-the-sphere.** Models were trained with the (NR/R; z = 16; AE) specification. The loss on validation data follows the same trend, but it is not shown for simplicity.

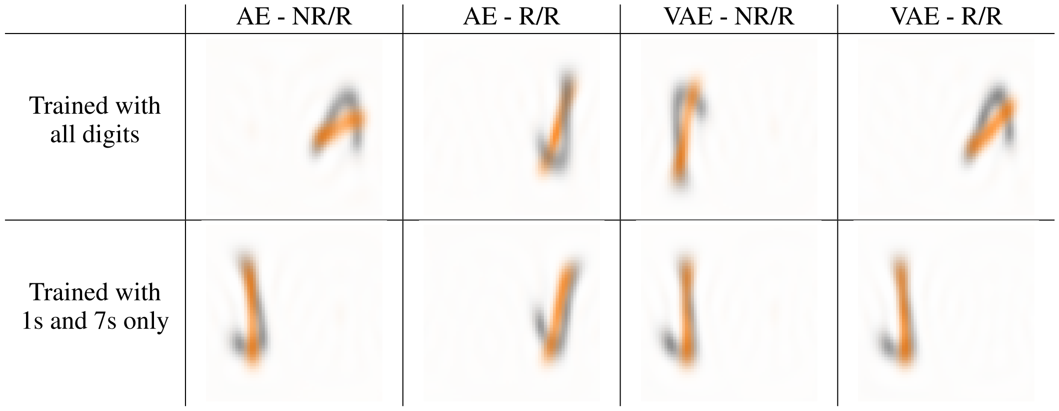

Figure A.4: **Empirical tests to show H-(V)AE learns a canonical frame in the MNIST-on-the-sphere.** For each of the 4 models with $z = 16$, we train a version using only images containing 1s and 7s. For each of the resulting 8 models, we visualize the sum of training images of digits 1 and 7, when rotated to the canonical frame. To achieve the canonical frame we rotate each image such that the frame learned by the encoder is the same for all images, and specifically, we make it correspond to the 3×3 identity matrix. We compute the sums of images with the same digit, and overlay them with different colors for ease of visualization. We test the hypothesis as whether H-(V)AE learns frames that align the training images such that they maximally overlap; we do so in two ways.
First, if the hypothesis were true, all canonical images of the same digit should maximally or near-maximally overlap - since they have very similar shape - and thus, their overlays would look like a "smooth" version of that digit. Indeed, we find this statement to be true for all models irrespective of their training strategy.
Second, we consider the alignment of images of different digits. We take 1s and 7s as examples given their similarity in shape. If the hypothesis were true, models trained with only 1s and 7s should align canonical 1s along the long side of canonical 7s; indeed we find this to be consistently the case. The same alignment between 1s and 7s, however, does not necessarily hold for models trained with all digits. This is because maximizing overlap across a set of diverse shapes does not necessarily maximize the overlap within any independent pair of such shapes. Indeed, we find that canonical 1s and canonical 7s do not overlap optimally with each other for models trained with all digits.
We note that these tests do not provide proof, but rather empirical evidence of the characteristics of frames learned by H-(V)AE on the MNIST-on-the-sphere task.

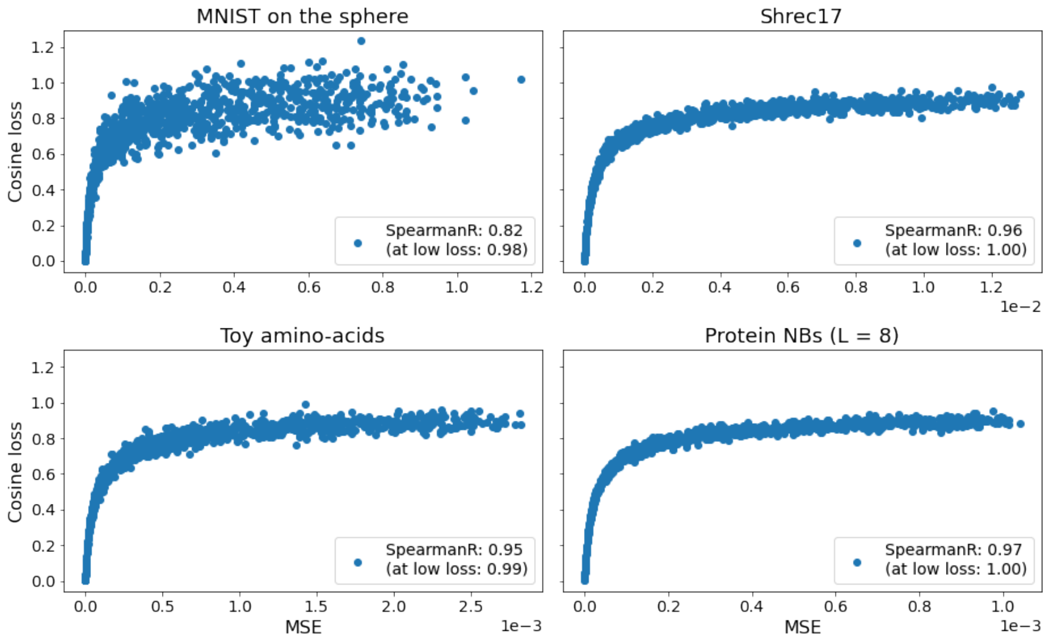

Figure A.5: **Correlation between Cosine loss and MSE values between pairs of random tensors**. For each dataset, we sample a batch of $N = 1000$ tensors with dataset-specific feature degrees and channel sizes, where each coefficient is sampled from a normal distribution. We mimic the normalization step performed in the real experiment and normalize each tensor by the average total norm of the batch. We then generate a "noisy" version of each tensor by adding (normalized) Gaussian noise to each coefficient with standard deviation sampled from a uniform distribution between 0 and some maximum noise level (10 in these plots). This procedure results in N pairs of tensors with varying degrees of similarity between them. We compute the MSE and Cosine loss for all N pairs of tensors and visualize them. The two loss values are well correlated in rank as measured by Spearman Correlation. The correlation is significantly stronger in the regime of reconstruction loss below a Cosine loss of 0.5 (SpearmanR ~ 0.99), a value well above the maximum Cosine loss achieved by H-(V)AE in all our experiments. All the p-values for the Spearman Correlations shown in the plot are significant (< 0.05).

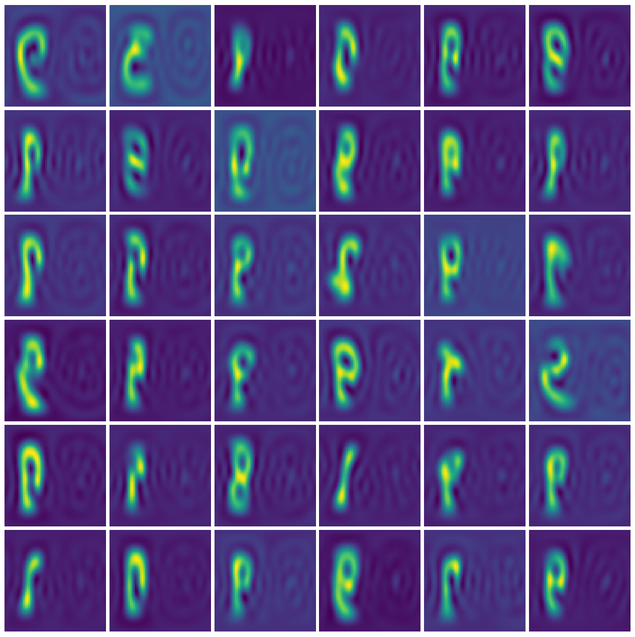

Figure A.6: **Random samples generated by the (NR/R; z = 16; VAE) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal) and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. The samples show a wide range of diversity in digit and style.

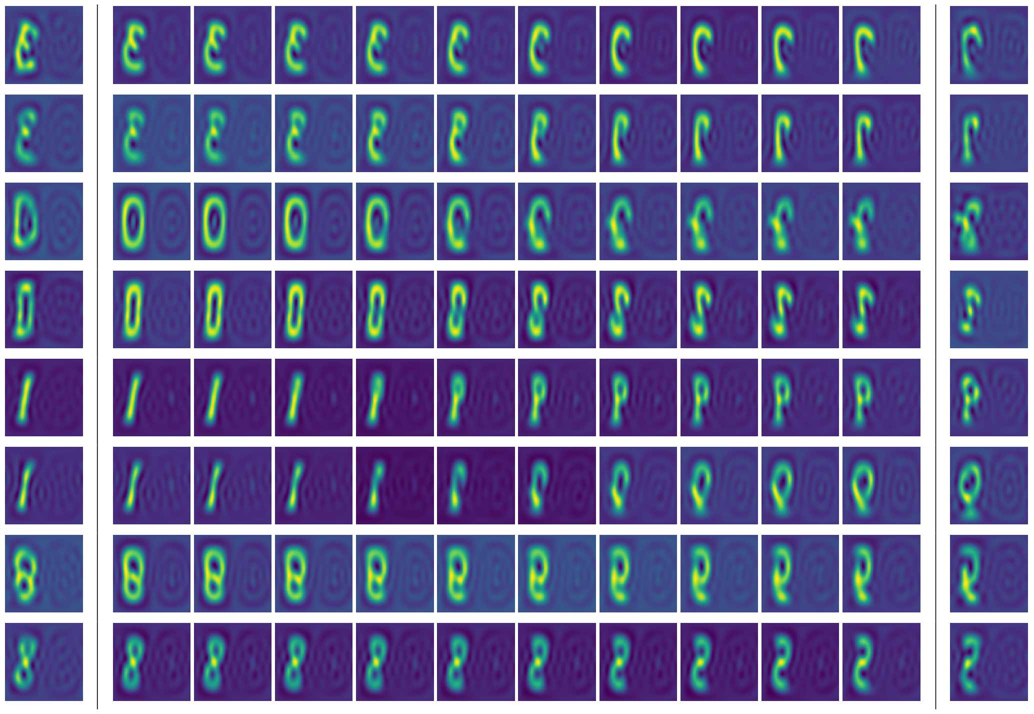

Figure A.7: **Trajectories across the latent space for the (NR/R; z = 16; VAE) MNIST-on-the-sphere model.** We compute pairs of invariant latent embeddings using the model's encoder, and linearly interpolate between them through the latent space. We then feed the interpolated embeddings into the decoder, together with the canonical frame, and compute the inverse SFT to get the image in real space. The left and right columns show the original images (after forward and inverse SFT) rotated to be placed in the learned canonical frame, whereas the center rows show the interpolated images. We can see that all trajectories are smooth, respecting spatial consistency, sign of a well structured latent space.

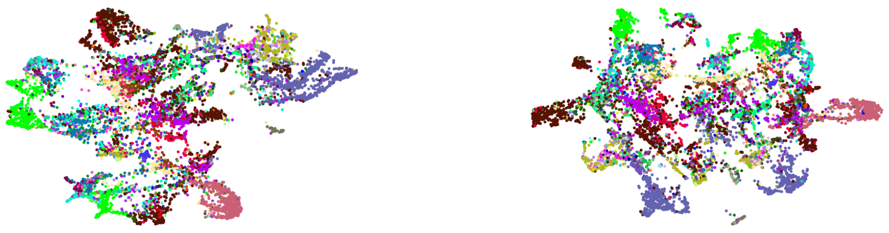

Figure A.8: **2D visualization via UMAP of the invariant latent embeddings of Shrec17 test data learned by H-(V)AE.** Left: H-AE, Right: H-VAE. Points are colored by class (55 classes).

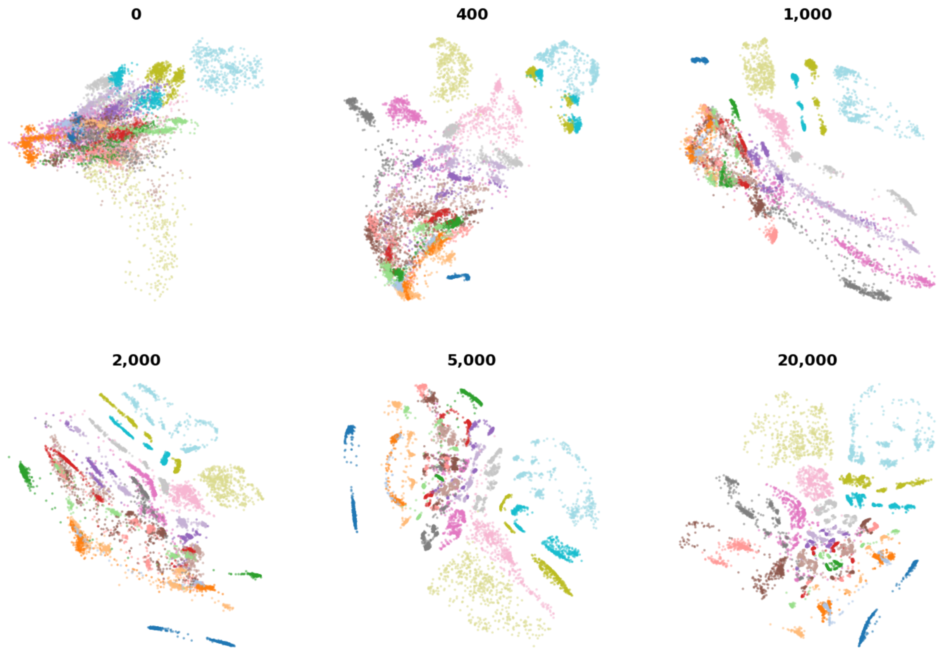

Figure A.9: **Amino acid latent space learned by H-AE.** Visualization of the test data's invariant latent space learned by H-AE trained with varying amounts of the training data. As more training data is added, the separation of clusters containing residues with most similar conformations becomes more distinct. Notably, even with no training data, conformation clusters can be identified.

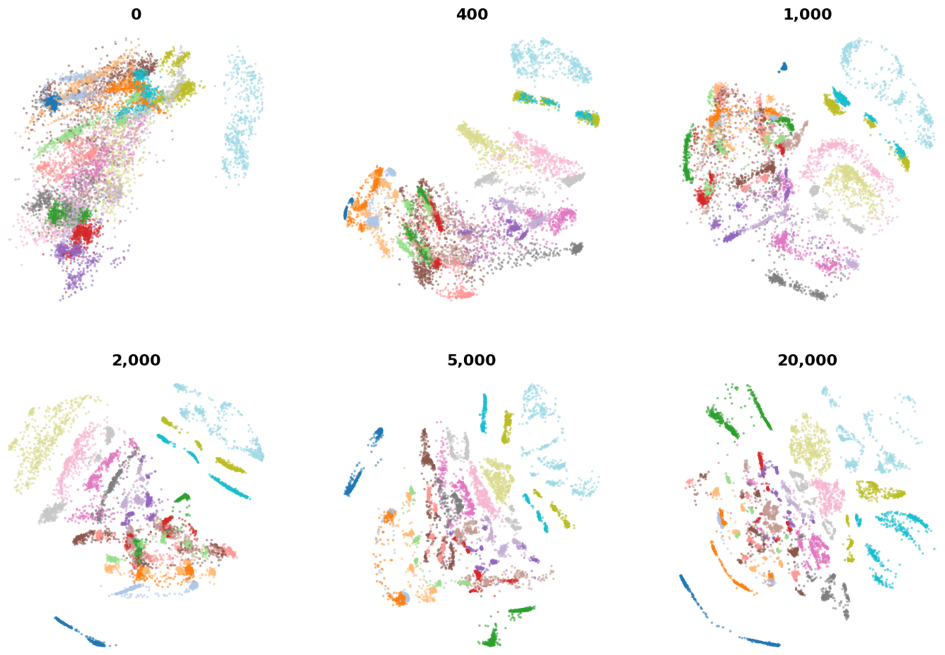

Figure A.10: **Amino acid latent space learned by H-VAE.** Visualization of the test data's invariant latent space learned by H-VAE ($\beta = 0.025$) trained with varying amounts of training data.

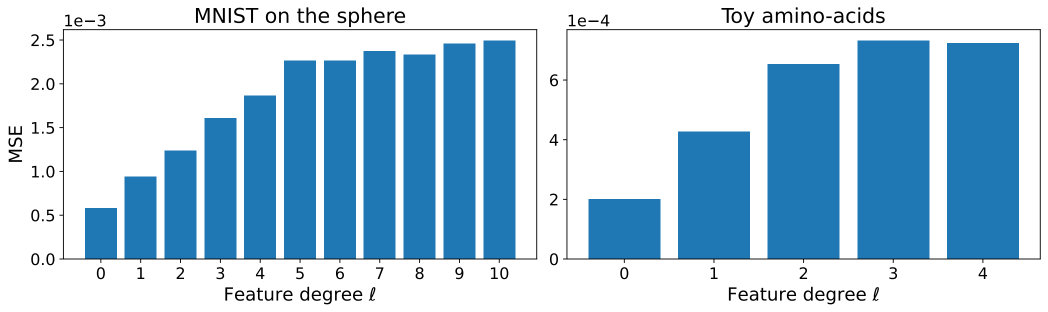

Figure A.11: **Reconstruction loss as a function of feature degree ℓ.** Test reconstruction loss (MSE) of H-VAE split by feature degree ℓ, for the MNIST-on-the-sphere (left) and Toy amino acids dataset (right). In both cases, features of larger degrees are harder to reconstruct accurately. The increase in loss is more steep for smaller degrees.

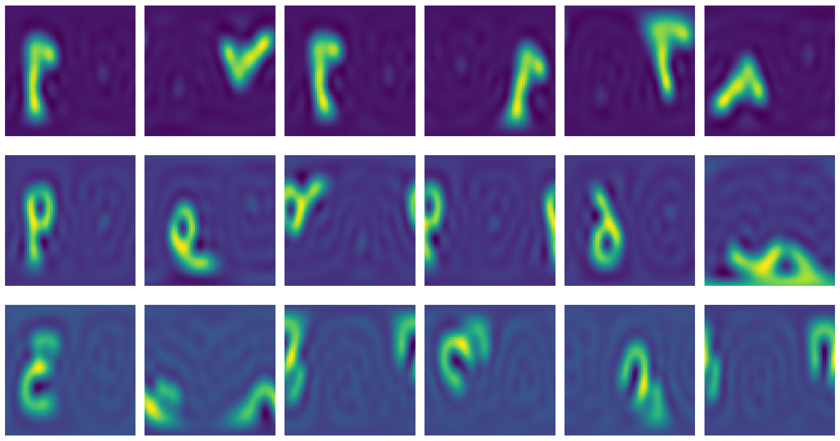

Figure A.12: **Visual proof of the disentanglement in the latent space of MNIST-on-the-sphere.** For each row, the invariant embedding $\mathbf{z}$ is held fixed, and a different frame (i.e., the rotation matrix) is used. Frames are sampled randomly and differ across rows, with the exception of the first column, which is always the identity frame. Then, $\mathbf{z}$ and the frame are fed to the decoder and the Inverse Fourier Transform is used to generate the reconstructed spherical image, which is projected onto a plane for the ease of visualization. Modulo the distortions given by projecting the image onto a plane, it is clear that the invariant embedding contains all semantic information, and the frame solely determines the orientation of the image.