
Full-Atom Protein Pocket Design via Iterative Refinement

Zaixi Zhang^{1,2,4}, Zepu Lu^{1,2}, Zhongkai Hao³, Marinka Zitnik⁴, Qi Liu^{1,2*}

¹ Anhui Province Key Lab of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China

² State Key Laboratory of Cognitive Intelligence, Hefei, Anhui, China

³ Dept. of Comp. Sci. and Tech., Institute for AI, THBI Lab, BNRist Center, Tsinghua-Bosch Joint ML Center, Tsinghua, ⁴ Harvard University
{zaixi, zplu}@mail.ustc.edu.cn, hzj21@mails.tsinghua.edu.cn, marinka@hms.harvard.edu, qiliuql@ustc.edu.cn

Abstract

The design and optimization of functional proteins that bind specific ligand molecules is paramount in therapeutics and bio-engineering. A critical yet formidable task in this endeavor is the design of the protein pocket, which is the cavity region of the protein where the ligand binds. Current methods are plagued by inefficient generation, inadequate context modeling of the ligand molecule, and the inability to generate side-chain atoms. Here, we present the Full-Atom Iterative Refinement (FAIR) method, designed to address these challenges by facilitating the co-design of protein pocket sequences, specifically residue types, and their corresponding 3D structures. FAIR operates in two steps, proceeding in a coarse-to-fine manner (transitioning from protein backbone to atoms, including side chains) for a full-atom generation. In each iteration, all residue types and structures are simultaneously updated, a process termed full-shot refinement. In the initial stage, the residue types and backbone coordinates are refined using a hierarchical context encoder, complemented by two structure refinement modules that capture both inter-residue and pocket-ligand interactions. The subsequent stage delves deeper, modeling the side-chain atoms of the pockets and updating residue types to ensure sequence-structure congruence. Concurrently, the structure of the binding ligand is refined across iterations to accommodate its inherent flexibility. Comprehensive experiments show that FAIR surpasses existing methods in designing superior pocket sequences and structures, producing average improvement exceeding 10% in AAR and RMSD metrics.

1 Introduction

Proteins are macromolecules that perform fundamental functions in living organisms. An essential and challenging step in functional protein design, such as enzymes [52, 27] and biosensors [5, 18], is to design protein pockets that bind specific ligand molecules. Protein pockets usually refer to residues binding with the ligands in the protein-ligand complexes. Nevertheless, such a problem is very challenging due to the tremendous search space of both sequence and structure and the small solution space that satisfies the constraints of binding affinity, geometric complementarity, and biochemical stability. Traditional approaches for pocket design use hand-crafted energy functions [42, 11] and template-matching methods [67, 8, 48], which can be less accurate and lack efficiency.

*Qi Liu is the corresponding author.

Deep learning approaches, especially deep generative models, have made significant progress in protein design [46, 17, 9, 20] using experimental and predicted 3D structure information to design amino acid sequences [26, 30, 7, 14]. Despite their strong performance, these approaches are not directly applicable to pocket design because the protein pocket structures to be designed are a priori unknown [8]. To this end, generative models with the capability to co-design both the sequence and structure of protein pockets are needed. In another area of protein design, pioneering research developed techniques to co-design sequence and structure of complementarity determining regions (CDRs) for antibodies (Y-shaped proteins that bind with specific pathogens) [29, 28, 39, 33, 56]. Nevertheless, these approaches are tailored specifically for antibodies and are unsuitable for the pocket design of general proteins. For example, prevailing methods generate only protein backbone atoms while neglecting side-chain atoms, which play essential roles in protein pocket-ligand interactions [43]. Further, binding targets (antigens) considered by these methods are fixed, whereas binding ligand molecules are more flexible and require additional modeling.

Here, we introduce a full-atom iterative refinement approach (FAIR) for protein pocket sequence-structure co-design, addressing the limitations of prior research. FAIR consists of two primary steps: initially modeling the pocket backbone atoms to generate coarse-grained structures, followed by incorporating side-chain atoms for a complete full-atom pocket design. (1) In the first step, since the pocket residue types and the number of side-chain atoms are largely undetermined, FAIR progressively predicts and updates backbone atom coordinates and residue types via soft assignment over multiple rounds. Sequence and structure information are jointly updated in each round for efficiency, which we refer to as full-shot refinement. A hierarchical encoder with residue- and atom-level encoders captures the multi-scale structure of protein-ligand complexes. To update the pocket structure, FAIR leverages two structure refinement modules to predict the interactions within pocket residues and between the pocket and the binding ligand in each round. (2) In the second step, FAIR initializes residue types based on hard assignment and side-chain atoms based on the results of the first step. The hierarchical encoder and structure refinement modules are similar to the first step for the full-atom structure update. To model the influence of structure on residue types, FAIR randomly masks a part of the pocket residues and updates the masked residue types based on the neighboring residue types and structures in each round. After several rounds of refinement, the residue type and structure update gradually converge, and the residue sequence-structure becomes consistent. The ligand molecular structure is also updated along with the above refinement processes to account for the flexibility of the ligand.

Overall, FAIR is an $E(3)$ -equivariant generative model that generates both pocket residue types and the pocket-ligand complex 3D structure. Our study presents the following main contributions:

- **New task:** We preliminarily formulate protein pocket design as a co-generation of both the sequence and full-atom structure of a protein pocket (i.e., co-design), conditioned on a binding ligand molecule and the protein backbone (see Sec. 3.1).
- **Novel method:** FAIR is an end-to-end generative framework that generates both the pocket sequence and structure through iterative refinement (see Sec. 3). It addresses the limitations of previous approaches and takes into account side-chain effects, ligand flexibility, and sequence-structure consistency to enhance prediction efficiency.
- **Strong performance:** FAIR outperforms existing methods on various protein pocket design quality metrics, producing average improvements of 15.5% (AAR) and 13.5% (RMSD). FAIR's pocket-generation process is over ten times faster than traditional methods (see Sec. 4).

2 Related Work

Protein Design. Computational protein design has brought the capability of inventing proteins with desired structures and functions [22, 35, 60]. Existing studies focus on designing amino acid sequences that fold into target protein structures, i.e., inverse protein folding. Traditional approaches mainly rely on hand-crafted energy functions to iteratively search for low-energy protein sequences with heuristic sampling algorithms, which are inefficient, less accurate, and less generalizable [1, 6]. Recent deep generative approaches have achieved remarkable success in protein design [15, 26, 30, 7, 14, 51, 20, 3]. Some protein design methods also benefit from the latest advances in protein structure prediction techniques, e.g., AlphaFold2 [31]. Nevertheless, they usually assume the target protein structure is given, which is unsuitable for our pocket design case. Another

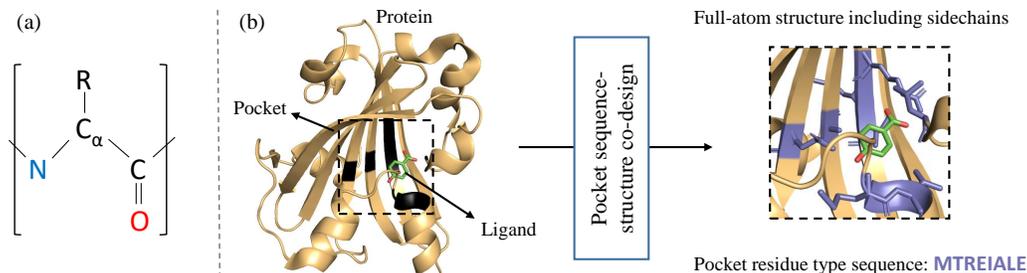


Figure 1: (a) The residue structure, where the backbone atoms are C_α, N, C, O . R represents a side chain that determines the residue types. (b) The protein pocket design problem. Pocket (colored in black) consists of a subsequence of residues closest to the binding ligand molecule (formal definition in Sec. 3.1). FAIR aims to co-design the full-atom structures and residue types of the pocket.

line of recent works proposes to co-design the sequence and 3D structure of antibodies as graphs [29, 28, 39, 33, 56]. However, they are designed explicitly for antibody CDR topologies and are primarily unable to consider side-chain atoms and the flexibility of binding targets. For more detailed discussions on protein design, interested readers may refer to the comprehensive surveys [22, 35].

Pocket Design. Protein pockets are the cavity region of protein where the ligand binds to the protein. In projects to design functional proteins such as biosensors [52, 27] and enzymes [5, 18], pocket design is the central task [8, 48]. Compared with the general protein design/antibody design problems, the pocket design brings unique challenges: (1) protein pocket has a smaller spatial scale than whole proteins/antibodies. Therefore, the structure and interactions of side-chain atoms play essential roles in pocket design and require careful modeling [36, 12]; (2) the influence and structural flexibility of target ligand molecules need to be considered to achieve high binding affinity [42]. Traditional methods for pocket design are physics-based methods [42, 11] or template-matching methods [67, 8, 48], which often suffer from the large computational complexity and limited accuracy. So far, the pocket design problem has rarely been tackled with deep learning.

Structure-Based Drug Design. Structure-based drug design is another line of research inquiry that explores 3D ligand molecule design inside protein pockets [49, 40, 37, 47, 70, 69, 19]. These studies can be regarded as the *dual* problem of pocket design studied in our paper. However, the methods primarily focus on generating 3D molecular graphs based on a fixed protein pocket structure and may not be easily adapted to the pocket sequence-structure co-design problem examined in our study. We refer interested readers to [71] for a comprehensive survey on structure-based drug design.

3 Full-Atom Protein Pocket Design

Next, we introduce FAIR, our method for pocket sequence-structure co-design. The problem is first formulated in Sec. 3.1. We then describe the hierarchical encoder that captures residue- and atom-level information in Sec. 3.2. Furthermore, we show the iterative refinement algorithms with two main steps in Sec. 3.3. Finally, we show the procedures for model training and sampling in Sec. 3.4 and demonstrate the E(3)-equivariance property of FAIR in Sec. 3.5.

3.1 Notation and Problem Statement

We aim to co-design residue types (sequences) and 3D structures of the protein pocket that can fit and bind with the target ligand molecule. As shown in Figure 1, a protein-ligand binding complex typically consists of a protein and a ligand molecule. The ligand molecule can be represented as a 3D point cloud $\mathcal{M} = \{(\mathbf{x}_k, \mathbf{v}_k)\}_{k=1}^{N_l}$ where \mathbf{x}_k denotes the 3D coordinates and \mathbf{v}_k denotes the atom feature. The protein can be represented as a sequence of residues $\mathcal{A} = \mathbf{a}_1 \cdots \mathbf{a}_{N_s}$, where N_s is the length of the sequence. The 3D structure of the protein can be described as a point cloud of protein atoms $\{\mathbf{a}_{i,j}\}_{1 \leq i \leq N_s, 1 \leq j \leq n_i}$ and let $\mathbf{x}(\mathbf{a}_{i,j}) \in \mathbb{R}^3$ denote the 3D coordinate of the pocket atoms. n_i is the number of atoms in a residue determined by the residue types. The first four atoms in any residue correspond to its backbone atoms (C_α, N, C, O), and the rest are its side-chain atoms. In our work, the protein pocket is defined as a set of residues in the protein closest to the binding ligand

molecule: $\mathcal{B} = \mathbf{b}_1 \cdots \mathbf{b}_m$. The pocket \mathcal{B} can thus be represented as an amino acid subsequence of a protein: $\mathcal{B} = \mathbf{a}_{e_1} \cdots \mathbf{a}_{e_m}$ where $e = \{e_1, \dots, e_m\}$ is the index of the pocket residues in the protein. The index e can be formally given as: $e = \{i \mid \min_{1 \leq j \leq n_i, 1 \leq k \leq N_l} \|\mathbf{x}(\mathbf{a}_{i,j}) - \mathbf{x}_k\|_2 \leq \delta\}$, where $\|\cdot\|_2$ is the L_2 norm and δ is the distance threshold. According to the distance range of pocket-ligand interactions [43], we set $\delta = 3.5 \text{ \AA}$ in the default setting. Note that the number of residues (i.e., m) varies for pockets due to varying structures of protein-ligand complexes. To summarize this subsection, our preliminary problem statement is formulated as below:

Problem Statement: In FAIR, our objective is to learn a conditional generative model for full-atom protein pocket generation conditioned on the protein backbone and the binding ligand. Given the availability of existing protein pockets, we may leverage them for reference to construct residue graphs and calculate geometric features. Our preliminary problem statement is formulated considering practical situations such as enzyme optimization.

3.2 Hierarchical Encoder

Leveraging this intrinsic multi-scale protein structure [59], we adopt a hierarchical graph transformer [44, 38, 68] to encode the hierarchical context information of protein-ligand complexes for pocket sequence-structure co-design. The transformer includes both atom- and residue-level encoders, as described below.

3.2.1 Atom-Level Encoder

The atom-level encoder considers all atoms in the protein-ligand complex and constructs a 3D context graph connecting the K_a nearest neighboring atoms in $\mathcal{A} \cup \mathcal{M}$. The atomic attributes are firstly mapped to node embeddings $\mathbf{h}_i^{(0)}$ with two MLPs, respectively, for protein and ligand molecule atoms. The edge embeddings \mathbf{e}_{ij} are obtained by processing pairwise Euclidean distances with Gaussian kernel functions [53]. The 3D graph transformer consists of L Transformer layers [62] with two modules in each layer: a multi-head self-attention (MHA) module and a position-wise feed-forward network (FFN). Specifically, in the MHA module of the l -th layer ($1 \leq l \leq L$), the queries are derived from the current node embeddings $\mathbf{h}_i^{(l)}$ while the keys and values come from the relational representations: $\mathbf{r}_{ij}^{(l)} = [\mathbf{h}_j^{(l)}, \mathbf{e}_{ij}^{(l)}]$ ($[\cdot, \cdot]$ denotes concatenation) from neighboring nodes:

$$\mathbf{q}_i^{(l)} = \mathbf{W}_Q \mathbf{h}_i^{(l)}, \mathbf{k}_{ij}^{(l)} = \mathbf{W}_K \mathbf{r}_{ij}^{(l)}, \mathbf{v}_{ij}^{(l)} = \mathbf{W}_V \mathbf{r}_{ij}^{(l)}, \quad (1)$$

where $\mathbf{W}_Q, \mathbf{W}_K$ and \mathbf{W}_V are learnable transformation matrices. Then, in each attention head $c \in \{1, 2, \dots, C\}$ (C is the total number of heads), the scaled dot-product attention mechanism works as follows:

$$\text{head}_i^c = \sum_{j \in \mathcal{N}(i)} \text{Softmax} \left(\frac{\mathbf{q}_i^{(l)\top} \cdot \mathbf{k}_{ij}^{(l)}}{\sqrt{d}} \right) \mathbf{v}_{ij}^{(l)}, \quad (2)$$

where $\mathcal{N}(i)$ denotes the neighbors of the i -th atom in the constructed graph and d is the dimension size of embeddings for each head. Finally, the outputs from different heads are further concatenated and transformed to obtain the final output of MHA:

$$\text{MHA}_i = \left[\text{head}_i^1, \dots, \text{head}_i^C \right] \mathbf{W}_O, \quad (3)$$

where \mathbf{W}_O is the output transformation matrix. The output of the atom-level encoder is a set of atom representations $\{\mathbf{h}_i\}$. Appendix A includes the details of the atom-level encoder.

3.2.2 Residue-Level Encoder

The residue-level encoder only keeps the C_α atoms of residues and a coarsened ligand node at the ligand's center of mass to supplement binding context information (the coarsened ligand node is appended at the end of the residue sequence as a special residue). Then, a K_r nearest neighbor graph is constructed at the residue level. We use the original protein pocket backbone atoms for reference to construct the K_r nearest neighbor graph and help calculate geometric features. The i -th residue \mathbf{a}_i can be represented by a feature vector $\mathbf{f}_{\mathbf{a}_i}^{res}$ describing its geometric and chemical characteristics such as volume, polarity, charge, hydrophathy, and hydrogen bond interactions according to its residue

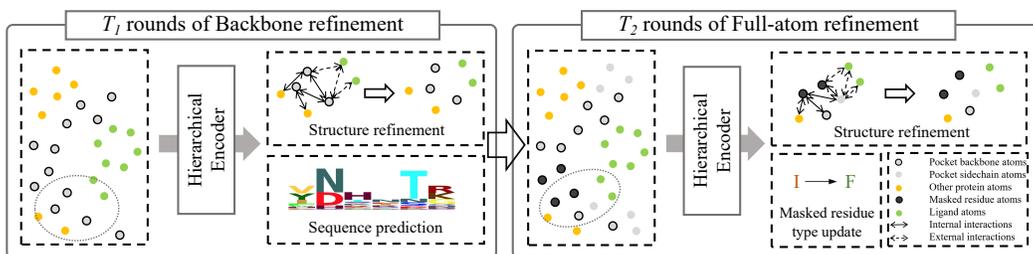


Figure 2: Overview of FAIR with two main steps (T_1 rounds of backbone refinement and T_2 rounds of full-atom refinement). FAIR co-designs pocket residue types and structures via iterative full-shot refinement. The structure refinement is illustrated with the atoms in the dotted ovals as examples.

type a_i . We concatenate the residue features with the sum of atom-level embeddings h_k within each residue to initialize the residue representations; the ligand node representation is initialized by summing up all the ligand atom embeddings:

$$\hat{f}_i^{res} = [\mathbf{f}_{a_i}^{res}, \sum_{j \in res_i} \mathbf{h}_j]; \hat{f}_i^{lig} = \sum_{k \in \mathcal{M}} \mathbf{h}_k. \quad (4)$$

As for the edge features e_{ij}^{res} , local coordinate frames are built for residues, and e_{ij}^{res} are computed according to the distance, direction, and orientation between neighboring residues [26]. Lastly, the encoder takes the node and edge features into the residue-level graph transformer to compute the residue-level representations. The residue-level graph transformer architecture is similar to that of the atom-level encoder, and details are shown in Appendix A.

In summary, the output of our hierarchical encoder is a set of residue representations $\{\mathbf{f}_i\}$ and atom representations $\{\mathbf{h}_i\}$, which are used in the following subsections for structure refinement and residue type prediction. Since our encoder is based on the atom/residue attributes and pairwise relative distances, the obtained representations are E(3)-invariant².

3.3 Iterative Refinement

For efficiency, FAIR predicts pocket residue types and structures via an iterative full-shot refinement scheme: all the pocket residue types and structures are updated together in each round. Since the pocket residue types and the number of side-chain atoms are largely unknown at the beginning rounds, FAIR is designed to have two main steps that follow a coarse-to-fine pipeline: FAIR firstly only models the backbone atoms of pockets to generate the coarse-grained structures and then fine-adjusts full-atom residues to achieve sequence-structure consistency. Figure 2 shows the overview of FAIR.

3.3.1 Backbone Refinement

Initialization. Before backbone refinement, the residue types to predict are initialized with uniform distributions over all residue type categories. Since the residue types and the number of side-chain atoms are unknown, we only initialize the 3D coordinates of backbone atoms (C_α, N, C, O) with linear interpolations and extrapolations based on the known structures of nearest residues in the protein. Details are shown in the Appendix A.4. Following previous research [8, 28, 33], we initialize the ligand structures with the reference structures from the dataset in the generation process.

Residue Type Prediction. In each round of iterative refinement, we first encode the current protein-ligand complex with the hierarchical encoder in Sec. 3.2. We then predict and update the sequences and structures. To predict the probability of each residue type, we apply an MLP and SoftMax function on the obtained pocket residue embeddings $\mathbf{f}_i: p_i = \text{Softmax}(\text{MLP}(\mathbf{f}_i))$, where $p_i \in \mathbb{R}^{20}$ is the predicted distribution over 20 residue type categories. The corresponding residue features can then be updated with weighted averaging: $\mathbf{f}_{b_i}^{res} \leftarrow \sum_j p_{i,j} \mathbf{f}_j^{res}$, where $p_{i,j}$ and \mathbf{f}_j^{res} are the probability and feature vector for the j -th residue category respectively.

Structure Refinement. Motivated by the force fields in physics [50] and the successful practice in previous works [28, 33], we calculate the interactions between atoms for the coordinate update.

²E(3) is the group of Euclidean transformations: rotations, reflections, and translations.

According to previous works [32, 12, 10], the internal interactions within the protein and the external interactions between protein and ligand are different. Therefore, FAIR uses two separate modules for interaction predictions. Firstly, the internal interactions are calculated based on residue embeddings:

$$\mathbf{g}_{i,j,kj} = g(\mathbf{f}_i, \mathbf{f}_k, e_{\text{RBF}}(\|\hat{\mathbf{x}}(\mathbf{b}_{i,1}) - \mathbf{x}(\mathbf{a}_{k,1})\|))_j \cdot \frac{\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j})}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j})\|}, \quad (5)$$

where $e_{\text{RBF}}(\|\hat{\mathbf{x}}(\mathbf{b}_{i,1}) - \mathbf{x}(\mathbf{a}_{k,1})\|)$ is the radial basis distance encoding of pairwise C_α distances, the function g is a feed-forward neural network with four output channels for four backbone atoms, and $\frac{\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j})}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j})\|} \in \mathbb{R}^3$ is the normalized vector from the residue atom $\mathbf{a}_{k,j}$ to $\mathbf{b}_{i,j}$. For the stability of training and generation, we use original protein backbone coordinates to calculate the RBF encodings. Similarly, for the external interactions between pocket \mathcal{B} and ligand \mathcal{M} , we have:

$$\mathbf{g}_{i,j,s} = g'(\mathbf{h}_{\mathbf{b}_{i,j}}, \mathbf{h}_s, e_{\text{RBF}}(\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s\|)) \cdot \frac{\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s\|}, \quad (6)$$

which is calculated based on the atom representations. g' is a feed-forward neural network with one output channel. For the conciseness of expression, here we use $\mathbf{h}_{\mathbf{b}_{i,j}}$ and \mathbf{h}_s to denote the pocket and ligand atom embeddings, respectively. Finally, the coordinates of pocket atoms $\mathbf{x}(\mathbf{b}_{i,j})$ and ligand molecule atoms \mathbf{x}_s are updated as follows:

$$\mathbf{x}(\mathbf{b}_{i,j}) \leftarrow \mathbf{x}(\mathbf{b}_{i,j}) + \frac{1}{K_r} \sum_{k \in \mathcal{N}(i)} \mathbf{g}_{i,j,kj} + \frac{1}{N_l} \sum_s \mathbf{g}_{i,j,s}, \quad (7)$$

$$\mathbf{x}_s \leftarrow \mathbf{x}_s - \frac{1}{|\mathcal{B}|} \sum_{i,j} \mathbf{g}_{i,j,s}, \quad 1 \leq i \leq m, 1 \leq j \leq 4, 1 \leq s \leq N_l, \quad (8)$$

where $\mathcal{N}(i)$ is the neighbors of the constructed residue-level K_r -NN graph; $|\mathcal{B}|$ denotes the number of pocket atoms. Details of the structure refinement modules are shown in the Appendix A.

3.3.2 Full-Atom Refinement

Initialization and Structure Refinement. After rounds of backbone refinements, the predicted residue type distribution and the backbone coordinates become relatively stable (verified in experiments). We then apply the full-atom refinement procedures to determine the side-chain structures. The residue types are initialized by sampling the predicted residue type p_i from backbone refinement (hard assignment). The number of side-chain atoms and the atom types can then be determined. The coordinates of the side-chain atoms are initialized with the corresponding α -carbon coordinates. The full-atom structure refinement is similar to backbone refinement (Sec. 3.3.1) with minor adjustments to accommodate the side-chain atoms. We include the details in the Appendix A.

Residue Type Update. Since the structures and interactions of side-chain atoms can further influence the residue types, proper residue type update algorithms are needed to achieve sequence-structure consistency. Considering the hard assignment of residue types in the full-atom refinement, we randomly sample and mask one residue in each protein pocket and update its residue type based on the surrounding environment in each iteration. Specifically, we mask the residue types, remove the sampled residues' side-chain atoms in the input to the encoder, and use the obtained corresponding residue types for residue type prediction. If the predicted result is inconsistent with the original residue type, we will update the residue type with the newly predicted one and reinitialize the side-chain structures. As our experiments indicate, the residue types and structures become stable after several refinement iterations, and sequence-structure consistency is achieved.

3.4 Model Training and Sampling

The loss function for FAIR consists of two parts. We apply cross-entropy loss l_{ce} for (sampled) pocket residues over all iterations for the sequence prediction. For the structure refinement, we adopt the Huber loss l_{huber} for the stability of optimization following previous works [28, 56]:

$$\mathcal{L}_{seq} = \frac{1}{T} \sum_t \sum_i l_{ce}(p_i^t, \hat{p}_i); \quad (9)$$

$$\mathcal{L}_{struct} = \frac{1}{T} \sum_t \left[\sum_i l_{huber}(\mathbf{x}(\mathbf{b}_i)^t, \hat{\mathbf{x}}(\mathbf{b}_i)) + \sum_j l_{huber}(\mathbf{x}_j^t, \hat{\mathbf{x}}_j) \right], \quad (10)$$

where $T = T_1 + T_2$ is the total rounds of refinement. \hat{p}_i , $\hat{\mathbf{x}}(\mathbf{b}_i)$, and $\hat{\mathbf{x}}_j$ are the ground-truth residue types, residue coordinates, and ligand coordinates. p_i^t , $\mathbf{x}(\mathbf{b}_i)^t$, and \mathbf{x}_j^t are the predicted ones at the t -th round. In the training process, we aim to minimize the sum of the above two loss functions $\mathcal{L} = \mathcal{L}_{seq} + \mathcal{L}_{struct}$ similar to previous works [56, 34]. After the training of FAIR, we can co-design pocket sequences and structures. Algorithm 1 and 2 in Appendix A outline model training and sampling.

3.5 Equivariance

FAIR is E(3)-equivariant, which is a desirable property of generative protein models [33, 64]:

Theorem 1. Denote the E(3)-transformation as T_g and the generative process of FAIR as $\{(\mathbf{b}_i, \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m = p_\theta(\mathcal{A} \setminus \mathcal{B}, \mathcal{M})$, where $\{(\mathbf{b}_i, \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m$ indicates the designed pocket sequence and structure. We have $\{(\mathbf{b}_i, T_g \cdot \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m = p_\theta(T_g \cdot (\mathcal{A} \setminus \mathcal{B}), T_g \cdot \mathcal{M})$.

Proof. The main idea is that the E(3)-invariant encoder and E(3)-equivariant structure refinement lead to an E(3)-equivariant generative process of FAIR. We give the full proof in Appendix A. \square

4 Experiments

We proceed by describing the experimental setups (Sec. 4.1). We then assess our method for ligand-binding pocket design (Sec. 4.2) on two datasets. We also perform further in-depth analysis of FAIR, including sampling efficiency analysis and ablations in Sec. 4.3.

4.1 Experimental Setting

Datasets. We consider two widely used datasets for experimental evaluations. **CrossDocked** dataset [16] contains 22.5 million protein-molecule pairs generated through cross-docking. We filter out data points with binding pose RMSD greater than 1 Å, leading to a refined subset with around 180k data points. For data splitting, we use mmseqs2 [58] to cluster data at 30% sequence identity, and randomly draw 100k protein-ligand structure pairs for training and 100 pairs from the remaining clusters for testing and validation, respectively. **Binding MOAD** dataset [21] contains around 41k experimentally determined protein-ligand complexes. We further filter and split the Binding MOAD dataset based on the proteins' enzyme commission number [4], resulting in 40k protein-ligand pairs for training, 100 pairs for validation, and 100 pairs for testing following previous work [54]. More data split schemes based on graph clustering [66, 65] may be explored.

Considering the distance ranges of protein-ligand interactions [43], we redesign all the residues that contain atoms within 3.5 Å of any binding ligand atoms, leading to an average of around eight residues for each pocket. We sample 100 sequences and structures for each protein-ligand complex in the test set for a comprehensive evaluation.

Baselines and Implementation. FAIR is compared with five state-of-the-art representative baseline methods. **PocketOptimizer** [45] is a physics-based method that optimizes energies such as packing and binding-related energies for ligand-binding protein design. **DEPACT** [8] is a template-matching method that follows a two-step strategy [67] for pocket design. It first searches the protein-ligand complexes in the database with similar ligand fragments. It then grafts the associated residues into the protein pocket with PACMatch [8]. **HSRN** [28], **Diffusion** [39], and **MEAN** [33] are deep-learning-based models that use auto-regressive refinement, diffusion model, and graph translation method respectively for protein sequence-structure co-design. They were originally proposed for antibody design, and we adapted them to our pocket design task with proper modifications (see Appendix. B for more details).

We use AMBER ff14S force field [41] for energy computation and the Dunbrack rotamer library [55] for rotamer sampling in PocketOptimizer. For the template-based method DEPACT, the template databases are constructed based on the training datasets for fair comparisons. For the deep-learning-based models, including our FAIR, we train them for 50 epochs and select the checkpoint with the lowest loss on the validation set for testing. We use the Adam optimizer with a learning rate of 0.0001 for optimization. In FAIR, the default setting sets T_1 and T_2 as 5 and 10. Descriptions of baselines and FAIR are provided in the Appendix. B.

Table 1: Evaluation of different approaches on the pocket design task.

Model	CrossDocked			Binding MOAD		
	AAR (\uparrow)	RMSD (\downarrow)	Vina (\downarrow)	AAR (\uparrow)	RMSD (\downarrow)	Vina (\downarrow)
PocketOptimizer	27.89 \pm 14.9%	1.75 \pm 0.08	-6.905 \pm 2.39	28.78 \pm 11.3%	1.68 \pm 0.12	-7.829 \pm 2.41
DEPACT	22.58 \pm 8.48%	1.97 \pm 0.14	-6.670 \pm 2.13	26.12 \pm 8.97%	1.76 \pm 0.15	-7.526 \pm 2.05
HSRN	31.62 \pm 10.4%	2.15 \pm 0.17	-6.565 \pm 1.95	33.70 \pm 10.1%	1.83 \pm 0.18	-7.349 \pm 1.93
Diffusion	34.62 \pm 13.7%	1.68 \pm 0.12	-6.725 \pm 1.83	36.94 \pm 12.9%	1.47 \pm 0.09	-7.724 \pm 2.36
MEAN	35.46 \pm 8.15%	1.76 \pm 0.09	-6.891 \pm 1.86	37.16 \pm 14.7%	1.52 \pm 0.09	-7.651 \pm 1.97
FAIR	40.17\pm12.6%	1.42\pm0.07	-7.022\pm1.75	43.75\pm15.2%	1.35\pm0.10	-7.978\pm1.91

Performance Metrics. To evaluate the generated sequences and structures, we employ Amino Acid Recovery (AAR), defined as the overlapping ratio between the predicted 1D sequences and ground truths, and Root Mean Square Deviation (RMSD) regarding the 3D predicted structure of residue atoms. Due to the different residue types and numbers of side-chain atoms in the generated pockets, we calculate RMSD with respect to backbone atoms following [28, 33]. To measure the binding affinity between the protein pocket and ligand molecules, we calculate **Vina Score** with QVina [61, 2]. The unit is kcal/mol; a lower Vina score indicates better binding affinity. Before feeding to Vina, all the generated protein structures are firstly refined by AMBER force field [41] in OpenMM [13] to optimize the structures. For the baseline methods that predict backbone structures only (Diffusion and MEAN), we use Rosetta [1] to do side-chain packing following the instructions in their papers.

4.2 Ligand-Binding Pocket Design

Comparison with Baselines. Table 1 shows the results of different methods on the CrossDocked and Binding MOAD dataset. FAIR overperforms previous methods with a clear margin on AAR, RMSD, and Vina scores, which verifies the strong ability of FAIR to co-design pocket sequences and structures with high binding affinity. For example, the average improvements on AAR and RMSD are 15.5% and 13.5% respectively. Compared with the energy-based method PocketOptimizer and the template-matching-based method DEPACT, deep-learning-based methods such as FAIR have better performance due to their stronger ability to model the complex sequence-structure dependencies and pocket-ligand interactions. However, these deep-learning-based methods still have limitations that restrict their performance improvement. For example, the autoregressive generation manner in HSRN inevitably incurs error accumulation; Diffusion and MEAN fail to model the influence of side chain atoms; all these methods ignore the flexibility of ligands. In contrast, FAIR addresses the limitations with properly designed modules, further validated in Sec. 4.3. Moreover, FAIR performs relatively better on Binding MOAD than the CrossDocked dataset, which may be explained by the higher quality and diversity of Binding MOAD [21] (e.g. Vina score on CrossDocked -7.022 vs. -7.978 on Binding MOAD). For example, Binding MOAD has 40k unique protein-ligand complexes, while CrossDocked only contains 18k complexes.

Comparison of Generated and Native Sequences. Figure 4(a) shows the residue type distributions of the designed pockets and the native ones from two datasets. Generally, we observe that the distributions align very well. Some residue types, such as Tyrosine and Asparagine, have been used more frequently in designed than naturally occurring sequences. In contrast, residue types with obviously reduced usages in the designed sequences include Lysine, Isoleucine, and Leucine.

Convergence of Iterative Refinement Process. During the iterations of FAIR, we monitor the ratio of unconverged residues (residue types different from those in the final designed sequences) and AAR, shown in Figure 4(b) ($T_1 = 5$ and $T_2 = 15$). We observe that both curves converge quickly during refinement. We find that the AAR becomes stable in just several iterations (typically less than 3 iterations in our experiments) of backbone refinement and then gradually increases with the complete atom refinement, validating the effectiveness of our design. We show the curves of RMSD in Appendix C.3, which also converges rapidly. Therefore, we set $T_1 = 5$ and $T_2 = 10$ for efficiency as default setting.

Case Studies. In Figure. 3, we visualize two examples of designed pocket sequences and structures from the CrossDocked dataset (PDB: 2pc8) and the Binding MOAD dataset (PDB: 6p7r), respectively. The designed sequences recover most residues, and the generated structures are valid and realistic. Moreover, the designed pockets exhibit comparable or higher binding affinities (lower Vina scores) with the target ligand molecules than the reference pockets in the test set.

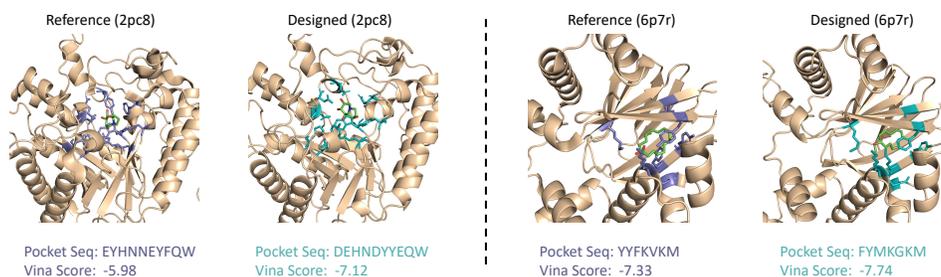


Figure 3: Case studies of Pocket design. We show the reference and designed structures/sequences of two protein pockets from the CrossDocked (PDB: 2pc8) and Binding MOAD datasets (PDB: 6p7r).

4.3 Additional Analysis of FAIR’s Performance

Sampling Efficiency Analysis. To evaluate the efficiency of FAIR, we considered the generation time of different approaches using a single V100 GPU on the same machine. Average runtimes are shown in Figure 4(c). First, the generation time of FAIR is more than ten times faster (FAIR: 5.6s vs. DEPACT: 107.2s) than traditional PocketOptimizer and DEPACT methods. Traditional methods are time-consuming because they require explicit energy calculations and combinatorial enumerations. Second, FAIR is more efficient than the autoregressive HSRN and diffusion methods. The MEAN sampling time is comparable to FAIR since both use full-shot decoding schemes. However, FAIR has the advantage of full-atom prediction over MEAN.

Local Geometry of Generated Pockets. Since RMSD values in Tables 1 and 2 reflect only the global geometry of generated protein pockets, we also consider the RMSD of bond lengths and angles of predicted pockets to evaluate their local geometry [33]. We measure the average RMSD of the covalent bond lengths in the pocket backbone (C-N, C=O, and C-C). We consider three conventional dihedral angles in the backbone structure, i.e., ϕ , ψ , ω [57] and calculate the average RMSD of their cosine values. The results in Table 3 of Appendix C.3 show that FAIR achieves better performance regarding the local geometry than baseline methods. The RMSD of bond lengths and angles drop 20% and 8% respectively with FAIR.

Ablations. In Table 2, we evaluate the effectiveness and necessity of the proposed modules in FAIR. Specifically, we removed the residue-level encoder, the masked residue type update procedure, the full-atom refinement, and the ligand molecule context, denoted as w/o the encoder/consistency/full-atom/ligand. The results in Table 2 demonstrate that the removal of any of these modules leads to a decrease in performance. We observe that the performance of FAIR w/o hierarchical encoder drops the most. This is reasonable, as the hierarchical encoder captures essential representations for sequence/structure prediction. Moreover, the masked residue type update procedure helps ensure sequence-structure consistency. Additionally, the full-atom refinement module and the binding ligand are essential for modeling side-chain and pocket-ligand interactions, which play important roles in protein-ligand binding.

Further, we replaced the iterative full-shot refinement modules in FAIR with the autoregressive decoding method used in HSRN [28] as a variant of FAIR. However, this variant’s performance is also worse than FAIR’s. As discussed earlier, autoregressive generation can accumulate errors and is more time-consuming than the full-shot refinement in FAIR. For example, AAR drops from 40.17% to 31.04% on CrossDocked with autoregressive generation.

Table 2: Ablation studies of FAIR.

Model	CrossDocked			Binding MOAD		
	AAR (\uparrow)	RMSD (\downarrow)	Vina (\downarrow)	AAR (\uparrow)	RMSD (\downarrow)	Vina (\downarrow)
w/o hier encoder	25.30 \pm 9.86%	2.36 \pm 0.24	-6.694 \pm 2.28	29.21 \pm 10.8%	1.82 \pm 0.14	-7.235 \pm 2.10
w/o consistency	36.75 \pm 9.21%	1.49 \pm 0.08	-6.937 \pm 1.72	37.40 \pm 15.0%	1.49 \pm 0.12	-7.752 \pm 1.94
w/o full-atom	35.52 \pm 10.4%	1.62 \pm 0.13	-6.412 \pm 2.37	35.91 \pm 13.5%	1.54 \pm 0.12	-7.544 \pm 2.23
w/o ligand	33.16 \pm 15.3%	1.71 \pm 0.11	-6.353 \pm 1.91	34.54 \pm 14.6%	1.50 \pm 0.09	-7.271 \pm 1.79
autoregressive	31.04 \pm 14.2%	1.56 \pm 0.16	-6.874 \pm 1.90	33.65 \pm 12.7%	1.46 \pm 0.11	-7.765 \pm 1.84
FAIR	40.17\pm12.6%	1.42\pm0.07	-7.022\pm1.75	43.75\pm15.2%	1.35\pm0.10	-7.978\pm1.91

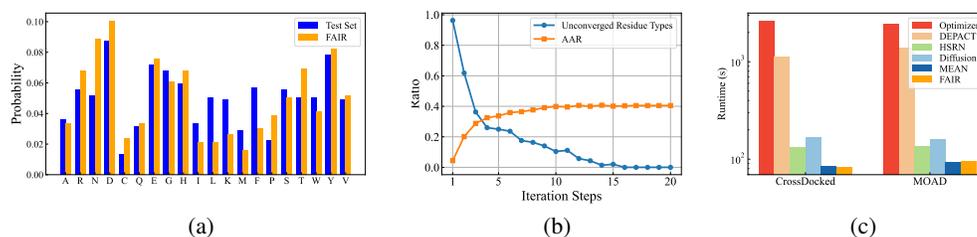


Figure 4: (a) The distributions of residue types in the designed pockets and the test sets. (b) The ratio of unconverged residues (residue types different from those in the final designed sequences) and AAR (Amino Acid Recovery) during the iterations of FAIR on the CrossDocked dataset. (c) Comparisons of average generation time for 100 pockets with baseline methods.

5 Limitations and Future Research

While we have made significant strides in addressing the limitations of prior research, several avenues remain for future research. First, we preliminarily formulate the protein pocket co-design problem in this paper. In the future, more realistic problem formulations may be constructed according to the requirements of different scenarios. Second, beyond the design of protein pockets, optimizing properties such as the binding affinity of existing pockets is crucial, especially for therapeutic applications. In forthcoming research, we aim to combine reinforcement learning and Bayesian optimization with FAIR to refine pocket design. Third, leveraging physics-based and template-matching methodologies can pave the way for the development of hybrid models that are both interpretable and generalizable. For instance, physics-based techniques can be employed to sample various ligand conformations during ligand initialization. The inherent protein-ligand interaction data within templates can guide pocket design. Lastly, integrating FAIR into protein/enzyme design workflows has the potential to be immensely beneficial for the scientific community. Conducting wet-lab experiments to assess the efficacy of protein pockets designed using FAIR would be advantageous. Insights gleaned from these experimental outcomes can be instrumental in refining our model, thereby establishing a symbiotic relationship between computation and biochemical experiments.

6 Conclusion

We develop a full-atom iterative refinement framework (FAIR) for protein pocket sequence and 3D structure co-design. Generally, FAIR has two refinement steps (backbone refinement and full-atom refinement) and follows a coarse-to-fine pipeline. The influence of side-chain atoms, the flexibility of binding ligands, and sequence-structure consistency are well considered and addressed. We empirically evaluate our method on cross-docked and experimentally determined datasets to show the advantage over existing physics-based, template-matching-based, and deep generative methods in efficiently generating high-quality pockets. We hope our work can inspire further explorations of pocket design with deep generative models and benefit the broad community.

7 Acknowledgements

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2021YFF0901003) and the University Synergy Innovation Program of Anhui Province (GXXT-2021-002). We wish to express our sincere appreciation to Dr. Yaoxi Chen and Yifei Hu for their constructive discussions, which greatly enrich this research.

References

- [1] Rebecca F Alford, Andrew Leaver-Fay, Jeliasko R Jeliaskov, Matthew J O’Meara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 13(6):3031–3048, 2017.

- [2] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015.
- [3] Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, et al. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.
- [4] Amos Bairoch. The enzyme database in 2000. *Nucleic acids research*, 28(1):304–305, 2000.
- [5] Matthew J Bick, Per J Greisen, Kevin J Morey, Mauricio S Antunes, David La, Banumathi Sankaran, Luc Reymond, Kai Johnsson, June I Medford, and David Baker. Computational design of environmental sensors for the potent opioid fentanyl. *Elife*, 6:e28909, 2017.
- [6] F Edward Boas and Pehr B Harbury. Potential energy functions for protein design. *Current opinion in structural biology*, 17(2):199–204, 2007.
- [7] Yue Cao, Payel Das, Vijil Chenthamarakshan, Pin-Yu Chen, Igor Melnyk, and Yang Shen. Fold2seq: A joint sequence (1d)-fold (3d) embedding-based generative model for protein design. In *International Conference on Machine Learning*, pages 1261–1271. PMLR, 2021.
- [8] Yaoxi Chen, Quan Chen, and Haiyan Liu. Depact and pacmatch: A workflow of designing de novo protein pockets to bind small molecules. *Journal of Chemical Information and Modeling*, 62(4):971–985, 2022.
- [9] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning-based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.
- [10] Christopher M Dobson. Protein folding and misfolding. *Nature*, 426(6968):884–890, 2003.
- [11] Jiayi Dou, Lindsey Doyle, Per Jr Greisen, Alberto Schena, Hahnbeom Park, Kai Johnsson, Barry L Stoddard, and David Baker. Sampling and energy evaluation challenges in ligand binding protein design. *Protein Science*, 26(12):2426–2437, 2017.
- [12] Xing Du, Yi Li, Yuan-Ling Xia, Shi-Meng Ai, Jing Liang, Peng Sang, Xing-Lai Ji, and Shu-Qun Liu. Insights into protein–ligand interactions: mechanisms, models, and methods. *International journal of molecular sciences*, 17(2):144, 2016.
- [13] Peter Eastman, Jason Swails, John D Chodera, Robert T McGibbon, Yutong Zhao, Kyle A Beauchamp, Lee-Ping Wang, Andrew C Simmonett, Matthew P Harrigan, Chaya D Stern, et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7):e1005659, 2017.
- [14] Noelia Ferruz and Birte Höcker. Controllable protein design with language models. *Nature Machine Intelligence*, 4(6):521–532, 2022.
- [15] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.
- [16] Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of chemical information and modeling*, 60(9):4200–4215, 2020.
- [17] Wenhao Gao, Sai Pooja Mahajan, Jeremias Sulam, and Jeffrey J Gray. Deep learning in protein structural modeling and design. *Patterns*, 1(9):100142, 2020.
- [18] Anum A Glasgow, Yao-Ming Huang, Daniel J Mandell, Michael Thompson, Ryan Ritterson, Amanda L Loshbaugh, Jenna Pellegrino, Cody Krivacic, Roland A Pache, Kyle A Barlow, et al. Computational design of a modular protein sense-response system. *Science*, 366(6468):1024–1028, 2019.
- [19] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *International Conference on Learning Representations*, 2023.
- [20] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *ICML*, 2022.

- [21] Liegi Hu, Mark L Benson, Richard D Smith, Michael G Lerner, and Heather A Carlson. Binding moad (mother of all databases). *Proteins: Structure, Function, and Bioinformatics*, 60(3):333–340, 2005.
- [22] Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, 2016.
- [23] Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. *arXiv preprint arXiv:2203.06442*, 2022.
- [24] Peter J Huber. Robust estimation of a location parameter. *Breakthroughs in statistics: Methodology and distribution*, pages 492–518, 1992.
- [25] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.
- [26] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.
- [27] Lin Jiang, Eric A. Althoff, Fernando R. Clemente, Lindsey Doyle, Daniela Röthlisberger, Alexandre Zanghellini, Jasmine L. Gallaher, Jamie L. Betker, Fujie Tanaka, Carlos F. Barbas, Donald Hilvert, Kendall N. Houk, Barry L. Stoddard, and David Baker. De novo computational design of retro-aldol enzymes. *Science*, 319(5868):1387–1391, 2008.
- [28] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Antibody-antigen docking and design via hierarchical structure refinement. In *ICML*, pages 10217–10227. PMLR, 2022.
- [29] Wengong Jin, Jeremy Wohlwend, Regina Barzilay, and Tommi Jaakkola. Iterative refinement graph neural network for antibody sequence-structure co-design. *ICLR*, 2022.
- [30] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *ICLR*, 2021.
- [31] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- [32] Judith Klein-Seetharaman, Maki Oikawa, Shaun B Grimshaw, Julia Wirmer, Elke Duchardt, Tadashi Ueda, Taiji Imoto, Lorna J Smith, Christopher M Dobson, and Harald Schwalbe. Long-range interactions within a nonnative protein. *Science*, 295(5560):1719–1722, 2002.
- [33] Xiangzhe Kong, Wenbing Huang, and Yang Liu. Conditional antibody design as 3d equivariant graph translation. *ICLR*, 2023.
- [34] Xiangzhe Kong, Wenbing Huang, and Yang Liu. End-to-end full-atom antibody design. *arXiv preprint arXiv:2302.00203*, 2023.
- [35] Ivan V Korendovych and William F DeGrado. De novo protein design, a retrospective. *Quarterly reviews of biophysics*, 53:e3, 2020.
- [36] Brian Kuhlman and Philip Bradley. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology*, 20(11):681–697, 2019.
- [37] Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3d molecules for target protein binding. *ICML*, 2022.
- [38] Shengjie Luo, Tianlang Chen, Yixian Xu, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. One transformer can understand both 2d & 3d molecular data. *ICLR*, 2022.
- [39] Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models. *NeurIPS*, 2022.
- [40] Youzhi Luo and Shuiwang Ji. An autoregressive flow model for 3d molecular geometry generation from scratch. In *ICLR*, 2021.
- [41] James A Maier, Carmenza Martinez, Koushik Kasavajhala, Lauren Wickstrom, Kevin E Hauser, and Carlos Simmerling. ff14sb: improving the accuracy of protein side chain and backbone parameters from ff99sb. *Journal of chemical theory and computation*, 11(8):3696–3713, 2015.
- [42] Christoph Malisi, Marcel Schumann, Nora C Toussaint, Jorge Kageyama, Oliver Kohlbacher, and Birte Höcker. Binding pocket optimization by computational protein design. *PLoS one*, 7(12):e52505, 2012.

- [43] Gilles Marcou and Didier Rognan. Optimizing fragment and scaffold docking by use of molecular interaction fingerprints. *Journal of chemical information and modeling*, 47(1):195–207, 2007.
- [44] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.
- [45] Jakob Noske, Josef Paul Kynast, Dominik Lemm, Steffen Schmidt, and Birte Höcker. Pocketoptimizer 2.0: A modular framework for computer-aided ligand-binding design. *Protein Science*, 32(1):e4516, 2023.
- [46] Sergey Ovchinnikov and Po-Ssu Huang. Structure-based protein design with deep learning. *Current opinion in chemical biology*, 65:136–144, 2021.
- [47] Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. *ICML*, 2022.
- [48] Nicholas F Polizzi and William F DeGrado. A defined structural unit enables de novo design of small-molecule-binding proteins. *Science*, 369(6508):1227–1233, 2020.
- [49] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Generating 3d molecules conditional on receptor binding sites with deep generative models. *Chemical science*, 13(9):2701–2713, 2022.
- [50] Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American chemical society*, 114(25):10024–10035, 1992.
- [51] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.
- [52] Daniela Röthlisberger, Olga Khersonsky, Andrew M Wollacott, Lin Jiang, Jason DeChancie, Jamie Betker, Jasmine L Gallaher, Eric A Althoff, Alexandre Zanghellini, Orly Dym, et al. Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195, 2008.
- [53] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.
- [54] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022.
- [55] Maxim V Shapovalov and Roland L Dunbrack Jr. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*, 19(6):844–858, 2011.
- [56] Chence Shi, Chuanrui Wang, Jiarui Lu, Bozитай Zhong, and Jian Tang. Protein sequence and structure co-design with equivariant translation. *ICLR*, 2023.
- [57] Ryan K Spencer, Glenn L Butterfoss, John R Edison, James R Eastwood, Stephen Whitlam, Kent Kirshenbaum, and Ronald N Zuckermann. Stereochemistry of polypeptoid chain configurations. *Biopolymers*, 110(6):e23266, 2019.
- [58] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.
- [59] H Stephen Stoker. *General, organic, and biological chemistry*. Cengage Learning, 2015.
- [60] Arthur G Street and Stephen L Mayo. Computational protein design. *Structure*, 7(5):R105–R109, 1999.
- [61] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

- [62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [63] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tiejian Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.
- [64] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *ICLR*, 2022.
- [65] Xihong Yang, Jiaqi Jin, Siwei Wang, Ke Liang, Yue Liu, Yi Wen, Suyuan Liu, Sihang Zhou, Xinwang Liu, and En Zhu. Dealmvc: Dual contrastive calibration for multi-view clustering. In *Proceedings of the 31th ACM International Conference on Multimedia*, 2023.
- [66] Xihong Yang, Cheng Tan, Yue Liu, Ke Liang, Siwei Wang, Sihang Zhou, Jun Xia, Stan Z Li, Xinwang Liu, and En Zhu. Convert: Contrastive graph clustering with reliable augmentation. In *Proceedings of the 31th ACM International Conference on Multimedia*, 2023.
- [67] Alexandre Zanghellini, Lin Jiang, Andrew M Wollacott, Gong Cheng, Jens Meiler, Eric A Althoff, Daniela Röthlisberger, and David Baker. New algorithms and an in silico benchmark for computational enzyme design. *Protein Science*, 15(12):2785–2794, 2006.
- [68] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 2022.
- [69] Zaixi Zhang, Qi Liu, Chee-Kong Lee, Chang-Yu Hsieh, and Enhong Chen. An equivariant generative framework for molecular graph-structure co-design. *Chemical Science*, 14(31):8380–8392, 2023.
- [70] Zaixi Zhang, Yaosen Min, Shuxin Zheng, and Qi Liu. Molecule generation for target protein binding with structural motifs. In *ICLR*, 2023.
- [71] Zaixi Zhang, Jiaxian Yan, Qi Liu, Enhong Chen, and Marinka Zitnik. A systematic survey in geometric deep learning for structure-based drug design. *arXiv preprint arXiv:2306.11768*, 2023.

A Details on FAIR Model

A.1 Training and Sampling Algorithms

Here, we show the pseudo-codes of the training and sampling of FAIR in the Algorithm. 1 and 2.

Algorithm 1 Training of FAIR

Input: protein sequences $\mathcal{A} = \mathbf{a}_1 \cdots \mathbf{a}_{N_s}$ and structures $\{\mathbf{x}(\mathbf{a}_i)\}_{i=1}^{N_s}$, ligand molecule \mathcal{M} .
Initialize the coordinates of pocket residues $\{\mathbf{x}(\mathbf{b}_i)\}$.
Add Gaussian noise to the ligand molecular structure $\{\mathbf{x}_j\}$.
Initialize the sequences of pocket residues with uniform distributions over 20 residue categories.
 $\mathcal{L}_{pred} = 0$; $\mathcal{L}_{struct} = 0$.
Backbone refinement
for t in $1, \dots, T_1$:
 Obtain the residue embeddings $\{\mathbf{f}_i\}$ and atom embeddings $\{\mathbf{h}_i\}$ with the hierarchical encoder.
 Predict residue types and update coordinates with structure refinement modules.
 $\mathcal{L}_{seq} + = \sum_i l_{ce}(p_i^t, \hat{p}_i)$.
 $\mathcal{L}_{struct} + = \sum_i l_{huber}(\mathbf{x}(\mathbf{b}_i)^t, \hat{\mathbf{x}}(\mathbf{b}_i)) + \sum_j l_{huber}(\mathbf{x}_j^t, \hat{\mathbf{x}}_j)$.
end for
Sample residue types from the predicted residue type distributions.
Initialize the coordinates of side-chain atoms with the coordinates of corresponding C_α .
Full-atom refinement
for t in $1, \dots, T_2$:
 Randomly mask a subset of pocket residues.
 Obtain the residue embeddings $\{\mathbf{f}_i\}$ and atom embeddings $\{\mathbf{h}_i\}$ with the hierarchical encoder.
 Predict the residue types of masked residues.
 $\mathcal{L}_{seq} + = \sum_i l_{ce}(p_i^t, \hat{p}_i)$.
 Reinitialize the masked residue if its residue type conflicts with the predicted result.
 Update coordinates with structure refinement modules.
 $\mathcal{L}_{struct} + = \sum_i l_{huber}(\mathbf{x}(\mathbf{b}_i)^t, \hat{\mathbf{x}}(\mathbf{b}_i)) + \sum_j l_{huber}(\mathbf{x}_j^t, \hat{\mathbf{x}}_j)$.
end for
Minimize $\frac{1}{T}(\mathcal{L}_{seq} + \mathcal{L}_{struct})$.

Algorithm 2 Sampling of FAIR

Input: protein sequences $\mathcal{A} = \mathbf{a}_1 \cdots \mathbf{a}_{N_s}$ and structures $\{\mathbf{x}(\mathbf{a}_i)\}_{i=1}^{N_s}$, ligand molecule \mathcal{M} .
Initialize the coordinates of pocket residues $\{\mathbf{x}(\mathbf{b}_i)\}$.
Initialize the sequences of pocket residues with uniform distributions over 20 residue categories.
Backbone refinement
for t in $1, \dots, T_1$:
 Obtain the residue embeddings $\{\mathbf{f}_i\}$ and atom embeddings $\{\mathbf{h}_i\}$ with the hierarchical encoder.
 Predict residue types and update coordinates with structure refinement modules.
end for
Sample residue types from the predicted residue type distributions.
Initialize the coordinates of side-chain atoms with the coordinates of corresponding C_α .
Full-atom refinement
for t in $1, \dots, T_2$:
 Randomly mask a subset of pocket residues.
 Obtain the residue embeddings $\{\mathbf{f}_i\}$ and atom embeddings $\{\mathbf{h}_i\}$ with the hierarchical encoder.
 Predict the residue types of masked residues.
 Reinitialize the masked residue if its residue type conflicts with the predicted result.
 Update coordinates with structure refinement modules.
end for

A.2 Additional Information on Hierarchical Graph Transformer Encoder

Our hierarchical encoder includes the atom-level encoder and the residue-level encoder. The powerful 3D graph transformer is used as the model backbone. Here, we first provide additional information on the graph transformer. Residue-level node and edge feature representations follow prior research [26, 28].

Graph transformer architecture. The atom/residue-level encoder contains L graph transformer layers. Let $\mathbf{h}^{(l)}$ be the set of node representations at the l -th layer. In each graph transformer layer, there is a multi-head self-attention (MHA) and a feed-forward block (FFN). The layer normalization (LN) is applied before the two blocks [63, 68]. The details of MHA have been shown in Sec 3.2, and the graph transformer layer is formally characterized as:

$$\mathbf{h}'^{(l-1)} = \text{MHA}(\text{LN}(\mathbf{h}^{(l-1)})) + \mathbf{h}^{(l-1)} \quad (11)$$

$$\mathbf{h}^{(l)} = \text{FFN}(\text{LN}(\mathbf{h}'^{(l-1)})) + \mathbf{h}'^{(l-1)}, \quad (0 \leq l < L). \quad (12)$$

Residue-level node features. The residue-level encoder only keeps the C_α atoms to represent residues and constructs a K_r nearest neighbor graph at the residue level. We use the original protein pocket backbone atoms for reference to construct the K_r nearest neighbor graph and help calculate geometric features. Each residue node is represented by six features: polarity $f_p \in \{0, 1\}$, hydrophathy $f_h \in [-4.5, 4.5]$, volume $f_v \in [60.1, 227.8]$, charge $f_c \in \{-1, 0, 1\}$, and whether it is a hydrogen bond donor $f_d \in \{0, 1\}$ or acceptor $f_a \in \{0, 1\}$. We expand hydrophathy and volume features into radial basis with interval sizes 0.1 and 10, respectively. Overall, the dimension of the residue-level feature vector \mathbf{f}_i is 112.

Residue-level edge features. For the i -th residue, we let $\mathbf{x}(\mathbf{a}_{i,1})$ denote the coordinate of its C_α and define its local coordinate frame $\mathbf{O}_i = [\mathbf{c}_i, \mathbf{n}_i, \mathbf{c}_i \times \mathbf{n}_i]$ as:

$$\mathbf{u}_i = \frac{\mathbf{x}(\mathbf{a}_{i,1}) - \mathbf{x}(\mathbf{a}_{i-1,1})}{\|\mathbf{x}(\mathbf{a}_{i,1}) - \mathbf{x}(\mathbf{a}_{i-1,1})\|}, \quad \mathbf{c}_i = \frac{\mathbf{u}_i - \mathbf{u}_{i+1}}{\|\mathbf{u}_i - \mathbf{u}_{i+1}\|}, \quad \mathbf{n}_i = \frac{\mathbf{u}_i \times \mathbf{u}_{i+1}}{\|\mathbf{u}_i \times \mathbf{u}_{i+1}\|}. \quad (13)$$

Based on the local frame, the edge features between residues i and j can be computed as:

$$\mathbf{e}_{ij}^{res} = \text{Concat} \left(E_{\text{pos}}(i-j), e_{\text{RBF}}(\|\mathbf{x}(\mathbf{a}_{i,1}) - \mathbf{x}(\mathbf{a}_{j,1})\|), \mathbf{O}_i^\top \frac{\mathbf{x}(\mathbf{a}_{j,1}) - \mathbf{x}(\mathbf{a}_{i,1})}{\|\mathbf{x}(\mathbf{a}_{i,1}) - \mathbf{x}(\mathbf{a}_{j,1})\|}, \mathbf{q}(\mathbf{O}_i^\top \mathbf{O}_j) \right). \quad (14)$$

The edge feature \mathbf{e}_{ij}^{res} contains four parts. The positional encoding $E_{\text{pos}}(i-j)$ encodes the relative sequence distance between two residues. The second term $e_{\text{RBF}}(\cdot)$ is a distance encoding with radial basis functions. The third term is a direction encoding corresponding to the relative direction of $\mathbf{x}(\mathbf{a}_{j,1})$ in the local frame of the i -th residue. The last term $\mathbf{q}(\mathbf{O}_i^\top \mathbf{O}_j)$ is the orientation encoding of the quaternion representation $\mathbf{q}(\cdot)$ of the spatial rotation matrix $\mathbf{O}_i^\top \mathbf{O}_j$ [25]. Overall, the dimension of the residue-level edge feature \mathbf{e}_{ij}^{res} is 39.

A.3 More Details of the Full-atom Structure Refinement

The full-atom structure refinement is similar to the backbone structure refinement in Sec. 3.3.1 with minor modifications to accommodate side-chain atoms. We consider the internal interactions within the protein and the external interactions between ligand molecules and protein residues. Firstly, the internal interactions within the protein are calculated based on residue embeddings:

$$\mathbf{g}_{ij,kj} = g(\mathbf{f}_i, \mathbf{f}_k, e_{\text{RBF}}(\|\hat{\mathbf{x}}(\mathbf{b}_{i,1}) - \mathbf{x}(\mathbf{a}_{k,1})\|))_j \cdot \frac{\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k)}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k)\|}, \quad (15)$$

where $e_{\text{RBF}}(\|\hat{\mathbf{x}}(\mathbf{b}_{i,1}) - \mathbf{x}(\mathbf{a}_{k,1})\|)$ is the radial basis distance encoding of pairwise C_α distances, the function g is a feed-forward neural network with 14 channels for all side-chain atoms (the maximum number of atoms in a residue is 14). Due to the different numbers of side-chain atoms in different kinds of residues, the normalized direction vector here is $\frac{\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k)}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k)\|} \in \mathbb{R}^3$ where $\bar{\mathbf{x}}(\mathbf{a}_k)$ is the average coordinates of residue \mathbf{a}_k . The external interactions between pocket \mathcal{B} and ligand \mathcal{M} is the same as the backbone refinement:

$$\mathbf{g}_{ij,s} = g'(\mathbf{h}_{\mathbf{b}_{i,j}}, \mathbf{h}_s, e_{\text{RBF}}(\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s\|)) \cdot \frac{\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s}{\|\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s\|}, \quad (16)$$

which is calculated based on the atom representations. For the conciseness of expression, here we use $\mathbf{h}_{\mathbf{b}_{i,j}}$ and \mathbf{h}_s to denote the pocket and ligand atom embeddings, respectively. Finally, the coordinates of pocket atoms $\mathbf{x}(\mathbf{b}_{i,j})$ and ligand molecule atoms \mathbf{x}_s are updated as follows:

$$\mathbf{x}(\mathbf{b}_{i,j}) \leftarrow \mathbf{x}(\mathbf{b}_{i,j}) + \frac{1}{K_r} \sum_{k \in \mathcal{N}(i)} \mathbf{g}_{ij,kj} + \frac{1}{N_l} \sum_s \mathbf{g}_{ij,s}, \quad (17)$$

$$\mathbf{x}_s \leftarrow \mathbf{x}_s - \frac{1}{|\mathcal{B}|} \sum_{i,j} \mathbf{g}_{ij,s}, \quad 1 \leq i \leq m, 1 \leq j \leq n_i, 1 \leq s \leq N_l, \quad (18)$$

where $\mathcal{N}(i)$ indicates the neighbors in the constructed residue-level K_r -NN graph; $|\mathcal{B}|$ denotes the number of pocket atoms.

A.4 Structure initialization based on interpolation and extrapolation

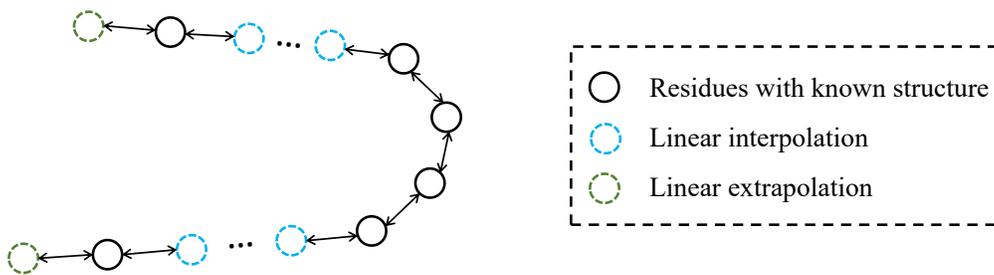


Figure 5: Structure initialization based on interpolation and extrapolation

Figure 5 illustrates our structure initialization strategy. We initialize the residue coordinates with linear interpolations and extrapolations based on the nearest residues with known structures in the protein. Denote the sequence of residues as $\mathcal{A} = \mathbf{a}_1 \cdots \mathbf{a}_{N_s}$, where N_s is the length of the sequence. Let $\mathbf{Z}(\mathbf{a}_i) \in \mathbb{R}^{3 \times 4}$ denote the backbone coordinates of the i -th residue. If we want to initialize the coordinates of the i -th residue, we take the following initialization strategy: (1) We use linear interpolation if there are residues with known coordinates at both sides of the i -th residue. Specifically, assume p and q are the indexes of the nearest residues with known coordinates at each side of the i -th residue ($p < i < q$), we have: $\mathbf{Z}(\mathbf{a}_i) = \frac{1}{q-p} [(i-p)\mathbf{Z}(\mathbf{a}_q) + (q-i)\mathbf{Z}(\mathbf{a}_p)]$. (2) We conduct linear extrapolation if the i -th residue is at the ends of the chain, i.e., no residues with known structures at one side of the i -th residue. Specifically, let p and q denote the index of the nearest and the second nearest residue with known coordinates. The position of the i -th residue can be initialized as $\mathbf{Z}(\mathbf{a}_i) = \mathbf{Z}(\mathbf{a}_p) + \frac{i-p}{p-q} (\mathbf{Z}(\mathbf{a}_p) - \mathbf{Z}(\mathbf{a}_q))$. We initialize the side-chain atoms' coordinates with the coordinate of their corresponding C_α . In practice, we can further apply local geometric constraints to the initialized backbone atoms in each residue.

A.5 Equivariance.

FAIR has the desirable property of E(3)-equivariance as follow:

Theorem 1. Denote the E(3)-transformation as T_g and the generative process of FAIR as $\{(\mathbf{b}_i, \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m = p_\theta(\mathcal{A} \setminus \mathcal{B}, \mathcal{M})$, where $\{(\mathbf{b}_i, \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m$ indicates the designed pocket sequence and structure. We have $\{(\mathbf{b}_i, T_g \cdot \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m = p_\theta(T_g \cdot (\mathcal{A} \setminus \mathcal{B}), T_g \cdot \mathcal{M})$.

The E(3)-transformation on the euclidean coordinate $\mathbf{x} \in \mathbb{R}^3$ can be represented as: $T_g \cdot \mathbf{x} = O\mathbf{x} + t$, where $O \in \mathbb{R}^3$ is the orthogonal transformation matrix, $t \in \mathbb{R}^3$ is the translation vector. Before we give the proof, we present the following lemmas.

Lemma 1. The hierarchical graph transformer encoder is E(3)-invariant.

Proof. As shown in Sec. 3.2, the input to the hierarchical graph transformer encoders are scalar node features and pairwise distances, and the output is atom and residue scalar features. Therefore, the hierarchical encoder is E(3)-invariant by construction. \square

Lemma 2. *The backbone and full-atom structure refinement are E(3)-equivariant.*

Proof. Firstly, with the E(3)-invariant encoder, the obtained residue embeddings $\{f_i\}$, atom embeddings $\{h_i\}$, and the output of functions g and g' are E(3)-invariant. Furthermore, $\forall T_g \in E(3)$, it is easy to have $T_g \cdot (\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j})) = O\mathbf{x}(\mathbf{b}_{i,j}) + t - O\mathbf{x}(\mathbf{a}_{k,j}) - t = O(\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}(\mathbf{a}_{k,j}))$. Similarly, we have $T_g \cdot (\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s) = O(\mathbf{x}(\mathbf{b}_{i,j}) - \mathbf{x}_s)$ and $T_g \cdot (\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k)) = O(\mathbf{x}(\mathbf{b}_{i,j}) - \bar{\mathbf{x}}(\mathbf{a}_k))$. Let $\mathbf{x}(\mathbf{b}_{i,j})'$ denote the updated pocket coordinates in Equation. 7 and 17. It is then possible to derive that the updated pocket coordinates are E(3)-equivariant as follows:

$$T_g \cdot \mathbf{x}(\mathbf{b}_{i,j}) + T_g \cdot \frac{1}{K_r} \sum_{k \in \mathcal{N}(i)} \mathbf{g}_{ij,kj} + T_g \cdot \frac{1}{N_l} \sum_s \mathbf{g}_{ij,s} \quad (19)$$

$$= O\mathbf{x}(\mathbf{b}_{i,j}) + t + O\frac{1}{K_r} \sum_{k \in \mathcal{N}(i)} \mathbf{g}_{ij,kj} + O\frac{1}{N_l} \sum_s \mathbf{g}_{ij,s} \quad (20)$$

$$= O[\mathbf{x}(\mathbf{b}_{i,j}) + \frac{1}{K_r} \sum_{k \in \mathcal{N}(i)} \mathbf{g}_{ij,kj} + \frac{1}{N_l} \sum_s \mathbf{g}_{ij,s}] + t \quad (21)$$

$$= T_g \cdot \mathbf{x}(\mathbf{b}_{i,j})'. \quad (22)$$

Similarly, for the updated ligand coordinates, we have:

$$T_g \cdot \mathbf{x}_s - T_g \cdot \frac{1}{|\mathcal{B}|} \sum_{i,j} \mathbf{g}_{ij,s} \quad (23)$$

$$= O\mathbf{x}_s + t - O\frac{1}{|\mathcal{B}|} \sum_{i,j} \mathbf{g}_{ij,s} \quad (24)$$

$$= O[\mathbf{x}_s - \frac{1}{|\mathcal{B}|} \sum_{i,j} \mathbf{g}_{ij,s}] + t \quad (25)$$

$$= T_g \cdot \mathbf{x}'_s. \quad (26)$$

Here, we use \mathbf{x}'_s to denote the updated ligand coordinates in Equation. 8 and 18. Therefore, the structure refinement modules in backbone and full-atom refinement are E(3)-equivariant. \square

Proof. The predicted residue types are obtained by applying softmax and MLPs on the hidden representations from the E(3)-invariant encoder. With the above lemmas, Theorem 1 can be proved since the E(3)-invariant encoder and the E(3)-equivariant structure refinement constitute the E(3)-equivariant generative process of FAIR. Formally, we have:

$$\{(\mathbf{b}_i, T_g \cdot \mathbf{x}(\mathbf{b}_i))\}_{i=1}^m = p_\theta(T_g \cdot (\mathcal{A} \setminus \mathcal{B}), T_g \cdot \mathcal{M}), \quad (27)$$

which concludes Theorem 1. \square

A.6 Huber Loss

In Equation 10, we use Huber loss [24] as the structure refinement loss for the stability of optimization, which is defined as follows:

$$l_{huber}(x, y) = \begin{cases} 0.5(x - y)^2, & \text{if } |x - y| < \epsilon, \\ \epsilon \cdot (|x - y| - 0.5 \cdot \epsilon), & \text{else,} \end{cases} \quad (28)$$

where x and y represent the predicted and ground-truth coordinates. The Huber loss has the following property: if the L1 norm of $|x - y|$ is smaller than ϵ , it is MSE loss, otherwise it is L1 loss. At the beginning of the model training, the deviation between the predicted and ground-truth coordinates is large, and the L1 term makes the loss less sensitive to outliers than MSE loss. The deviation is small when the training is almost complete, and the MSE loss is near 0. In practice, we find that directly using MSE loss sometimes leads to NaN at the beginning of the training, while Huber loss makes the training procedure more stable. Following the suggestion of previous works [28, 33], we set $\epsilon = 1$ in all our experiments.

B Implementation Details

PocketOptimizer [45]³ is a physic-based computational protein design method that predicts mutations in the binding pockets of proteins to increase affinity for a specific ligand. Its previous version is [42]. Generally, there are four main steps in PocketOptimizer: structure preparation, flexibility sampling, energy calculations, and computation of design solutions. Specifically for the energy calculations, both packing-related energies and binding-related energies are considered. As for the output design solutions, we select the top 100 designs identified by PocketOptimizer based on integer linear programming for downstream metric calculations. Following the suggestions in the original paper, we use AMBER ff14S force field [41] for energy computation and the Dunbrack rotamer library [55] for rotamer sampling for PocketOptimizer in our implementation.

DEPACT [8]⁴ is a template-matching method that follows the two-step strategy of RosettaMatch [67] for pocket design and has better performance. It first searches the protein-ligand complexes in the database with similar ligand fragments and constructs a cluster model (a set of pocket residues). It then grafts the cluster model into the protein pocket with PACMatch, which places residues in pocket clusters on protein scaffolds by matching key atoms in the cluster model with key atoms of the protein scaffold. The qualities of the generated pockets are evaluated and ranked based on a statistical scoring function. We take the top 100 designed pockets for evaluation. The template databases are constructed in our experiments based on the corresponding training datasets for fair comparisons.

HSRN [28]⁵ was originally developed for antibody design. It autoregressively predicts the residue sequence and docks the generated structure. To adapt HSRN to our pocket design task, we replace the antigen with the target ligand molecule to provide the context information. We finetune HSRN's hyperparameters based on its performance on the validation set. The hidden size in the message-passing network is 256, the number of layers is 4, and the number of RBF kernels is 16.

Diffusion [39]⁶ jointly models both the sequence and structure of an antibody using diffusion probabilistic estimates and an equivariant neural network. The diffusion model first aggregates information from the antigen and the antibody during generation. It then iteratively updates each amino acid's type, position, and orientation. Since the side-chain atoms are not modeled, they are reconstructed at the last step with side-chain packing. To adapt Diffusion to our pocket design task, we replace the antigen with the target ligand molecule to provide the context information. We adjust the key hyperparameters based on the results of the validation sets. The number of layers, hidden dimension, and diffusion steps are 6, 128, and 100, respectively.

MEAN [33]⁷ tackles antibody design as a conditional graph translation problem with the modified GMN [23] as encoders. Specifically, MEAN co-designs antibody sequence and structure via a multi-round progressive full-shot scheme, which is more efficient than auto-regressive or diffusion-based approaches. However, it only models protein backbone atoms and neglects side chains, which are important for protein-protein and protein-ligand interactions. To adapt MEAN to our pocket design task, we replace the antigen with the target ligand molecule to provide the context information. The hyperparameters are finetuned based on the validation set for better performance. We set the hidden size as 128, the number of layers as 3, and the number of iterations for decoding as 3.

FAIR is our full atom pocket sequence-structure co-design method. The number of layers for the atom and residue-level encoder are 6 and 2, respectively. K_a and K_r are set as 24 and 8 respectively. The number of attention heads is set as 4; The hidden dimension d is set as 128. The standard deviation of the Gaussian noise added to the ligand coordinates in model training is 0.1.

³<https://github.com/Hoecker-Lab/pocketoptimizer>

⁴https://github.com/chenyaoxi/DEPACT_PACMatch

⁵<https://github.com/wengong-jin/abdockgen>

⁶<https://github.com/luost26/diffab>

⁷<https://github.com/THUNLP-MT/MEAN>

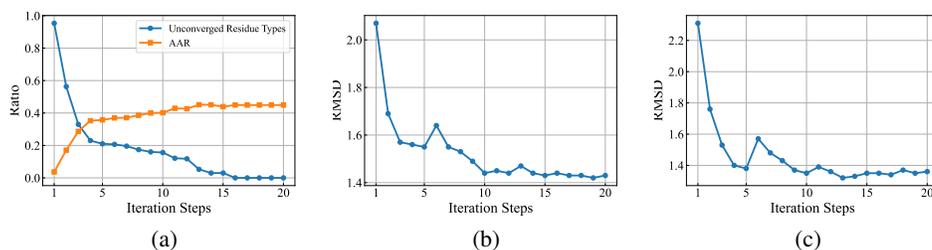


Figure 6: (a) The ratio of unconverged residue types and AAR during the iterations of FAIR on Binding MOAD; (b) The convergence plot of RMSD on CrossDocked; (c) The convergence plot of RMSD on Binding MOAD.

C More Experimental Results

C.1 Pocket Generation without Backbone Reference

In FAIR, the original protein backbone coordinates are leveraged for residue-level KNN graph construction for training and generation stability. However, FAIR can also work well without protein backbone coordinates as a reference. In our supplementary experiments, FAIR can still achieve comparable performance.

C.2 Convergence of Iterative Refinement

We show the RMSD curves in Table 6 (b) & (c). We observe the RMSD converges rapidly as the protein pocket is refined. The curve has fluctuations near step 5, potentially caused by the initialization of the side-chain atom at the beginning of full-atom refinement. The coordinates of side-chain atoms are randomly initialized near the C_{α} and may distort the backbone atoms with the structural refinement. However, the RMSD of FAIR becomes steady quickly after step 10, which again shows the effectiveness of FAIR.

C.3 Local Geometry of the Generated Pockets

Since the RMSD used in Tables 1 and 2 reflects only the correctness of global geometry in generated protein pockets, we also calculate the RMSD of bond lengths and angles in the designed pocket to validate the local geometry following previous work [33]. We measure the average RMSD of the covalent bond lengths in the pocket backbone (C-N, C=O, and C-C). The three conventional dihedral angles in the backbone structure, i.e., ϕ, ψ, ω [57] are considered, and the average RMSD of their cosine values are calculated. The results in Table 3 demonstrate that FAIR still achieves much better performance regarding the local geometry than the baseline methods.

Table 3: The average RMSD of bond lengths and dihedral angles of the generated backbone structures on the two datasets.

Models	Bond lengths	Angles (ϕ, ψ, ω)
PocketOptimizer	0.47	0.328
DEPACT	0.68	0.360
HSRN	0.76	0.437
Diffusion	0.49	0.344
MEAN	0.44	0.312
FAIR	0.35	0.287

C.4 Hyperparameter Analysis

Here, we evaluate the influence of hyperparameter choices on FAIR performance. We focus on two hyperparameters, T_1 and T_2 , for iterative refinements. In Figure 7, we observe that the performance

of FAIR is stable when T_1 and T_2 are larger than 5 and 10, respectively, indicating the refinement processes gradually converge. We set T_1 and T_2 to 5 and 10 throughout our experiments.

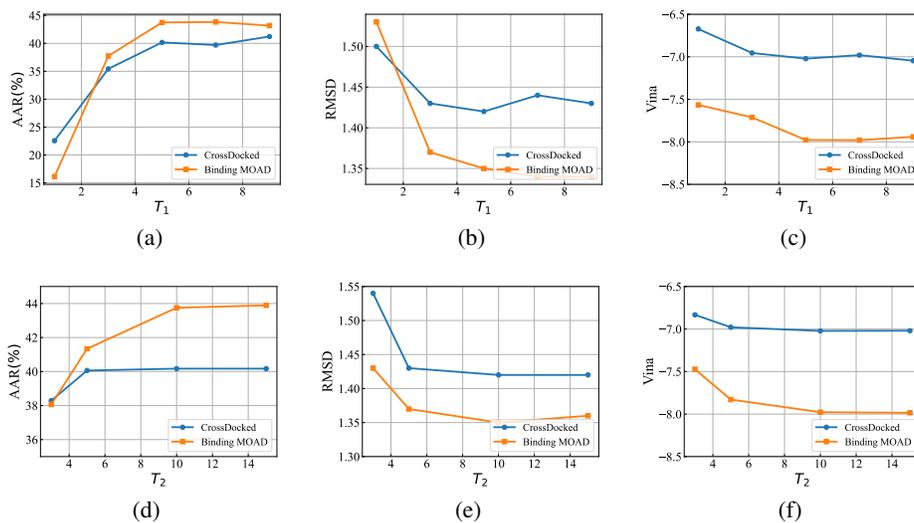


Figure 7: Hyperparameter analysis with respect to T_1 (first row) and T_2 (second row).