Holographic-(V)AE: an end-to-end SO(3)-Equivariant (Variational) Autoencoder in Fourier Space

Gian Marco Visani*
Paul G. Allen School of Computer Science and Engineering, University of Washington, 85 E Stevens Way NE, Seattle, WA 98195, USA

Michael N. Pun
Department of Physics, University of Washington, 3910 15th Avenue Northeast, Seattle, WA 98195, USA

Arman Angaji
Institute for Biological Physics, University of Cologne, Zülpicher Str. 77, 50937 Cologne, Germany

Armita Nourmohammad*
Department of Physics, University of Washington, 3910 15th Avenue Northeast, Seattle, WA 98195, USA
Paul G. Allen School of Computer Science and Engineering, University of Washington, 85 E Stevens Way NE, Seattle, WA 98195, USA
Department of Applied Mathematics, University of Washington, 4182 W Stevens Way NE, Seattle, WA 98105, USA and
Fred Hutchinson Cancer Center, 1241 Eastlake Ave E, Seattle, WA 98102, USA

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*-(Variational) Auto Encoder (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 in 3D. H-(V)AE is trained to reconstruct the spherical Fourier encoding of data, learning in the process a low-dimensional representation of the data (i.e., a latent space) with a maximally informative rotationally invariant embedding alongside an equivariant frame describing the orientation of the data. We extensively test the performance of H-(V)AE on diverse datasets. We show that the learned latent space efficiently encodes the categorical features of spherical images. Moreover, H-(V)AE's latent space can be used to extract compact embeddings for protein structure microenvironments, and when paired with a Random Forest Regressor, it enables state-of-the-art predictions of protein-ligand binding affinity.

I. INTRODUCTION

In supervised learning, e.g. for classification tasks, 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 should be translationally invariant, in that the output should not depend on the position of the object in the image. 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 [1]). Alternatively, one could use invariant features of the data (e.g. pairwise distance between different features) as input to train any machine learning algorithm [2]. However, the choice of these invariants is arbitrary and the resulting network could lack in expressiveness.

Recent advances have brought concepts from group theory to develop symmetry-aware neural network architectures that are equivariant under actions of different symmetry groups [3–12]. Equivariance with respect to a symmetry group is the property that, if the input is transformed via a group action, then the output is transformed according to a linear operation determined by the symmetry group itself; it is easy to see that invariance is a special case of equivariance, where the linear operation is simply the identity. These group equivariant networks can systematically treat and interpret various transformation in data, and learn models that are agnostic to the specified transformations. For example, models equivariant to euclidean transformations have recently advanced the state-of-the-art on many supervised tasks for classification and regression [3–12]. These models are more flexible and data-efficient compared to their purely in-

* Correspondence should be addressed to: Gian Marco Visani: gvisan01@cs.washington.edu, and Armita Nourmohammad: armita@uw.com.

variant counterparts [1, 13].

Extending such group invariant and equivariant paradigms to unsupervised learning (i.e., for modeling the data distribution) could provide compact representations of data that are agnostic to a specified symmetry transformation. In machine learning, auto-encoders (AE's) and their probabilistic version, variational auto-encoders (VAE's), are among the artificial neural networks that are commonly used for unsupervised learning, in that they provide an efficient representation of unlabeled data [14, 15]. However, little work has been done to implement group equivariance in auto-encoder architectures [16].

Here, we focus on developing neural network architectures for unsupervised learning that are equivariant to rotations around a specified origin in 3D, denoted by the group SO(3). To define rotationally equivariant transformations, it is convenient to project data to spherical Fourier space [6]. Accordingly, we encode the data in spherical Fourier space by constructing *holograms* of the data that are conveniently structured for equivariant operations. These data holograms are inputs to our 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 (or 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 and properties of H-(V)AE on two domains. First, we focus on spherical images, demonstrating high accuracy in unsupervised classification and clustering tasks. Second, we focus on structural biology, and demonstrate that H-(V)AE can be effectively used to construct compact, informative, and symmetry-aware representations of protein structures, which can be used for downstream tasks. Specifically, we leverage H-(V)AE trained on a large corpus of protein structure micro-environments to construct local representations of protein-ligand binding pockets that are both rotationally and translationally equivariant (i.e., SE(3) equivariant). When combined with a simple Random Forest Regressor, we achieve state-of-the-art accuracy on the task of predicting the binding affinity between a protein and a ligand in complex. Our code and pre-trained models are available at https://github.com/gvisani/Holographic-VAE.

II. MODEL

A. Representation and transformation of 3D data in spherical bases

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 [17]. We consider functions distributed around a specified origin, which we express by the resulting spherical coordinates (r, θ, ϕ) around the origin; θ and ϕ are the azimuthal and the polar angles and r defines the distance to the reference point in the spherical coordinate system. 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).

To define rotationally equivariant transformations, it is convenient to project data to spherical Fourier space. We use spherical harmonics to encode the angular information of the data. Spherical harmonics are a class of functions that form a complete and orthonormal basis for functions $f(\theta, \phi)$ defined on a unit sphere ($r = 1$). In their complex form, spherical harmonics are defined as,

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

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 quantum mechanics, spherical harmonics are used to represent the orbital angular momenta, e.g. for an electron in a hydrogen atom. In this context, the degree ℓ relates to the eigenvalue of the square of the angular momentum, and the order m is the eigenvalue of the angular momentum about the azimuthal axis.

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

$$\hat{Z}_{\ell m}^n = \int \rho(r, \theta, \phi)\, Y_{\ell m}(\theta, \phi) R_\ell^n(r)\, \mathrm{d}V \tag{2}$$

where $Y_{\ell m}(\theta, \phi)$ is the spherical harmonics of degree ℓ and order m, and $R_\ell^n(r)$ is the radial Zernike polynomial in 3D (Eq. A.7) with radial frequency $n \geq 0$ and degree ℓ. $R_\ell^n(r)$ 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 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.

A class of functions that we consider in this work are 3D point clouds, e.g., the atomic composition of a protein in space. We represent point clouds $\rho(\boldsymbol{r}) \equiv \rho(r, \theta, \phi)$ by the sum of Dirac-δ functions centered at each point:

$$\rho(\boldsymbol{r}) = \sum_{i \in \text{points}} \delta(\rho(\boldsymbol{r}_i) - \rho(\boldsymbol{r})) \tag{3}$$

where $\delta(x) = 1$ for $x = 0$ and it is zero, otherwise. The resulting ZFT of a point cloud follows a closed form, and

notably, it does not require a discretization of 3D space for numerical computation,

$$\hat{Z}^n_{\ell m} = \sum_{i \in \text{points}} R^\ell_n(r_i) Y_{\ell m}(\theta_i, \varphi_i) \tag{4}$$

We can reconstruct the data using the inverse ZFT and define approximations by truncating the angular and radial frequencies at L and N (see Section A 1).

Conveniently, the angular representation of the data by spherical harmonics in a ZFT transform form an equivariant basis under rotation in 3D, implying that if the input (i.e., atomic coordinates of a protein) is rotated, then the output is transformed according to a linear operation determined by the rotation. The linear operator that describes how spherical harmonics transform under rotations are called the Wigner D-matrices. Notably, Wigner-D matrices are the irreducible representations (irreps) of SO(3). Therefore, 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. For example, for 3D atomic point clouds, these features include the identity and chemical properties of the constituent atoms. Throughout the paper, we refer to generic steerable tensors as $\boldsymbol{h}$ and index them by ℓ (degree of $Y_{\ell m}$), m (order of $Y_{\ell m}$) and c (channel type). We adopt the "hat" notation for individual entries (e.g. $\hat{h}_{\ell m}$) to remind ourselves of the analogy with Fourier coefficients; see Figure 1A for a graphical illustration of a tensor.

One key feature of neural networks is applying nonlinear activations, which enable a network to approximately model complex and non-linear phenomena. Commonly used nonlinearities include reLU, tanh, and softmax functions. However, these conventional non-linearities can break rotational equivariance in the Fourier space. To construct expressive rotationally equivariant neural networks we can use the Clebsch-Gordan (CG) tensor product $\otimes_{cg}$, which is the natural nonlinear (more specifically, bilinear in the case of using two sets of Fourier coefficients) operation in the space of spherical harmonics [18].

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 tensor, with individual components $\hat{h}_{\ell m}$ for $-\ell \leq m \leq \ell$. The CG tensor product is given by,

$$\begin{aligned} \hat{h}_{\ell_3 m_3} &= (\boldsymbol{h}_{\ell_1} \otimes_{cg} \boldsymbol{h}_{\ell_2})_{\ell_3 m_3} \\ &= \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C^{(\ell_3 m_3)}_{(\ell_1 m_1)(\ell_2 m_2)} \hat{h}_{\ell_1 m_1} \hat{h}_{\ell_2 m_2} \end{aligned} \tag{5}$$

where $C^{(\ell_3 m_3)}_{(\ell_1 m_1)(\ell_2 m_2)}$ are the Clebsch-Gordan coefficients, and can be precomputed for all degrees of spherical tensors [18]. Similar to spherical harmonics, Clebsch-Gordan tensor products also appear in quantum mechanics, and they are used to express couplings between angular momenta. In following with recent work on group-equivariant machine learning [6], we will use Clebsch-Gordan products to express nonlinearities in 3D rotationally equivariant neural networks for protein structures.

III. HOLOGRAPHIC-(V)AE (H-(V)AE)

Network architecture and training. H-(V)AE consists of an encoder that, through learned linear projections and pre-set nonlinear operations, project the data onto a compressed rotationally equivariant latent space. A trained decoder that is similarly constructed then takes this latent projection and reconstructs the input data. The combination of leaned linear and pre-set nonlinear operations form equivariant Clebsch-Gordan blocks (CG bl.) both for the encoder and the decoder; see Figure 1 and below for details on the structure of a Clebsch-Gordan block.

Using the Clebsch-Gordan blocks, we construct a fully rotationally equivariant architecture for unsupervised learning. Specifically, the encoder takes as input a steerable tensor with maximum degree $\ell_{\max} = L$ 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 $\ell_{\max} = 1$, resulting in the invariant ($\ell = 0$) and the frame-defining equivariant ($\ell = 1$) embeddings. The frame is constructed by learning two vectors from the $\ell = 1$ embedding in the final layer of the encoder and using Gram-Schmidt to find the corresponding orthonormal basis [19]. The third orthonomal basis vector is then calculated as the cross product of the first two.

The decoder learns to reconstruct the input from the invariant ($\ell = 0$) embedding of the encoder's final layer $\mathbf{z}$ and the frame, iteratively increasing the maximum degree $\ell_{\max}$ of the intermediate representations by leveraging the CG Tensor Product within the Clebsch-Gordan blocks (Figure 1C). We refer the reader to Section A 2 e for further details on the design choices of the network.

To add stochasticity and make the model variational (i.e., constructing H-VAE as opposed to H-AE), 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

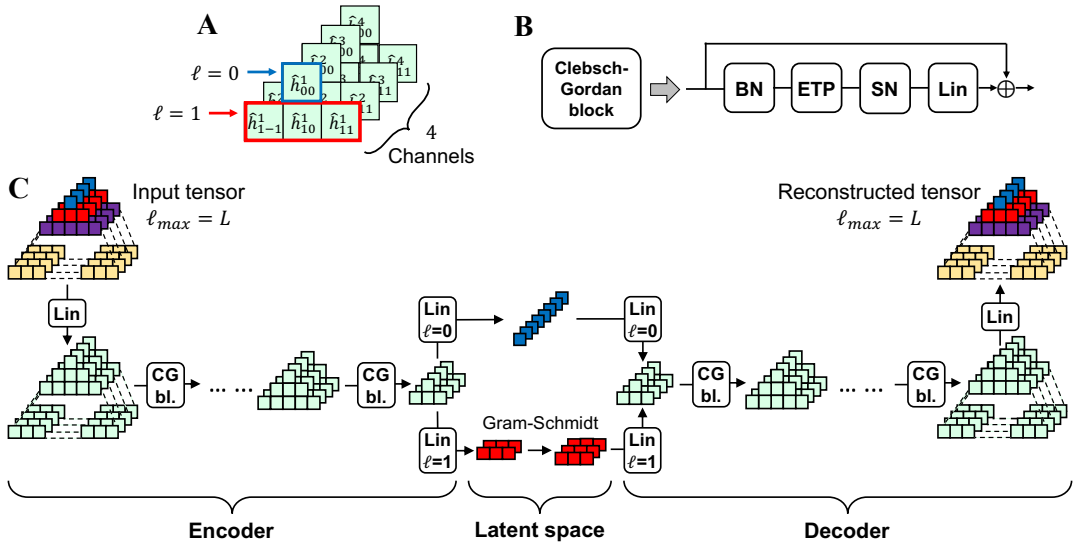

FIG. 1. **Schematic of the Network architecture.** **A:** Schematic of a steerable tensor with $\ell_{\max} = 1$ and 4 channels per feature degree. We choose a pyramidal representation that naturally follows the expansion in size of features of higher degree. **B:** Schematic of a Clebsch-Gordan Block (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).

deviations.

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 Kullback-Leibler divergence D_{KL} of the posterior invariant latent space distribution $q(\boldsymbol{z}|\boldsymbol{x})$ from the selected prior $p(\boldsymbol{z})$ [20]:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = \alpha \mathcal{L}_{\text{rec}}(\boldsymbol{x}, \boldsymbol{x}') + \beta D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{6}$$

We use mean square error (MSE) for $\mathcal{L}_{\text{rec}}$, which as we show in Section A 2 h, 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 regularization [21]; see Section A 2 f for details on tuning of these rates during training.

As a result of this training, H-(V)AE 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; see Figure S1 for a visual example. We discuss the equivariant properties of H-(V)AE in more detail in Figure 3, and empirically verify the equivariance of our model up to numerical errors in Table A.9.

Architecture of a Clebsch-Gordan block. Each Clebsch-Gordan block (CG bl.) consists of a trained linear layer (*Linearity* (Lin)), an efficient tensor product (ETP) to inject nonlinearity in the network, and normalization steps by (*Batch Norm* (BN)) and (*Signal Norm* (SN)) to respectively speed-up convergence and stabilize training.

Linearity (Lin). A linear layer acts on steerable tensors by learning degree-specific linear operations. Linear layers are trained in that 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. Specifically, let us consider a vector $\mathbf{h}_\ell$, containing features of the same degree ℓ. We train the network to learn the weight matrix $\mathbf{W}_\ell$ to map $\mathbf{h}_\ell$ to $\mathbf{h}'_\ell$ between the network's layers, i.e., $\mathbf{h}'_\ell = \mathbf{W}_\ell \mathbf{h}_\ell$ (see Section A 2 a for details).

Nonlinearity with efficient tensor product (ETP). One key feature of neural networks is applying nonlinear activations, which enable a network to approximately model complex and non-linear phenomena. As noted above, we use the Clebsch-Gordan tensor product to inject rotationally equivariant nonlinearities in the network. Specifically, within a Clebsch-Gordan block the output of a linear layer is acted upon by bi-linear CG tensor product, as was originally prescribed by in ref. [6] for SO(3)-equivariant convolutional neural networks. This bilinear operation enables information flow between features of

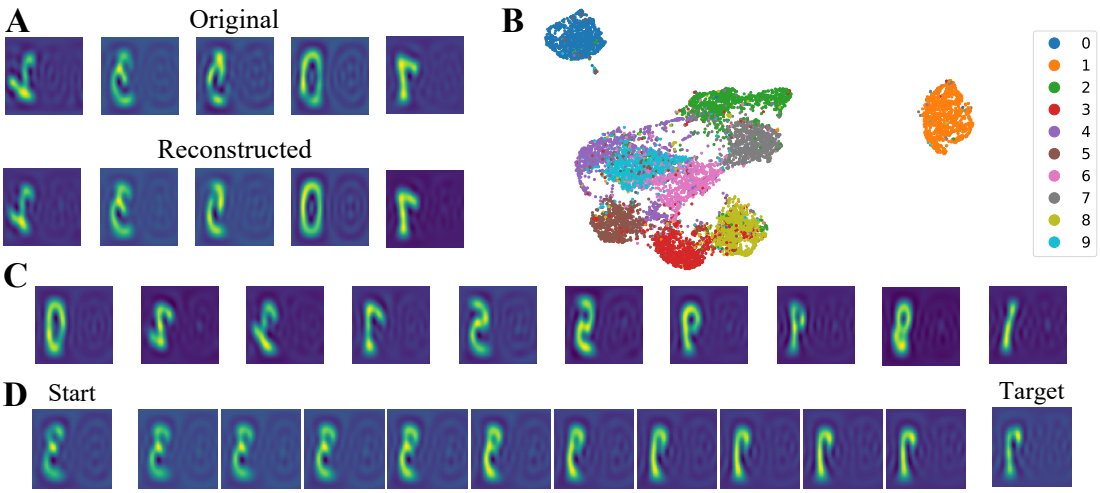

FIG. 2. **H-VAE on MNIST-on-the-sphere.** Evaluation on rotated digits for an H-VAE trained on non-rotated digits with $z = 16$. **(A)** Original and reconstructed images in the canonical frame after inverse transform from Fourier space. The images are projected onto a plane. Distortions at the edges and flipping are side-effects of the projection. **(B)** visualization of the latent space via 2D UMAP [22]. Data points are colored by digit identity. **(C)** 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. Interpolated invariant embeddings are fed to the decoder alongside the canonical frame. MNIST-on-the-sphere dataset is created by projecting data from the planar MNIST on a discrete unit sphere, using the Driscoll-Healey (DH) method with a bandwidth (bw) of 30 [4].

different degrees, which is necessary for constructing expressive models, and for transferring higher-ℓ information to $\ell = 0$ in H-(V)AE's invariant encoder, and back in the decoder.

To significantly reduce the computational and memory costs of the tensor products, we perform *efficient tensor products* (ETPs) by leveraging some of the modifications proposed in ref. [23]. 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. 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 b for details, and Table A.8 for an ablation study showing the improvement in parameter efficiency provided by the ETP.

Batch and signal Norm. We normalize intermediate tensor representations degree-wise and channel-wise by the batch-averaged norms of the features, as initially proposed in ref. [6]; see Figure 1B, Section A 2 c, and Fig. S2 for details. 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 the square root of its *total* norm, defined as the sum of the norms of each of the tensor's features, and apply a degree-specific affine transformation for added flexibility; see Section A 2 d for mathematical details. Signal Norm can be seen as a form of the classic Layer Normalization that respects SO(3) equivariance [24].

IV. RESULTS

A. Rotated MNIST on the sphere

We extensively test the performance of H-(V)AE on the MNIST-on-the-sphere dataset [25]. Following ref. [4], we project the MNIST dataset, which includes images of handwritten numbers, onto the lower hemisphere of a discrete unit sphere. We consider two variants of training/test set splits, NR/R and R/R, differing in whether the training/test images have been randomly rotated (R) or not (NR). For each dataset, we map the images to steerable tensors via the Zernike Fourier Transform (ZFT) in eq. 2 and train models with different different sizes of latent spaces and model types (AE vs. VAE). In all cases the model architecture follows from Fig. 1C; see Section A 5 b for details.

We use the metric *Cosine loss* to measure a model's reconstruction ability. Cosine loss 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. Importantly, unlike MSE, Cosine loss is dimensionless, and therefore, comparable across different datasets and encodings of data in tensors of different sizes (network hyperparameters); see Section A 2 j for details.

All trained models achieve very low reconstruction Cosine loss (Table I) 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 (since they do not need to

find a trade-off between reconstruction error and KL divergence, Eq. 6), and so do models with a larger latent space. Nonetheless, H-VAE achieves reliable reconstructions, as shown in Figure 2A and Table I.

Interpretability of the latent space is also an important feature of a model. 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. 2B, and quantitatively by clustering the data via K-Means with 10 centroids and computing standard clustering metrics of Purity [26] and V-measure [27] in Table I. Crucially, the built-in SO(3)-equivariance enables models trained on the non-rotated images to seamlessly generalize to images that have been randomly rotated, as seen by the equivalent performance between models trained and evaluated on the NR/R and R/R splits (Table I).

All trained models achieve much better clustering metrics than Rot-Inv AE [28], with the VAE models consistently outperforming the 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 K-nearest neighbor (KNN) classifier instead of LC further improves performance (Table A.3).

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. 2C and Fig. S3). 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. 2D and Fig. S4), which is a sign of a semantically well-structured latent space.

To understand the meaning of the learned frames, we ask ourselves what the output of the decoder looks like if the frame is held constant; for simplicity, we set it equal to the 3×3 identity matrix. We 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 Figure 3. We call this frame the **canonical** frame. 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; in fact in our experiments, when visualizing reconstructed or sampled MNIST images, we first rotate them to the canonical frame for ease of visualization.

B. Shrec17

The Shrec17 dataset consists of 51k colorless 3D models belonging to 55 object classes, with a 70/10/20 train/valid/test split [31]. We use the variant of the dataset in which each object is randomly rotated. Converting 3D shapes into spherical images preserves topological surface information, while significantly simplifying the representation. We follow ref. [4] and project surface information for each image onto an enclosing spherical grid via a ray-casting scheme and apply ZFT (eq. 2) on these transformed images.

We train an AE and a VAE model on ZFT transformed data (Section A 5 c); Fig. S5 shows the resulting latent embeddings for both H-AE and H-VAE on this dataset. Similarly to the MNIST dataset, we compute cosine loss, clustering metrics and classification accuracy via a linear classifier. We also compute the standard Shrec17 object retrieval metrics via the latent space linear classifier's predictions (see [31] for a description of the retrieval metrics). 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 I). Using KNN classification instead of a linear classifier further improves performance (Table A.4). 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.

C. Structural embeddings of amino acid neighborhoods to predict function

Here, we provide a strong use case for H-(V)AE in structural biology. Specifically, we learn expressive embeddings for amino acids neighborhoods within protein structures that can be used to learn protein function.

Embeddings of amino acid conformations. As a first, propedeutic step, we train H-(V)AE to reconstruct the 3D structure of individual amino acids, represented as atomic point clouds, extracted from protein structures in the Protein Data Bank (PDB) [32]. Residues of the same type have different conformations and naturally have noisy coordinates, making this problem a natural benchmark for our rotationally equivariant method.

We represent an amino acid by atom-type-specific clouds (C, O, N and S; we exclude H) centered at the residue's Cα and compute the ZFT (eq. 2) with $L = 4$ and $N = 20$ within a radius of 10Å from the residue's Cα, and concatenate features of the same degree ℓ, resulting in a tensor with 940 coefficients. We train several H-AE and H-VAE with different architectures, but all with the latent space sizes $z = 2$; see Sec. A 5 d for details.

We consistently find that the latent space clusters by amino acid conformations (Figure 4A), with sharper cluster separations as more training data is added (Figure S6 and S7). 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

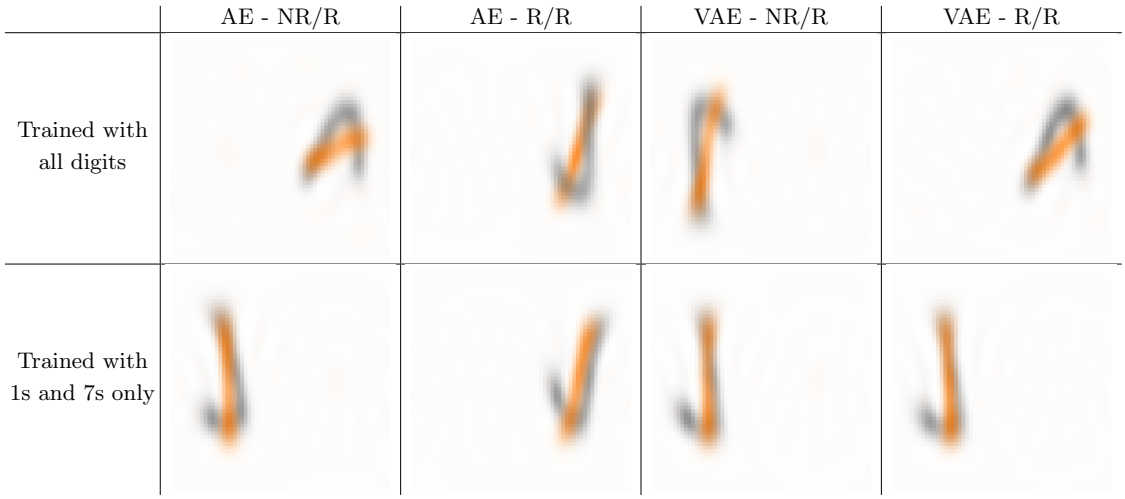

FIG. 3. **H-(V)AE implicitly learns to maximally overlap training images on MNIST-on-the-sphere.** For each of the 4 models with $z = 16$, we train a version using only images containing 1s and 7s. For each of the resulting 8 models, we visualize the sum of training images of digits 1 and 7, when rotated to the canonical frame. 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 a formal proof, but rather empirical evidence of the characteristics of frames learned by H-(V)AE on the MNIST-on-the-sphere task.

20,000); Table A.5. 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.5). 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.6.

Embeddings of amino acid structure neighborhoods. The structural neighborhood surrounding an amino acid provides a context for its function (e.g., whether it takes part in interaction with other proteins at a protein-protein interface or not). Indeed, our previous work has shown that supervised learning algorithms can accurately classify focal amino acids based on the composition of their surrounding neighborhoods [33]. Here, we train H-(V)AE to reconstruct residue-specific protein *neighborhoods* - which we define as the point clouds of atoms within 10Å of a residue's Cα - across the protein universe (Fig. 4A). We extract these protein neighborhoods from all proteins in ProteinNet [34]. We then construct each neighborhood's Fourier representation by computing the ZFT (Eq. 2) over the point clouds associated with each atom type within the neighrbohood (C, N, O and S) and concatenating atom-specific features of the same degree ℓ.

We train several H-AE models with varying architectures with different maximum spherical degree L and latent space sizes z (see details in Sec. A 5 e); note that we do not experiment with variational models for this task. H-AE shows strong reconstruction ability, but its accuracy worsens with smaller latent space sizes and higher maximum spherical degree L (Table A.7). Notably, the learned latent space is structured according to the secondary structure of the neighborhood's focal amino acid (Fig. 4B).

Predicting protein-Ligand Binding Affinity (LBA). The binding interaction between proteins and ligands should be primarily determined by the composition of the protein's binding pocket in complex with the ligand. Therefore, we hypothesized that the inferred protein structure embeddings (Fig. 4B) should contain information about protein-ligand binding interactions, if the neighborhood is defined along the binding pocket and contains atoms from the ligand. To test this hypothe-

Dataset	Type	Method	z	bw	Cosine	Purity	V-meas.	Class. Acc.	P@N	R@N	F1@N	mAP	NDCG
MNIST	Supervised	[23] NR/R	-	30	-	-	-	0.993	-	-	-	-	-
	Unsupervised	[28] 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	[7]	-	128	-	-	-	-	0.717	0.737	-	0.685	-
		[23]	-	128	-	-	-	-	0.719	0.710	0.708	0.679	0.758
	Unsupervised	[28]	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

TABLE I. **Performance metrics on MNIST-on-the-sphere and Shrec17**. Reconstruction Cosine loss, clustering metrics (Purity and V-measure), classification accuracy in the latent space using a linear classifier, and retrieval metrics (Shrec17 only) are shown. For MNIST, we follow ref. [4] to create the MNIST-on-the-sphere dataset by projecting data from the planar MNIST on a discrete unit sphere, using the Driscoll-Healey method with a bandwidth (bw) of 30. We then 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 different sizes of latent spaces z (16 vs. 120) and model types (AE vs. VAE). For Shrec17, we follow ref. [4] and project surface information from each model onto an enclosing Driscoll-Healey 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 only report scores presented in the corresponding papers, and only the best-performing supervised method from the literature; see Fig. S5 for visualization of the latent embeddings for the Shrec17 dataset.

sis, we follow the pipeline in Fig. 4C. Specifically, given a protein-ligand structure complex, we identify residues in the binding pocket (i.e., residues with C-α 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 rotationally-invariant embeddings. We highlight that, since the neighborhood centers are well-defined at the residues' Cα's, the embeddings are not only rotationally invariant about their center, but also translationally invariant with the respect to translations of the whole system, i.e., they are effectively SE(3)-invariant.

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; 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.

To test the performance of our method, we use the LBA dataset in ATOM3D [30] that provides the PDB structure of the protein-ligand complex together with either the measured dissociation constant K_d or the inhibition constant K_i; see Sec. A 6 for further details. To map between the learned pocket embeddings to the log dissociation (or inhibition) constants we train both a simple Linear and a Random Forest Regressor on a training sets provided by ATOM3D. Fig. 4D shows the performance of the model with the Random Forest Regressor and Table II provide a detailed benchmark of our methods against prior approaches. The Linear model achieves competitive results, whereas the Random Forest Regressor achieves state-of-the-art.

These results demonstrate 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 II) learn complex graph-based functions on top of simple atomic representation, whereas our method uses simpler machine learning models over rich residue-level representations. Notably, ProtMD (the method competitive to ours) performs a pre-training scheme using expensive molecular dynamics simulations that informs the model about conformational flexibility, information that our method does not have access to. 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.

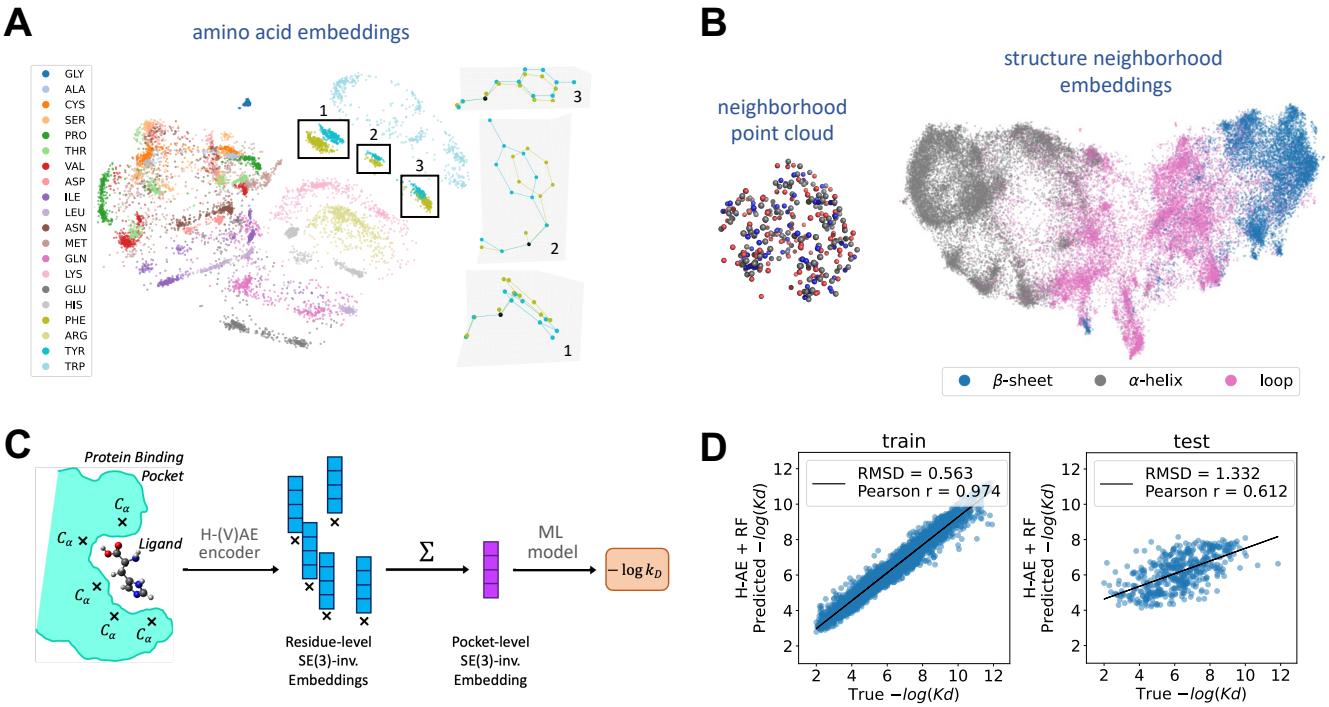

FIG. 4. **Structural embeddings to predict protein-ligand binding affinities with H-(V)AE. (A)** H-VAE was trained to reconstruct the Fourier representation of 3D atomic point clouds representing amino acids (colors). The invariant latent space clusters by amino acid conformations. The highlighted clusters 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. For this plot, 1,000 amino acids were used as training data, with network parameters: $\beta = 0.025$ and $z = 2$. **(B)** Left: An example protein neighborhood (point cloud of atoms) of 10 Å around a central residue, used to train the H-AE models, is shown. Right: 2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$ can separate neighborhoods by the secondary structure of their focal amino acid (colors). Each point represents a neighborhood; see Section A 5 e for details on the network architecture and training procedure. **(C)** We use H-AE to extract the residue-level SO(3)-invariant embeddings in the binding pocket of a protein-ligand structure complex (data from PDBbind [29]). 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. **(D)** The predictions on the protein-ligand binding affinities from (C) is shown against the true values for the training (left) and the test (right) sets. We use the data split provided by ATOM3D [30], which devises training and test sets respectively containing 3507, and 490 protein-ligand complexes, with maximum 30% sequence similarly between training and test proteins; see Table II for a comparison against state-of-the-art methods.

V. DISCUSSION

In this work, we have developed the first end-to-end SO(3)-equivariant unsupervised algorithm, termed Holographic (Variational) Auto-Encoder (H-(V)AE), suitable for data distributed in three dimensions around a given central point. 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.

Prior works have attempted to learn representations that are invariant to certain transformations. For example, in refs. [35, 36] general "shape" embeddings are learned by characterizing a separate "deformation" embedding. However, these networks are not explicitly equivariant to the transformations of interest.

Others proposed 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* [37] learns in quotient space by pooling together the latent encodings of copies of the data that have been transformed with sampled group actions, and back-propagate the minimum reconstruction loss between the decoded element and the transformed copies of the data. This approach is best suited for discrete and finite groups, for which it does not require approximations, and it is computationally expensive as it is akin to data augmentation. *Lohit et al* [28] construct an SO(3)-invariant autoencoder for spherical

Model	Ligand Binding Affinity 30% Similarity		
	RMSD ↓	Pearson's r ↑	Spearman's r ↑
DeepDTA	1.565	0.573	0.574
3DCNN	1.414 ± 0.021	0.550	0.553
GNN	1.570 ± 0.025	0.545	0.533
MaSIF	1.484 ± 0.018	0.467 ± 0.020	0.455 ± 0.014
EGNN	1.492 ± 0.012	0.489 ± 0.017	0.472 ± 0.008
GBPNet	1.405 ± 0.009	0.561	0.557
EGNN + PLM	1.403 ± 0.013	0.565 ± 0.016	0.544 ± 0.005
ProtMD	<u>1.367 ± 0.014</u>	<u>0.601 ± 0.036</u>	<u>0.587 ± 0.042</u>
H-AE + L.R.	1.397 ± 0.019	0.560 ± 0.017	0.568 ± 0.018
H-AE + R.F.	$\mathbf{1.332 \pm 0.012}$	$\mathbf{0.612 \pm 0.009}$	$\mathbf{0.619 \pm 0.009}$

TABLE II. **Results on the Ligand Binding Affinity task.** Prediction accuracies using H-AE embeddings with Linear Regression (L.R.), and Random Forest (R.F.) Regression are benchmarked against other methods. 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. Best scores are in **bold** and second-best scores are <u>underlined</u>. H-AE+R.F. delivers state-of-the-art predictions. Methods are ordered by date of release; see Table A.10 for more details.

signals by learning an invariant latent space and minimizing a loss which first finds the rotation that best aligns the true and reconstructed signals. Although this approach is effective for non-discrete data, it still manually imposes rotational invariance, and can only reconstruct signals up to a global rotation. In contrast, H-(V)AE is fully equivariant and only requires simple MSE for reconstruction of data in its original orientation.

A small body of work went beyond invariance to develop equivariant autoencoders. Several methods construct data and group-specific architectures to autoencode data equivariantly, learning an equivariant representation in the process [38, 39]. Others use supervision to extract class-invariant and class-equivariant representations [40]. 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 [16]. Our method in SO(3) is closely related to this work, with the crucial differences that our network is end-to-end rotationally equivariant in 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 (Section A 3) and in Table A.1.

There is also a diverse body of literature on using Fourier transforms and and CG tensor products to construct representations of atomic systems that are invariant/equivariant to euclidean symmetries [41–43], but without reducing the dimensionality of the representations in a data-driven way.

H-(V)AE's learned embeddings are highly expressive. For example, we used the learned invariants to achieve state-of-the-art unsupervised clustering and classification results on various spherical image datasets. By making our model variational in its invariant latent space, we enhanced the quality of clustering and made the model generative. Our model is defined fully in spherical Fourier space, and thus, can reach a desired expressiveness without a need for excessive computational resources.

H-(V)AE also produces rich residue-level representations of local neighborhoods in protein structures, which we use 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 learning the binding affinity between proteins and small molecule ligands.

More broadly, we expect that H-(V)AE can be used to extract rich, symmetry-aware features from local neighborhoods in spherical images and complex 3D objects, to be used in more complex downstream tasks that benefit from the symmetry constraints. For example, we expect our method can be leveraged for modeling diffusion MRI data, for which rotation-equivariant methods have recently proven to be highly beneficial [44]. In structural biology, we expect our method to be useful for coarse-graining full-atom representations of protein structures - or other biomolecules - to facilitate structure-based predictions of 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 embeddings can be leveraged to predict complex protein functions, as we show for predicting ligand binding affinity. This approach is akin to using protein embeddings for sequence data, learned by language models, to inform (few-shot) predictions for protein function [45].

VI. ACKNOWLEDGEMENTS

This work has been supported by the National Institutes of Health MIRA award (R35 GM142795), the CAREER award from the National Science Foundation (grant No: 2045054), the Royalty Research Fund from the University of Washington (no. A153352), the Microsoft Azure award from the eScience institute at the University of Washington, and the Allen School Computer Science & Engineering Research Fellowship from the Paul G. Allen School of Computer Science & Engineering at the University of Washington. This work is also supported, in part, through the Departments of Physics and Computer Science and Engineering, and the

College of Arts and Sciences at the University of Washington.

[1] M. Geiger and T. Smidt, e3nn: Euclidean Neural Networks (2022), arXiv:2207.09453 [cs].

[2] A. Capecchi, D. Probst, and J.-L. Reymond, One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome, Journal of Cheminformatics **12**, 43 (2020).

[3] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen, 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data (2018), arXiv:1807.02547 [cs, stat].

[4] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, Spherical CNNs (2018), arXiv:1801.10130 [cs, stat].

[5] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds (2018), arXiv:1802.08219 [cs].

[6] R. Kondor, Z. Lin, and S. Trivedi, Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network (2018), arXiv:1806.09231 [cs, stat].

[7] C. Esteves, A. Makadia, and K. Daniilidis, Spin-Weighted Spherical CNNs (2020), arXiv:2006.10731 [cs].

[8] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling, SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks (2020), arXiv:2006.10503 [cs, stat].

[9] J. Brandstetter, R. Hesselink, E. van der Pol, E. J. Bekkers, and M. Welling, Geometric and Physical Quantities Improve E(3) Equivariant Message Passing (2022), arXiv:2110.02905 [cs, stat].

[10] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics (2022), arXiv:2204.05249 [cond-mat, physics:physics].

[11] V. G. Satorras, E. Hoogeboom, and M. Welling, E(n) Equivariant Graph Neural Networks (2022), arXiv:2102.09844 [cs, stat].

[12] Y.-L. Liao and T. Smidt, Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs (2022), arXiv:2206.11990 [physics].

[13] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials, Nat. Commun. **13**, 1 (2022).

[14] M. A. Kramer, Autoassociative neural networks, Computers & Chemical Engineering Neutral network applications in chemical engineering, **16**, 313 (1992).

[15] D. P. Kingma and M. Welling, An Introduction to Variational Autoencoders, Foundations and Trends® in Machine Learning **12**, 307 (2019), arXiv:1906.02691 [cs, stat].

[16] R. Winter, M. Bertolini, T. Le, F. Noé, and D.-A. Clevert, Unsupervised Learning of Group Invariant and Equivariant Representations (2022), arXiv:2202.07559 [cs].

[17] A. Einstein, Die Grundlage der allgemeinen Relativitätstheorie, Annalen der Physik , 770 (1916).

[18] W.-K. Tung, *Group Theory in Physics* (1985).

[19] E. Schmidt, Zur Theorie der linearen und nichtlinearen Integralgleichungen, Mathematische Annalen **63**, 433 (1907).

[20] D. P. Kingma and M. Welling, Auto-Encoding Variational Bayes 10.48550/arXiv.1312.6114 (2013).

[21] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework (2022).

[22] L. McInnes, J. Healy, and J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction (2020), arXiv:1802.03426 [cs, stat].

[23] O. J. Cobb, C. G. R. Wallis, A. N. Mavor-Parker, A. Marignier, M. A. Price, M. d'Avezac, and J. D. McEwen, Efficient Generalized Spherical CNNs (2021), arXiv:2010.11661 [astro-ph].

[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, Layer Normalization (2016), arXiv:1607.06450 [cs, stat].

[25] L. Deng, The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web], IEEE Signal Processing Magazine **29**, 141 (2012), conference Name: IEEE Signal Processing Magazine.

[26] M. Aldenderfer and R. Blashfield, *Cluster Analysis* (SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America, 1984).

[27] A. Rosenberg and J. 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)* (Association for Computational Linguistics, Prague, Czech Republic, 2007) pp. 410–420.

[28] S. Lohit and S. Trivedi, Rotation-Invariant Autoencoders for Signals on Spheres (2020), arXiv:2012.04474 [cs].

[29] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li, and R. Wang, Comparative Assessment of Scoring Functions: The CASF-2016 Update, Journal of Chemical Information and Modeling **59**, 895 (2019), publisher: American Chemical Society.

[30] R. Townshend, M. Vögele, P. Suriana, A. Derry, A. Powers, Y. Laloudakis, S. Balachandar, B. Jing, B. Anderson, S. Eismann, R. Kondor, R. Altman, and R. Dror, ATOM3D: Tasks on Molecules in Three Dimensions, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks **1** (2021).

[31] SHREC 2017: Large-scale 3D Shape Retrieval from ShapeNet Core55.

[32] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, The Protein Data Bank, Nucleic Acids Research **28**, 235 (2000).

[33] M. N. Pun, A. Ivanov, Q. Bellamy, Z. Montague, C. LaMont, P. Bradley, J. Otwinowski, and A. Nourmohammad, Learning the shape of protein micro-environments with a holographic convolutional neural network (2022), pages: 2022.10.31.514614 Section: New Results.

[34] M. AlQuraishi, ProteinNet: a standardized data set for machine learning of protein structure, BMC Bioinformat-

ics **20**, 311 (2019).

[35] Z. Shu, M. Sahasrabudhe, A. Guler, D. Samaras, N. Paragios, and I. Kokkinos, Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance (2018), arXiv:1806.06503 [cs].

[36] K. Koneripalli, S. Lohit, R. Anirudh, and P. Turaga, Rate-Invariant Autoencoding of Time-Series, in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020) pp. 3732–3736, iSSN: 2379-190X.

[37] E. Mehr, A. Lieutier, F. S. Bermudez, V. Guitteny, N. Thome, and M. Cord, Manifold Learning in Quotient Spaces, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018) pp. 9165–9174, iSSN: 2575-7075.

[38] G. E. Hinton, A. Krizhevsky, and S. D. Wang, Transforming Auto-Encoders, in *Artificial Neural Networks and Machine Learning – ICANN 2011*, Lecture Notes in Computer Science, edited by T. Honkela, W. Duch, M. Girolami, and S. Kaski (Springer, Berlin, Heidelberg, 2011) pp. 44–51.

[39] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton, Stacked Capsule Autoencoders (2019), arXiv:1906.06818 [cs, stat].

[40] I. Feige, Invariant-equivariant representation learning for multi-class data, (2022).

[41] R. Drautz, Atomic cluster expansion for accurate and transferable interatomic potentials, Physical Review B **99**, 014104 (2019), publisher: American Physical Society.

[42] F. Musil, A. Grisafi, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti, Physics-Inspired Structural Representations for Molecules and Materials, Chemical Reviews **121**, 9759 (2021), publisher: American Chemical Society.

[43] M. Uhrin, Through the eyes of a descriptor: Constructing complete, invertible descriptions of atomic environments, Physical Review B **104**, 144110 (2021), arXiv:2104.09319 [cond-mat].

[44] P. Müller, V. Golkov, V. Tomassini, and D. Cremers, Rotation-Equivariant Deep Learning for Diffusion MRI (2021), arXiv:2102.06942 [cs].

[45] K. Swanson, H. Chang, and J. Zou, Predicting Immune Escape with Pretrained Protein Language Model Embeddings, in *Proceedings of the 17th Machine Learning in Computational Biology meeting* (PMLR, 2022) pp. 110–130, iSSN: 2640-3498.

[46] J. P. Boyd and F. 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**, 1408 (2011).

[47] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, Generating Sentences from a Continuous Space (2016), arXiv:1511.06349 [cs].

[48] M. Shanker, M. Y. Hu, and M. S. Hung, Effect of data standardization on neural network training, Omega **24**, 385 (1996).

[49] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in *Proceedings of the 25th international conference on Machine learning*, ICML '08 (Association for Computing Machinery, New York, NY, USA, 2008) pp. 1096–1103.

[50] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization (2017), arXiv:1412.6980 [cs].

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research **12**, 2825 (2011).

[52] H. Öztürk, A. Özgür, and E. Ozkirimli, DeepDTA: deep drug–target binding affinity prediction, Bioinformatics **34**, i821 (2018).

[53] M. Karimi, D. Wu, Z. Wang, and Y. Shen, DeepAffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks, Bioinformatics **35**, 3329 (2019).

[54] B. Anderson, T.-S. Hy, and R. Kondor, Cormorant: Covariant Molecular Neural Networks (2019), arXiv:1906.04015 [physics, stat].

[55] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost, ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning, IEEE transactions on pattern analysis and machine intelligence **44**, 7112 (2022).

[56] 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**, 184 (2020), number: 2 Publisher: Nature Publishing Group.

[57] T. Nguyen, H. Le, T. P. Quinn, T. Nguyen, T. D. Le, and S. Venkatesh, GraphDTA: predicting drug–target binding affinity with graph neural networks, Bioinformatics **37**, 1140 (2021).

[58] B. Jing, S. Eismann, P. Suriana, R. J. L. Townshend, and R. Dror, Learning from Protein Structure with Geometric Vector Perceptrons (2021), arXiv:2009.01411 [cs, q-bio, stat].

[59] V. R. Somnath, C. Bunne, and A. Krause, Multi-Scale Representation Learning on Proteins (2022), arXiv:2204.02337 [cs, q-bio].

[60] S. Aykent and T. Xia, GBPNet | Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2022).

[61] F. Wu, Y. Tao, D. Radev, and J. Xu, When Geometric Deep Learning Meets Pretrained Protein Language Models (2022), arXiv:2212.03447 [cs, q-bio].

[62] F. Wu, S. Jin, Y. Jiang, X. Jin, B. Tang, Z. Niu, X. Liu, Q. Zhang, X. Zeng, and S. Z. Li, Pre-training of Equivariant Graph Matching Networks with Conformation Flexibility for Drug Binding, Advanced Science **9**, 2203796 (2022), arXiv:2204.08663 [cs].

Appendix A: Appendix

1. Expanded background on SO(3)-equivariance

a. *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 $\boldsymbol{D}_Y(\mathfrak{g}) = \boldsymbol{I}, \forall \mathfrak{g} \in \mathfrak{G}$.

b. *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 [18]. 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.

c. *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 ℓth 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: $\boldsymbol{D}(\mathfrak{g}) = \boldsymbol{D}_{\ell_1}(\mathfrak{g}) \oplus \boldsymbol{D}_{\ell_2}(\mathfrak{g}) \oplus ... \oplus \boldsymbol{D}_{\ell_n}(\mathfrak{g})$.

d. *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 ℓth spherical harmonic via the ℓth 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.

e. 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 ref. [46], 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. ref. [43] also uses Zernike to construct invariant descriptors of atomic environments. However, this choice is not unique and other special functions such as Bessel functions can be used to encode the data [10].

2. Holographic-(V)AE (H-(V)AE) details

a. *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 $\overline{h}_\ell \in \mathbb{R}^{K\times(2\ell+1)}$:

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

b. *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 ref. [23] 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 [6, 23]) 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 in [23], 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 $\ell_{\max}$ from $\mathcal{O}(\ell_{\max}^2)$ - when all possible pairs are used - down to $\mathcal{O}(\ell_{\max})$. We refer the reader to ref. [23] for more details.

c. 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:

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

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

d. Signal Norm

Similar to for Batch Norm above, let us consider a batch of steerable tensors h which we index by batch b, degree ℓ, order m and channel c. Formally, the total norm for an individual tensor h is computed as:

$$N_{tot} = \sum_\ell \frac{\sum_c \sum_{m=-\ell}^{\ell} (\hat{h}_{\ell m}^c)^2}{2\ell+1} \tag{A.16}$$

Then, the tensor features are updated as $\overline{\hat{h}_{\ell m}^c} = \hat{h}_{\ell m}^c w_\ell / \sqrt{N_{tot}}$, where w_ℓ is a degree-specific affine transformation for added flexibility

e. Network architecture

Certain features of our problem formulation pose constraints upon the specific design choice of the network. For example, within the decoder, the maximum degree $\ell_{\max,b}$ that can be outputted by each block b is constrained by the sparsity of the CG tensor product. Specifically, $\ell_{\max,b} \leq 2^b$ 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 $\ell_{\max,B} \geq L$, or $B \geq \log_2 L$. In our experiments, we set $\ell_{\max,b} = \min\{2^b, L\}$ and do not let $\ell_{\max,b}$ 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. However this is required for input data that does not have the same number of channels per degree since the ETP operates channel-wise.

f. Tuning the hyperparameters of the training objective

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 ref. [47].

g. Data normalization

As per standard machine learning practice [48], 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.

h. 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):

$$\begin{aligned}
\text{MSE}(\boldsymbol{Dx}, \boldsymbol{Dy}) &= (\boldsymbol{Dx} - \boldsymbol{Dy})^T(\boldsymbol{Dx} - \boldsymbol{Dy}) \\
&= (\boldsymbol{D}(\boldsymbol{x} - \boldsymbol{y}))^T(\boldsymbol{D}(\boldsymbol{x} - \boldsymbol{y})) \\
&= (\boldsymbol{x} - \boldsymbol{y})^T\boldsymbol{D}^T\boldsymbol{D}(\boldsymbol{x} - \boldsymbol{y}) \\
&= (\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y}) \qquad \text{since Wigner-D matrices are unitary} \\
&= \text{MSE}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned} \tag{A.17}$$

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.

i. Dependence of the reconstruction loss on feature degree

We observe that features of larger degrees ℓ are harder to reconstruct accurately. Specifically, Fig. S8 shows that the test reconstruction loss (MSE) increases with degree ℓ for both the MNIST-on-the-sphere and the Toy amino acids problems.

j. Cosine loss

We use the metric *Cosine loss* to measure a model's reconstruction ability. Cosine loss 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 (see Sec. A 2 j for details),

$$\begin{aligned}
\text{Cosine}(\boldsymbol{x}, \boldsymbol{y}) &= 1 - \frac{\boldsymbol{x} \odot \boldsymbol{y}}{\sqrt{(\boldsymbol{x} \odot \boldsymbol{x})(\boldsymbol{y} \odot \boldsymbol{y})}}, \\
\text{with} \quad \boldsymbol{x} \odot \boldsymbol{y} &= \sum_{\ell'} (\boldsymbol{x}_{\ell'} \otimes_{cg} \boldsymbol{y}_{\ell'})_{\ell=0}
\end{aligned} \tag{A.18}$$

Importantly, unlike MSE, which depends on the characteristics of the data (e.g., the size of the data tensors), Cosine loss is dimensionless and therefore, interpretable and comparable across datasets. A measure with these characteristics is practically useful for evaluating 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.7, 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. S9). Nonetheless, as a measure of a model's reconstruction ability, Cosine loss correlates almost perfectly with MSE for a given dataset and network, especially in the mid-to-low reconstruction quality regime (SpearmanR = 0.99, Fig. S9 and Table A.7). Crucially, however, as Cosine loss ignores the relative norms of data features, it is unable to reconstruct norms and thus, not suitable as a training objective.

Proof: Invariance of Cosine loss. The generalized dot product $\odot$ from Eq. A.18 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.

3. Using a non-equivariant decoder

Winter et al [16] 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 [1] 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 [49] 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).

4. Implementation details

Without loss of generality, we use real spherical harmonics for implementation of H-(V)AE. We leverage `e3nn` [1], 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.

Method	z	Speed	MSE	Cosine	Purity	V-meas.	LC Class. Acc.	KNN Class. Acc.
H-AE NR/R	16	**1.0**	$\mathbf{0.9 \times 10^{-3}}$	**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**

TABLE A.1. **Performance comparison between our H-(V)AE and a H-(V)AE constructed with ref. [16]'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 5).

5. Experimental details

a. *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.

b. *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 [50] 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; see Fig. S2 for the behavior of the training loss. 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.

c. *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.

d. Embedding of amino acid conformations

Pre-processing of protein structures. We sample residues from the set of training structures pre-processed as described in Sec. A 5 e.

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 $10.0\mathring{A}$ 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 $\sim$ 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.

e. Embedding of protein structure neighborhoods

Pre-processing of protein structures. We model protein neighborhoods extracted from tertiary protein structures from the Protein Data Bank (PDB) [32]. We use ProteinNet's splittings for training and validation sets to avoid redundancy, e.g. due to similarities in homologous protein domains [34]. 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 b).

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.

z	L	Tensor size	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

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

f. 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` [51] implementation with default parameters. At evaluation time, we select the class with the highest probability value.

g. Clustering metrics

Purity [26]. 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 [27]. 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.

h. 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.

6. Protein-Ligand Binding Affinity Prediction

a. *Data Preprocessing Details*

We leverage the H-AE models trained on Protein Neighborhoods (Sec. A 5 e) 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 [29], containing $\sim 5,000$ structures. We use the dataset splits provided by ATOM3D [30] 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. 2) 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, 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 II), the Spearman's correlation over the ligands containing only protein atoms is higher than for the ligands containing other kinds of atoms by about 0.3 points. 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.

b. *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` [51] 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]

c. Extended Discussion of Results

Table A.10 shows benchmarking results on the split of ATOM3D with 30% sequence similarity, extending Table II in the main text with more baselines. Notably, we show the prediction results for the models that only use the SO(3)-invariant ($\ell = 0$) 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.

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.3. 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	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.4. Evaluation 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.

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

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

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.7. **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	TP-type	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.8. **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 5, 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	# 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	-	$(0.09 \pm 0.07)\times10^{-5}$	$(0.1 \pm 1.3)\times10^{-2}$

TABLE A.9. **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).

Method	Ligand Binding Affinity 30% Similarity			
	RMSD ↓	Pearson's r ↑	Spearman's r ↑	Kendall's τ ↑
DeepDTA [52]	1.565	0.573	0.574	-
DeepAffinity [53]	1.893 ± 0.650	0.415	0.426	-
Cormorant [54]	1.568 ± 0.012	0.389	0.408	-
ProtTrans [55]	1.544 ± 0.015	0.438 ± 0.053	0.434 ± 0.058	-
3DCNN [30]	1.414 ± 0.021	0.550	0.553	-
GNN [30]	1.570 ± 0.025	0.545	0.533	-
MaSIF [56]	1.484 ± 0.018	0.467 ± 0.020	0.455 ± 0.014	-
DGAT [57]	1.719 ± 0.047	0.464	0.472	-
DGIN [57]	1.765 ± 0.076	0.426	0.432	-
DGAT-GCN [57]	1.550 ± 0.017	0.498	0.496	-
GVP-GNN [58]	1.648 ± 0.014	0.213 ± 0.013	0.164 ± 0.009	0.110 ± 0.012
EGNN [11]	1.492 ± 0.012	0.489 ± 0.017	0.472 ± 0.008	0.329 ± 0.014
HoloProt [59]	1.464 ± 0.006	0.509 ± 0.002	0.500 ± 0.005	-
GBPNet [60]	1.405 ± 0.009	0.561	0.557	-
EGNN + PLM [61]	1.403 ± 0.013	0.565 ± 0.016	0.544 ± 0.005	0.379 ± 0.007
ProtMD [62]	1.367 ± 0.014	<u>0.601 ± 0.036</u>	<u>0.587 ± 0.042</u>	-
Zernike Inv. + Linear Regression	1.455 ± 0.005	0.513 ± 0.005	0.516 ± 0.006	0.357 ± 0.005
Zernike Inv. + Random Forest	<u>1.361 ± 0.011</u>	0.587 ± 0.009	0.584 ± 0.010	<u>0.408 ± 0.008</u>
H-AE + Linear Regression	1.397 ± 0.019	0.560 ± 0.017	0.568 ± 0.018	0.397 ± 0.016
H-AE + Random Forest	$\mathbf{1.332 \pm 0.012}$	$\mathbf{0.612 \pm 0.009}$	$\mathbf{0.619 \pm 0.009}$	$\mathbf{0.436 \pm 0.006}$

TABLE A.10. **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 <u>underlined</u>. Errors for our models are computed as the standard deviation in prediction by 10 Machine Learning models trained with bootstrapped data.