# Invariant point message passing for protein side chain packing and design

Nicholas Randolph[1,2] and Brian Kuhlman[1,2]

[1]Department of Bioinformatics and Computational Biology, University of North Carolina School of Medicine, Chapel Hill, North Carolina, USA.

[2]Department of Biochemistry and Biophysics, University of North Carolina School of Medicine, Chapel Hill, North Carolina, USA.

**Abstract**
Protein side chain packing (PSCP) is a fundamental problem in the field of protein engineering, as high confidence and low energy conformations of amino acid side chains are crucial for understanding (and designing) protein folding, protein-protein interactions, and protein-ligand interactions. Traditional PSCP methods (such as the Rosetta Packer) often rely on a library of discrete side chain conformations, or rotamers, and a forcefield to guide the structure to low energy conformations. Recently, deep learning (DL) based methods (such as DLPacker, AttnPacker, and DiffPack) have demonstrated state-of-the-art predictions and speed in the PSCP task. Building off the success of graph and message passing neural networks for protein modeling, we present the Protein Invariant Point Packer (PIPPack) which effectively processes local structural and sequence information to produce realistic, idealized side chain coordinates using $\chi$-angle distribution predictions and geometry-aware invariant point message passing (IPMP). To demonstrate its broad applicability to protein-related tasks, IPMP was additionally incorporated in a fixed backbone protein design method, which enabled the generation of more native-like sequences than common message passing schemes. On a test set of ~1,400 high-quality protein chains, PIPPack performs competitively with other state-of-the-art PSCP methods in rotamer recovery and per-residue RMSDs but is significantly faster.

## Introduction

The myriad of complex functions facilitated by proteins as well as many intrinsic properties of proteins, such as folding and stability, are dependent on the interactions and conformations adopted by the protein's amino acid side chains. Accurate modeling of side chains is therefore important for understanding structure-function relationships as well as designing new protein functions. The protein side chain packing (PSCP) problem has been traditionally formulated as a guided search over a library of discrete side chain conformations, or rotamers, given a protein backbone and its amino acid sequence[1]. There have been decades of research into and development of rotamer libraries that effectively capture the distribution of conformations observed for each amino acid in naturally occurring proteins[2–11]. Evaluating the favorability of individual rotamers in a residue's environment often entails an energy function that models various physical phenomena such as hydrogen bonding and van der Waals interactions[12–17]. The final component of many traditional PSCP methods is a search strategy by which rotamers are sampled and evaluated across the entire protein[11,18–20]. Research in each of these individual components has led to the development of many traditional physics-based PSCP methods that have been successfully employed in a variety of applications.

Recently, the protein modeling field has been experiencing remarkable breakthroughs largely due to deep learning (DL) methods taking advantage of the growing amount of experimental protein data. Of particular significance, protein structure prediction networks, such as AlphaFold2 (AF2)[21] and RoseTTAFold2[22], have made large strides in predicting overall protein fold to near-experimental accuracy in many cases. While these methods produce coordinates for all heavy atoms in the protein and are, therefore, capable of side chain packing, there is no way to pre-specify and hold the backbone conformation fixed to mimic the PSCP task. On the other hand, DL-based

methods designed specifically for PSCP have shown significant accuracy improvements over traditional approaches, while often being less time-consuming[23–29]. These methods either directly predict the location of side chain atoms[25,27] or torsion angles from which the side chain can be reconstructed with idealized geometry[23,24,28]. Some of these methods rely on a rotamer library to select a specific conformation[25,26] and some require subsequent post-processing to correct chemical and geometric violations and/or atomic clashes[27,29]. Importantly, all these methods are trained and evaluated on data to which no quality filters or only structure-level (e.g., resolution and R-factor) quality filters have been applied. Global structure-level filters provide some level of certainty in the overall model's integrity but provide no guarantees for the quality of local substructure or even individual residues, many of which may be unresolved or exist in multiple conformations.

Building off the recent success of graph neural networks for encoding and propagating structural information within proteins [27,30–33], we present the Protein Invariant Point Packer (PIPPack) which can rapidly and accurately predict side chain conformations. To combat potentially unreliable or uncertain side chains within training and evaluation data, we apply runtime B-factor filters and take advantage of the Top2018 "best-parts" dataset[34] that contains about 15,000 protein chains and was specifically curated at a chain- and residue-level such that there is excellent agreement between experimental data and the final structural model. Our model was trained on about 12,000 Top2018 chains to jointly predict binned $\chi$ dihedral angles for each residue, refining its previous predictions with recycling and explicitly incorporating, throughout the network, the geometry of the protein backbone through a novel message passing scheme. This message passing scheme can be viewed as a generalization of the invariant point attention (IPA) module introduced in AF2[21] and, as such, is named invariant point message passing (IPMP). We demonstrate that incorporating

IPMP for rotamer prediction provides a boost in performance over standard message passing and neighborhood-based IPA. To evaluate the applicability of IPMP to other protein-related tasks, we replaced the message passing layers of the recent protein design method ProteinMPNN[32] with IPMP layers and observed improvements in sequence perplexity and recovery. The training and inference code of our PyTorch implementation of PIPPack (and the modified ProteinMPNN) is publicly available on GitHub at https://github.com/Kuhlman-Lab/PIPPack.

## Methods

### *Top2018 Dataset Preparation*

The data used for training and evaluation was the Top2018 main chain-filtered dataset (v2.01) created as described in Williams *et al*.[34]. Briefly, protein chains released prior to the start of 2019 that were solved by x-ray crystallography with a resolution of 2.0 or better were selected from the Protein Data Bank (PDB)[35]. They were subsequently filtered at a chain level to a set with low MolProbity[36] scores and few structural geometry outliers. Filtered chains were then clustered using MMseqs2[37] to various sequence identity levels to reduce redundancy. For our models, we trained and evaluated using protein chains clustered at 40% identity. Next, the chains were subjected to filters applied to the main chain (N, Cα, C, O, and Cβ) atoms of each residue. Specifically, residues were removed from each chain if any atom under consideration had a high B-factor, geometry outlier, alternate location, steric clash, or did not agree with the experimental data. Finally, chains with more than 40% of its residues removed were discarded. In the end at 40% sequence identity, there were 10,449 clusters, of which 8,361 were used for training, 1,044 for validation, and 1,044 for testing. Finally, we removed any chains that had 40% sequence identity or more to a chain in the CASP13/14 test sets (see below, $n = 439$), as determined by MMseqs2's `easy-search` workflow.

Note Williams *et al.*[34] additionally created a full residue-filtered dataset wherein the same residue-level filters were applied to all heavy atoms in a residue. We decided to train on the main chain filtered data primarily for two reasons. First, stricter filters in the full residue-filtered data results in fewer residues and, therefore, fewer training examples. Second, removal of residues whose side chain atoms do not pass the filters, but whose backbone atoms do results in loss in valuable training signal that may influence rotamer placement as the N and O backbone atoms can participate in hydrogen bonding.

*BC40 Dataset Preparation*

To further assess the impact of high-quality data on the PSCP, we additionally experimented with training models on the BC40 data[38], which has been used recently for training data in PSCP methods. This dataset consists of 36,970 protein chains released before August 2020 that are nonredundant at 40% sequence identity but have no other filters. It was originally constructed for the protein secondary structure prediction task. We obtained the PDB code and chain id for each chain in the dataset, downloaded the coordinate files directly from the PDB, and extracted the appropriate chain. Prior to randomly splitting the data into training (90%) and validation (10%) sets, we removed any chains that had 40% sequence identity or more to a chain in the Top2018 test set (see above, $n = 5{,}066$) and the CASP13/14 test sets (see below, $n = 736$), as determined by MMseqs2's `easy-search` workflow.

*CASP13 and CASP14 Test Set Preparation*

In addition to the high-quality Top2018 test set, we evaluated our method on protein targets from the CASP13 ($n = 82$) and CASP14 ($n = 64$) competitions, like other recent PSCP methods[23,24,27,29]. Similar to the BC40 dataset, there are no structure-, chain-, nor residue-level filters, but this data was originally used to evaluate protein structure prediction methods. We

acquired the PDB files for each target directly from the data archives of the CASP prediction center.

*Architectural Considerations*

Graph neural networks (GNNs) have shown remarkable promise in modeling proteins and have been successfully applied to various protein tasks, including fold classification[39,40], property prediction[39–41], fixed backbone sequence design[27,30,32,33,42], and PSCP[24,27–29]. With the rationale that the specific side chain conformations are primarily dependent upon the local environment of the amino acid, we decided to model the PSCP problem with a GNN, wherein each residue is modelled as a node and is connected to its $k$ nearest neighbors. Due to the various symmetries of proteins in 3D Euclidean space, special considerations must be taken in the formulation of the network, either preserving equivariance or invariance to global rotations and translations. Most networks for proteins that preserve equivariance due so by operating on equivariant features and predictions (e.g., coordinates) with specialized, equivariant neural network layers (e.g., SE(3) transformer[43]) to ensure that the effects of global transformations are propagated throughout the network[22,33,44]. Invariance, on the other hand, ensures that global transformations do not affect the network and is often maintained using invariant features and predictions (e.g., relative orientations) and invariant layers (e.g., IPA or standard multi-layer perceptron)[21,42,45]. PIPPack is an invariant GNN that maintains its invariance through the choice of features, predictions, and layers (Fig. 1).

*Initializing the protein graph.* PIPPack represents an input protein as a graph $G = (V, E)$ with a node for each residue and edges connecting nodes to their nearest neighbors (Fig. 1A). Node features include a one-hot embedding of the amino acid sequence and the sine and cosine of the backbone dihedral angles $\phi$, $\psi$, and $\omega$. The edges between residues contains a one-hot embedding of the relative sequence position and backbone-backbone atomic distances encoded with Gaussian

radial basis functions. Edges are formed between each residue and its $k$ nearest neighbors (we use $k = 30$), determined by Cα-Cα distances. Note that all these features are invariant to global transformations. Additionally, for IPMP (see below, Fig. 1B), we obtain the rigid transformations that define the backbones of each residue.

*Forming the predictions*. The output of PIPPack is a packed protein structure, complete with all heavy atoms for each residue. We parameterize the PSCP task by predicting the $\chi$ dihedral angles for each residue, reconstructing the side chain with ideal bond lengths and angles. Most neural networks that predict side chain conformations via dihedrals perform a regression task on $\sin(\chi)$ and $\cos(\chi)$ for all $\chi$ angles, but we instead framed PSCP as a classification task by predicting the distribution across bins from $[-\pi, \pi)$ (we used a bin width of 5°) as well as an offset value to precisely place the angle within the bin. Because the side chains are modelled with torsion angles, the prediction is also invariant.

*Conditioning on previous predictions.* AF2 utilized the concept of "recycling" whereby previous predictions were provided to the model, enabling multiple passes or attempts through the network[21]. This effectively conditions the model on its previous predictions, allowing them to be refined a few times. We incorporated recycling into PIPPack by augmenting the initial edge representation with additional side chain-backbone and side chain-side chain encoded distances derived from the previously predicted side chains.

*Ensembling predictions.* Due to the lightweight and one-shot nature of PIPPack, inference can be performed rapidly. We hypothesized that predictions may benefit from combining knowledge from an ensemble of trained models, so we experimented with ensembling three randomly seeded models.

*Exchanging information with message passing*. Graph neural networks extract and process information from example graphs by performing convolution or passing messages between neighboring nodes to update node and/or edge representations[46]. This complete update step is usually made up of three functions, that is, to perform a node update for $n_i$, we perform

$$n_i' = U\left(n_i, Agg\left(\left\{M\left(n_i, n_j, e_{ij}\right)\right\}_{j\in\mathcal{N}(i)}\right)\right),$$

where $M$ is the message function that computes the message between nodes $i$ and $j$ based on node information $n_i, n_j$ and edge information $e_{ij}$, $Agg$ is a permutation-invariant aggregation function (e.g., mean or sum) that combines messages from all neighbors $j$ in the neighborhood $\mathcal{N}(i)$ of node $i$, and $U$ is an update function that computes the new node state based on the previous state and the aggregated messages. The neighborhood $\mathcal{N}(i)$ is defined as all the nodes connected to node $i$.

We explore the use of three types of message passing layers: a standard messaging passing layer (MPNN), a neighborhood-based IPA layer (IPA), and a novel invariant point message passing layer (IPMP, see next section). The update function of the MPNN layer used in this paper is given as

$$n_i' = n_i + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} MLP\left(n_i|\, n_j|e_{ij}\right),$$

where $|\mathcal{N}(i)|$ is the number of neighbors in $\mathcal{N}(i)$, $MLP$ is a multi-layer perceptron, and $\cdot|\cdot$ denotes the concatenation operation. The update function of the IPA layer used in this paper is given as

$$n_i' = n_i + IPA\left(n_i, \{n_j\}_{j\in\mathcal{N}(i)}, \{e_{ij}\}_{j\in\mathcal{N}(i)}\right),$$

where $IPA$ is the invariant point attention module introduced in AF2 (Algorithm 22 in the supplementary information of AF2[21]) but restricted to computing attention on the $|\mathcal{N}(i)|$ neighbors of node $i$ rather than the entire set of nodes.

*Invariant Point Message Passing (IPMP)*

The protein structure prediction network AF2 introduced a geometry-aware node representation update termed "invariant point attention"[21]. This operation performs attention across each node (residue) biased by information contained in the edges and geometric proximity. To capture this geometry awareness, IPA utilizes rigid transformations $T_i = (\boldsymbol{R}_i, \vec{t}_i)$, where $\boldsymbol{R}_i$ is a rotation matrix and $\vec{t}_i$ is a translation vector, that represent the backbone of each residue $i$ and places "invariant points" (points in the local frame of $i$). This information is aggregated via an edge- and geometry-biased attention mechanism and used as an update for each node. Note that the rigid transformations are not invariant with respect to global transformations and, therefore, must be applied appropriately to maintain invariance. In AF2, the protein is essentially represented as a densely connected graph and, therefore, messages come from every pair of nodes, but IPA can easily be adapted to form messages within local neighborhoods. In the message passing framework, IPA uses a message function that consists of geometry- and edge-biased attention.

While attention is a powerful and performant operation, it may be useful to be able to consider other functions that operate in a geometry-aware manner like IPA but without the attention. To this end, we generalize IPA to "invariant point message passing" (IPMP) wherein the message function becomes some invariant function of the connected nodes, the edge between them, and their rigid transformations, i.e., $M(n_i, T_i, n_j, T_j, e_{ij})$ (Fig. 1B). We specifically experiment with a message function that concatenates the node embeddings $n_i$ and $n_j$, the edge embedding $e_{ij}$, and five components derived from local points for each node and their rigid transforms. That is, first,

each node computes $N_{points}$ invariant points from its node representation with a learnable function $f_{points}$: $\{p_i^a\}_{a=1}^{N_{points}} = f_{points}(n_i)$. The five additional components are computed by combining the invariant points and the rigid transforms of each residue:

- Invariant points in node $i$'s local frame: $\{p_i^a\}_{a=1}^{N_{points}}$
- Squared distance between node $i$'s origin and node $i$'s invariant points: $\{\|p_i^a\|_2^2\}_{a=1}^{N_{points}}$
- Invariant points from node $j$ in node $i$'s local frame: $\{T_i^{-1} \circ T_j \circ p_i^a\}_{a=1}^{N_{points}}$
- Squared distance between node $i$'s origin and node $j$'s invariant points in node $i$'s local frame: $\left\{\left\|T_i^{-1} \circ T_j \circ p_j^a\right\|_2^2\right\}_{a=1}^{N_{points}}$
- Squared distances between points in global frame: $\left\{\left\|T_i \circ p_i^a - T_j \circ p_j^a\right\|_2^2\right\}_{a=1}^{N_{points}}$

These components are concatenated with the node and edge embeddings and are processed with an MLP to obtain the message $m_{ij}$. The same aggregation and update functions as shown for the MPNN layer are then applied to create the new node representation $n_i'$. Note that each time the transformations $T_i$ are applied, it is in an invariant manner (e.g., distance/norm calculation).

In addition to incorporating IPMP layers into PIPPack, we investigated their applicability in another protein task: inverse folding. For this experiment, we modified the fixed backbone protein design method ProteinMPNN[32] by simply swapping the MPNN layers with IPMP and retrained the network (we call this ProteinIPMP).

*Training PIPPack and ProteinIPMP*

PIPPack was trained using PyTorch on the Top2018 dataset for 150 epochs with early stopping on validation $\chi$ perplexity and the Adam optimizer with learning rate schedule described in Vaswani *et al* [47]. Using an NVIDIA A100 80G GPU, the network trained for approximately 19 hours and is

relatively lightweight with about 1.9 M learnable parameters. ProteinIPMP was trained using the same dataset, optimizer, and learning rate as described in the ProteinMPNN paper[32] for about 24 hours and has about 1.77 M parameters (vs the 1.66 M in ProteinMPNN).

*Evaluation of Performance*

Following other PSCP methods, we evaluate the performance of our method (and other methods) on our Top2018 test set using residue-level root mean squared deviation (RMSD), $\chi$ dihedral angle mean absolute error (MAE), and rotamer recovery (RR). Residue-level RMSD is determined by aligning the backbone atoms (N, Cα, C, and O) of the predicted and ground-truth residues and computing the RMSD over the side chain heavy atoms, and we report the mean RMSD value (in Å) over all residues. $\chi$ MAE (in °) is computed by determining the absolute error for each $\chi_i$ and averaging over all $\chi_i$ in the dataset. RR is the percentage of recovered rotamers within the dataset, where a rotamer is considered recovered if *all* the predicted $\chi_i$ for a particular residue are within 20° of the native $\chi_i$. These metrics are further stratified across the different centrality levels: all, core, and surface. A residue is considered in the core if the number of neighboring residues (determined by Cβ-Cβ distance < 10 Å) is at least 20, whereas it is considered on the surface if there are at most 15 neighbors. Additionally, we report the mean clashscore and rotamer evaluations, both determined via MolProbity[36]. Clashscore refers to the number of serious steric clashes (atoms with overlap of van der Waals radii > 0.4 Å) per 1000 atoms, whereas rotamer evaluations determine if a specific rotamer is considered "favored", "allowed", or an "outlier" based on the statistics of occurrences within the PDB.

**Results**

*Importance of Data Quality*

To evaluate the effect of training on data subjected to different quality filters, we trained models on the Top2018 data and the BC40 data. Moreover, we experimented with applying B-factor filters to the side chain atoms, discarding any $\chi$ angle that depends on an atom with B-factor > 40 $Å^2$. This results in four different models: Top2018 data with and without B-factor filter (Top2018-BF and Top2018) and BC40 data with and without B-factor filter (BC40-BF and BC40). These models were then evaluated on the Top2018 test set and the CASP13/14 test sets. While other PSCP methods report performance on test sets using all residues regardless of quality, we additionally report performance on our test sets using residues filtered such that the side chain atoms have low B-factors (< 40 $Å^2$).

As seen in Table 1, applying B-factor filters to the test sets results in fewer total residues for consideration with the largest differences occurring in the CASP datasets where most of the residues (> 60%) are filtered out. This filtered subset of the test sets represents the residues whose side chain conformations are reasonably reliable and, as such, comprise a better estimate of the true performance of PSCP methods. This is further evidenced by the similarities in values across B-factor filtered test sets, as opposed to the larger discrepancies within the unfiltered test sets. On the other hand, training models with or without B-factor cutoffs appears to have little to no effect on the overall performance of the method, independent of the training and testing dataset. Across training datasets, the performances are relatively similar despite the BC40 dataset containing about three-fold more chains than Top2018. Differences between the BC40- and Top2018-trained models are most apparent in the CASP13/14 test sets, but due to the 30-50 times more rotamers, we consider the results from the Top2018 datasets to be more accurate and robust estimates of the true model performance. As such, PIPPack trained on Top2018 data with B-factor filters is the

model we pursue in subsequent analyses and comparisons to other methods, and we refer to this model as PIPPack for the rest of the paper (unless otherwise noted).

*PIPPack Ablation Studies*

To investigate some of the architectural decisions and determinants of PIPPack's success, we systematically removed components and retrained our network. Specifically, we explored the importance of the $\chi$ angle representation (discretized bins vs sine and cosine), the role of iterative prediction refinement via recycling, the benefit of geometry-aware updates (i.e., MPNN layers vs IPA layers vs IPMP layers), and the effects of ensembling predictions from multiple models (Fig. 2). The largest contributor to PIPPack's success was the transformation of the PSCP problem from regression to classification, affecting the model's RR by more than 8%. Concretely, for each $\chi_i$, this transforms the prediction from the continuous $\sin(\chi_i), \cos(\chi_i)$ target to predicting discrete probabilities for each bin across $[-\pi, \pi)$. This resulted in additional benefits in terms of RMSD (~0.06 Å) and clashscore (~4.5).

The second largest contributor (1.5-2% for RR, 0.022-0.03 Å for RMSD, and 3.3-3.8 for clashscore) was the inclusion of IPMP layers instead of MPNN layers or IPA layers. Interestingly, the complex attention computations in IPA led to slightly worse performance on average than the simpler message passing layers (similar to [30]). Moreover, ensuring roughly the same parameter count between these variant models suggests that IPMP's performance isn't simply due to larger model capacity. The success of IPMP layers suggests that explicit incorporation of geometry-aware updates (without attention-weighted messages) provides better inductive reasoning over the protein structure. To further investigate this hypothesis, we augmented a recent protein inverse folding network ProteinMPNN[32] by replacing the MPNN layers with IPMP layers. Briefly, the protein "inverse folding" problem can be formulated as given a protein backbone, predict a

sequence that would fold into the same structure. By integrating IPMP into ProteinMPNN (ProteinIPMP) and retraining the network from scratch, we were able to demonstrate an improvement over native ProteinMPNN in terms of both sequence perplexity and sequence recovery (Table 2) and for almost all amino acid types (Fig. S1) on the ProteinMPNN test set. These results suggest broader applicability of IPMP for protein-related tasks.

Although not contributing quite as much, recycling provides an improvement of about 1.2% in RR, 0.016 Å in RMSD, and 3.1 in clashscore by enabling PIPPack to iteratively refine its previous predictions. While the model was trained with a specific number of recycling iterations in mind, the actual sampling procedure can perform an arbitrary amount of recycles. The default protocol utilizes the number of recycles that the model was trained for (in our case, 3 recycles), but it has been shown that recycling past this default in other models that incorporate recycling, such as AF2, can have some benefits in terms of the prediction accuracy[48,49]. To evaluate this effect for PIPPack, we sampled rotamers for the Top2018 test set from the model while varying the number of recycling iterations from 0 to 10 (Fig. S2). Increasing the number of recycling iterations appears to have little effect on the mean performance metrics past the default value, suggesting a limit to the model's refinement capabilities. Increasing recycling iterations also has the downside of increasing runtimes, requiring $n + 1$ passes through the network for $n$ recycles. Interestingly, however, PIPPack's performance is greatly improved as the number of recycles increases from 0 to 1, suggesting that model inference with just a single recycle may strike a balance between speed and prediction quality.

Without compromising too much speed, ensembled PIPPack benefits by averaging the predicted of $\chi$ distributions of several trained models. We experimented with averaging the outputted logits (prior to the softmax computation for the probabilities) from three randomly seeded versions of

our best model. The combined knowledge enabled improvements across rotamer recovery, RMSD, and clashscore of about 1.3%, 0.022 Å, and 0.8, respectively.

*Post-Prediction Minimization of PIPPack*

Although PIPPack can rapidly produce accurate side chains, it occasionally produces steric overlaps between atoms (usually when forming hydrogen bonds, but not exclusively, see Fig. S3). Just as AttnPacker uses a post-processing step to correct potential violations in bond geometries and atom clashes, we reasoned that some form of minimization may reduce the clashes in PIPPack's predictions. We experimented with applying Rosetta's MinMover protocol (referred to as PIPPack+RM) that reduces the energy of Rosetta's energy function (ref2015) by gradient descent and manipulating specific degrees of freedom (in our case, side chain torsional angles) and the same post-processing procedure (referred to as PIPPack+PP) as AttnPacker which applied gradient-based minimization to reduce clashes while not straying too far from the original torsion predictions (Table 3). Interestingly, both post-processing procedures improve PIPPack's predictions in terms of RMSD, while reducing its overall RR performance. AttnPacker's post-processing, however, improves the clashscore more than the Rosetta MinMover, presumably because the minimization objective function in Rosetta contains more terms than just a repulsive clash energy. These energy terms may also provide some explanation as to the slight improvements in MAE of longer side chains and rotamer evaluations that more closely match the native distribution (Table S1).

*Performance Comparison with PSCP Methods*

We sought to evaluate PIPPack's performance in the context of other successful PSCP methods, specifically Rosetta Packer[13,50], DLPacker[26], AttnPacker[27], and DiffPack[24]. DLPacker[26] is a PSCP method that sequentially captures the local environment of each residue by performing 3D

convolutions, predicts a probability density for the location of side chain atoms for a residue, and then selects a rotamer that fits into the predicted density from a rotamer library. Because DLPacker operates on each residue within the protein one at a time, there can be different orders by which the rotamers are sampled. We follow the recommendation by Misiura *et al.*[26] that assigns rotamers sequentially from the most crowded residues to the least crowded residues, as this order serves as a compromise between speed and quality. AttnPacker[27] is an attention-based GNN that processes the protein backbone through equivariant updates to finally predict the locations of side chain atoms all at once. Because the network is predicting coordinates of all side chain atoms simultaneously, AttnPacker sometimes violates chemical bond geometries and produces atomic clashes, therefore requiring a post-processing step to idealize the side chains and reduce these violations. Moreover, McPartlon *et al* [27] also introduced an inverse folding variant of AttnPacker that designs an amino acid sequence and packs the rotamers. As the two variants of AttnPacker perform similarly for the PSCP task, we only consider the packing variant with and without subsequent post-processing in our comparison. DiffPack[24] is a diffusion-based method that iteratively denoises the torsional distribution of $\chi$ angles, utilizing a series of SE(3)-invariant GNNs as score networks to autoregressively build up each side chain. Note that unlike any of the other methods benchmarked here, DiffPack is the only network that applies diffusion-based generative PSCP and builds the side chain of each residue one $\chi$ angle at a time. A confidence-aware version of DiffPack uses another network to predict error in the modelled side chains and allows for sampling of multiple trajectories and combining predictions that are the most confident. We benchmark both DiffPack with no confidence and DiffPack with confidence. As mentioned in the previous section, steric clashes generated by PIPPack are reduced when post-prediction

minimization is applied, so we additionally consider PIPPack with Rosetta minimization (PIPPack+RM).

As above with the evaluation of the datasets, we evaluate the packing solutions on the Top2018 test data with low B-factors in residue RMSD and RR but also consider $\chi$ MAE, clashscore, and rotamer evaluations, stratifying some of these metrics by centrality level (Table 3 and Table S2). With respect to the $\chi$ angle performance metrics ($\chi$ MAE and RR), ensembled PIPPack and PIPPack+RM outperforms other PSCP methods except confidence aware DiffPack, which performs better than all other methods for nearly all metrics. Prior to post-processing, PIPPack performs slightly worse than AttnPacker and DLPacker with respect to RMSD, but Rosetta minimization and ensembling predictions eventually surpasses these models. This highlights an important distinction between these two methods and PIPPack, as PIPPack was trained to match the distribution of native $\chi$ angles while DLPacker and AttnPacker were trained to capture the distribution of atomic coordinates, and AttnPacker was trained specifically to minimize the RMSD between the predicted and native coordinates. Interestingly, DiffPack which was also trained to match $\chi$ distributions via denoising, albeit conditional $\chi$ distributions rather than the joint distribution PIPPack aims to capture, achieves strikingly better RMSD and clashscore even without post-processing. The confidence-aware DiffPack predicts multiple conformations and then selects regions with highest confidence (i.e., lowest predicted RMSD). Both DLPacker and Rosetta Packer consider residues one at a time, while both AttnPacker and PIPPack produce entire rotamers for each residue all at once. As mentioned by the authors[24], the autoregressive nature of DiffPack may contribute to the reduced clashes in output models and overall performance.

We next looked at the performance of each of these methods on a per amino acid basis. As shown in Figure 3 and Table S1, ensembled PIPPack improves the $\chi$ angle prediction in terms of rotamer

recovery for most amino acid types (ASN, CYS, HIS, LEU, MET, PHE, SER, THR, TRP, and TYR) over the other PSCP methods, even DiffPack with confidence. Moreover, on individual $\chi_i$ basis, PIPPack demonstrates robust and competitive performance, achieving top-1 for 33% and top-2 for 79.5% of all $\chi$. Ensembled PIPPack even produces improved RMSD for four amino acids (ASN, HIS, SER, and TRP).

As PSCP is a crucial step in many computational workflows such as protein design and protein-protein docking, having rapid access to accurate side chains can dramatically impact the scale and performance of the simulations. To this end, we evaluated the runtimes of the various PSCP methods. While also being highly accurate, PIPPack additionally achieves the fastest runtimes (with and without post-prediction minimization, see above) amongst the methods evaluated and for almost every protein size tested (Table 4). Moreover, because of the lightweight model and low resource demand, PIPPack can be run efficiently on both CPU and GPU.

*PIPPack Captures Complex Physical Interactions*

Protein amino acid side chains are known to make many different types of interactions between one another and other biomolecules. These include the well-known electrostatic interactions, van der Waals forces, and hydrogen bonding, but also π-π stacking and π-cation interactions plus other less understood interactions. A desirable property of any PSCP method is the recapitulation of these interactions in the solutions that it produces. Physics-based methods, like Rosetta Packer, explicitly incorporates some of these interactions in specific score terms, such as van der Waals attractive and repulsive energies, hydrogen-bonding energy, electrostatic energy. These interactions should also be able to be learned directly from the data, which is the assumption made by most DL-based methods including PIPPack. To investigate how well PIPPack can capture these complex physical interactions, we sought to find examples of several of these interactions. As

shown in Figure 4, PIPPack reproduces van der Waals forces between well-packed hydrophobic residues (Fig. 4A), coordination of an "invisible ligand" by four cysteines (Fig. 4B), formation of a salt bridge between a lysine and an aspartic acid (Fig. 4C), hydrogen bonding between a serine and aspartate (Fig. 4D), π-π stacking interactions between aromatic rings (Fig. 4E), and π-cation interactions between an aromatic ring and a positively-charged lysine (Fig. 4F).

**Discussion**

Protein side chain packing is an important step in many computational protein simulations and can provide key insights into interactions and functional mechanisms. Because of its role in simulation, PSCP can heavily impact the performance of algorithms and, ideally, can provide rapid and accurate access to side chain conformations. Here, we present PIPPack which is an invariant GNN with novel message passing layers that has been trained to capture the native distribution of $\chi$ dihedral angles from high quality protein structures. Our model is the fastest among other state-of-the-art PSCP methods and produces competitive residue-level RMSDs and rotamer recovery, demonstrating its ability to recapitulate native side chain conformations.

Contributing to PIPPack's success, we reframed the PSCP task as classification instead of regression, introduced iterative refinement via recycling, and developed a geometry-aware message passing scheme. The latter two were inspired by the success of the protein structure prediction network AF2. The novel message passing scheme, called invariant point message passing (IPMP), can be viewed as a generalization of AF2's invariant point attention, as it can accommodate arbitrary residue neighborhoods and invariant message functions. Furthermore, IPMP appears to better capture and facilitate reasoning over protein structure, as demonstrated through an experiment where we replaced the MPNN layers within the protein inverse folding method ProteinMPNN with IPMP layers and achieved improved sequence predictions. Since the

predictions from PIPPack are a probability distribution over $\chi$ angle bins, it is also possible to sample these distributions to generate ensembles of side chain conformations.

PIPPack effectively captures the local environment of residues within a protein by propagating information along the protein graph, like AttnPacker[27] and DiffPack[24]. In contrast, DLPacker[26] voxelates the environment of each residue, performs convolutions to extract information, and sequentially assigns rotamers. These factors are most likely responsible for the overall speed of these methods. Other DL-based PSCP methods have been announced, but we were unable to benchmark them with PIPPack because of unreleased code and/or model weights. OPUS-Rota4[23] is a series of neural networks that processes local environmental features, evolutionary information in the form of a multiple sequence alignment, and the 3D-voxelized representation of the environment produced by DLPacker. ZymePackNet[28] is a series of GNNs that build up the side chains of each amino acid by iteratively predicting $\chi$ angles given the partial context of the angles in the chain and then refines the previous predictions given the full context of the side chain.

We believe that since PIPPack produces rapid, accurate rotamer predictions, it will be a valuable resource that can speed up computational simulations without compromising on quality. Moreover, the generality of IPMP for protein representations may provide additional benefits for other protein-related tasks. Although PIPPack quickly generates reasonable predictions, it can still violate physical constraints through steric clashes. It remains future work to be able to reduce these clashes without secondary post-prediction minimization while also maintaining high accuracy.

**Conclusion**

Protein side chains are responsible for the broad functions of proteins through their flexible interactions with each other and other biomolecules, highlighting the need for rapid and accurate protein side chain packing (PSCP) methods in *in silico* simulations and design. Here we present

the Protein Invariant Point Packer (PIPPack) which utilizes a novel message passing scheme to learn high-quality distributions of the $\chi$ dihedral angles and outperforms other physics- and deep learning-based PSCP methods in rotamer recovery. Although reconstructing rotamers assuming ideality and iteratively refining its predictions through recycling, PIPPack still benefits from post-prediction minimization to reduce minor clashes, revealing a direction for future studies.

## References


1. Huang X, Pearce R, Zhang Y. Toward the Accuracy and Speed of Protein Side-Chain Packing: A Systematic Study on Rotamer Libraries. *J Chem Inf Model*. 2020;60(1):410-420. doi:10.1021/acs.jcim.9b00812

2. Bhuyan MSI, Gao X. A protein-dependent side-chain rotamer library. *BMC Bioinformatics*. 2011;12 Suppl 14(Suppl 14):S10. doi:10.1186/1471-2105-12-S14-S10

3. Scouras AD, Daggett V. The Dynameomics rotamer library: amino acid side chain conformations and dynamics from comprehensive molecular dynamics simulations in water. *Protein Sci*. 2011;20(2):341-352. doi:10.1002/pro.565

4. Shapovalov MV, Dunbrack RL. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*. 2011;19(6):844-858. doi:10.1016/j.str.2011.03.019

5. Renfrew PD, Craven TW, Butterfoss GL, Kirshenbaum K, Bonneau R. A rotamer library to enable modeling and design of peptoid foldamers. *J Am Chem Soc*. 2014;136(24):8772-8782. doi:10.1021/ja503776z

6. Lovell SC, Word JM, Richardson JS, Richardson DC. The penultimate rotamer library. *Proteins*. 2000;40(3):389-408. doi:10.1002/1097-0134(20000815)40:3<389::AID-PROT50>3.0.CO;2-2

7. Watkins AM, Craven TW, Renfrew PD, Arora PS, Bonneau R. Rotamer Libraries for the High-Resolution Design of β-Amino Acid Foldamers. *Structure*. 2017;25(11):1771-1780.e3. doi:10.1016/j.str.2017.09.005

8. Dunbrack RL. Rotamer libraries in the 21st century. *Curr Opin Struct Biol*. 2002;12(4):431-440. doi:10.1016/s0959-440x(02)00344-5

9. Dunbrack RL, Cohen FE. Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Sci*. 1997;6(8):1661-1681. doi:10.1002/pro.5560060807

10. Dunbrack RL, Karplus M. Backbone-dependent rotamer library for proteins. Application to side-chain prediction. *J Mol Biol*. 1993;230(2):543-574. doi:10.1006/jmbi.1993.1170

11. De Maeyer M, Desmet J, Lasters I. All in one: a highly detailed rotamer library improves both accuracy and speed in the modelling of sidechains by dead-end elimination. *Fold Des*. 1997;2(1):53-66. doi:10.1016/s1359-0278(97)00006-0

12. Alford RF, Leaver-Fay A, Jeliazkov JR, et al. The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *J Chem Theory Comput*. 2017;13(6):3031-3048. doi:10.1021/acs.jctc.7b00125

13. Maguire JB, Haddox HK, Strickland D, et al. Perturbing the energy landscape for improved packing during computational protein design. *Proteins*. 2021;89(4):436-449. doi:10.1002/prot.26030

14. Huang X, Pearce R, Zhang Y. EvoEF2: accurate and fast energy function for computational protein design. *Bioinformatics*. 2020;36(4):1135-1142. doi:10.1093/bioinformatics/btz740

15. Jumper JM, Faruk NF, Freed KF, Sosnick TR. Accurate calculation of side chain packing and free energy with applications to protein molecular dynamics. *PLoS Comput Biol*. 2018.

16. Liang S, Grishin NV. Side-chain modeling with an optimized scoring function. *Protein Sci*. 2002;11(2):322-331. doi:10.1110/ps.24902

17. O'Meara MJ, Leaver-Fay A, Tyka MD, et al. Combined covalent-electrostatic model of hydrogen bonding improves structure prediction with Rosetta. *J Chem Theory Comput*. 2015;11(2):609-622. doi:10.1021/ct500864r

18. Desmet J, De Maeyer M, Hazes B, Lasters I. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*. 1992;356(6369):539-542. doi:10.1038/356539a0

19. Holm L, Sander C. Database algorithm for generating protein backbone and side-chain coordinates from a C alpha trace application to model building and detection of co-ordinate errors. *J Mol Biol*. 1991;218(1):183-194. doi:10.1016/0022-2836(91)90883-8

20. Kuhlman B, Baker D. Native protein sequences are close to optimal for their structures. *Proc Natl Acad Sci USA*. 2000;97(19):10383-10388. doi:10.1073/pnas.97.19.10383

21. Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*. 2021;596(7873):583-589. doi:10.1038/s41586-021-03819-2

22. Baek M, Anishchenko I, Humphreys I, Cong Q, Baker D, DiMaio F. Efficient and accurate prediction of protein structure using RoseTTAFold2. *BioRxiv*. May 25, 2023. doi:10.1101/2023.05.24.542179

23. Xu G, Wang Q, Ma J. OPUS-Rota4: a gradient-based protein side-chain modeling framework assisted by deep learning-based predictors. *Brief Bioinformatics*. 2022;23(1). doi:10.1093/bib/bbab529

24. Zhan Y, Zhang Z, Zhong B, Misra S, Tang J. DiffPack: A Torsional Diffusion Model for Autoregressive Protein Side-Chain Packing. *arXiv*. 2023. doi:10.48550/arxiv.2306.01794

25. Jindal A, Kotelnikov S, Padhorny D, et al. Side-chain Packing Using SE(3)-Transformer. *Pac Symp Biocomput*. 2022;27:46-55. doi:10.1142/9789811250477_0005

26. Misiura M, Shroff R, Thyer R, Kolomeisky AB. DLPacker: Deep learning for prediction of amino acid side chain conformations in proteins. *Proteins*. 2022;90(6):1278-1290. doi:10.1002/prot.26311

27. McPartlon M, Xu J. An end-to-end deep learning method for protein side-chain packing and inverse folding. *Proc Natl Acad Sci USA*. 2023;120(23):e2216438120. doi:10.1073/pnas.2216438120

28. Mukhopadhyay A, Kadan A, McMaster B, McWhirter JL, Dixit SB. ZymePackNet: rotamer-sampling free graph neural network method for protein sidechain prediction. *BioRxiv*. May 6, 2023. doi:10.1101/2023.05.05.539648

29. Yan J, Li S, Zhang Y, Hao A, Zhao Q. ZetaDesign: an end-to-end deep learning method for protein sequence design and side-chain packing. *Brief Bioinformatics*. July 10, 2023. doi:10.1093/bib/bbad257

30. Ingraham J, Garg VK, Barzilay R, Jaakkola T. Generative models for graph-based protein design. *NeurIPS*. 2019.

31. Zhang Z, Xu M, Jamasb A, et al. Protein Representation Learning by Geometric Structure Pretraining. *arXiv*. 2022. doi:10.48550/arxiv.2203.06125

32. Dauparas J, Anishchenko I, Bennett N, et al. Robust deep learning-based protein sequence design using ProteinMPNN. *Science*. 2022;378(6615):49-56. doi:10.1126/science.add2187

33. Hsu C, Verkuil R, Liu J, et al. Learning inverse folding from millions of predicted structures. *BioRxiv*. April 10, 2022. doi:10.1101/2022.04.10.487779

34. Williams CJ, Richardson DC, Richardson JS. The importance of residue-level filtering and the Top2018 best-parts dataset of high-quality protein residues. *Protein Sci*. 2022;31(1):290-300. doi:10.1002/pro.4239

35. Berman HM, Westbrook J, Feng Z, et al. The protein data bank. *Nucleic Acids Res*. 2000;28(1):235-242. doi:10.1093/nar/28.1.235

36. Williams CJ, Headd JJ, Moriarty NW, et al. MolProbity: more and better reference data for improved all-atom structure validation. *Protein Sci*. 2018;27(1):293-315. doi:10.1002/pro.3330

37. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol*. 2017;35(11):1026-1028. doi:10.1038/nbt.3988

38. Wang Q, Wang B, Xu Z, et al. PSSM-Distil: Protein Secondary Structure Prediction (PSSP) on Low-Quality PSSM by Knowledge Distillation with Contrastive Learning. *AAAI*. 2021;35(1):617-625. doi:10.1609/aaai.v35i1.16141

39. Chen CS, Zhou J, Wang F, Liu X, Dou D. Structure-aware protein self-supervised learning. *Bioinformatics*. 2023;39(4). doi:10.1093/bioinformatics/btad189

40. Zhang Z, Xu M, Chenthamarakshan V, Lozano A, Das P, Tang J. Enhancing Protein Language Models with Structure-based Encoder and Pre-training. *arXiv*. 2023. doi:10.48550/arxiv.2303.06275

41. Zhang Z, Xu M, Lozano A, Chenthamarakshan V, Das P, Tang J. Pre-Training Protein Encoder via Siamese Sequence-Structure Diffusion Trajectory Prediction. *arXiv*. 2023. doi:10.48550/arxiv.2301.12068

42. Gao Z, Tan C, Chacón P, Li SZ. PiFold: Towards Effective and Efficient Protein Inverse Folding. *arXiv*. 2023.

43. Fuchs F, Worrall D, Fischer V, Welling M. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. *arXiv*. 2020.

44. Watson JL, Juergens D, Bennett NR, et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *BioRxiv*. December 10, 2022. doi:10.1101/2022.12.09.519842

45. Ingraham J, Baranov M, Costello Z, et al. Illuminating protein space with a programmable generative model. *BioRxiv*. December 2, 2022. doi:10.1101/2022.12.01.518682

46. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural Message Passing for Quantum Chemistry . *arXiv*. 2017.

47. Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need. *31st Conference on Neural Information Processing Systems*. 2017.

48. Wallner B. AFsample: Improving Multimer Prediction with AlphaFold using Aggressive Sampling. *BioRxiv*. December 20, 2022. doi:10.1101/2022.12.20.521205

49. Mirdita M, Schütze K, Moriwaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: making protein folding accessible to all. *Nat Methods*. 2022;19(6):679-682. doi:10.1038/s41592-022-01488-1

50. Leaver-Fay A, Tyka M, Lewis SM, et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. *Meth Enzymol*. 2011;487:545-574. doi:10.1016/B978-0-12-381270-4.00019-6

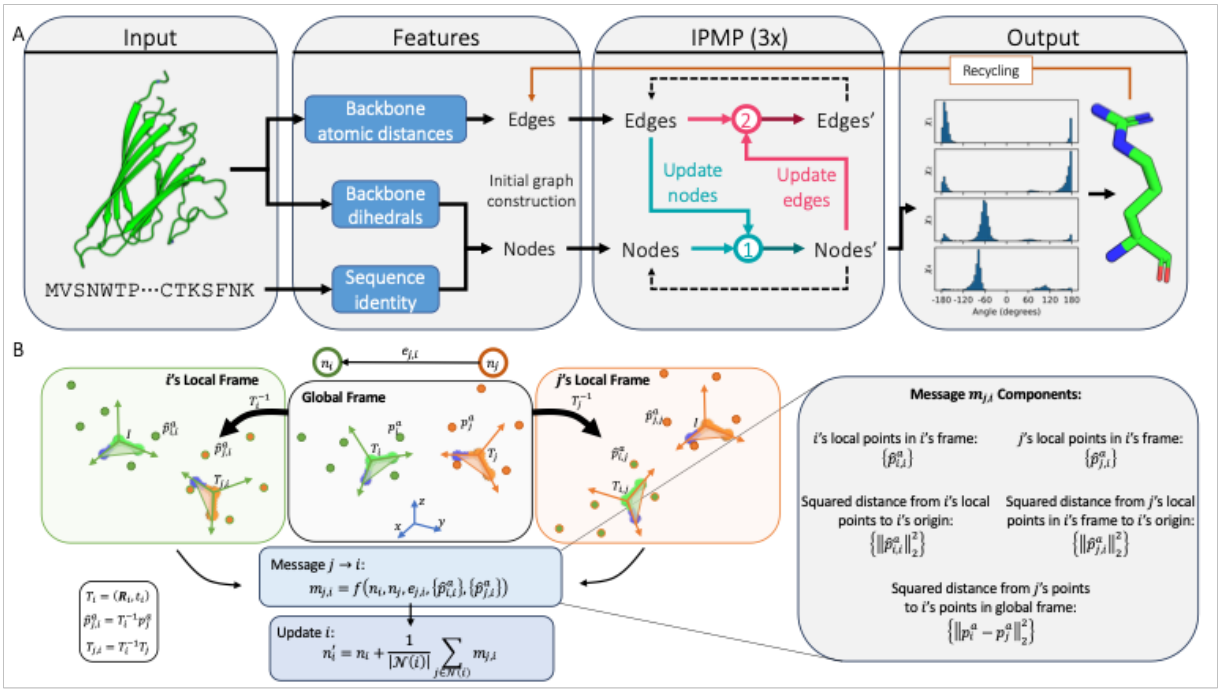


**Figure 1: Architecture of PIPPack and invariant point message passing.** (A) PIPPack is a graph neural network that processes protein backbone features (atomic distances, backbone dihedrals, and sequence) through geometry-aware message passing to iteratively refine and predict $\chi$ dihedral distributions for each residue. (B) PIPPack uses invariant point message passing (IPMP) to inject geometric information into each node and edge processing step. IPMP relies on invariant points that are produced in the local coordinate frame of each residue and transformed to obtain invariant features that depend on the protein backbone geometry.

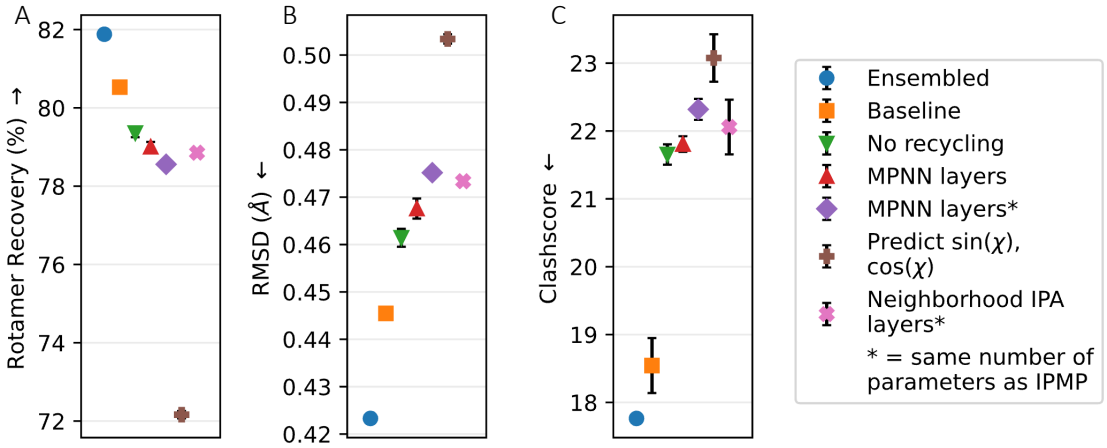


**Figure 2: PIPPack ablation studies.** Components were systematically removed from PIPPack to assess their contributions to its performance, specifically with respect to (A) rotamer recovery, (B) root mean squared deviation (RSMD), and (C) clashscore. The baseline model refers to PIPPack trained on Top2018 with binned $\chi$ prediction, 3 recycles, and IPMP layers.

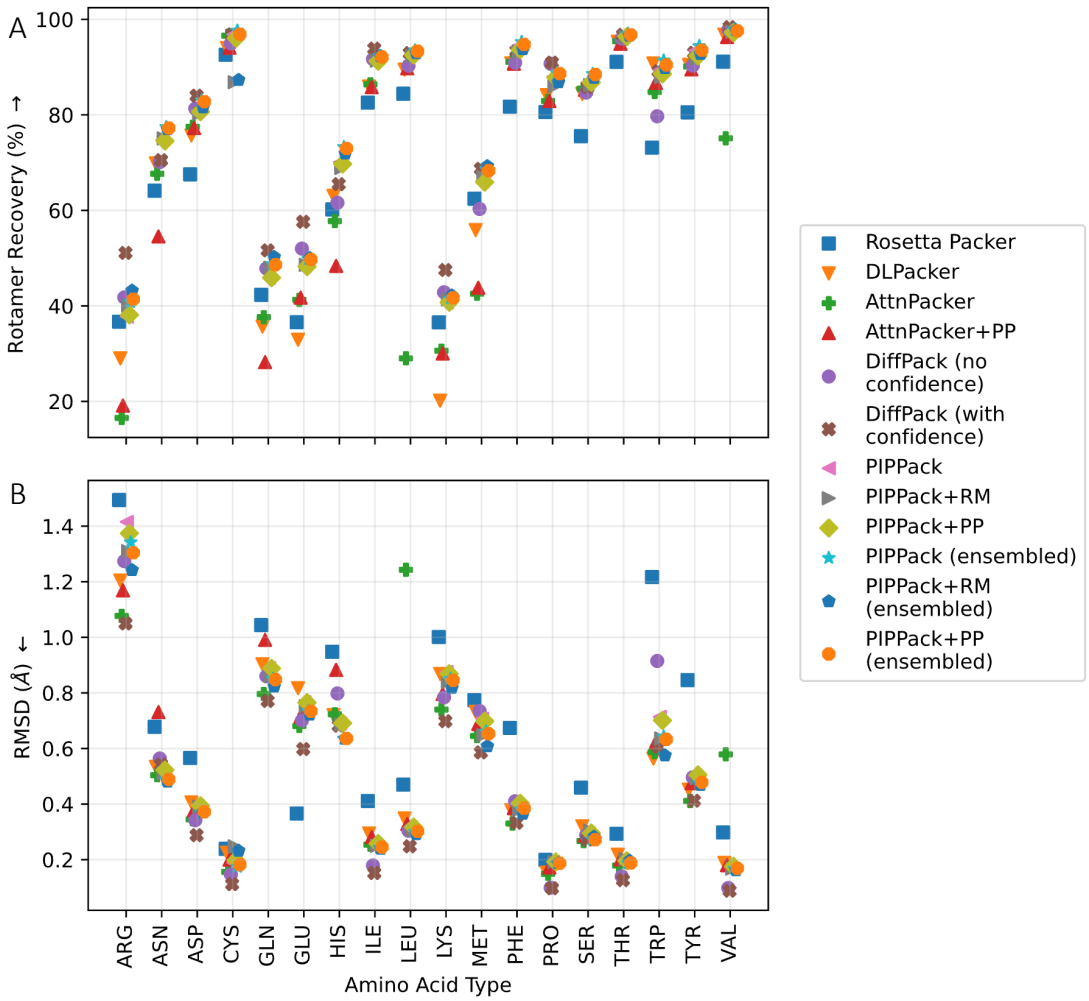


**Figure 3: Side chain packing performance across amino acid type.** (A) Ensembled PIPPack performs very competitively with other PSCP methods in rotamer recovery, achieving top-1 performance for over half of all amino acid types (ASN, CYS, HIS, LEU, MET, PHE, SER, THR, TRP, and TYR). (B) With respect to RMSD, ensembled PIPPack also performs competitively for all amino acid types, even improving for certain residues (ASN, HIS, SER, and TRP).

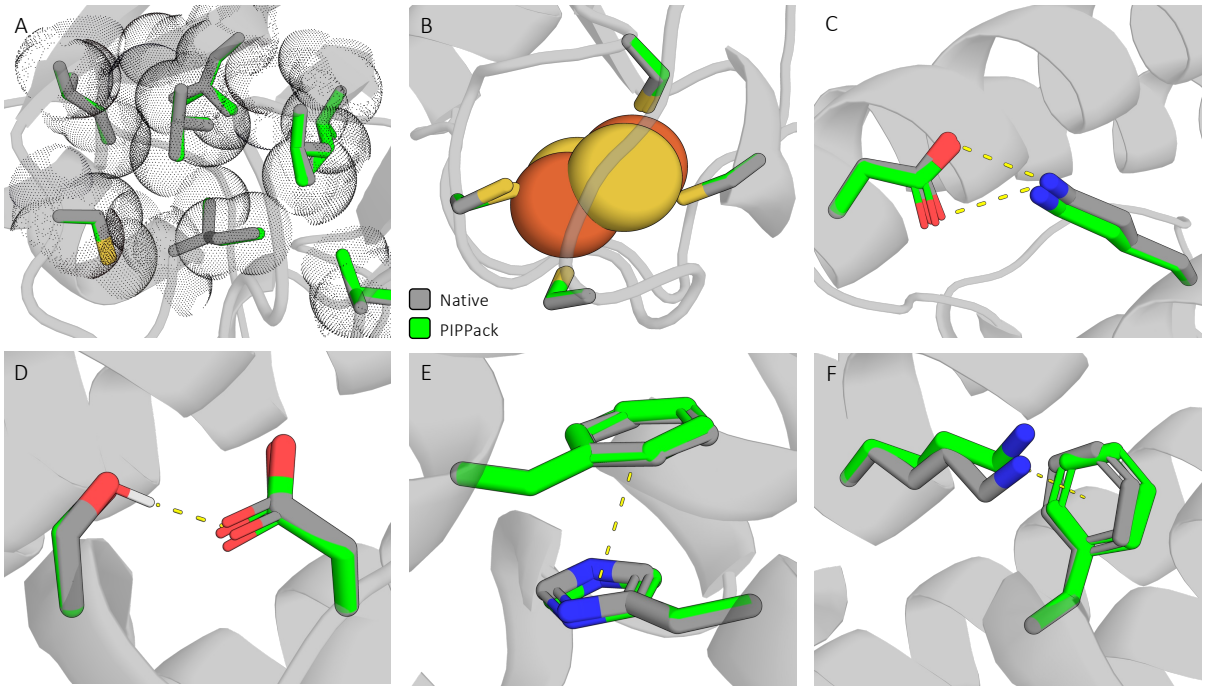


**Figure 4: PIPPack reproduces physical interactions involving side chains.** Numerous types of complex interactions are made by side chains within proteins, and PIPPack can reproduce many of them, including: (A) van der Waals forces, (B) coordination of invisible ligands, (C) ionic salt bridges, (D) hydrogen bonding, (E) π-π stacking, and (F) π-cation interactions.

**Table 1: Impact of B-factor filters on training and testing data.**

| Model | Top2018 | | Top2018-BF | | CASP13 | | CASP13-BF | | CASP14 | | CASP14-BF | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSD | RR | RMSD | RR | RMSD | RR | RMSD | RR | RMSD | RR | RMSD | RR |
| PIPP Top2018 | 0.506 | 76.72 | 0.445 | 80.29 | 0.746 | 61.30 | 0.601 | 70.65 | 0.911 | 51.35 | 0.576 | 69.63 |
| PIPP Top2018-BF | 0.502 | 76.80 | 0.441 | 80.53 | 0.756 | 61.07 | 0.611 | 70.83 | 0.916 | 51.15 | 0.588 | 69.57 |
| PIPP BC40 | 0.516 | 76.10 | 0.454 | 79.82 | 0.695 | 64.28 | 0.569 | 72.91 | 0.854 | 51.69 | 0.574 | 70.47 |
| PIPP BC40-BF | 0.519 | 75.90 | 0.457 | 79.64 | 0.696 | 64.21 | 0.568 | 72.77 | 0.852 | 53.81 | 0.570 | 70.60 |
| Number of Rotamers | 259912 | | 239168 (92.02%) | | 18020 | | 7199 (39.95%) | | 13356 | | 4449 (33.31%) | |

**Table 2: Sequence design task on ProteinMPNN test set.**

| Method* | Perplexity ↓ | Sequence Recovery (%) ↑ | | |
|---|---|---|---|---|
| | | All | Core | Surface |
| ProteinMPNN | 5.455 ± 0.038 | 49.07 ± 0.11 | 60.11 ± 0.11 | 36.87 ± 0.06 |
| ProteinIPMP | 4.555 ± 0.032 | 54.01 ± 0.16 | 65.64 ± 0.09 | 41.01 ± 0.19 |

* Methods were sampled with temperature T=0.0001

**Table 3: Protein side chain packing results on B-factor-filtered Top2018 test set.**

| Method | RMSD (Å) ↓ | | | $\chi$ MAE (°) ↓ | | | | RR (%) ↑ | | | Rotamer Outliers* (%) | Clashscore* ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | All | Core | Surface | $\chi_1$ | $\chi_2$ | $\chi_3$ | $\chi_4$ | All | Core | Surface | | |
| Rosetta Packer | 0.589 | 0.423 | 0.733 | 18.28 | 22.35 | 37.22 | 45.41 | 71.97 | 80.85 | 63.24 | 0.06 | 22.66 |
| DLPacker | 0.444 | 0.342 | 0.536 | 12.18 | 19.62 | 39.42 | 56.85 | 75.12 | 81.61 | 68.65 | 1.18 | 10.56 |
| AttnPacker | 0.519 | 0.450 | 0.574 | 12.93 | 29.68 | 35.82 | 45.35 | 65.68 | 70.43 | 61.19 | 4.01 | 21.85 |
| AttnPacker+PP | 0.440 | 0.349 | 0.520 | 10.95 | 20.62 | 40.67 | 45.18 | 73.80 | 79.78 | 67.91 | 2.29 | 11.17 |
| DiffPack (no confidence) | 0.407 | 0.308 | 0.490 | 11.47 | 17.45 | 34.21 | 43.51 | 79.72 | 85.55 | 74.02 | 0.55 | 9.84 |
| DiffPack (with confidence) | **0.347** | **0.256** | **0.426** | **9.59** | 15.44 | **30.72** | **39.43** | **82.52** | **88.13** | **76.98** | 0.43 | **6.74** |
| PIPPack† | 0.445 | 0.346 | 0.533 | 10.48 | 16.24 | 33.26 | 44.48 | 80.53 | 86.58 | 74.49 | 0.11 | 18.54 |
| PIPPack+RM† | 0.431 | 0.328 | 0.523 | 11.05 | 16.51 | 32.92 | 43.97 | 80.13 | 86.19 | 74.07 | 0.40 | 14.63 |
| PIPPack+PP† | 0.444 | 0.343 | 0.531 | 10.63 | 16.34 | 33.30 | 44.47 | 80.27 | 86.22 | 74.33 | 0.32 | 12.66 |
| PIPPack (ensembled) | 0.423 | 0.325 | 0.510 | 9.63 | **15.21** | 31.72 | 42.77 | 81.88 | 87.93 | 75.82 | 0.08 | 17.76 |
| PIPPack+RM (ensembled) | 0.411 | 0.309 | 0.501 | 10.29 | 15.53 | 31.43 | 42.23 | 81.30 | 87.33 | 75.20 | 0.34 | 14.29 |
| PIPPack+PP (ensembled) | 0.422 | 0.322 | 0.509 | 9.78 | 15.31 | 31.74 | 42.78 | 81.66 | 87.64 | 75.66 | 0.25 | 12.19 |

* The native Top2018 test set has 0.69% rotamer outliers and a clashscore of 1.27.

† The mean values of three randomly seeded models are shown.

**Table 4: Runtime comparison between side chain packing methods.**

| Method | Runtime* (sec / protein) | | | | |
|---|---|---|---|---|---|
| | $N_{res} = 100$ | $N_{res} = 250$ | $N_{res} = 500$ | $N_{res} = 750$ | $N_{res} = 1005$ |
| Rosetta Packer | 72.37 | 542.19 | 911.93 | 1836.12 | 2563.08 |
| DLPacker | 19.70 | 48.99 | 110.31 | 172.09 | 242.73 |
| AttnPacker | 1.70 | 2.24 | 3.91 | 6.20 | 8.82 |
| AttnPacker+PP | 7.78 | 12.65 | 14.21 | 16.74 | 21.10 |
| DiffPack (no confidence) | 5.98 | 7.42 | 11.27 | 15.66 | 21.36 |
| DiffPack (with confidence) | 8.12 | 13.64 | 24.89 | 35.89 | 45.38 |
| PIPPack | 0.36 | 0.42 | 0.49 | 0.55 | 0.66 |
| PIPPack+RM | 2.06 | 4.33 | 9.24 | 9.40 | 28.83 |
| PIPPack+PP | 8.23 | 9.17 | 9.81 | 10.66 | 11.51 |
| PIPPack (ensembled) | 0.83 | 0.88 | 0.96 | 1.00 | 1.16 |
| PIPPack+RM (ensembled) | 2.52 | 4.79 | 9.72 | 9.86 | 29.32 |
| PIPPack+PP (ensembled) | 8.69 | 9.64 | 10.28 | 11.11 | 12.01 |

* Mean values of 25 unbatched predictions.

† Only method solely running on the CPU.

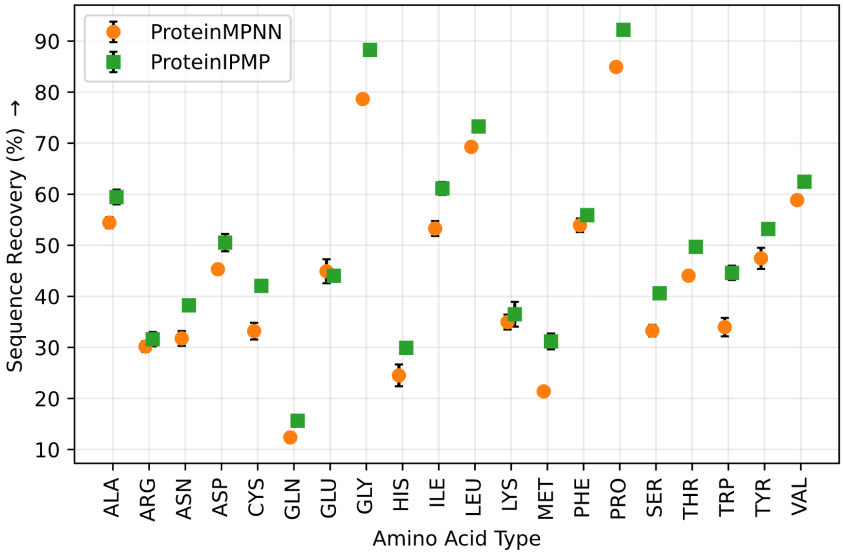


**Figure S1: Native sequence recovery across amino acid types.** Incorporating invariant point message passing in the protein fixed backbone sequence design ("inverse folding") task results in improved recovery rates across almost all amino acid types (except GLU).

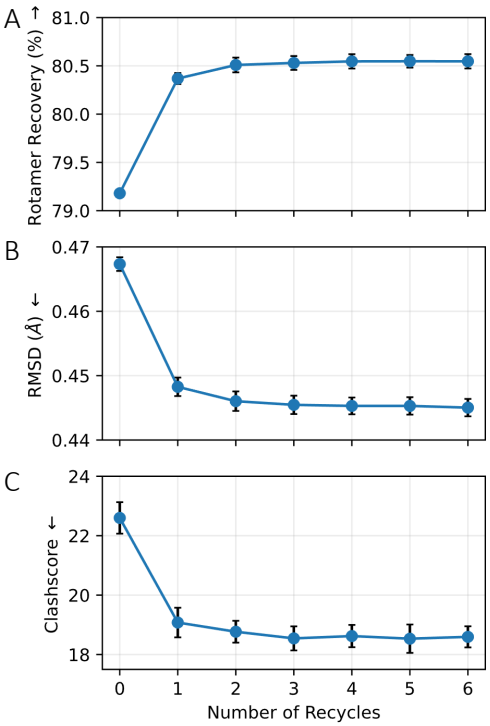


**Figure S2: PIPPack performance across recycles.** PIPPack uses recycling to iteratively refine its side chain conformation predictions, resulting in improved performance metrics. Recycling past the value PIPPack was trained for (in our case 3) does not result in any significant differences in performance, but the model performs well even with a single recycle.

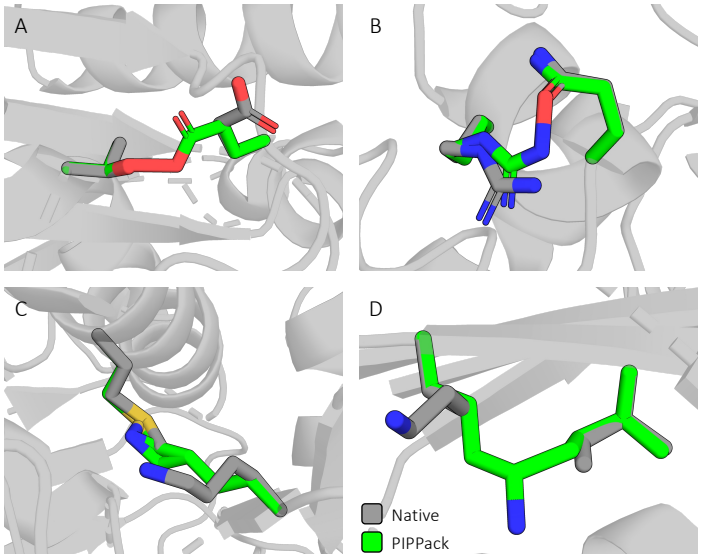


**Figure S3: Representative clashes produced by PIPPack**. Although not common, PIPPack can produce atomic clashes due to the one-shot nature of the network. Egregious clashes, as shown here, are usually fixed via post-prediction minimization. Most of these clashes occur between hydrogen bonding partners (A and B) and/or long side chains (B, C, and D).

**Table S1: Side chain $\chi$ angle errors across amino acid type.**

| AA Type | Method | MAE (°) ↓ | | | | AA Type | Method | MAE (°) ↓ | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\chi_1$ | $\chi_2$ | $\chi_3$ | $\chi_4$ | | | $\chi_1$ | $\chi_2$ |
| ARG | Rosetta Packer | 23.00 | 24.34 | 48.62 | 48.54 | ASN | Rosetta Packer | 20.35 | 28.72 |
| | DLPacker | 17.11 | 22.75 | 45.97 | 53.64 | | DLPacker | 13.19 | 28.56 |
| | AttnPacker | 15.61 | 20.11 | 43.56 | 48.96 | | AttnPacker | 12.07 | 27.69 |
| | AttnPacker+PP | 17.22 | 21.08 | 42.78 | 48.56 | | AttnPacker+PP | 12.36 | 53.56 |
| | DiffPack (no confidence) | 16.50 | 19.86 | 41.19 | 46.56 | | DiffPack (no confidence) | 14.29 | 33.27 |
| | DiffPack (with confidence) | **13.30** | **17.18** | **34.86** | **40.30** | | DiffPack (with confidence) | 11.74 | 36.16 |
| | PIPPack† | 15.37 | 20.19 | 41.67 | 48.79 | | PIPPack† | 12.38 | 24.91 |
| | PIPPack+RM† | 16.00 | 20.56 | 41.74 | 48.00 | | PIPPack+RM† | 12.63 | 24.63 |
| | PIPPack+PP† | 15.89 | 20.49 | 41.88 | 48.73 | | PIPPack+PP† | 12.47 | 24.95 |
| | PIPPack (ensembled) | 14.20 | 19.14 | 39.17 | 45.96 | | PIPPack (ensembled) | **11.20** | 23.16 |
| | PIPPack+RM (ensembled) | 14.82 | 19.31 | 39.16 | 45.12 | | PIPPack+RM (ensembled) | 11.62 | **23.14** |
| | PIPPack+PP (ensembled) | 14.68 | 19.33 | 39.29 | 45.91 | | PIPPack+PP (ensembled) | 11.29 | 23.2 |
| LYS | Rosetta Packer | 23.40 | 25.05 | 32.14 | 41.89 | ASP | Rosetta Packer | 19.72 | 18.1 |
| | DLPacker | 17.03 | 24.37 | 38.45 | 60.48 | | DLPacker | 12.86 | 15.4 |
| | AttnPacker | 14.77 | 20.50 | 30.16 | 41.27 | | AttnPacker | 10.40 | 14.09 |
| | AttnPacker+PP | 15.64 | 21.08 | 30.50 | 41.37 | | AttnPacker+PP | 10.80 | 13.96 |
| | DiffPack (no confidence) | 15.73 | 19.67 | 28.48 | 40.06 | | DiffPack (no confidence) | 11.95 | 12.67 |
| | DiffPack (with confidence) | **13.35** | **17.83** | **25.73** | **38.45** | | DiffPack (with confidence) | **9.59** | **11.55** |
| | PIPPack† | 15.67 | 20.84 | 27.93 | 39.63 | | PIPPack† | 11.14 | 12.87 |
| | PIPPack+RM† | 15.85 | 20.70 | 27.68 | 39.42 | | PIPPack+RM† | 11.53 | 13.1 |
| | PIPPack+PP† | 15.84 | 20.87 | 27.94 | 39.66 | | PIPPack+PP† | 11.24 | 12.91 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | PIPPack (ensembled) | 14.72 | 20.10 | 27.30 | 39.18 | | PIPPack (ensembled) | 10.20 | 12.08 |
| | PIPPack+RM (ensembled) | 14.95 | 19.97 | 27.05 | 38.96 | | PIPPack+RM (ensembled) | 10.68 | 12.5 |
| | PIPPack+PP (ensembled) | 14.89 | 20.14 | 27.32 | 39.24 | | PIPPack+PP (ensembled) | 10.28 | 12.14 |
| **GLU** | Rosetta Packer | 29.33 | 33.46 | 27.00 | | **LEU** | Rosetta Packer | 12.64 | 18.53 |
| | DLPacker | 23.16 | 29.94 | 30.35 | | | DLPacker | 8.33 | 14.25 |
| | AttnPacker | 18.82 | 24.44 | 26.82 | | | AttnPacker | 7.13 | 87.63 |
| | AttnPacker+PP | 19.34 | 24.57 | 26.36 | | | AttnPacker+PP | 7.36 | 13.44 |
| | DiffPack (no confidence) | 19.55 | 25.16 | 22.54 | | | DiffPack (no confidence) | 9.57 | 11.77 |
| | DiffPack (with confidence) | **16.35** | **21.31** | **20.48** | | | DiffPack (with confidence) | 7.33 | **9.49** |
| | PIPPack† | 19.81 | 25.11 | 23.41 | | | PIPPack† | 6.67 | 11.04 |
| | PIPPack+RM† | 19.88 | 25.16 | 23.20 | | | PIPPack+RM† | 6.99 | 11.07 |
| | PIPPack+PP† | 19.93 | 25.16 | 23.39 | | | PIPPack+PP† | 6.74 | 11.11 |
| | PIPPack (ensembled) | 18.49 | 23.85 | 22.68 | | | PIPPack (ensembled) | **6.16** | 10.25 |
| | PIPPack+RM (ensembled) | 18.58 | 23.90 | 22.53 | | | PIPPack+RM (ensembled) | 6.58 | 10.37 |
| | PIPPack+PP (ensembled) | 18.59 | 23.88 | 22.66 | | | PIPPack+PP (ensembled) | 6.22 | 10.32 |
| **GLN** | Rosetta Packer | 23.20 | 32.63 | 42.28 | | **ILE** | Rosetta Packer | 12.48 | 19.03 |
| | DLPacker | 18.17 | 27.87 | 43.74 | | | DLPacker | 6.86 | 17.78 |
| | AttnPacker | 15.67 | 24.47 | 42.01 | | | AttnPacker | 6.06 | 14.92 |
| | AttnPacker+PP | 16.29 | 24.68 | 71.75 | | | AttnPacker+PP | 6.43 | 15.07 |
| | DiffPack (no confidence) | 16.95 | 24.76 | 46.42 | | | DiffPack (no confidence) | 4.66 | 11.52 |
| | DiffPack (with confidence) | **13.99** | **21.43** | 46.27 | | | DiffPack (with confidence) | **4.02** | **9.64** |
| | PIPPack† | 16.13 | 24.76 | 42.14 | | | PIPPack† | 5.08 | 12.19 |
| | PIPPack+RM† | 16.17 | 24.58 | 41.10 | | | PIPPack+RM† | 5.47 | 12.21 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | PIPPack+PP† | 16.23 | 24.78 | 42.11 | | | PIPPack+PP† | 5.22 | 12.24 |
| | PIPPack (ensembled) | 14.93 | 23.33 | 40.30 | | | PIPPack (ensembled) | 4.65 | 11.28 |
| | PIPPack+RM (ensembled) | 15.01 | 23.13 | **39.50** | | | PIPPack+RM (ensembled) | 5.14 | 11.37 |
| | PIPPack+PP (ensembled) | 15.01 | 23.39 | 40.26 | | | PIPPack+PP (ensembled) | 4.77 | 11.32 |
| **MET** | Rosetta Packer | 17.77 | 21.39 | 39.55 | | **PRO** | Rosetta Packer | 10.90 | 15.14 |
| | DLPacker | 13.81 | 20.12 | 42.15 | | | DLPacker | 8.45 | 12.42 |
| | AttnPacker | 12.16 | 17.52 | 43.65 | | | AttnPacker | 8.10 | 11.74 |
| | AttnPacker+PP | 13.02 | 17.90 | 43.06 | | | AttnPacker+PP | 8.43 | 12.28 |
| | DiffPack (no confidence) | 15.53 | 20.05 | 40.67 | | | DiffPack (no confidence) | 5.28 | 6.35 |
| | DiffPack (with confidence) | 11.55 | 15.30 | 32.51 | | | DiffPack (with confidence) | **5.17** | **6.2** |
| | PIPPack† | 11.92 | 15.29 | 35.71 | | | PIPPack† | 7.08 | 9.83 |
| | PIPPack+RM† | 12.07 | 15.28 | 35.05 | | | PIPPack+RM† | 9.40 | 12.14 |
| | PIPPack+PP† | 12.14 | 15.41 | 35.62 | | | PIPPack+PP† | 7.14 | 9.88 |
| | PIPPack (ensembled) | **10.89** | **13.82** | 32.91 | | | PIPPack (ensembled) | 6.68 | 9.22 |
| | PIPPack+RM (ensembled) | 11.10 | 13.94 | **32.40** | | | PIPPack+RM (ensembled) | 9.09 | 11.68 |
| | PIPPack+PP (ensembled) | 11.12 | 13.95 | 32.86 | | | PIPPack+PP (ensembled) | 6.74 | 9.27 |
| **PHE** | Rosetta Packer | 13.14 | 11.70 | | | **THR** | Rosetta Packer | 14.46 | |
| | DLPacker | 6.45 | 8.85 | | | | DLPacker | 10.05 | |
| | AttnPacker | 5.26 | 8.56 | | | | AttnPacker | 8.15 | |
| | AttnPacker+PP | 5.81 | 8.59 | | | | AttnPacker+PP | 8.44 | |
| | DiffPack (no confidence) | 8.32 | 7.56 | | | | DiffPack (no confidence) | 7.78 | |
| | DiffPack (with confidence) | 6.49 | **6.50** | | | | DiffPack (with confidence) | **6.91** | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | PIPPack† | 5.63 | 6.84 | | | | PIPPack† | 7.54 | |
| | PIPPack+RM† | 6.24 | 7.21 | | | | PIPPack+RM† | 8.37 | |
| | PIPPack+PP† | 5.82 | 7.06 | | | | PIPPack+PP† | 7.63 | |
| | PIPPack (ensembled) | **5.11** | 6.76 | | | | PIPPack (ensembled) | 7.05 | |
| | PIPPack+RM (ensembled) | 5.88 | 6.96 | | | | PIPPack+RM (ensembled) | 7.99 | |
| | PIPPack+PP (ensembled) | 5.30 | 6.92 | | | | PIPPack+PP (ensembled) | 7.13 | |
| **TYR** | Rosetta Packer | 14.98 | 11.63 | | | **SER** | Rosetta Packer | 32.99 | |
| | DLPacker | 7.06 | 8.82 | | | | DLPacker | 22.08 | |
| | AttnPacker | 5.99 | 8.92 | | | | AttnPacker | 18.20 | |
| | AttnPacker+PP | 6.39 | 8.96 | | | | AttnPacker+PP | 18.46 | |
| | DiffPack (no confidence) | 8.99 | 7.47 | | | | DiffPack (no confidence) | 21.53 | |
| | DiffPack (with confidence) | 7.32 | **6.62** | | | | DiffPack (with confidence) | 20.17 | |
| | PIPPack† | 6.53 | 7.09 | | | | PIPPack† | 19.16 | |
| | PIPPack+RM† | 7.28 | 7.52 | | | | PIPPack+RM† | 19.90 | |
| | PIPPack+PP† | 6.73 | 7.40 | | | | PIPPack+PP† | 19.20 | |
| | PIPPack (ensembled) | **5.87** | 7.03 | | | | PIPPack (ensembled) | **17.18** | |
| | PIPPack+RM (ensembled) | 6.75 | 7.21 | | | | PIPPack+RM (ensembled) | 18.04 | |
| | PIPPack+PP (ensembled) | 6.09 | 7.30 | | | | PIPPack+PP (ensembled) | 17.23 | |
| **TRP** | Rosetta Packer | 17.63 | 30.62 | | | **CYS** | Rosetta Packer | 11.73 | |
| | DLPacker | 6.99 | 14.97 | | | | DLPacker | 9.21 | |
| | AttnPacker | 6.84 | 15.76 | | | | AttnPacker | 6.83 | |
| | AttnPacker+PP | 7.11 | 15.39 | | | | AttnPacker+PP | 8.58 | |
| | DiffPack (no confidence) | 10.53 | 26.73 | | | | DiffPack (no confidence) | 8.27 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | DiffPack (with confidence) | 8.30 | 14.77 | | | | DiffPack (with confidence) | **6.14** | |
| | PIPPack† | 7.54 | 15.75 | | | | PIPPack† | 7.10 | |
| | PIPPack+RM† | 7.93 | 16.33 | | | | PIPPack+RM† | 11.00 | |
| | PIPPack+PP† | 7.82 | 15.95 | | | | PIPPack+PP† | 7.85 | |
| | PIPPack (ensembled) | **6.57** | **13.16** | | | | PIPPack (ensembled) | 6.15 | |
| | PIPPack+RM (ensembled) | 7.09 | 14.06 | | | | PIPPack+RM (ensembled) | 9.98 | |
| | PIPPack+PP (ensembled) | 6.88 | 13.47 | | | | PIPPack+PP (ensembled) | 6.89 | |
| **HIS** | Rosetta Packer | 18.61 | 45.62 | | | **VAL** | Rosetta Packer | 14.02 | |
| | DLPacker | 10.48 | 47.03 | | | | DLPacker | 7.54 | |
| | AttnPacker | 9.54 | 52.95 | | | | AttnPacker | 32.93 | |
| | AttnPacker+PP | 10.32 | 69.95 | | | | AttnPacker+PP | 7.18 | |
| | DiffPack (no confidence) | 12.77 | 53.06 | | | | DiffPack (no confidence) | 5.21 | |
| | DiffPack (with confidence) | 9.74 | 49.34 | | | | DiffPack (with confidence) | **4.62** | |
| | PIPPack† | 9.89 | 40.40 | | | | PIPPack† | 6.23 | |
| | PIPPack+RM† | 10.73 | 40.46 | | | | PIPPack+RM† | 6.41 | |
| | PIPPack+PP† | 10.06 | 40.46 | | | | PIPPack+PP† | 6.37 | |
| | PIPPack (ensembled) | **8.73** | **36.68** | | | | PIPPack (ensembled) | 5.79 | |
| | PIPPack+RM (ensembled) | 9.68 | 36.94 | | | | PIPPack+RM (ensembled) | 6.04 | |
| | PIPPack+PP (ensembled) | 8.90 | 36.72 | | | | PIPPack+PP (ensembled) | 5.91 | |

**Table S2: Rotamer Evaluations for PSCP Methods.**

| Method | Rotamer Evaluation (%) | | |
|---|---|---|---|
| | **Favored** | **Allowed** | **Outlier** |
| Native | 96.77 | 2.54 | 0.69 |
| Rosetta Packer | 99.29 | 0.65 | 0.06 |
| DLPacker | 96.32 | 2.50 | 1.18 |
| AttnPacker | 87.65 | 8.34 | 4.01 |
| AttnPacker+PP | 93.62 | 4.09 | 2.29 |
| DiffPack (no confidence) | 97.97 | 1.48 | 0.55 |
| DiffPack (with confidence) | 98.34 | 1.23 | 0.43 |
| PIPPack | 99.19 | 0.70 | 0.11 |
| PIPPack+RM | 98.39 | 1.20 | 0.40 |
| PIPPack+PP | 98.72 | 0.96 | 0.32 |
| PIPPack (ensembled) | 99.35 | 0.57 | 0.08 |
| PIPPack+RM (ensembled) | 98.57 | 1.08 | 0.34 |
| PIPPack+PP (ensembled) | 98.94 | 0.81 | 0.25 |