H-Packer: Holographic Rotationally Equivariant Convolutional Neural Network for Protein Side-Chain Packing

Gian Marco Visani[1] **William Galvin**[1] **Michael N. Pun**[2] **Armita Nourmohammad**[1 2 3 4]

Abstract

Accurately modeling protein 3D structure is essential for the design of functional proteins. An important sub-task of structure modeling is protein side-chain packing: predicting the conformation of side-chains (rotamers) given the protein's backbone structure and amino-acid sequence. Conventional approaches for this task rely on expensive sampling procedures over hand-crafted energy functions and rotamer libraries. Recently, several deep learning methods have been developed to tackle the problem in a data-driven way, albeit with vastly different formulations (from image-to-image translation to directly predicting atomic coordinates). Here, we frame the problem as a joint regression over the side-chains' true degrees of freedom: the dihedral χ angles. We carefully study possible objective functions for this task, while accounting for the underlying symmetries of the task. We propose *Holographic Packer* (H-Packer), a novel two-stage algorithm for side-chain packing built on top of two light-weight rotationally equivariant neural networks. We evaluate our method on CASP13 and CASP14 targets. H-Packer is computationally efficient and shows favorable performance against conventional physics-based algorithms and is competitive against alternative deep learning solutions.

1 Introduction

Proteins are macromolecules composed of residues (amino-acids) that are linked consecutively to form an amino-acid *sequence*. Each residue is conceptually divided into two parts: (i) a *backbone* structure common to all amino acids, which is comprised of the alpha carbon (C-α) bounded to an amino group (-NH_2) and a carboxyl group (-COOH); and (ii) a residue-specific *side-chain*. Backbones are connected by peptide bonds between the amino and carboxyl groups of consecutive residues. Physical interactions between the freely-moving side-chains cause the protein chain to *fold* into a complex 3D structure, which confers the protein its function.

Conceptually, a protein's full atomic structure can be divided into its backbone structure (the coordinates of its backbone atoms) and its side-chains conformations (the coordinates of its side-chain atoms). Side-chain conformations are relatively flexible, while the backbone structure is more rigid and confers the protein its main 3D topology, and thus, its main function. Nonetheless, the interaction between a protein's backbone and side-chains is essential for the stability of the fold and protein function.

Determining amino acid side-chain conformations in a protein, known as Protein Side-Chain Packing (or Rotamer Packing), is an essential step in protein folding and the de-novo design of proteins. Computational approaches to protein folding often divide the structure inference problem into two steps: first, they characterize the rigid backbone structure, and then they pack the side-chains associated with the amino acids at each residue. The flexibility of the side-chain makes the search in the space of possible conformations inevitably complex and computationally expensive. The de-novo protein design protocols also rely on similar logical steps: Often an amino acid sequence compatible with a desirable backbone structure is to be inferred (designed) [1] and then the associated side-chains should be packed to form the full atomic composition of a protein.

[1]Paul G. Allen School of Computer Science and Engineering, University of Washington
[2]Department of Physics, University of Washington
[3]Department of Applied Mathematics, University of Washington
[4]Fred Hutch Cancer Research Center, Seattle, WA

Machine Learning in Computational Biology, PMLR track (MLCB 2023)

Many of the conventional methods for side-chain packing rely on physical models through which they find a rotamer that minimizes a physically-reasoned heuristic energy of the protein fold [2, 3, 4]. However, these computational methods often lack accuracy and speed in their predictions. As deep learning makes strides in protein science, there is a growing effort in developing machine learning methods for rotamer packing. Among these methods is DLPacker [5], which treats the packing problem as an image transformation. This algorithm characterizes the local environment of a given amino acid backbone within a structure as a 3D image, and uses this model to predict the atomic coordinates of the side chain. It then compares the predicted side-chain to a pre-set library of rotamers to select the closest conformation. AttnPacker [6], a more recently developed method, uses a deep graph attention network to model the local geometry of a residue within a structure and is trained to predict the coordinates of the side-chain atoms. Recently, diffusion models over side-chain torsional angles have also being applied, such as DiffPack [7].

Here, we tackle the problem of side-chain packing by learning to directly regress over χ (torsional) angles, which are main degrees of freedom determining side-chain conformations. We derive and discuss three possible parameterizations of the χ angles, ultimately settling on regressing over the Sine and Cosine transforms of the angles. We introduce Holographic-Packer (H-Packer), a deep learning method that packs rotamers by first predicting candidate χ angles from backbone and sequence, and then refines the predictions with a model trained on full-atom structures. Our approach relies on our previously developed holographic convolutions neural network (H-CNN) to characterize amino acid preferences, given their local atomic environment within a structure [8]. H-CNN, and by extension H-packer, are locally rotationally (i.e., SO(3)) equivariant, in that they can physically reason about the local geometry of protein structures. Specifically, they achieve their rotational equivariance by operating fully in the spherical Fourier space.

By directly predicting the side-chain χ-angles, H-packer does not rely on comparing its output with a pre-set library of rotamers, making it computationally more efficient than methods like DLPacker. Furthermore, H-Packer is light-weight (2×3M parameters vs. 208M of AttnPacker) and requires few resources to train (single vs. multiple GPUs for diffusion models like DiffPack). We evaluate the packing performance of H-Packer on standard datasets, and show that it has generally better performance than conventional physics-based methods, and competitive against machine learning solutions. In general, our results suggest that H-Packer has learned complementary features to alternative methods. Our code is freely available at `https://github.com/gvisani/hpacker`.

2 Methods

In this work, we study the problem of amino-acid side-chain packing using rotationally equivariant neural networks. We introduce H-Packer, a novel yet simple algorithm that predicts side-chain conformations by jointly predicting the values of the key degrees of freedom of a side-chains, i.e., its χ angles.

2.1 Modeling side-chain conformations with χ Angles

While amino acids are composed of a maximum of 10 heavy atoms (in the case of Tryptophan), their 3D conformations can be uniquely described by the value of at most 4 dihedral angles, referred to as the χ angles (Figure 1A). This reduction in the number of degrees of freedom is granted due to the physical constraints posed on the remaining internal coordinates (bond angles, bond lengths, and dihedral angles - *redundant internal coordinates*). Specifically, the inter-atomic physical interactions within amino acids often constrain these redundant coordinates to a constant, or a well-defined function of the residue's χ angles. Therefore, predicting χ angles is the key step in side-chain packing.

H-packer addresses the side-chain packing problem in two steps: (i) it predicts the value of χ angles, and (ii) it reconstructs the atomic coordinates using the predicted χ angles and the constrained values of the redundant internal coordinates. Specifically, we evaluate the redundant internal coordinates from a subset of training data (1,700 structures) by leveraging the `internal_coords` feature of the `biopython` package. We empirically verified that the distributions of values were Gaussian with low variance, and resolved to take their *medians* as the ground truth. Substituting these values for the original ones yields a negligible Null Reconstruction error of approximately 0.127Å (Figure A.1). Notably, this error remains unchanged even when using only 100 reference structures instead of 1,700 (Figure A.2).

2.2 Predicting χ angles using H-Packer

We aim to predict the χ angles associated with a side-chain conformation from the configuration of atoms surrounding a given residue. This atomic neighborhood is associated with the backbone and the side-chain of the neighboring residues in the structure.

During inference only the coordinates of the backbone atoms are known a priori - alongside the identity of the amino acids they belong to. However, physical interactions with the atoms of other side-chains are the true determinants of a

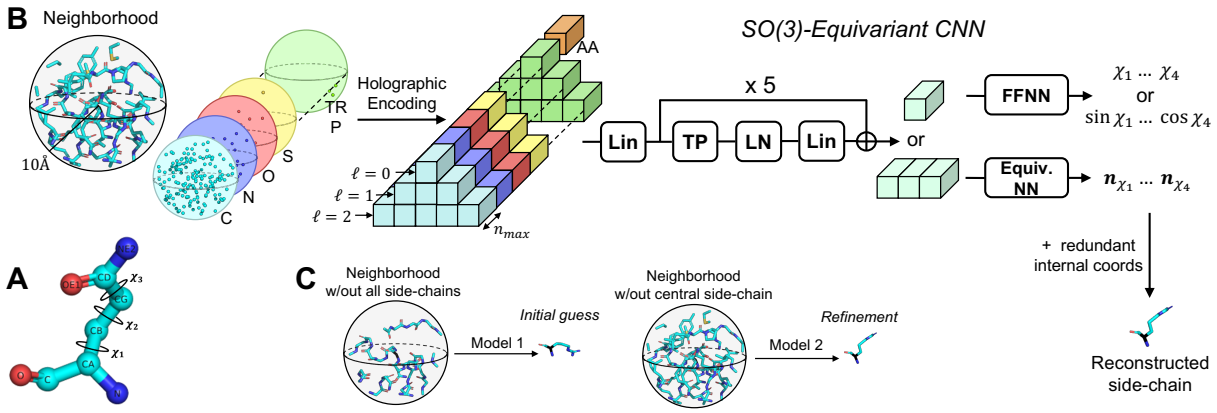

Figure 1: **Overview of H-Packer. A:** Illustration of Glutamine's χ angles, of which there are three. **B:** Schematic shows the H-CNN style network for side-chain packing by first predicting the missing residue's χ angles from its surrounding atomic environment, and then using the χ angles to reconstruct the residue's side-chain. As illustrated in **C**, H-Packer consists of two H-CNN networks, one trained on backbone atoms only and used to make an initial guess, and one trained on full side-chain neighborhoods and used to refine the predictions.

residue's conformation. Therefore, we develop H-Packer into a two-step solution (Figure 1C). Specifically, we two train models: one to predict χ angles from the backbone atoms and amino-acid identity alone, the another to predict χ angles from full neighborhoods, i.e., by including the true side-chain atoms of the surrounding residues (minus the residue of interest). At inference time, we use the first model to make an *initial guess* of the side-chain conformations, and then a second model to iteratively *refine* the predictions.

To build the individual models that predict χ angles, we start by considering their symmetries. Notably, χ angles are *invariant* to rigid-body transformations (translations and rotations) of the protein (i.e., they are SE(3) invariant). Translation invariance can be satisfied by choosing a well-defined center for a residue of interest; we choose the residue's C-α, as it is a common component of all residues and is at the beginning of the side-chain. Then, we still need to take into account rotational invariance about the specified center, which is associated with transformations under the rotation group SO(3).

To respect such rational symmetry, we build SO(3)-*equivariant* models to predict a residue's χ angles from its surrounding atomic environment. Equivariance is a generalization of invariance whereby when a function's input is transformed by the action of a certain group element (in this case rotation group SO(3)), the output is transformed by the same group element in a well-defined way; equivariant layers ensure both expressivity and efficiency when fitting both invariant and equivariant functions (see Appendix A.1 for details). To develop these models, we use an approach inspired by our previous work [9, 8]. We consider as input the point cloud of atoms within a radius $r = 10$ of the residue's C-α (with or without the neighboring side-chains). To ensure rotational equivariance, we both encode the input in a rotationally equivariant fashion (i.e., a holographic encoding), and use SO(3)-equivariant layers to predict the χ angles.

2.2.1 Holographic encoding of the data

We represent the point clouds of atoms within a structural neighborhood with a density function by summing over (weighted) Dirac-δ functions, indicating the presence of atoms at a given position in space: $\rho(r, \theta, \phi) = \sum_{i \in \text{points}} \omega_i \delta(\boldsymbol{r}_i - \boldsymbol{r})$; here, ω_i indicates the weight associated with point i at position $\boldsymbol{r}_i$. We then use 3D Zernike Fourier Transform (ZFT) of the density function to encode the neighborhood into a convenient SO(3) equivariant basis,

$$\hat{Z}^n_{\ell m} = \sum_{i \in \text{points}} \omega_i R^\ell_n(r_i) Y_{\ell m}(\theta_i, \varphi_i) \tag{1}$$

where $Y_{\ell m}(\theta, \phi)$ is the spherical harmonics of degree ℓ and order m, and $R^n_\ell(r)$ is the radial Zernike polynomial in 3D with radial frequency $n \geq 0$ and degree ℓ. $R^n_\ell(r)$ is non-zero only for even values of $n - \ell \geq 0$. Notably, the spherical harmonics that describe the angular component of ZFT arise from the irreducible representations of the 3D rotation group SO(3), and form a convenient basis under rotation in 3D (see Appendix A.1). Zernike projections in spherical

Fourier space can be understood as a superposition of spherical holograms of an input point cloud, and thus, we term this operation as *holographic encoding* of the data [9, 8].

We truncate the Fourier expansion by the maximum degree ℓ_{max} and a maximum radial frequency n_{max}. Additionally, we normalize the Fourier coefficients of each Dirac-δ function by the sum of the square of its coefficients. We found this normalization to be beneficial for training, likely due to the avoidance of singularities close to the boundaries.

Following [8] and [9] we incorporate atom-level input features by dividing the holographic encoding into different *channels* (see Figure 1). We consider the following two sets: (i) **Atomic channels**: C, N, O, S, wildcard element excluding hydrogens, partial charge from the Amber99sb force field [10], and (ii) **Amino-Acid channels**: one for each of the 20 canonical amino-acids, plus a wildcard channel. We include the charge value in its dedicated channel as the weights ω_i coupled to the point cloud's density function. While we train the *initial guess* model using both sets of channels (atomic and amino-acid) as input, we only consider the atomic channels for the *refinement* model. We do this in an effort to make the model's predictions more grounded in physical interactions. We condition both models with the identity of the residue of interest by concatenating a linear embedding of its one-hot encoding to the input's invariant ($\ell = 0$) features. This is particularly necessary for the refinement model - which is trained only with atomic channels - since it wouldn't otherwise know about the identity of the residue of interest.

2.2.2 SO(3)-Equivariant neural network architecture

We use the resulting holograms as inputs to an SO(3)-Equivariant Convolutional Neural Network (Figure 1B). The key is to transform the inputs through the network such that all intermediate outputs of the network remain rotationally equivariant. Our resulting model is conceptually divided into three parts:

First, a linear layer that projects data and conditioning to a hidden representation with same number of features per ℓ.
Second, a stack of equivariant blocks connected via additive skip connections, each composed of: (i) feature-wise tensor product nonlinearity, (ii) layer norm with silu nonlinearity, and (iii) a linear layer whose output dimensions are the same as the input's. After the final block, we retain only the features of type $\ell = 0$ or $\ell = 1$ depending on the training objective (Section 2.2.3). It should be noted that features of type $\ell = 0$ are rotationally invariant scalars, whereas those associated with $\ell = 1$ are equivariant vectors that transform consistently with the input under rotation. We use $\ell = 1$ features to directly learn the orientation of the intersecting planes that define a side-chain's dihedral angles χ (see Section 2.2.3).
Third, optionally and only for the models with invariant ($\ell = 0$ output), we apply a standard feed-forward neural network with dropout regularization and silu nonlinearity. We refer to Section A.2 in the appendix for more details on the architecture components.

2.2.3 Training objectives to infer χ angles

We consider three alternative parameterizations of χ angles, i.e. three possible objective functions:

(i) The angle itself. χ angles are defined between $-180°$ and $180°$ with a periodicity such that the angles $-179°$ and $179°$ are to be considered $2°$ apart, not $358°$. Thus, plain MSE loss would pose strong and unnatural constraints on the model. To account for this, we mod the predictions to fall in the valid range, and compute the loss between two angles as the minimum between the computed error and $360°$ minus the error, resulting in the following loss function:

$$\mathcal{L}_{\text{angles}}(\{\hat{\chi}_i\}_{i=1}^{N_\chi}, \{\chi_i\}_{i=1}^{N_\chi}) = \frac{1}{N_\chi}\sum_{i=1}^{N_\chi} \min(E_{\chi_i},\, 2\pi - E_{\chi_i}) \quad \text{where} \quad E_{\chi_i} = (\text{mod}(\hat{\chi}_i, 2\pi) - \chi_i)^2 \tag{2}$$

where $\hat{\chi}_i$ and χ_i are the predicted and the true values of the i^{th} χ, respectively, and N_χ is the number of χ angles associated with the residue of interest. In our implementation, the χ angle domain is scaled and shifted to fall in $[0, 2]$ to make the scale of the loss functions comparable between the three representations of the angles.

(ii) Sine and Cosine transforms of the angle. A pair of sine and cosine transformation provides an alternative representation for a χ angle that accounts for its periodicity and is also rotationally invariant; a similar approach is also considered in concurrent work [11]. We directly predict sine and cosine values by feeding 8 outputs from the network to a tanh activation function, which then form the arguments of a MSE loss function:

$$\mathcal{L}_{\text{sin-cos}}(\{\hat{\chi}_i\}_{i=1}^{N_\chi}, \{\chi_i\}_{i=1}^{N_\chi}) = \frac{1}{2N_\chi}\sum_{i=1}^{N_\chi} (\cos\hat{\chi}_i - \cos\chi_i)^2 + (\sin\hat{\chi}_i - \sin\chi_i)^2 \tag{3}$$

Notably, this loss function is justified by a nice geometric interpretation, whereby it is equivalent to computing the cosine loss between the 2D vectors that describe the χ angles on the unit circle (proof in Eq. A.6).

(iii) Normal vectors to the dihedral plane. χ angles are examples of dihedral angles, meaning that they are defined as the angle between two planes. For χ angles, the two planes are described by subsequent triplets of atoms along the side-chains. Any two subsequent χ angles share one plane. Therefore, any conformation with N_χ angles can be alternatively described by $N_\chi + 1$ planes (or their normal vectors); one of these normal vectors is a redundant internal coordinate (defined by backbone + Cβ atoms), while others specify the N_χ independent degrees of freedom.

We consider training models to predict the dihedral planes' normal vectors: $\boldsymbol{n}_{\chi_1} \dots \boldsymbol{n}_{\chi_4}$. It should be noted that unlike the sine/cosine transformation, the vectors are not invariant to rotations, but *equivariant* of type $\ell = 1$ (geometric vectors) which can be extracted from the H-Packer equivariant network. We use a cosine loss over the true and predicted vectors:

$$\mathcal{L}_{\text{norms}}(\{\hat{\boldsymbol{n}}_{\chi_i}\}_{i=1}^{N_\chi}, \{\boldsymbol{n}_{\chi_i}\}_{i=1}^{N_\chi}) = \frac{1}{N_\chi} \sum_{i=1}^{N_\chi} 1 - \langle \hat{\boldsymbol{n}}_{\chi_i}, \boldsymbol{n}_{\chi_i} \rangle \tag{4}$$

Relevant symmetries in computing loss functions. Some amino acid conformations exhibit a rotation symmetry by π in some of their χ angles. For example, χ_2 of Phenylalanine and Tyrosine indicates the torsion of their benzene rings, thus a rotation by π leaves the conformation physically unchanged. However, as χ angles are formally defined by internal atom names, these equivalent conformations are associated with different χ angle values. We correct for this degeneracy by considering the minimum loss value between considering χ and $\pi - \chi$ as targets during training and evaluation. When computing the error on the atomic coordinates (generally via Root Mean Square Deviation, RMSD) for the full side-chain, we need to consider other such symmetries between non-χ atoms, as listed in Table A.1.

3 Related Work

Protein side-chain packing. Methods for side-chain packing can be divided into (older) physics-based algorithms [2, 4, 3, 12, 13] and (newer) machine learning (ML) approaches [5, 6, 7, 11, 14, 15]. Physics-based approaches generally work by minimizing a hand-crafted energy function over the side-chain conformational space, usually with the help of a rotamer (i.e., side-chain conformation) library to discretize and reduce the dimensionality of such space. Popular algorithms include RosettaPacker from the rosetta suite [2], FASPR [4], and SCWRL [3]. Among ML methods, the most related to this work include: DLPacker [5], which frames the problem as an image-to-image translation (with "channels" analogous to ours) to predict a 3D "image" of the desired rotamer, which is then matched against a rotamer library to return a valid representation; AttnPacker [6], which uses a large (208M) model derived from the SE(3)-Transformer [16] to directly predict the coordinates of side-chain atoms from the backbone structure and the amino-acid sequence. The concurrent ZymePackNet [11] (open source code not available) which autoregressively predicts the sine and cosine of χ angles, using two graph neural networks in a two-step procedure similar to ours; and DiffPack [7], which consists of four expensive diffusion models over each of the χ angles, autoregressively used together at inference time.

Equivariant neural networks for protein structures. In recent years, great successes has been achieved in structural biology by leveraging the underlying geometric symmetries in modeling protein structure and surface in the form of developing neural networks that are equivariant to the relevant symmetry transformations [17, 18, 19, 8, 9]. Specifically, a great deal of literature has been devoted to efficiently modeling 3D atomistic systems using neural networks equivariant to euclidean symmetries [16, 20, 21, 22, 23]. The drawback is that most such methods are computationally expensive due to computing expensive tensor products between all pairs of neighboring atoms (see Section A.2 and [20, 21]). Here, we greatly reduce computational complexity by constructing equivariant representations of a system about a single natural center (the central residue's C-α), following an approach originally designed to model spherical images [23]. Applying this approach to residue-level structure modeling has been proven effective in predicting amino-acid propensities in protein structures [8], as well as compactly encoding residue environments in an unsupervised way for downstream tasks [9].

4 Experiments

4.1 Toy task: inferring χ angles from atomic coordinates

We start by studying the behavior of our model on a simple task: predicting (or rather, calculating) χ angles from the true atomic coordinates of the conformation. We found this to be a useful benchmark to study our model's behavior.

Setup. We randomly select 160 structures from our real task's training set (see below) and split them into 100/30/30 for training/validation/testing, respectively. We collect conformations of all residues presenting χ angles, and consider only their heavy atoms (C, N, O, S). We then apply the Zernike encoding varying ℓ_{max} from 1 to 5 and train models

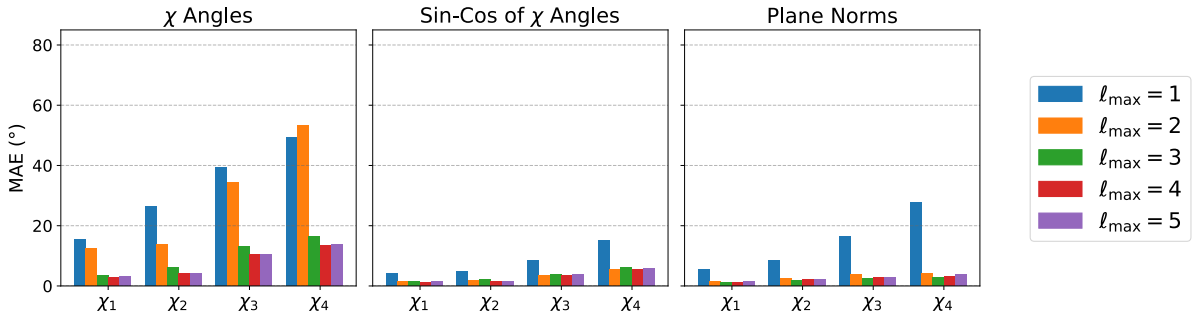

Figure 2: **Test MAE for the simple task of predicting χ angles from atomic conformation.** Panels show reconstruction accuracies using three loss functions: the angle χ itself (left), the sin/cos transform of the angle (center), and the normal vectors to the dihedral planes (right), for different maximum angular degrees ℓ_{max} (colors).

with varying ℓ_{max} consistent with that of the input, as well as with different prediction objectives (angles, sin-cos of angles, plane norms). Crucially, we vary the number of hidden channels (decreasing it with higher ℓ_{max}) to keep the number of parameters constant around 330k, and thus, removing differences in model capacity as a contributing factor to performance. We do not condition the models with amino-acid identity to make the problem more challenging, and therefore more interesting. We refer to Section A.4 for more details.

Results. Test Mean Absolute Error (MAE) per χ angle for all models is shown in Figure 2, and training curves are shown in the Appendix (Figure A.3). Notably, the Angle model performs the worst, and is unable to recover the true χ angle with negligible error. The Sin-Cos and Plane Norm models instead recover all χ angles with very low error (< 5 Å) with $\ell_{\text{max}} > 1$. It appears that $\ell_{\text{max}} = 2$ is the minimum sufficient degree nedded to solve this task with high accuracy. We note that error is higher for later χ angles. We hypothesise that this is expected for two reasons: (i) later χ angles depend on atoms that are farther way from the center of the neighborhood, thus having lower angular resolution within the Zernike representation, and (ii) there is simply less training data for them. Weighting χ angles in the loss function according to their average frequency partially mitigates the second issue (Figure A.4). Notably, the fact that the model performs well without explicit knowledge of amino acid identities implies that it can easily infer the amino acid type from the the number and the relative location of the atoms.

4.1.1 Side-Chain Packing

Dataset. We consider the training and validation datasets used in DLpacker [5], consisting of 19,436 structures with a maximum inter-protein sequence similarity of 50%. Unlike DLPacker, we do not remodel structures with PDB-redo [24] and do not convert selenomethionine residues into methionine. For testing our model, we use the CASP13 and CASP14 targets (82 and 64 structures, respectively). We remove from the training and validation sets any protein that has sequence similarity above 50% with any of the proteins in the test set.

H-Packer training. We used the Sin/Cos loss function (Eq. 3) as it was the best-performing loss in our toy-task; while the Plane Norms loss (Eq. 4) also performed well in the toy-task, we found that models trained with the Sin/Cos objective were easier to regularize via dropout in the final invariant feed-forward neural network. The *initial guess* and the *refinement* networks were trained with the same ℓ_{max} of 5 and $n_{\text{max}} = 12$; the latter was chosen such that it included at least one radial function with wavelength lower than the minimum interatomic distance. We also considered models trained with $\ell_{\text{max}} = 4$, tuning the number of hidden features to keep the number of trainable parameters the same as the $\ell_{\text{max}} = 5$ models, and equal to $\sim$3M. All models were trained for 10 epochs, keeping the model with lowest validation loss at the end of an epoch; see further details in Section A.4. Throughout our experiments, we consider the performance of H-Packer models with different number of rounds of refinement. For example, H-Packer$_0$ denotes the model with *no* refinement. For each model, we also compute an *upper bound* in performance of the refinement process by tasking the *refinement* model to predict χ angles from the ground truth neighboring structures (i.e., the toy task); we denote this by H-Packer$_{up}$.

Metrics. In line with previous work [6, 7], we evaluate our models on three main metrics. (i) Angle-specific Mean Absolute Error (MAE), (ii) residue-level angle accuracy, defined as the proportion of residues for which the prediction of all χ angles is within 20° of the true value, and (iii) average atomic Root Mean Square Deviation (RMSD) of side-chain atoms across residues. We further distinguish between *Surface* and *Core* residues, as conformations occurring on the surface of proteins are notoriously harder to predict. Surface residues are defined as having at most 15 β-C within 10 Å of their β-C, whereas core residues must have at least 20 β-C's in this range.

CASP13	Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å↓		
Method	χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
SCWRL	27.64	28.97	49.75	61.54	56.2	71.3	43.4	0.934	0.495	1.027
FASPR	27.04	28.41	50.30	60.89	56.4	70.3	43.6	0.910	0.502	1.002
RosettaPacker	25.88	28.25	48.13	59.82	58.6	75.3	35.7	0.872	0.422	1.001
DLPacker	22.18	27.00	51.22	70.04	58.8	73.9	45.4	0.772	0.402	0.876
AttnPacker	18.92	23.17	44.89	58.98	62.1	73.7	47.6	0.669	0.366	0.775
DiffPack	**15.35**	**19.19**	**37.30**	**50.19**	**69.5**	**82.7**	**57.3**	**0.579**	**0.298**	**0.696**
H-Packer$_0^{\ell_{\max}=5}$	26.89	31.95	47.51	52.75	49.3	61.5	40.4	0.961	0.726	1.131
H-Packer$_2^{\ell_{\max}=5}$	23.64	29.47	45.17	53.26	54.4	69.9	43.6	0.863	0.575	1.070
H-Packer$_5^{\ell_{\max}=5}$	23.60	29.40	44.91	52.91	54.7	70.7	43.7	0.858	0.564	1.067
H-Packer$_{up}^{\ell_{\max}=5}$	*20.03*	*26.88*	*42.74*	*52.07*	*58.4*	*75.4*	*46.4*	*0.765*	*0.483*	*0.980*

CASP14	Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å↓		
Method	χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
SCWRL	33.50	33.05	51.61	55.28	45.4	62.5	33.2	1.062	0.567	1.216
FASPR	33.04	32.49	50.15	54.82	46.3	62.4	34.0	1.048	0.594	1.205
RosettaPacker	31.79	28.25	50.54	56.16	47.5	67.2	33.5	1.006	0.501	1.183
DLPacker	29.01	33.00	53.98	72.88	48.0	66.9	33.9	0.929	0.476	1.107
AttnPacker	25.34	28.19	48.77	51.92	50.9	66.2	36.3	0.823	0.438	1.001
DiffPack	**21.91**	**25.54**	**44.27**	55.03	**57.5**	**77.8**	**43.5**	**0.770**	**0.356**	**0.956**
H-Packer$_0^{\ell_{\max}=5}$	32.31	35.90	49.05	50.34	40.8	57.0	31.6	1.087	0.762	1.297
H-Packer$_2^{\ell_{\max}=5}$	29.96	34.32	47.46	50.50	45.0	65.1	34.1	1.011	0.629	1.250
H-Packer$_5^{\ell_{\max}=5}$	29.61	34.03	46.72	**50.35**	45.2	65.5	34.0	1.002	0.626	1.244
H-Packer$_{up}^{\ell_{\max}=5}$	*26.58*	*31.54*	*45.67*	*49.46*	*48.1*	*69.5*	*36.2*	*0.915*	*0.534*	*1.160*

Table 1: **Comparative assessment on CASP13 and CASP14.** We present best results in **bold** and second-best underlined. *Italicized* results represent an upper bound to our algorithm's performance. Performance of models other than H-Packer is taken from [7].

	CASP13				CASP14			
Method	base	Rec.	Sym.	Rec.+Sym.	base	Rec.	Sym.	Rec.+Sym.
H-Packer$_0^{\ell_{\max}=5}$	0.961	0.943	0.923	0.906	1.087	1.070	1.050	1.034
H-Packer$_2^{\ell_{\max}=5}$	0.863	0.842	0.826	0.805	1.011	0.992	0.972	0.953
H-Packer$_5^{\ell_{\max}=5}$	0.858	0.837	0.821	0.800	1.002	0.984	0.964	0.945
H-Packer$_{up}^{\ell_{\max}=5}$	0.765	0.741	0.730	0.706	0.915	0.895	0.880	0.860

Table 2: **Atom RMSD (Å↓) across all residues with different treatments of the true structure. Rec:** reconstructing the true structure with our data-derived redundant internal coordinates. **Sym:** considering the additional non-natural symmetries used by AttnPacker.

Comparative Evaluation on CASP13 and CASP14 targets. Table 1 compare H-Packer's performance in side-chain packing with other computational methods [6, 7, 5, 3, 2, 4]. Despite its simplicity, H-Packer$_5$ is competitive against the state-of-the-art at predicting χ_3 and χ_4, but falls behind on χ_1 and χ_2 predictions. This discrepancy indicates that H-Packer has likely learned complementary features to the other models. Moreover, H-Packer mostly outperforms the physics-based computational algorithms [3, 2, 4] and is competitive with DLPacker [5] in all our performance metrics. Interestingly, H-Packer is consistently better than physics-based approaches in terms of overall Atom RMSD, but tends to fall shorter on Angle Accuracy. We present error distrubtions for H-Packer in Figures A.5, A.6, A.7, and A.8.

Interestingly, while H-Packer predictions are improved upon using refinement networks, the performance saturates after 2 steps of refinement; the accuracies after 5 iterations of refinement are comparable to those after only 2 steps (Table 1). Therefore, it is unlikely that further refinement could improve H-Packer's performance to reach its upper bound performance. We hypothesise that training H-Packer to produce confidence scores might help in developing site-specific convergence criteria to help bridge the gap [6, 7].

Ablation in $\ell_{\max}$. Table 3 shows how changing $\ell_{\max}$ (from 4 to 5) impacts the performance of H-Packer. For the same model capacity, using higher $\ell_{\max}$ consistently yields better performance, indicating that higher angular resolutions of the input can be beneficial for learning this task. This performance improvement comes with a trade-off in

		Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å↓		
	H-Packer$_5$	χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
CASP13	$\ell_{\text{max}} = 4$	24.27	29.76	46.32	**52.56**	53.5	68.8	43.0	0.878	0.594	1.081
	$\ell_{\text{max}} = 5$	**23.60**	**29.40**	**44.91**	52.91	**54.7**	**70.7**	**43.7**	**0.858**	**0.564**	**1.067**
CASP14	$\ell_{\text{max}} = 4$	30.36	34.38	48.76	50.62	43.5	64.1	32.5	1.024	0.648	1.260
	$\ell_{\text{max}} = 5$	**29.61**	**34.03**	**46.72**	**50.35**	**45.2**	**65.5**	**34.0**	**1.002**	**0.626**	**1.244**

Table 3: **Ablation in ℓ_{max}.** Metrics for the other H-packer models can be found in Table A.2.

Method	HPacker$_5^{\ell_{\text{max}}=5}$	HPacker$_2^{\ell_{\text{max}}=5}$	HPacker$_0^{\ell_{\text{max}}=5}$	DiffPack	AttnPack	DLPack	RosPack	FASPR	SCWRL4
Rel. Time	1.00	0.51	0.18	0.45	0.05	5.70	6.96	0.02	0.67

Table 4: **Relative times to undertake full atomic reconstruction.** In our current (unoptimized) implementation, HPacker$_5^{\ell_{\text{max}}=5}$ takes 1,482s to reconstruct the 82 CASP13 targets on a single NVIDIA A40 GPU. Times for the other methods were taken from [6], with the exception of DiffPack which was run locally on a single NVIDIA A40 GPU.

training and inference time, which scale superlinearly with ℓ_{max} unless the Tensor Product computation is adequately constrained [25]. For reference, training our models with $\ell_{\text{max}} = 5$ takes ∼40% longer than those with $\ell_{\text{max}} = 4$. We leave the hyperparameter optimization of ℓ_{max} to future work.

On computing RMSD fairly. In Table 1 we report RMSD computed by measuring the distance between the coordinates of true and predicted atoms, modulo the symmetries we report in A.1. However, other algorithms such as AttnPacker [6] consider other symmetries as well, sometimes even between atoms of differing chemical elements. Though these symmetries reflect spatially similar conformations (such as a flip of the Histidine ring), they result in inflated RMSD scores. We show the effect of this inflation on H-Packer predictions in Table 2. In the same table, we also show the RMSD computed against true structures that have been "reconstructed" using the true χ angles and the constant values that we use for redundant internal coordinate within H-Packer; we do this in an effort to disentangle the Null Reconstruction Error (Figure A.1) from the error given by mistakes in χ angle prediction.

Speed. Table 4 shows relative reconstruction speeds for several packing algorithms. Using the current implementation of the reconstruction algorithm, the best-performing H-Packer model is about 7x faster than the popular algorithm RosettaPacker and 6x faster than DLPacker; however, it is considerably slower than AttnPacker. Speed can be considerably cut down by half at the expense of minor performance degradation using two refinement iterations instead of five. However, more considerable speed gains may be achieved by CPU parallelization, when computing holographic encodings of structural neighborhood during initial data processing. Indeed, each initial guessing and refinement step of H-Packer predicts all χ angles at once, but in the current implementation holographic encodings are computed in series, creating a bottleneck that currently accounts for 88% of the inference time (10% is atom placement, and only 2% is making the actual predictions on GPU). We plan on optimizing this aspect in future iterations of the model.

5 Discussion

In this paper, we present H-Packer, a novel algorithm for predicting side-chain conformations by jointly regressing over the side-chain's χ angles. H-packer is composed of two simple and fast rotationally equivariant neural networks, the first one is used for making an *initial guess* using the coordinates of backbone atoms alongside residue identity information, while the second one *refines* the predictions by considering the predicted coordinates of the neighboring side-chain atoms. We carefully study three alternative objective functions, eventually deciding on using a geometrically justified loss function over the sine and cosine of χ angles. Our experiments show that H-packer is competitive against physics-based methods and some machine-learning solutions, but its performance still lags behind the state-of-the-art at predicting χ angles closer to the backbone. Overall, the lack of consistent comparative patterns in performance metrics suggests that H-Packer learns features complementary to other approaches. In addition, the formulation of H-packer makes it amenable to easy-to-achieve CPU parallelization to speed up its already fast inference predictions. We further emphasize that H-Packer is remarkably lightweight - 2×3M parameters vs. 208M of AttnPacker - and requires few resources to train - single GPU at < 1 hour per epoch vs. 4 GPUs for 400 epochs for DiffPack (unknown total time). Limitations of the model include: its inability to distinguish between covalent and non-covalent interactions as atomic interactions are not explicitly encoded into the network, and its inherently lower angular resolution further away from a neighborhood's center. Future areas of improvement include: enhancing angular resolution by scaling up ℓ_{max} while adjusting the architecture to reduce the resulting computational complexity, and training a confidence model for the predictions and using it to inform the refinement process.

6 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), 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.

References

[1] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, October 2022. Publisher: American Association for the Advancement of Science.

[2] Rebecca F. Alford, Andrew Leaver-Fay, Jeliazko R. Jeliazkov, Matthew J. O'Meara, Frank P. DiMaio, Hahnbeom Park, Maxim V. Shapovalov, P. Douglas Renfrew, Vikram K. Mulligan, Kalli Kappel, Jason W. Labonte, Michael S. Pacella, Richard Bonneau, Philip Bradley, Roland L. Dunbrack, Rhiju Das, David Baker, Brian Kuhlman, Tanja Kortemme, and Jeffrey J. Gray. The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *Journal of Chemical Theory and Computation*, 13(6):3031–3048, June 2017.

[3] Georgii G. Krivov, Maxim V. Shapovalov, and Roland L. Dunbrack Jr. Improved prediction of protein side-chain conformations with SCWRL4. *Proteins: Structure, Function, and Bioinformatics*, 77(4):778–795, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.22488.

[4] Xiaoqiang Huang, Robin Pearce, and Yang Zhang. FASPR: an open-source tool for fast and accurate protein side-chain packing. *Bioinformatics (Oxford, England)*, 36(12):3758–3765, June 2020.

[5] Mikita Misiura, Raghav Shroff, Ross Thyer, and Anatoly B. Kolomeisky. DLPacker: Deep learning for prediction of amino acid side chain conformations in proteins. *Proteins: Structure, Function, and Bioinformatics*, 90(6):1278–1290, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.26311.

[6] Matthew McPartlon and Jinbo Xu. An end-to-end deep learning method for protein side-chain packing and inverse folding. *Proceedings of the National Academy of Sciences*, 120(23):e2216438120, June 2023. Publisher: Proceedings of the National Academy of Sciences.

[7] Yangtian Zhang, Zuobai Zhang, Bozitao Zhong, Sanchit Misra, and Jian Tang. DiffPack: A Torsional Diffusion Model for Autoregressive Protein Side-Chain Packing, June 2023. arXiv:2306.01794 [cs, q-bio].

[8] Michael N. Pun, Andrew Ivanov, Quinn Bellamy, Zachary Montague, Colin LaMont, Philip Bradley, Jakub Otwinowski, and Armita Nourmohammad. Learning the shape of protein micro-environments with a holographic convolutional neural network, November 2022. arXiv:2211.02936 [physics, q-bio].

[9] Gian Marco Visani, Michael N. Pun, Arman Angaji, and Armita Nourmohammad. Holographic-(V)AE: an end-to-end SO(3)-Equivariant (Variational) Autoencoder in Fourier Space, June 2023. arXiv:2209.15567 [physics].

[10] Jay W. Ponder and David A. Case. Force fields for protein simulations. *Advances in Protein Chemistry*, 66:27–85, 2003.

[11] Abhishek Mukhopadhyay, Amit Kadan, Benjamin McMaster, J. Liam McWhirter, and Surjit B. Dixit. ZymePackNet: rotamer-sampling free graph neural network method for protein sidechain prediction, May 2023. Pages: 2023.05.05.539648 Section: New Results.

[12] Yang Cao, Lin Song, Zhichao Miao, Yun Hu, Liqing Tian, and Taijiao Jiang. Improved side-chain modeling by coupling clash-detection guided iterative search with rotamer relaxation. *Bioinformatics*, 27(6):785–790, March 2011.

[13] Shide Liang, Dandan Zheng, Chi Zhang, and Daron M. Standley. Fast and accurate prediction of protein side-chain conformations. *Bioinformatics*, 27(20):2913–2914, October 2011.

[14] Gang Xu, Qinghua Wang, and Jianpeng Ma. OPUS-Rota4: a gradient-based protein side-chain modeling framework assisted by deep learning-based predictors. *Briefings in Bioinformatics*, 23(1):bbab529, January 2022.

[15] Ken Nagata, Arlo Randall, and Pierre Baldi. SIDEpro: a novel machine learning approach for the fast and accurate prediction of side-chain conformations. *Proteins*, 80(1):142–153, January 2012.

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

[17] Stephan Eismann, Raphael J. L. Townshend, Nathaniel Thomas, Milind Jagota, Bowen Jing, and Ron O. Dror. Hierarchical, rotation-equivariant neural networks to select structural models of protein complexes. *Proteins: Structure, Function, and Bioinformatics*, 89(5):493–501, May 2021. arXiv:2006.09275 [cs, q-bio, stat].

[18] P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, February 2020. Number: 2 Publisher: Nature Publishing Group.

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

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

[21] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, May 2022. Number: 1 Publisher: Nature Publishing Group.

[22] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, February 2023. Number: 1 Publisher: Nature Publishing Group.

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

[24] R. P. Joosten, J. Salzemann, V. Bloch, H. Stockinger, A.-C. Berglund, C. Blanchet, E. Bongcam-Rudloff, C. Combet, A. L. Da Costa, G. Deleage, M. Diarena, R. Fabbretti, G. Fettahi, V. Flegel, A. Gisel, V. Kasam, T. Kervinen, E. Korpelainen, K. Mattila, M. Pagni, M. Reichstadt, V. Breton, I. J. Tickle, and G. Vriend. PDB_redo: automated re-refinement of X-ray structure models in the PDB. *Journal of Applied Crystallography*, 42(3):376–384, June 2009. Publisher: International Union of Crystallography.

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

[26] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, May 2021. arXiv:2104.13478 [cs, stat].

[27] Mario Geiger and Tess Smidt. e3nn: Euclidean Neural Networks, July 2022. arXiv:2207.09453 [cs].

[28] Wu-Ki Tung. *Group Theory in Physics*. 1985.

[29] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. arXiv:1607.06450 [cs, stat].

[30] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic Networks: Deep Translation and Rotation Equivariance, April 2017. arXiv:1612.04642 [cs, stat].

[31] SE(3)-Transformers for PyTorch | NVIDIA NGC.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

A Appendix

A.1 More rigorous mathematical background on SO(3)-Equivariance

Group Invariance and Equivariance. Intuitively, a function is said to be *invariant* to a certain group of transformations (e.g. 3D rotations) if applying one such transformation to the function's input does not change its output. *Equivariance* is a generalization of invariance whereby when the input is transformed by the action of a certain group element (or rather by a matrix representation parameterized by the group element) the output of the function is transformed by the same group element (i.e., by a matrix representation parameterized by the same group element, but that can be different from the input's representation). In short, equivariant functions transform the input in the same way regardless of its coordinate frame, but do not necessarily discard the coordinate frame information, whereas invariant functions also do the latter. Both of these concepts can be extended to *properties* as well, e.g. "the mass of a molecule remains constant (is invariant) when rotating it, whereas its dipole moment rotates alongside it (is equivariant)". More formally, a function between two vector spaces $f : X \to Y$ is said to be equivariant to a group of transformations $\mathfrak{G}$ iff applying any group transformation to the input space of f corresponds to applying the same transformation to the output space (i.e., via a *representation* parametrized by the same group element). Formally: $f(\boldsymbol{D}_X(\mathfrak{g})\boldsymbol{x}) = \boldsymbol{D}_Y(\mathfrak{g})f(\boldsymbol{x}), \forall \boldsymbol{x} \in X \wedge \forall \mathfrak{g} \in \mathfrak{G}$. The group acts on the input and output vector spaces with space-specific representations that are appropriate for the space (i.e., $\boldsymbol{D}_X$ and $\boldsymbol{D}_Y$). A group may have different representations, and a special one is the one that always maps to the identity: $\boldsymbol{D}_Y(\mathfrak{g}) = 1, \forall \mathfrak{g} \in \mathfrak{G}$; a function on whose output space $\mathfrak{G}$ acts with the identity representation is said to be *invariant* to $\mathfrak{G}$. In the context of machine learning, building models for which the output is provably invariant/equivariant to the same groups as the target function can avoid expensive data augmentation. However, even when fitting invariant functions, using equivariant layers is advisable - if not necessary [26].

Irreducible representations. How are equivariant layers generally achieved? The key is to look at the group's *irreducible representations* (irreps). These are the group's *smallest* representations, so that any possible representation can be provably decomposed into a direct sum of irreps. Therefore, the group's irreps can be used to describe how the group elements act on any vector space. We can use this fact to build group-equivariant functions by ensuring that both the input and output of the function are composed (via direct sum i.e., concatenation) of features that transform under the group's action under the group's irreps.

SO(3)-Equivariance. The above is often easier said than done, but it has been worked out for SO(3), which is a group describing 3D rotations about a fixed point [27, 23, 20]. Spherical Fourier space can be used to conveniently define equivariant transformation for rotations. For rotations about a given reference point, the points in 3D can be expressed by the resulting spherical coordinates (r, θ, ϕ) about the set origin. Since the radius r (i.e., the distance of a point from to the reference) does not change under rotations about the origin, we will ignore the radial component for now and consider a signal over the sphere of radius r, $f(\theta, \phi) : \ S^2(r) \to \mathbb{R}$. The Fourier transform $\hat{F}$ of the signal on the sphere follows,

$$\hat{F}_{\ell m} = \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi) Y_{\ell m}(\theta, \phi) \sin\theta \, d\theta \, d\phi \tag{A.1}$$

where $Y_{\ell m}(\theta, \phi)$ is the spherical harmonic of degree ℓ and order m 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. The operators that describe how spherical harmonics transform under rotations are called the Wigner D-matrices, denoted by $D^\ell_{mm'}(R)$ [28].

$$Y_{\ell m}(\theta, \phi) \overset{\text{rotation: } R}{\longrightarrow} \sum_{m'=-\ell}^{\ell} D^\ell_{m'm}(R) Y_{\ell m'}(\theta, \phi) \tag{A.3}$$

Indeed, Wigner-D matrices are the irreps of SO(3). Therefore, any vector space that "3D-rotates" can be decomposed into a direct sum of type-ℓ features that transform according to the irrep of type ℓ. For example, features of type 0 are invariant to rotation (e.g. atomic mass), while features of type 1 transform as geometric vectors (e.g. dipole moment). Thus, to build an SO(3)-equivariant model we start by projecting the data onto a convenient SO(3)-equivariant basis via the spherical harmonics. We then leverage a suite of rules that allows one to build learnable layers without breaking equivariance, and transform the input into a new representation composed of features within the same range of possible types. One key transformation rule is the Clebsch-Gordan Tensor Product [28] that is commonly used to inject nonlinearity (to be precise bi-linearity) in the SO(3) equivariant neural networks [23].

A.2 Equivariant architecture components

Linearity. The equivariant linear layer consists of a set of linear projections each acting on the set of features sharing the same type ℓ. It practically equates to $\ell_{\text{max}} + 1$ standard linear layers with no bias except for the $\ell = 0$ case, and with the consideration that all $2\ell + 1$ moments of the same feature are processed together. Formally, let $\boldsymbol{h}_\ell \in \mathbb{R}^{C\times(2\ell+1)}$ be a set of C features of type ℓ. Then, we learn weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{C\times K}$ that linearly maps h_ℓ to $\overline{h}_\ell \in \mathbb{R}^{K\times(2\ell+1)}$; if $\ell = 0$, $\mathbf{b}_\ell \in \mathbb{R}^K$ is also learned:

$$\overline{\boldsymbol{h}}_\ell = \boldsymbol{W}_\ell^T \boldsymbol{h}_\ell \ (+\mathbf{b}_\ell) \tag{A.4}$$

Tensor Product Nonlinearity. Arguably the most important SO(3)-Equivariant operation is the Clebsch-Gordan (CG) Tensor Product. It is the only known operation capable of nonlinearly coupling (i.e. mixing information between) features of different type ℓ and with different momentum index m. 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,

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

where $C_{(\ell_1 m_1)(\ell_2 m_2)}^{(\ell_3 m_3)}$ are the Clebsch-Gordan coefficients. Similar to spherical harmonics, Clebsch-Gordan tensor products also appear in quantum mechanics, and they are used to express couplings between angular momenta.

Here, we use the CG Tensor Product as the primary nonlinear activation of our networks, as originally prescribed by [23], and crucially the only operation that can transfer information across features of different types ℓ. Following [25] and [9], we compute the Tensor Product feature-wise (i.e. we do not compute it across features with a different index) to significantly decrease computation time. We refer the reader to [25] and [9] for details.

Layer Norm Nonlinearity. This consists of applying a standard Layer Norm [29] to the *norms* of type-ℓ features (which are invariant), and then feeding the normalized norm into a standard nonlinear activation function. This layer effectively combines the equivariant layer norm in `e3nn` [27] with the Norm Nonlinearity originally used in Harmonic Networks [30], and then adapted to the SO(3) domain by Tensor Field Networks [20]. It is also used in the SE(3)-Transformer [16, 31].

The SO(3)-equivariant architecture components were implemented with the help of `e3nn` primitives [27]. We emphasize that we do not ablate all the architectural components, and different choices would be possible. Specifically, we do not ablate over the choice of normalization and of invariant nonlinearity function: in preliminary experiments, other options seemed to perform equally. Other components are necessary or were otherwise found to be useful. For example the tensor product is necessary to ensure that information flows across different ℓs, and tuning dropout was found to be greatly useful to prevent overfitting.

A.3 Proof of equivalence between MSE and cosine loss in predicting Sine and Cosine of χ angles

Let χ_i and $\hat{\chi}_i$ be the true and the predicted angle values for a residue's i^{th} χ angle, respectively. Let $\mathbf{v}_i \equiv [\sin\chi_i, \cos\chi_i]$ and $\hat{\mathbf{v}}_i \equiv [\sin\hat{\chi}_i, \cos\hat{\chi}_i]$ denote their respective 2D vectors on the unit circle, whose two components are the sine and cosine of the angles themselves. Then we have:

$$\begin{aligned}\text{MSE}_{\text{sin-cos}}(\hat{\chi}_i, \chi_i) &= \frac{1}{2}(\cos\hat{\chi}_i - \cos\chi_i)^2 + \frac{1}{2}(\sin\hat{\chi}_i - \sin\chi_i)^2 \\ &= 1 - \cos\hat{\chi}_i\cos\chi_i - \sin\hat{\chi}_i\sin\chi_i \\ &= 1 - \langle\hat{\mathbf{v}}_i, \mathbf{v}_i\rangle \\ &= \text{CosineLoss}(\hat{\mathbf{v}}_i, \mathbf{v}_i)\end{aligned} \tag{A.6}$$

where the jump from step 1 to step 2 is granted by the trigonometric identity $\sin^2\theta + \cos^2\theta = 1$.

A.4 Training Details

A.4.1 Toy Task

All models were constructed with 5 equivariant blocks and no invariant feed-forward neural network (FFNN) for the two models predicting invariant quantities. All models were trained using Adam [32] for 30 epochs with learning rate 0.001 and a batch size of 32. The model exhibiting lowest validation loss was used.

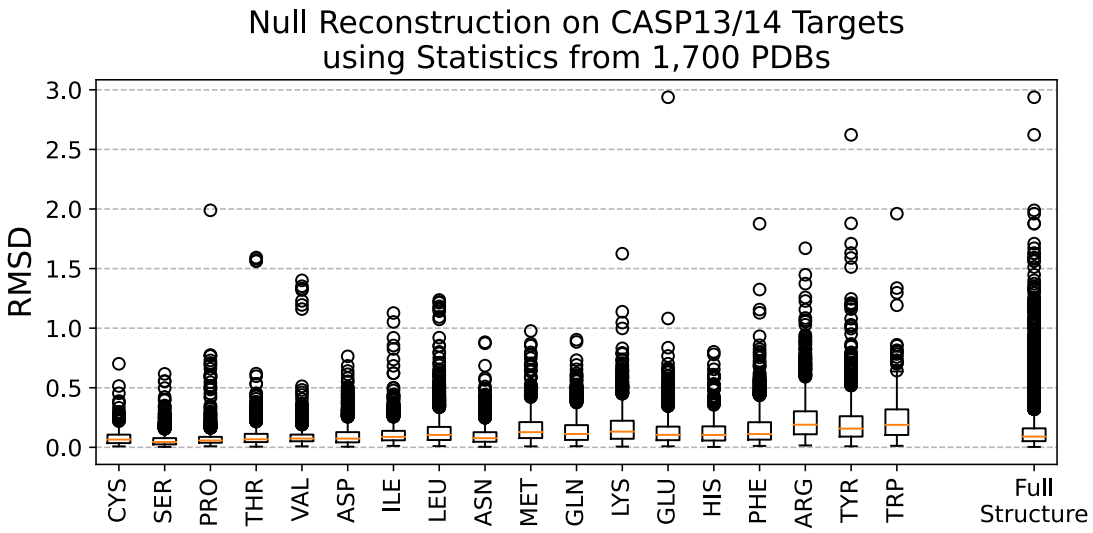

Figure A.1: **Null Reconstruction Error on Test data by using data-derived values as redundant internal coordinates.** Effectively, each redundant internal coordinate (i.e. excluding chi angles) gets substituted from a single value computed as the median of the corresponding value in a reference dataset. Across all structures, the average Null Reconstruction RMSD is 0.127 Å. We notice a small number of outliers (single residues with abnormally high RMSD), but we do not investigate the causes.

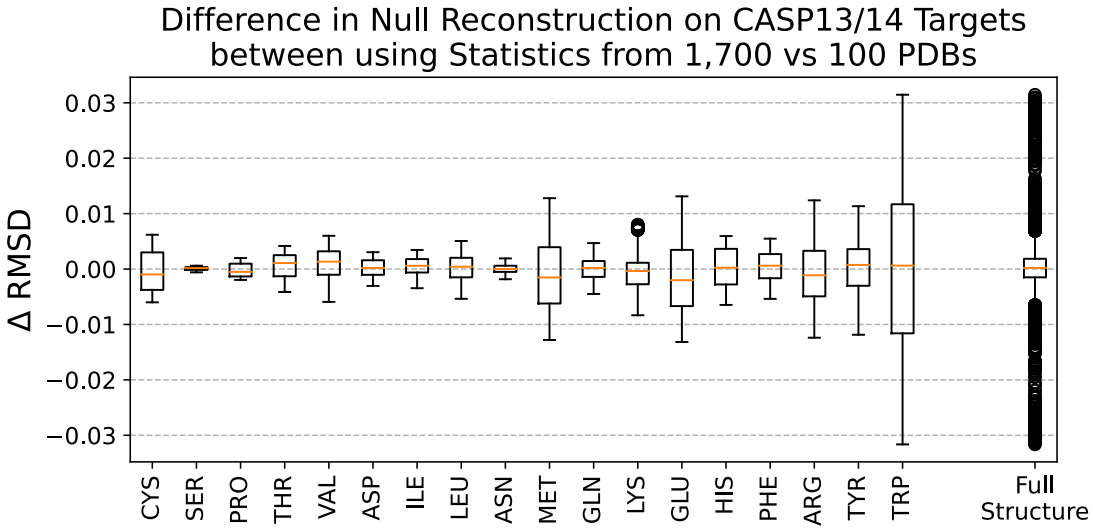

Figure A.2: **Difference in Null Reconstruction Error on Test data by using a smaller set of structures for reference.**

A.4.2 Side-chain Packing

Models with $\ell_{\text{max}} = 4$ have five equivariant blocks with a per-ℓ hidden feature side of 128. For models with $\ell_{\text{max}} = 5$ we use 5 blocks with per-ℓ size 96. In doing so, the two models have comparable number of parameters (3M). All models have a 3-layer FFNN with silu nonlinearity and dropout normalization rate of 0.1. We found it useful to tune he dropout rate to prevent overfitting. We train all models for 10 epochs, keeping the model with lowest validation loss at the end of an epoch (convergence usually happened by epoch $\sim$8); models with $\ell_{\text{max}} = 5$ took roughly 50 minutes per epoch to train on a single NVIDIA A40 GPU, while models with $\ell_{\text{max}} = 4$ took 35 minutes.

Table A.1: **List of conformational symmetries that we take into consideration.**

Amino-Acid	π-Symmetric χ	Pairs of Equivalent Atom names
ARG	-	NH1-NH2
TYR	χ_2	CD1-CE1, CD2-CE2
PHE	χ_2	CD1-CE1, CD2-CE2
ASP	χ_2	OD1-OD2
GLU	χ_3	OE1-OE2

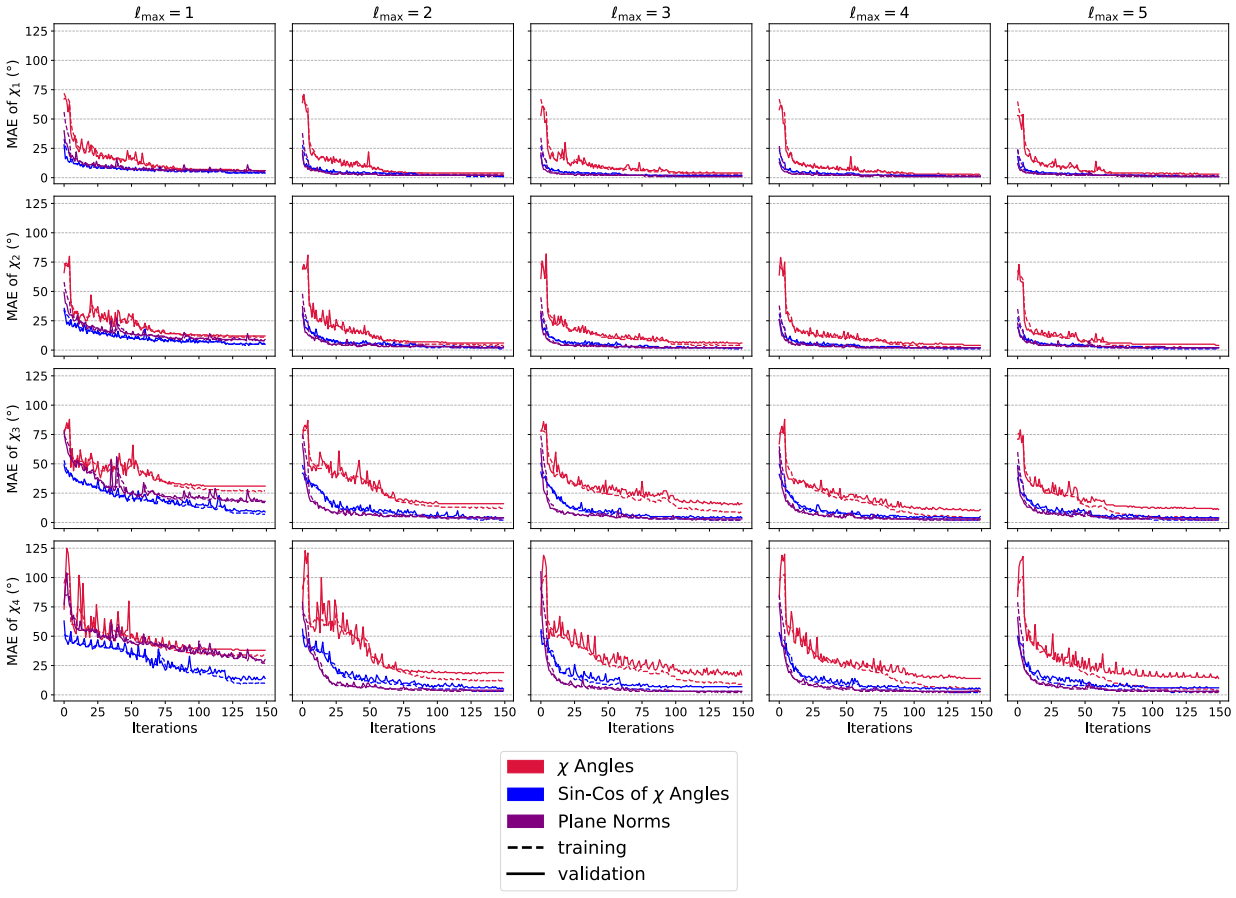

Figure A.3: χ **angles error trace during training for the simple task of predicting** χ **angles from true amino-acid conformations.** The Sin-Cos and Plane Norms models show comparable convergence curves, except for $\ell_{\text{max}} = 1$ where the Plane Norms model struggles with χ_3 and χ_4. The Angles model has the worst convergence trace, even overfitting for high ℓ_{max}.

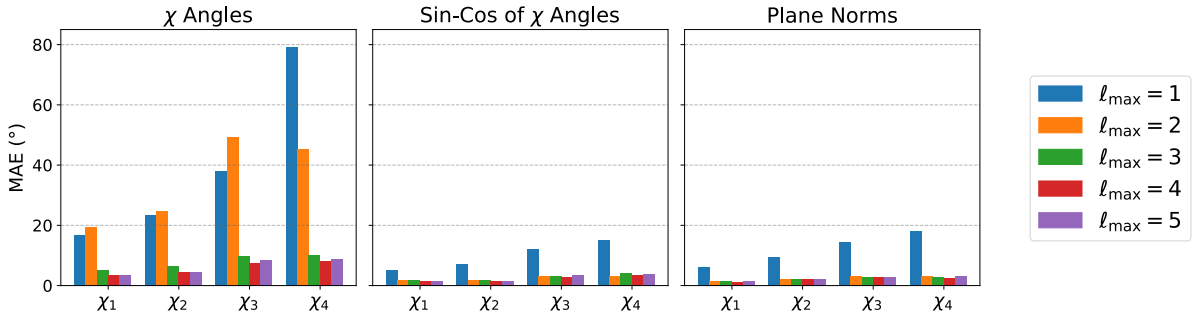

Figure A.4: **Test MAE for the simple task of predicting χ angles from atomic conformation, with loss weighted by frequency.** Models trained with weighting χ angles in the loss function based on their frequency. We use the following weights for χ_1 to χ_4, which are derived from how many residues have which χ angle: $[1.0, 18.0/14.0, 18.0/5.0, 18.0/2.0]$. We then re-scale them to make sure the absolute scale of the loss function is the same as it was in the un-weighted case.

		Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å↓		
	Method	χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
CASP13	H-Packer$_0$ $\ell_{\text{max}} = 4$	27.60	32.31	47.87	52.77	48.3	59.1	**40.5**	0.979	0.771	1.136
	H-Packer$_0$ $\ell_{\text{max}} = 5$	**26.89**	**31.95**	**47.51**	**52.75**	**49.3**	**61.5**	40.4	**0.961**	**0.726**	**1.131**
	H-Packer$_2$ $\ell_{\text{max}} = 4$	24.36	29.94	46.01	**52.89**	53.0	67.7	42.5	0.884	0.610	1.083
	H-Packer$_2$ $\ell_{\text{max}} = 5$	**23.64**	**29.47**	**45.17**	53.26	**54.4**	**69.9**	**43.6**	**0.863**	**0.575**	**1.070**
	H-Packer$_5$ $\ell_{\text{max}} = 4$	24.27	29.76	46.32	**52.56**	53.5	68.8	43.0	0.878	0.594	1.081
	H-Packer$_5$ $\ell_{\text{max}} = 5$	**23.60**	**29.40**	**44.91**	52.91	**54.7**	**70.7**	**43.7**	**0.858**	**0.564**	**1.067**
	H-Packer$_{up}$ $\ell_{\text{max}} = 4$	20.78	27.48	44.58	**51.97**	56.8	72.7	45.6	0.790	0.514	0.999
	H-Packer$_{up}$ $\ell_{\text{max}} = 5$	**20.03**	**26.88**	**42.74**	52.07	**58.4**	**75.4**	**46.4**	**0.765**	**0.483**	**0.980**
CASP14	H-Packer$_0$ $\ell_{\text{max}} = 4$	33.14	36.10	49.30	**50.22**	39.8	55.9	30.7	1.102	0.780	1.307
	H-Packer$_0$ $\ell_{\text{max}} = 5$	**32.31**	**35.90**	**49.05**	50.34	**40.8**	**57.0**	**31.6**	**1.087**	**0.762**	**1.297**
	H-Packer$_2$ $\ell_{\text{max}} = 4$	30.43	34.49	48.13	**50.39**	43.3	63.0	32.7	1.027	0.658	1.265
	H-Packer$_2$ $\ell_{\text{max}} = 5$	**29.96**	**34.32**	**47.46**	50.50	**45.0**	**65.1**	**34.1**	**1.011**	**0.629**	**1.250**
	H-Packer$_5$ $\ell_{\text{max}} = 4$	30.36	34.38	48.76	50.62	43.5	64.1	32.5	1.024	0.648	1.260
	H-Packer$_5$ $\ell_{\text{max}} = 5$	**29.61**	**34.03**	**46.72**	**50.35**	**45.2**	**65.5**	**34.0**	**1.002**	**0.626**	**1.244**
	H-Packer$_{up}$ $\ell_{\text{max}} = 4$	26.84	32.10	46.90	50.03	46.7	68.1	35.0	0.934	0.557	1.178
	H-Packer$_{up}$ $\ell_{\text{max}} = 5$	**26.58**	**31.54**	**45.67**	**49.46**	**48.1**	**69.5**	**36.2**	**0.915**	**0.534**	**1.160**

Table A.2: **Ablation in ℓ_{max}.**

	Atom RMSD Å↓					
	CASP13			CASP14		
Method	All	Core	Surface	All	Core	Surface
H-Packer$_0$ $\ell_{\text{max}} = 4$	0.943	0.733	1.099	1.065	0.743	1.273
H-Packer$_2$ $\ell_{\text{max}} = 4$	0.848	0.577	1.046	0.989	0.618	1.230
H-Packer$_5$ $\ell_{\text{max}} = 4$	0.841	0.561	1.043	0.986	0.608	1.223
H-Packer$_{up}$ $\ell_{\text{max}} = 4$	0.755	0.483	0.962	0.899	0.526	1.143
H-Packer$_0$ $\ell_{\text{max}} = 5$	0.923	0.689	1.092	1.050	0.723	1.262
H-Packer$_2$ $\ell_{\text{max}} = 5$	0.826	0.540	1.032	0.972	0.591	1.214
H-Packer$_5$ $\ell_{\text{max}} = 5$	0.821	0.530	1.029	0.964	0.589	1.208
H-Packer$_{up}$ $\ell_{\text{max}} = 5$	0.730	0.452	0.941	0.880	0.504	1.124

Table A.3: **Reconstruction RMSD when considering the additional (non-natural) symmetries used by AttnPacker.**

	Atom RMSD Å↓					
	CASP13			CASP14		
Method	All	Core	Surface	All	Core	Surface
H-Packer$_0$ $\ell_{\text{max}} = 4$	0.964	0.745	1.129	1.086	0.754	1.297
H-Packer$_2$ $\ell_{\text{max}} = 4$	0.863	0.576	1.074	1.010	0.628	1.255
H-Packer$_5$ $\ell_{\text{max}} = 4$	0.858	0.560	1.071	1.007	0.617	1.249
H-Packer$_{up}$ $\ell_{\text{max}} = 4$	0.767	0.477	0.987	0.915	0.526	1.167
H-Packer$_0$ $\ell_{\text{max}} = 5$	0.943	0.697	1.123	1.070	0.735	1.286
H-Packer$_2$ $\ell_{\text{max}} = 5$	0.842	0.540	1.060	0.992	0.598	1.239
H-Packer$_5$ $\ell_{\text{max}} = 5$	0.837	0.529	1.057	0.984	0.596	1.232
H-Packer$_{up}$ $\ell_{\text{max}} = 5$	0.741	0.444	0.968	0.895	0.500	1.147

Table A.4: **Reconstruction RMSD when reconstructing the true structure using our data-derived constant redundant internal coordinates.**

	Atom RMSD Å↓					
	CASP13			CASP14		
Method	All	Core	Surface	All	Core	Surface
H-Packer$_0$ $\ell_{\text{max}} = 4$	0.927	0.708	1.092	1.049	0.717	1.263
H-Packer$_2$ $\ell_{\text{max}} = 4$	0.828	0.543	1.038	0.973	0.588	1.220
H-Packer$_5$ $\ell_{\text{max}} = 4$	0.821	0.527	1.033	0.969	0.577	1.213
H-Packer$_{up}$ $\ell_{\text{max}} = 4$	0.733	0.447	0.950	0.881	0.494	1.132
H-Packer$_0$ $\ell_{\text{max}} = 5$	0.906	0.661	1.084	1.034	0.696	1.251
H-Packer$_2$ $\ell_{\text{max}} = 5$	0.805	0.506	1.023	0.953	0.560	1.202
H-Packer$_5$ $\ell_{\text{max}} = 5$	0.800	0.496	1.019	0.945	0.558	1.195
H-Packer$_{up}$ $\ell_{\text{max}} = 5$	0.706	0.414	0.930	0.860	0.471	1.111

Table A.5: **Reconstruction RMSD when considering both the additional (non-natural) symmetries used by AttnPacker, and reconstructing the true structure using our data-derived constant redundant internal coordinates.**

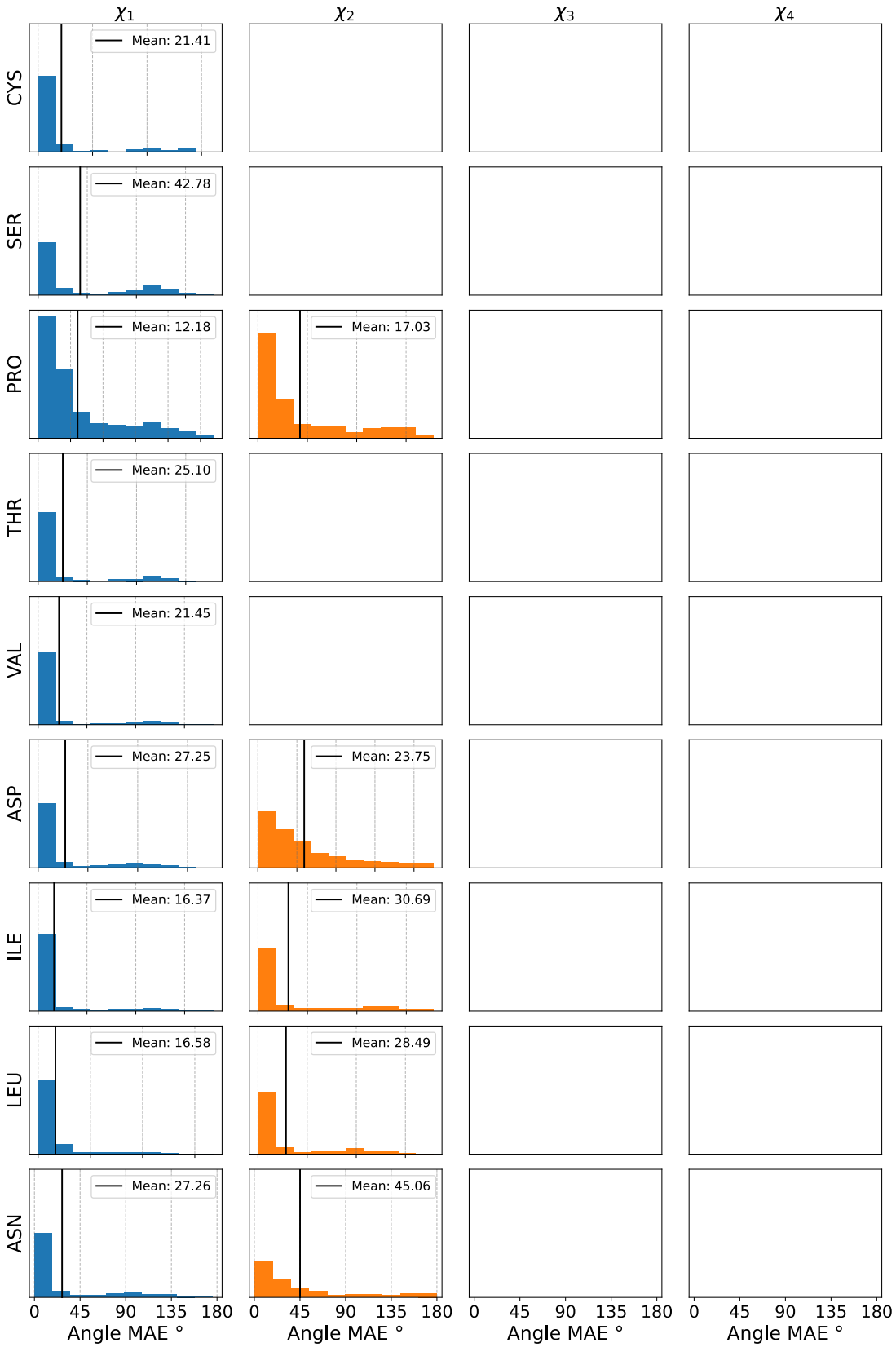

Figure A.5: **Angle Error distribution on CASP13 for H-PAcker$_5^{\ell_{\max}=5}$, part 1.**

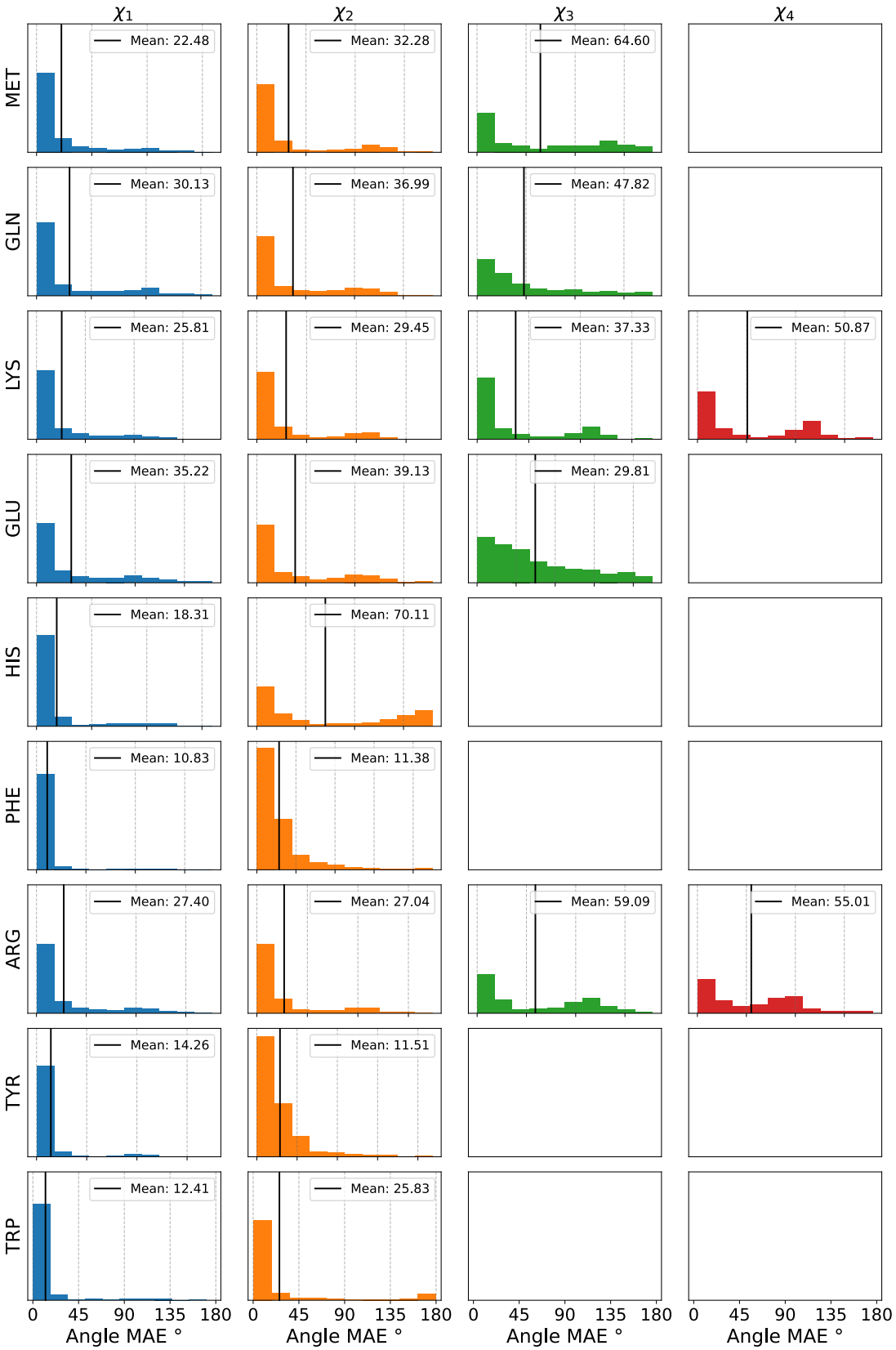

Figure A.6: **Angle Error distribution on CASP13 for H-PAcker$_5^{\ell_{\max}=5}$, part 2.**

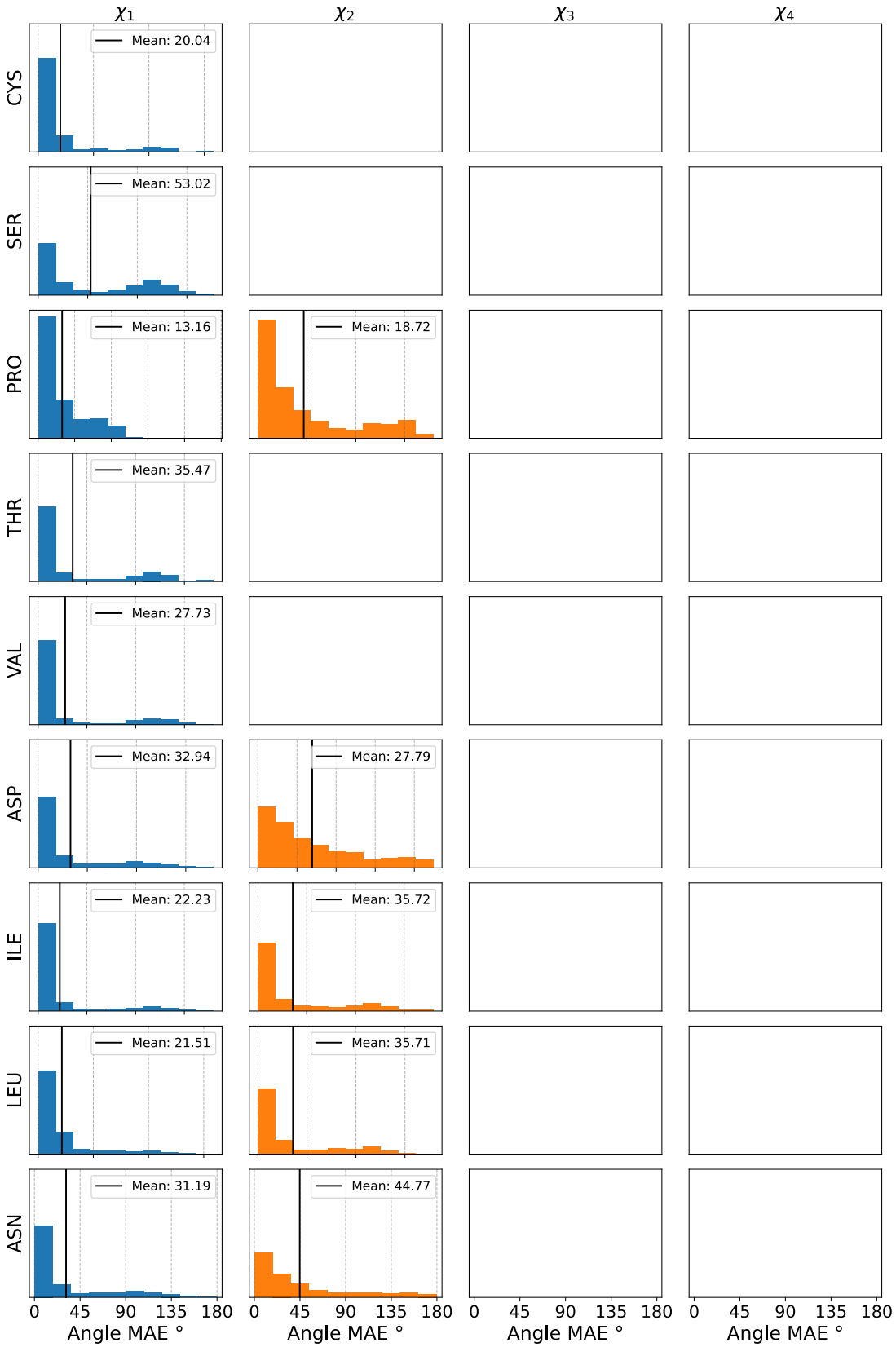

Figure A.7: **Angle Error distribution on CASP14 for H-PAcker$_5^{\ell_{\max}=5}$, part 1.**

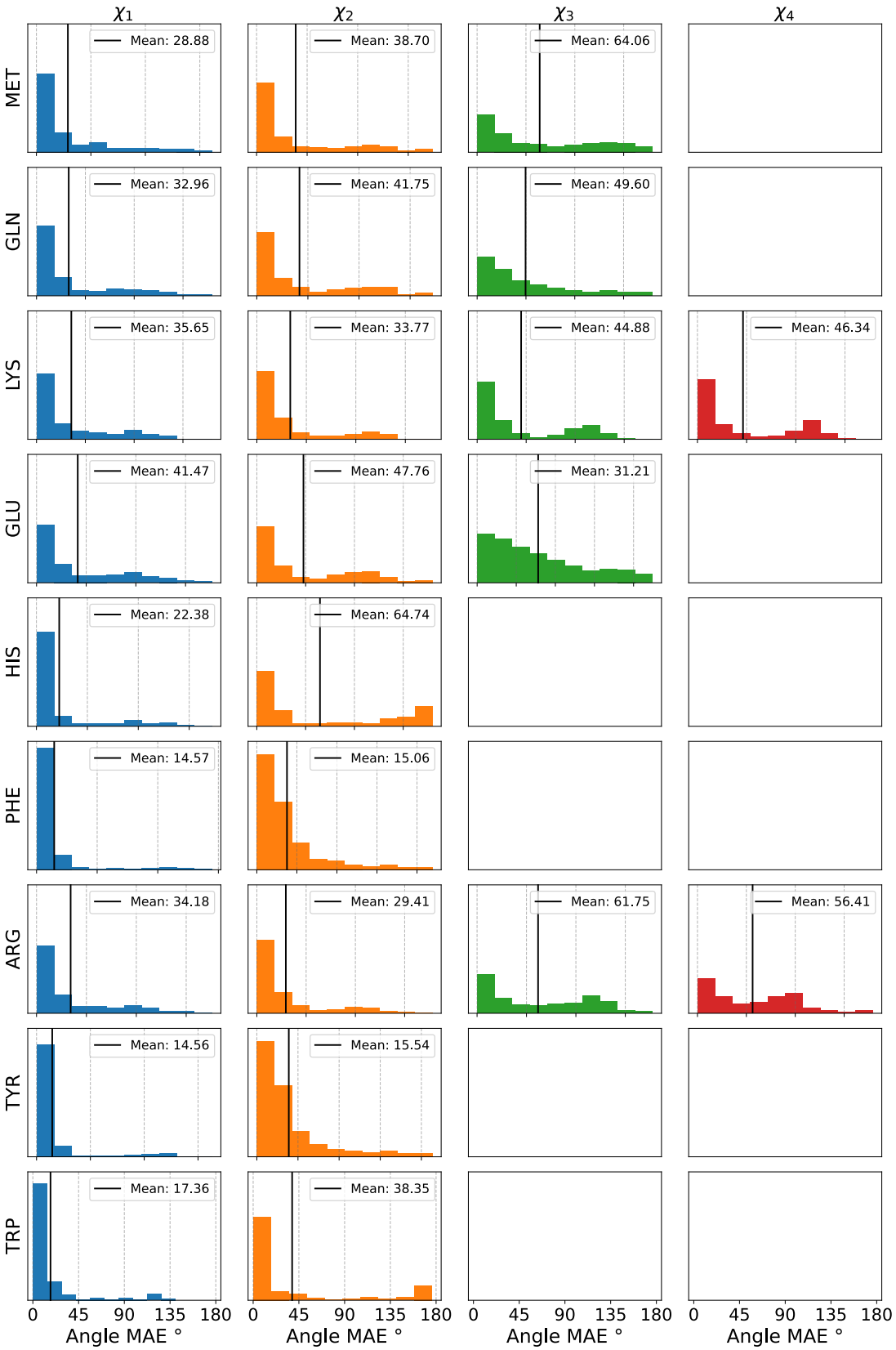

Figure A.8: **Angle Error distribution on CASP14 for H-PAcker$_5^{\ell_{\max}=5}$, part 2.**