PocketGen: Generating Full-Atom Ligand-Binding Protein Pockets

Zaixi Zhang[1,2,3], **Wanxiang Shen**[3], **Qi Liu**[1,2,✉], **and Marinka Zitnik**[3,4,5,6,✉]

[1]Anhui Province Key Lab of Big Data Analysis and Application, University of Science and Technology of China, Hefei, Anhui, China
[2]State Key Laboratory of Cognitive Intelligence, Hefei, Anhui, China
[3]Department of Biomedical Informatics, Harvard Medical School, Boston, MA, USA
[4]Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University, MA, USA
[5]Broad Institute of MIT and Harvard, Cambridge, MA, USA
[6]Harvard Data Science Initiative, Cambridge, MA, USA

✉qiliuql@ustc.edu.cn and marinka@hms.harvard.edu

ABSTRACT

Designing small-molecule-binding proteins, such as enzymes and biosensors, is essential in protein biology and bioengineering. Generating high-fidelity protein pockets—areas where proteins interact with ligand molecules—is challenging due to the complex interactions between ligand molecules and proteins, the flexibility of ligand molecules and amino acid side chains, and intricate sequence-structure dependencies. We introduce PocketGen, a deep generative method that produces the residue sequence and the full-atom structure within the protein pocket region, leveraging sequence-structure consistency. PocketGen comprises a bilevel graph transformer for structural encoding and a sequence refinement module utilizing a protein language model (pLM) for sequence prediction. The bilevel graph transformer captures interactions at multiple granularities (atom-level and residue/ligand-level) and aspects (intra-protein and protein-ligand) through bilevel attention mechanisms. A structural adapter employing cross-attention is integrated into the pLM for sequence refinement to ensure consistency between structure-based and sequence-based prediction. During training, only the adapter is fine-tuned, while the other layers of the pLM remain unchanged. Experiments demonstrate that PocketGen can efficiently generate protein pockets with higher binding affinity and validity than state-of-the-art methods. PocketGen is ten times faster than physics-based methods and achieves a 95% success rate (percentage of generated pockets with higher binding affinity than reference pockets) with an amino acid recovery rate exceeding 64%.

Introduction

A primary method for modulating protein functions involves the interaction between proteins and small molecule ligands[1–4]. These interactions are essential in biological processes, such as enzymatic catalysis, signal transduction, and regulatory mechanisms within cells. Binding small molecules to specific sites on proteins can induce conformational changes, modulate activity, or inhibit functions. This mechanism is a valuable tool for studying protein functions and designing small molecule-binding proteins with customized properties for therapeutic and industrial applications. These applications include designing enzymes to catalyze reactions without natural catalysts[5–8] and developing biosensors for detecting compounds in the environment. These biosensors transduce signals useful for environmental monitoring, clinical diagnostics, pathogen detection, drug delivery systems, and food industry applications[9–12]. The designs often involve modifying existing ligand-binding protein pockets to facilitate more precise interactions with specific ligands[13–15]. However, the complexity of ligand molecule-protein interactions, ligand and side chain flexibility, and sequence-structure relationships pose challenges for the computational generation of high-validity, ligand-binding protein pockets[3,15,16].

Traditional methods for pocket design focus on physics modeling or template-matching[10,11,13,17,18]. For example, PocketOptimizer[18–20] develops a pipeline that predicts mutations in protein pockets to increase binding affinity based on physics-inspired energy functions and search algorithms. Initiated with a bound protein-ligand complex, PocketOptimizer explores side chain structures and residue types. These mutants are evaluated using energy functions and ordered through integer linear programming techniques. Another line of research uses template matching and enumeration methods[11,13,14,17,21]. For instance, Polizzi et al.,[13] employ a two-step strategy for pocket design. They first identify and assemble disconnected protein motifs (van der Mer (vdM) structural units) surrounding the target molecule to build protein-ligand interactions (e.g., Hydrogen bonds). Subsequently, they graft these residues onto the protein scaffold and select the best combinations of protein-ligand pairs using scoring functions. This template-matching strategy enabled the de novo design of proteins binding the drug apixaban[22]. However, methods based on physics modeling and template-matching can be time-consuming, often requiring several hours for

a single protein pocket design. In practical protein engineering, evaluating thousands to millions of designed protein candidates to identify a successful candidate significantly increases the time required for protein pocket design. The focus on specific fold types (e.g., four-helix bundle[13] or NTF2[14]) by these methods further restricts their widespread application.

Recent advancements in protein pocket design have been facilitated by deep learning-based approaches[3,8,16,23–25]. For instance, RFDiffusion[26] employs denoising diffusion probabilistic models[27] in conjunction with RoseTTAFold[28] for *de novo* protein structure generation. Despite its capability to design pockets targeting binding with specific ligands, the approach's auxiliary guiding potentials lack precision in modeling protein-ligand interactions. RFdiffusion All-Atom (RFdiffusionAA)[16] represents a significant step forward, enabling the direct generation of binding proteins around small molecules through iterative denoising. This advance is attributed to modifications in model architecture to consider protein structures and ligand molecules concurrently. Nevertheless, the derivation of residue sequences in RFDiffusion and RFdiffusionAA involves post-processing with ProteinMPNN[29] or LigandMPNN[30], potentially leading to inconsistencies in modeling sequence and structure protein modalities. Conversely, our prior work FAIR[24] simultaneously designs the complete atom pocket structure and sequence using a two-stage refinement approach, employing a coarse-to-fine approach that initially updates the backbone protein structure before addressing the complete atom structure, including side chains. FAIR iteratively refines atom coordinates and residue sequence predictions until convergence is achieved. However, a gap between the two refinement steps occasionally results in instability and limited generation performance, indicating the need for an end-to-end generative approach for pocket design. Related research focused on the sequence and structure co-design of complementarity determining regions (CDRs) in antibodies[31–35]. These methods, devised explicitly for antibodies, face challenges when adapting to pocket designs conditioned on target ligand molecules. Hybrid approaches that combine deep learning models with traditional methods are also being explored[3,8]. For example, Yeh et al.[8] developed a novel Luciferase by employing a combination of protein hallucination[36], the trRosetta structure prediction neural network[37], hydrogen bonding networks, and RifDock[38], generating a multitude of idealized protein structures with diverse pocket shapes for subsequent filtering. Despite its success, this method's applicability is confined to specific protein scaffolds and substrates, lacking a generalized solution. Similarly, Lee et al.[3] merge deep learning with physics-based methods to create proteins featuring diverse and designable pocket geometries. This integration involves backbone generation via trRosetta hallucination, sequence design through ProteinMPNN[29] and LigandMPNN[30], and filtering with AlphaFold[39]. Current pocket generation models face challenges, including achieving sequence-structure consistency, accurately modeling complex protein-ligand interactions, and realizing generalized end-to-end pocket generation.

Here, we introduce PocketGen, a deep generative method for full-atom protein pocket generation that addresses existing limitations. PocketGen implements a co-design scheme (Figure 1(a)), where the model concurrently predicts the sequence and structure of the protein pocket based on the ligand molecule and the protein scaffold (excluding the pocket). PocketGen comprises two modules: the bilevel graph transformer (Figure 1(b)) and the sequence refinement module (Figure 1(c)). To achieve end-to-end molecular generation, PocketGen models the protein-ligand complex as a geometric graph of blocks to handle variable atom counts across different residues and ligands. Initially, the pocket residues are assigned the maximum possible number of atoms (14 atoms) and later are mapped back to specific residue types post-generation. The graph transformer module uses bilevel attention mechanisms to capture interactions at multiple granularities (atom-level and residue/ligand-level) and aspects (intra-protein and protein-ligand). To account for the influence of the redesigned pocket on the ligand, the ligand structure is updated during refinement to reflect potential changes in binding pose. To ensure consistency across protein sequence and structure domains and incorporate evolutionary information encoded in protein language models (the ESM series models[40,41]), PocketGen implements a structural adapter into protein sequence updates. This adapter facilitates cross-attention between sequence and structure features, promoting information flow and achieving sequence-structure consistency. Only the adapter is fine-tuned during training, while the remaining layers of the protein language models remain unchanged. Experimental results demonstrate that PocketGen significantly outperforms current state-of-the-art methods in protein pocket generation across two popular benchmarks. PocketGen achieves an average amino acid recovery rate of 63.40% and a Vina score of -9.655 for top-1 generated protein pockets on the CrossDocked dataset. Comprehensive analyses indicate that PocketGen can produce diverse and high-affinity protein pockets for functional molecules, showcasing its efficacy and potential in designing small-molecule binders and enzymes.

Results

Benchmarking generated protein pockets across multiple axes of molecular evaluation

PocketGen is benchmarked on two widely used datasets, following previous studies[24,42]. The **CrossDocked** dataset[43] comprises protein-molecule pairs generated through cross-docking and is divided into train, validation, and test sets based on a sequence identity threshold of 30%. The **Binding MOAD** dataset[44] includes experimentally determined protein-ligand complexes that are split into train, validation, and test sets according to the proteins' enzyme commission numbers[45]. Given the range of distances relevant to protein-ligand interactions[46], our default experimental setting considers all residues with atoms within 3.5 Å of any binding ligand atoms as part of the protein pocket, averaging about eight residues per pocket. We also investigate the capability

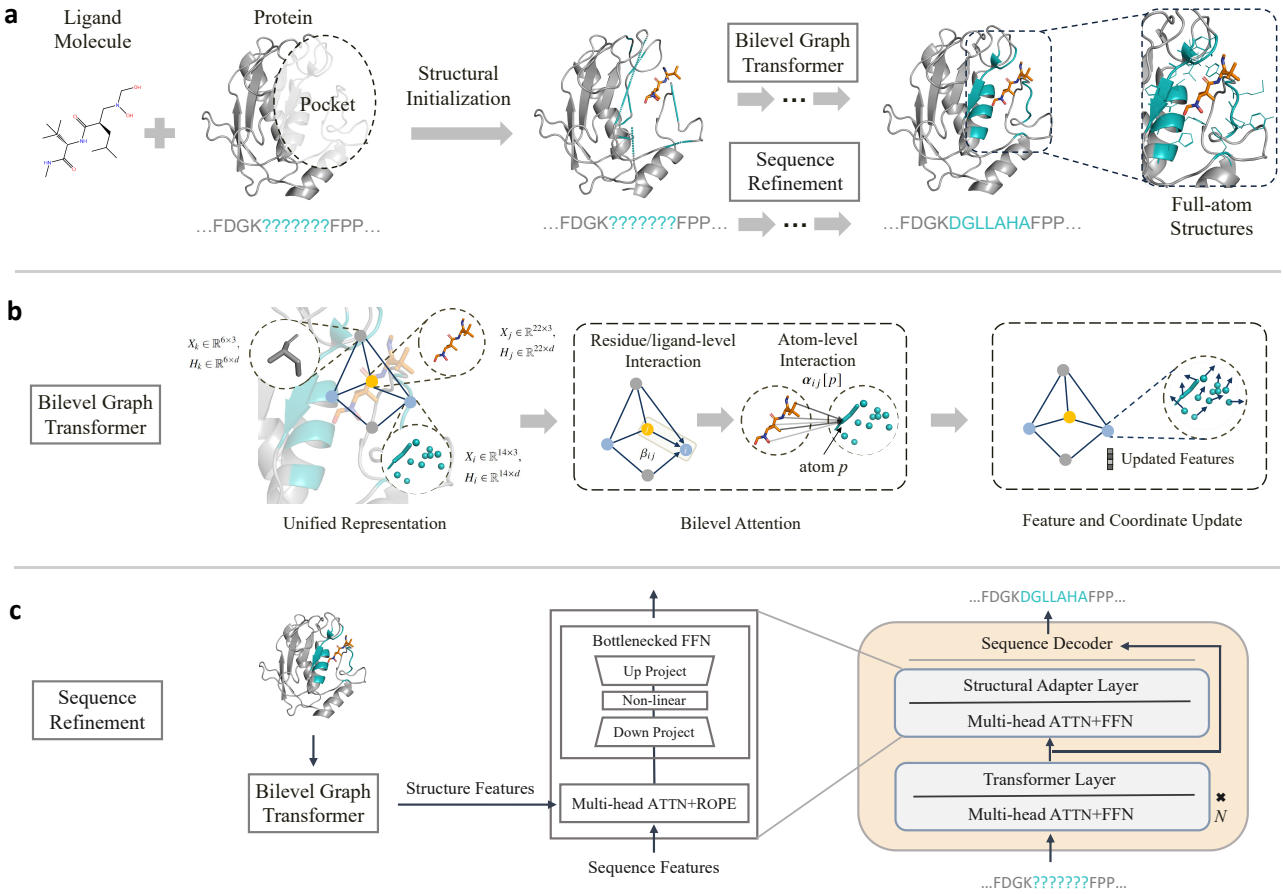


Figure 1. Overview of PocketGen generative model for the design of full-atom ligand-binding protein pockets. a, Conditioned on the binding ligand molecule and the rest part of the protein except the pocket region (i.e., scaffold), PocketGen aims to generate the full atom pocket structure (backbone and sidechain atoms) and the residue type sequence with iterative equivariant refinement. The ligand structure is also adjusted during the protein pocket refinement. **b**, Bilevel graph transformer is leveraged in PocketGen for all-atom structural encoding and update. The bilevel level attention captures both the residue/ligand and atom-level interactions. Both the protein pocket structure and the ligand molecule structure are updated in the refinement. **c**, Sequence refinement module adds lightweight structural adapter layers into pLMs for sequence prediction. Only the adapter's parameters are fine-tuned during training, and the other layers are fixed. In the adapter, the cross-attention between sequence and structure features is performed to achieve sequence-structure consistency.

of PocketGen to design larger pocket areas (e.g., 5.5 Å) and include more residues (see Figure 2 (c)). Methodological details on data processing are provided in the Methods section.

We use three groups of metrics to evaluate the quality of protein pockets generated by PocketGen. Firstly, AutoDock **Vina score**[47] measures the affinity between the generated pocket and the target ligand molecule. Secondly, we assess the structural validity of generated pockets using **scRMSD**, **scTM**, and **pLDDT**. We derive the amino acid sequence for the protein pocket structure using ProteinMPNN[29] and predict the pocket structure using ESMFold[48]. scRMSD is obtained by aligning and computing the *self-consistency* Root Mean Squared Deviation between the generated and the predicted structure's backbone atoms. Following established methodologies[49,50], eight sequences are predicted for each generated protein structure, and the sequence with the minimum scRMSD is kept for reporting the metric values. Similarly, scTM, the *self-consistency* Template Modeling score, is obtained by calculating the TM-scores[51] between the ESMFold-predicted structure and the generated structure, with scores ranging from 0 to 1, where higher values indicate greater designability. The predicted Local Distance Difference Test (pLDDT) score[39] reflects the confidence in structural predictions on a scale from 0 to 100, with higher scores indicating greater confidence. In our paper, we report the average pLDDT score from ESMFold[48] across pocket residues. Following previous works[26,52,53], a generated protein pocket structure is defined as **designable** if the overall structure scRMSD < 2Å and the pocket scRMSD < 1Å. Calculating the designability metric, Table 2 shows the percent of designable generated pockets. Supplementary Figure S1 describes the process of calculating the metrics. Finally, we determine the Amino Acid

Table 1. The top 1/3/5/10 generated protein pocket (ranked by Vina score) with designability on the CrossDocked dataset. The success rate measures the percentage of protein that the model can generate pockets with higher affinity than the reference ones in the datasets. We report the average plDDT of the predicted pocket, the scRMSD of the pocket backbone coordinates, and the scTM score of the whole protein. The plDDT, scRMSD, and scTM for PocketOpt are not reported, as PocketOpt keeps protein backbone structures fixed. We use to mark the results of affinity-related metrics, for pocket-structure related metrics, and for whole protein structure metrics.

	PocketOpt	DEPACT	dyMEAN	FAIR	RFDiffusion	RFDiffusionAA	PocketGen
Top-1 generated protein pocket							
Vina score (↓)	-9.216	-8.527	-8.540	-8.792	-9.037	-9.216	**-9.655**
Success Rate (↑)	0.92	0.75	0.76	0.80	0.89	0.93	**0.97**
pLDDT (↑)	-	82.1	83.3	83.2	84.5	86.3	**86.7**
scRMSD (↓)	-	0.705	0.703	0.680	0.676	0.654	**0.645**
scTM (↑)	-	0.901	0.906	0.899	0.924	0.931	**0.937**
Top-3 generated protein pockets							
Vina score (↓)	-8.878	-8.131	-8.196	-8.321	-8.876	-8.980	**-9.353**
pLDDT (↑)	-	81.9	82.8	83.1	84.6	**86.2**	**86.2**
scRMSD (↓)	-	0.706	0.724	0.685	0.679	**0.653**	0.657
scTM (↑)	-	0.896	0.892	0.897	0.929	0.930	**0.934**
Top-5 generated protein pockets							
Vina score (↓)	-8.702	-7.786	-7.974	-7.943	-8.510	-8.689	**-9.239**
pLDDT (↑)	-	82.2	82.9	83.3	84.3	85.7	**86.1**
scRMSD (↓)	-	0.717	0.725	0.690	0.680	0.656	**0.652**
scTM (↑)	-	0.892	0.903	0.886	0.926	0.929	**0.935**
Top-10 generated protein pockets							
Vina score (↓)	-8.556	-7.681	-7.690	-7.785	-8.352	-8.524	**-9.065**
pLDDT (↑)	-	81.5	82.7	83.0	84.2	85.3	**85.9**
scRMSD (↓)	-	0.710	0.734	0.705	0.684	**0.672**	0.678
scTM (↑)	-	0.895	0.896	0.884	0.924	0.929	**0.931**

Recovery (**AAR**), the percentage of correctly predicted pocket residue types, to evaluate the designed sequence's accuracy. A higher AAR signifies a deeper understanding of sequence-structure relationships.

We compare PocketGent to six methods, including deep learning methods RFDiffusion[26], RFDiffusionAA[16], FAIR[24] dyMEAN[25], a template-matching method DEPACT[17], and a physics modeling method PocketOpt[18] (refer to Methods for detailed descriptions of baselines). In Table 2, PocketGen and other methods are tasked to generate 100 sequences and structures for each protein-ligand complex in the CrossDocked and Binding MOAD dataset test set. We present the mean and standard deviation values over three different runs. PocketOpt is not compared in Table 2 because it focuses on mutating existing pockets for optimization and is time-consuming for generating many protein pockets. Table 2 shows that PocketGen surpasses all the baselines, including state-of-the-art methods RFDiffusion and RFDiffusionAA in designability (by 3% and 2% on CrossDocked, respectively) and Vina scores (by 0.199 and 0.123 on CrossDocked, respectively), reflecting its effectiveness in generating structurally valid pockets with high binding affinities. This superior performance can be attributed to PocketGen's capability to capture interactions at multiple granularities (atom-level and residue/ligand-level) and aspects (intra-protein and protein-ligand).

Additionally, PocketGen significantly improves the most competitive method RFDiffusionAA by 13.95% on average in AAR, benefiting from incorporating a protein language model that encompasses evolutionary sequence information. In contrast, RFDiffusion and RFDiffusionAA require post-processing to derive amino acid types, which may incur sequence-structure inconsistency and inferior performance in AAR. In protein engineering, the practice often involves mutating several key residues to optimize properties while keeping most residues unchanged to preserve protein folding stability[54,55]. A high AAR achieved by generated protein pockets aligns with this practice. In Table 1, the top-1, 3, 5, and 10 protein pockets (ranked by Vina score) generated by PocketGen demonstrate the lowest Vina scores, achieving 0.476 lower Vina scores on average than RFDiffusionAA, and competitive pLDDT, scRMSD, and scTM scores, showcasing PocketGen's ability to produce high-affinity pockets that maintain structural validity and sequence-structure consistency. With a 97% success rate in creating pockets with higher affinity than the reference cases (the success rate of the strongest baseline method is 93% of RFDiffusionAA) on CrossDocked, PocketGen's effectiveness and broad applicability across various ligand molecules are demonstrated.

Table 2. Benchmarking PocketGen and other approaches for pocket generation on two datasets. Reported are average and standard deviation values across three independent runs. The best results are bolded.

Model	CrossDocked			Binding MOAD		
	AAR (↑)	Designability (↑)	Vina (↓)	AAR (↑)	Designability (↑)	Vina (↓)
Test set	-	0.77	-7.016	-	0.79	-8.076
DEPACT	31.52±3.26%	0.68±0.04	-6.632±0.18	35.30±2.19%	0.67±0.06	-7.571±0.15
dyMEAN	38.71±2.16%	0.71±0.03	-6.855±0.06	41.22±1.40%	0.70±0.03	-7.675±0.09
FAIR	40.16±1.17%	0.73±0.02	-7.015±0.12	43.68±0.92%	0.72±0.05	-7.930±0.15
RFDiffusion	46.57±2.07%	0.74±0.01	-6.936±0.07	45.31±2.73%	0.75±0.05	-7.942±0.14
RFDiffusionAA	50.85±1.85%	0.75±0.03	-7.012±0.09	49.09±2.49%	0.78±0.03	-8.020±0.11
PocketGen	**63.40±1.64%**	**0.77±0.02**	**-7.135±0.08**	**64.43±2.35%**	**0.80±0.04**	**-8.112±0.14**

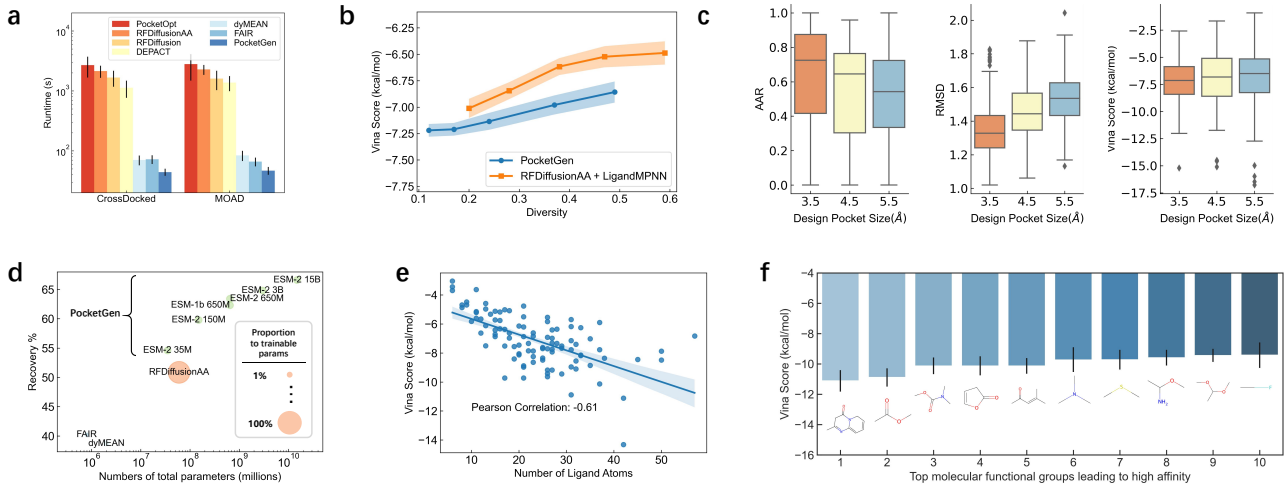


Figure 2. Exploring the capabilities of PocketGen. a, The average runtime of different methods for generating 100 protein pockets for a ligand molecule on the two benchmarks. **b**, The trade-off between quality (measured by Vina score) and diversity (1- average pairwise sequence similarity) of PocketGen. We can balance the trade-off by tuning the temperate hyperparameter τ. **c**, The influence of the design pocket size on the metrics. **d**, Performance w.r.t. model scales of pLMs using ESM series on CrossDocked dataset. The green dots represent PocketGen models with different ESMs. The bubble size is proportional to the number of trainable parameters. **e**, PocketGen tends to generate pockets with higher affinity for larger ligand molecules (Pearson Correlation -0.61, shadow indicates 95% confidence interval). **f**, The top molecular functional groups leading to high affinity.

To investigate the substructure validity and consistency with reference datasets, we conduct a qualitative substructure analysis (Table S2 and Figure S2). This analysis considers three covalent bonds in the residue backbone (C-N, C=O, and C-C), three dihedral angles in the backbone (ϕ, ψ, ω[56]), and four dihedral angles in the side chains ($\chi_1, \chi_2, \chi_3, \chi_4$[57]). Following previous research[58,59], we collect the bond length and angle distributions in the generated pockets and the test dataset and compute the Kullback-Leibler (KL) divergence to quantify the distribution distances. Lower KL divergence scores for PocketGen indicate its effectiveness in replicating the geometric features present in the data.

Probing generative capabilities of PocketGen

Next, we explore PocketGen's generative capabilities. In addition to designing high-quality protein pockets, generative models need to be efficient and maximize the yield of biochemical experiments, meaning that they can rapidly produce high-fidelity pocket candidates and that only a small number of designs are necessary to find a hit. Figure 2(a) offers a comparative analysis of average **runtime** across different methods. Traditional approaches, such as physics-based modeling (PocketOpt) and template-matching (DEPACT), require over 1,000 seconds to generate 100 pockets. Similarly, the runtime for the advanced protein backbone generation models RFDiffusion and RFDiffusionAA are significant due to their diffusion architecture, requiring 1633.5 and 2210.1 seconds on average to design 100 pockets. In contrast, methods that iteratively refine protein designs, including PocketGen, show a notable decrease in generation time; it takes 44.2 seconds for PocketGen to generate 100 pockets.

While recent methods for pocket generation prioritize maximizing binding affinity with target molecules, this strategy may not always align with practical needs, where pocket diversity also plays a critical role. Examining a batch of designed

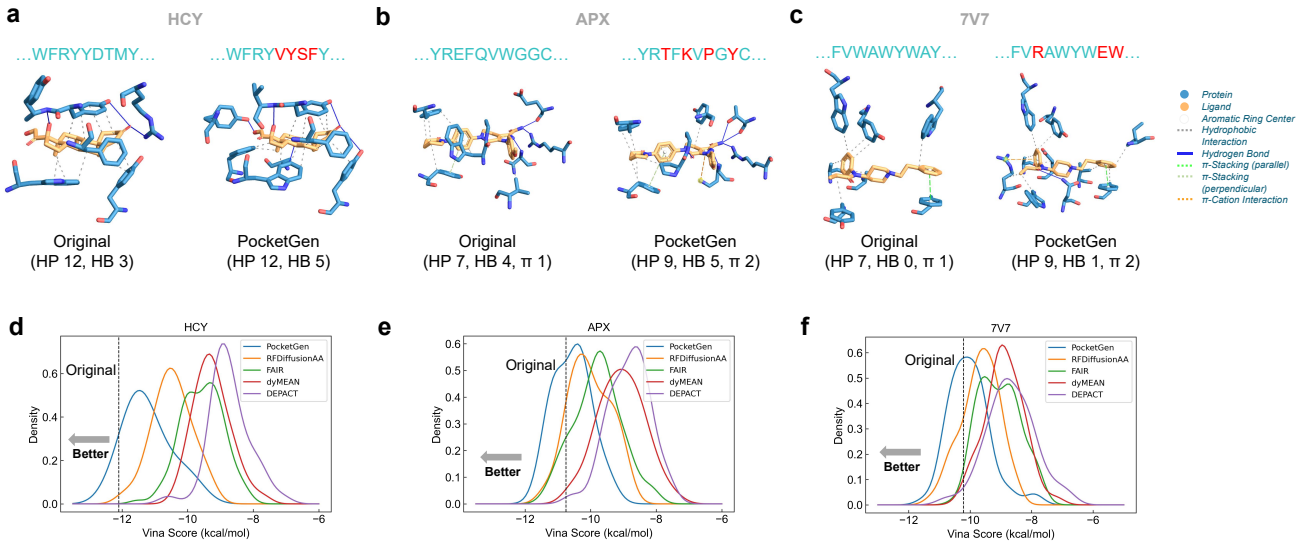


Figure 3. Using PocketGen to design protein pockets for binding with important ligands. **a**, **b**, **c** Illustrations of protein-ligand interaction analysis for three target molecules (HCY, APX, and 7V7, respectively). 'PocketGen' refers to the protein pocket designed by PocketGen, and 'Original' denotes the original protein-ligand structure. 'HP' indicates hydrophobic interactions, 'HB' signifies hydrogen bonds, and 'π' denotes the π-stacking/cation interactions. In the residue sequences, red ones denote the designed residues that differ from the original pocket. **d**, **e**, **f** The pocket binding affinity distributions of PocketGen and baseline methods for three target molecules (HCY, APX, and 7V7, respectively). We mark the Vina Score of the original pocket with the vertical dotted lines. For each method, we sample 100 pockets for each target ligand. The ratio of generated pockets by PocketGen with higher affinity than the corresponding reference pocket are 11%, 40%, and 45%, respectively.

pockets rather than a single design is beneficial to improve the success rate of pocket design. Thus, we examine the relationship between binding affinity and diversity of generated protein pockets in Figure 2(b). **Diversity** is quantified as (1 – average pairwise pocket residue sequence similarity) and can be adjusted by altering the sampling temperature τ (with higher τ resulting in greater diversity). Figure 2(b) compares PocketGen with the most competitive baseline RFDiffusionAA[16] + LigandMPNN[30], the latest version of ProteinMPNN[29]. Additional results for the RFDiffusion method are in Figure S4. We find that binding affinity and diversity present a trade-off. PocketGen can produce protein pockets with higher affinity than RFDiffusionAA at equivalent levels of diversity.

Figure 2(c) examines the impact of changes in the redesigned pocket size on PocketGen's performance. The redesign process targets all residues with atoms within 3.5 Å, 4.5 Å, and 5.5 Å of any binding ligand atoms. The average AAR, RMSD, and Vina scores show a slight decline for larger protein pockets. This trend is attributed to the increased complexity and reduced contextual information associated with expanding the redesigned protein pocket area. Furthermore, an observation of note is that larger pockets tend to enable the exploration of structures with potentially higher affinity, as indicated by the lowest Vina scores, reaching -20 kcal/mol for designs with a 5.5 Å radius. This phenomenon could be due to the enhanced structural complementarity achievable in larger pocket designs.

Incorporating protein language models (pLMs) distinguishes PocketGen from prior pocket generation models. Beyond employing ESM-2 650M[48] in the default configuration, various versions of ESM with parameter counts ranging from 8M to 15B were also evaluated. As illustrated in Figure 2(d), PocketGen's performance improves with the scaling of pLMs. Going from ESM-2 35 M to ESM-2 15B protein language models, the performance of PocketGen increases from 54.58% to 66.61%. Specifically, a logarithmic scaling law is observed, aligning with trends noted in large language models[60]. PocketGen achieves efficient training with large pLMs by fine-tuning adapter layers while keeping the majority of pLM layers fixed. PocketGen also has significantly fewer trainable parameters than RFDiffusionAA[16] (7.9 M versus 82.9 M trainable parameters).

The performance of PocketGen in generating binding pockets is influenced by the characteristics of the ligand molecule. Figure 2(e) illustrates the relationship between the average Vina score of generated pockets and the number of ligand atoms, indicating that PocketGen tends to create pockets with higher affinity for larger ligand molecules. This trend may be due to the increased surface area for interaction, the presence of additional functional groups, and enhanced flexibility in the conformation of larger molecules[61,62]. Additionally, functional groups in ligand molecules that contribute to high binding affinity were identified using IFG[63]. Figure 2(f) displays the top 10 molecular functional groups, which include hydrogen bond donors and

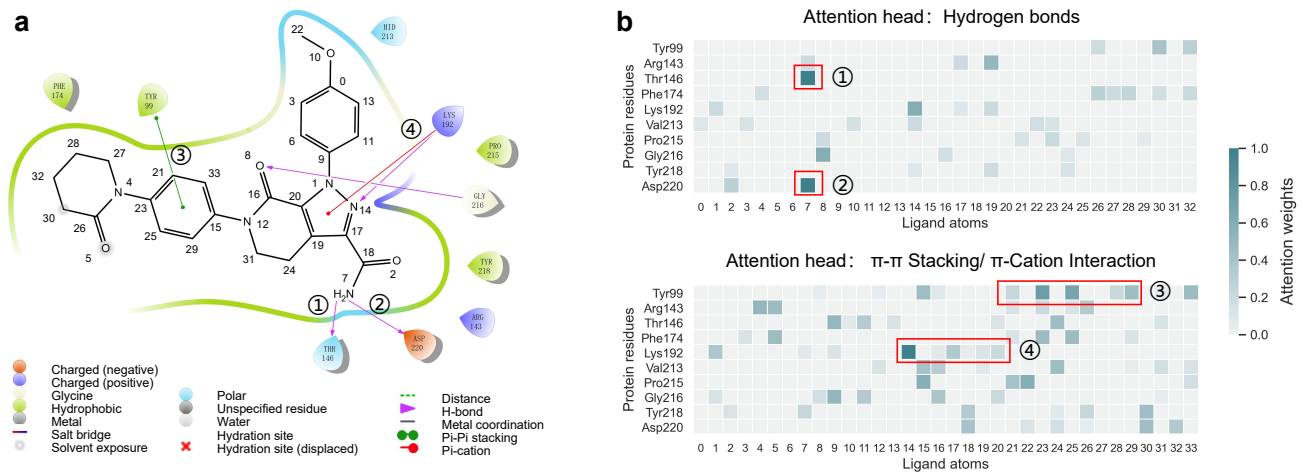


Figure 4. Attention maps in PocketGen capture interactions between atoms in protein and ligand molecules. a, The 2D interaction plot of the designed pocket by PocketGen for APX. **b**, The heatmap of attention matrices between residues and ligand atoms from the last layer of PocketGen. We show two selected attention heads with notable attention patterns marked with red rectangles. We notice that each head emphasizes different interactions. For example, PocketGen recognizes the **hydrogen bond** interaction and assigns a strong attention weight between residue ① THR146, ② ASP220, and ligand atom 7 in the first head. The **π-π stacking** and **π-Cation** interactions of ③ TYR99 and ④ LYS192 are well captured in the second head. The values are normalized by the maximum value ($v_{\max}$) and the minimum value ($v_{\min}$) in each heatmap (i.e., $v' = \frac{v - v_{\min}}{v_{\max} - v_{\min}}$).

acceptors (e.g., carbonyl groups), aromatic rings, sulfhydryl groups, and halogens. These groups can form favorable interactions with protein pockets, thereby increasing binding affinity.

In the supplementary material, ablation studies (Table S3) and hyperparameter analysis (Figure S3) were conducted to evaluate the contribution of each module in PocketGen and the impact of loss function hyperparameters on model performance. The results indicate that the bilevel attention module and the integration of pLM into PocketGen substantially improve performance. PocketGen exhibits robustness to variations in hyperparameters, consistently producing competitive results.

Generating protein pockets for therapeutic small molecule ligands

Next, we illustrate the capacity of PocketGen to redesign the pockets of antibodies, enzymes, and biosensors for target ligand molecules, drawing upon previous research[3,10,16]. Specifically, we consider the following molecules. **Cortisol (HCY)**[64] is a primary stress hormone that elevates glucose levels in the bloodstream and serves as a biomarker for stress and other conditions. We redesign the pocket of a cortisol-specific antibody (PDB ID 8cby) to potentially aid in developing immunoassays. **Apixaban (APX)**[65] is an oral anticoagulant approved by the FDA in 2012 for patients with non-valvular atrial fibrillation to reduce the risk of stroke and blood clots[66]. Apixaban targets Factor Xa (fXa) (PDB ID 2p16), a crucial enzyme in blood coagulation that transforms prothrombin into thrombin for clot formation. Redesigning the pocket of fXa could, therefore, have therapeutic significance. **Fentanyl (7V7)**[67] has become a widely abused drug contributing to the opioid crisis. Computational design of fentanyl-binding proteins (biosensors) can facilitate detection and neutralization of the toxin[10]. For example, Baker et al.[10] developed a biosensor (PDB ID 5tzo) for detecting fentanyl in plants. In Figure 3, PLIP[68] describes the interactions between the designed protein pocket and ligands, comparing these predicted interactions to the original patterns.

To generate pockets for the aforementioned small molecules, we pretrained PocketGen on the Binding MOAD dataset and carefully excluded the protein-ligand complexes used in the case studies. The pockets produced by PocketGen replicate most non-bonded interactions from experimentally measured protein-ligand complexes (e.g., achieving a 13/15 match for HCY) and introduce additional physically plausible interaction patterns not present in the original complexes (generated pockets for HCY, APX, and 7V7 molecules form 2, 3, and 4 extra interactions). Specifically, for HCY, PocketGen maintains essential interaction patterns such as hydrophobic interactions (TRP47, PHE50, TYR59, and TYR104) and hydrogen bonds (TYR59) and introduces two new hydrogen bond-mediated interactions with the pocket. For protein pockets designed to bind APX and 7V7 ligands while retaining key interaction patterns, including hydrophobic interactions, hydrogen bonds, and π-π stacking, PocketGen-generated molecules establish additional interactions (e.g., π-Cation interaction with LYS192 for APX and hydrogen bonds with ASN35 for 7V7), thus enhancing binding affinity with the target ligands. In conclusion, PocketGen can establish non-covalent interactions derived from protein-ligand structure data.

With the ability to establish favorable protein-ligand interactions, PocketGen generates high-affinity pockets for these drug ligands. In Figure 3 (d), (e), and (f), we further show the affinity distributions of the generated pockets by PocketGen and baseline

methods. The ratio of generated pockets with higher affinity than the reference pocket is 11 %, 40%, and 45%, respectively, for PocketGen. On the contrary, the best runner-up method, RFDiffusionAA, only has 0%, 10 %, and 18%, respectively.

Protein stability is a critical factor in protein design, ensuring the designed protein can fold into and maintain its specific three-dimensional structure[69]. Stability is typically quantified by the change in Gibbs free energy ($\Delta\Delta G$) of folding between the redesigned protein and the wild-type (original) protein, where $\Delta\Delta G = \Delta G_{\text{orig}} - \Delta G_{\text{redesign}}$. A positive $\Delta\Delta G$ value indicates increased protein stability, while a negative value suggests decreased. DDMut[70] is used to predict the change in stability for the three cases in Figure 3, with $\Delta\Delta G$ values of 0.09, 0.92, and 0.13, respectively. These results indicate that PocketGen can generate protein structures with enhanced stability, which is essential for functional activities such as binding with target ligand molecules.

Exploring interactions between atoms in protein and ligand molecules learned by PocketGen

Lastly, we analyze the attention matrices visualized in Fig. 4 to understand what PocketGen has learned, selecting a generated pocket for the ligand APX as a case study. In Figure 4(a), a 2D interaction plot is presented, drawn using Schrödinger Maestro software. To assess PocketGen's recognition of meaningful protein-ligand interactions, we plot the heatmap of attention weights produced by the final layer in PocketGen's neural architecture model. In Figure 4(b), two attention heads are chosen for illustration, with each row and column representing a protein residue or a ligand atom, respectively. The attention heatmaps are sparse, reflecting the use of sparse attention in PocketGen (refer to Methods for more details). The attention heads display diverse patterns, focusing on different aspects. For instance, it is hypothesized that the first head emphasizes hydrogen bonds, assigning high weights between residue THR146, ASP220, and ligand atom 7. The second attention head appears to capture π-π stacking and π-Cation interactions, specifically between residue TYR99 and ligand atoms 15, 21, 23, 25, 29, and 33; and residue LYS192 and ligand atoms 1, 14, 17, 19, and 20. These observations suggest that PocketGen, albeit trained via a data-driven approach, has acquired some biochemical insights to form beneficial interactions.

Discussion

Protein-ligand binding is critical in enzyme catalysis, immune recognition, cellular signal transduction, gene expression control, and other biological processes. Recent developments include deep generative models designed to study protein-ligand binding, like Lingo3DMol[71], ResGen[72], and PocketFlow[73] which generate *de novo* drug-like ligand molecules for fixed protein targets; NeuralPLexer[4] can create the structure of protein-ligand complexes given the protein sequence and ligand molecular graph. However, these models do not facilitate the *de novo* generation of protein pockets, the interfaces that bind with the ligand molecule for targeted ligand binding, critical in enzyme and biosensor engineering. We developed PocketGen, a deep generative method capable of generating both the residue sequence and the full atom structures of the protein pocket region for binding with the target ligand molecule. PocketGen includes two main modules: a bilevel graph transformer for structural encoding and updates and a sequence refinement module that uses protein language models (pLMs) for sequence prediction. To achieve sequence-structure consistency and effectively leverage evolutionary knowledge from pLMs, a structural adapter is integrated into protein language models for sequence updates. This adapter employs cross-attention between sequence and structure features to promote information flow and ensure sequence-structure consistency. Extensive experiments across benchmarks and case studies involving therapeutic ligand molecules illustrate PocketGen's ability to generate high-fidelity pocket structures with high binding affinity and favorable interactions with target ligands. Analysis of PocketGen's performance across various settings reveals its proficiency in balancing diversity and affinity and generalizing across different pocket sizes. Additionally, PocketGen offers computational efficiency, significantly reducing runtime compared to traditional physics-based methods, making it feasible to sample large quantities of pocket candidates. PocketGen surpasses existing methods in efficiently generating high-affinity protein pockets for target ligand molecules, finding important interactions between atoms on protein and ligand molecules, and attaining consistency in sequence and structure domains.

PocketGen creates several fruitful directions for future work. PocketGen could be expanded to design larger areas of the protein beyond the pocket area. While PocketGen has been evaluated on larger pocket designs, modifications will be required to enhance scalability and robustness for generating larger protein areas. Another fruitful future direction involves incorporating additional biochemical priors, such as subpockets[74] and interaction templates[17], to improve generalizability and success rates. For instance, despite overall dissimilarity, two protein pockets might still bind the same fragment if they share similar subpockets[75]. Moreover, conducting wet lab experiments could provide empirical validation of PocketGen's effectiveness. Approaches such as PocketGen have the potential to advance areas of machine learning and bioengineering and help with the design of small molecule binders and enzymes.

Methods

Overview of PocketGen

Unlike previous methods focusing on protein sequence or structure generation, we aim to co-design both residue types (sequences) and 3D structures of the protein pocket that can fit and bind with target ligand molecules. Inspired by previous works on structure-based drug design[72,74] and protein generation[34,35], we formulate pocket generation in PocketGen as a conditional generation problem that generates the sequences and structures of pocket conditioned on the protein scaffold (other parts of the protein except the pocket region) and the binding ligand. To be specific, let $\mathcal{A} = \boldsymbol{a}_1 \cdots \boldsymbol{a}_{N_s}$ denote the whole protein sequence of residues, 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 $\{\boldsymbol{a}_{i,j}\}_{1\le i\le N_s, 1\le j\le n_i}$ and let $\boldsymbol{x}(\boldsymbol{a}_{i,j}) \in \mathbb{R}^3$ denote the 3D coordinate of protein 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 the side-chain atoms. The ligand molecule can also be represented as a 3D point cloud $\mathcal{M} = \{\boldsymbol{v}_k\}_{k=1}^{N_l}$ where $\boldsymbol{v}_k$ denotes the atom feature. Let $\boldsymbol{x}(\boldsymbol{v}_k)$ denotes the 3D coordinates of atom $\boldsymbol{v}_k$. Our work defines the protein pocket as a set of residues in the protein closest to the binding ligand molecule: $\mathcal{B} = \boldsymbol{b}_1 \cdots \boldsymbol{b}_m$. The pocket $\mathcal{B}$ can thus be represented as an amino acid subsequence of a protein: $\mathcal{B} = \boldsymbol{a}_{e_1} \cdots \boldsymbol{a}_{e_m}$ where $\boldsymbol{e} = \{e_1, \cdots, e_m\}$ is the index of the pocket residues in the whole protein. The index $\boldsymbol{e}$ can be formally given as: $\boldsymbol{e} = \{i \mid \min_{1\le j\le n_i, 1\le k\le N_l} \|\boldsymbol{x}(\boldsymbol{a}_{i,j}) - \boldsymbol{x}(\boldsymbol{v}_k)\|_2 \le \delta\}$, where $\|\cdot\|_2$ is the L_2 distance norm and δ is the distance threshold. According to the distance range of pocket-ligand interactions[46], we set $\delta = 3.5$ Å in the default setting. With the above-defined notations, PocketGen aims to learn a conditional generative model formally defined as :

$$P(\mathcal{B} | \mathcal{A} \setminus \mathcal{B}, \mathcal{M}), \tag{1}$$

where $\mathcal{A} \setminus \mathcal{B}$ denotes the other parts of the protein except the pocket region. We also adjust the structure ligand molecule $\mathcal{M}$ in PocketGen to encourage protein-ligand interactions and reduce steric clashes.

To effectively generate the structure and the sequence of the protein pocket $\mathcal{B}$, the equivariant bilevel graph transformer and the sequence refinement module with pretrained protein language models and adapters are proposed, which will be discussed in the following paragraphs. The illustrative workflow is depicted in Fig. 1.

Equivariant bilevel graph transformer

It is critical to model the complex interactions in the protein pocket-ligand complexes for pocket generation. However, the multi-granularity (e.g., atom-level and residue-level) and multi-aspect (intra-protein and protein-ligand) nature of interactions brings a lot of challenges. Inspired by recent works on hierarchical graph transformer[74] and generalist equivariant transformer[76], we propose a novel equivariant bilevel graph transformer to well model the multi-granularity and multi-aspect interactions. Each residue or ligand is represented as a block (i.e., a set of atoms) for the conciseness of representation and ease of computation. Then the protein-ligand complex can be abstracted as a geometric graph of sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{\boldsymbol{H}_i, \boldsymbol{X}_i | 1 \le i \le B\}$ denotes the blocks and $\mathcal{E} = \{e_{ij} | 1 \le i, j \le B\}$ include all the edges between blocks. We added self-loops to the edges to capture interactions within the block (e.g., the interactions between ligand atoms). Our model adaptively assigns different numbers of channels to $\boldsymbol{H}_i$ and $\boldsymbol{X}_i$ to accommodate different numbers of atoms in residues and ligands. For example, given a block with n_i atoms, the corresponding block has $\boldsymbol{H}_i \in \mathbb{R}^{n_i \times d_h}$ indicating the atom features (d_h is the feature dimension size) and $\boldsymbol{X}_i \in \mathbb{R}^{n_i \times 3}$ denoting the atom coordinates. Specifically, the p-th row of H_i and $\boldsymbol{X}_i$ corresponds to the p-th atom's trainable feature (i.e., $\boldsymbol{H}_i[p]$) and coordinates (i.e., $\boldsymbol{X}_i[p]$) respectively. The trainable feature $\boldsymbol{H}_i[p]$ is first initialized with the concatenation of atom type embedding, residue/ligand embeddings, and the atom positional embeddings. To build $\mathcal{E}$, we connect the k-nearest neighboring residues according to the pairwise C_α distances. To reflect the interactions between the protein pocket and ligand, we add edges between all the pocket residue and the ligand block. We describe the modules in PocketGen's equivariant bilevel graph transformer, bilevel attention module, and equivariant feed-forward networks.

Bilevel attention module. Our model captures both **atom-level** and **residue/ligand-level** interactions with the bilevel attention module. Firstly, given two block i and j connected by an edge e_{ij}, we obtain the query, the key, and the value matrices with the following transformations:

$$\boldsymbol{Q}_i = \boldsymbol{H}_i \boldsymbol{W}_Q, \qquad \boldsymbol{K}_j = \boldsymbol{H}_j \boldsymbol{W}_K, \qquad \boldsymbol{V}_j = \boldsymbol{H}_j \boldsymbol{W}_V, \tag{2}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_h \times d_r}$ are trainable parameters.

To calculate the **atom-level attention** across the i-th and j-th block, we denote $\boldsymbol{X}_{ij} \in \mathbb{R}^{n_i \times n_j \times 3}$ and $\boldsymbol{D}_{ij} \in \mathbb{R}^{n_i \times n_j}$ as the relative coordinates and distances between atom pairs in block i and j, namely, $\boldsymbol{X}_{ij}[p,q] = \boldsymbol{X}_i[p] - \boldsymbol{X}_j[q], \boldsymbol{D}_{ij}[p,q] = \|\boldsymbol{X}_{ij}[p,q]\|_2$.

Then we have:

$$\boldsymbol{R}_{ij} = \frac{1}{\sqrt{d_r}} \left(\boldsymbol{Q}_i \boldsymbol{K}_j^\top \right) + \sigma_D \left(\mathrm{RBF}(\boldsymbol{D}_{ij}) \right), \tag{3}$$

$$\boldsymbol{\alpha}_{ij} = \mathrm{Softmax} \left(\boldsymbol{R}_{ij} \right), \tag{4}$$

where $\sigma_D(\cdot)$ is a Multi-Layer Perceptron (MLP) that adds distance bias to the attention calculation. RBF embeds the distance with radial basis functions. $\boldsymbol{\alpha}_{ij} \in \mathbb{R}^{n_i \times n_j}$ is the atom-level attention matrix obtained by applying row-wise Softmax on $\boldsymbol{R}_{ij} \in \mathbb{R}^{n_i \times n_j}$. To encourage sparsity in the attention matrix, we keep the top-k' elements of each row in $\boldsymbol{\alpha}_{ij}$ and set the others as zeros.

The **residue/ligand-level attention** from the j-th block to the i-th block is calculated as:

$$r_{ij} = \frac{\mathbf{1}^\top \boldsymbol{R}_{ij} \mathbf{1}}{n_i n_j}, \tag{5}$$

$$\beta_{ij} = \frac{\exp(r_{ij})}{\sum_{j \in \mathcal{N}(i)} \exp(r_{ij})}, \tag{6}$$

where $\mathbf{1}$ refers to the column vector with all elements set as ones and $\mathcal{N}(i)$ denotes the neighboring blocks of i. r_{ij} sums up all values in $\boldsymbol{R}_{ij}$ to represent the overall correlation between blocks i and j. Subsequently, β_{ij} denotes the attention across blocks at the block level.

We can update the representations and coordinates using the above atom-level and the residue/ligand-level attentions. PocketGen only updates the coordinates of the residues in the pocket and the ligand molecule. The other protein residues are fixed. Specifically, for the p-th atom in block i:

$$m_{ij,p} = \beta_{ij} \left(\boldsymbol{\alpha}_{ij}[p] \odot \phi_x(\boldsymbol{Q}_i[p] || \boldsymbol{K}_j || \mathrm{RBF}(\boldsymbol{D}_{ij}[p])) \right), \tag{7}$$

$$\boldsymbol{H}'_i[p] = \boldsymbol{H}_i[p] + \sum_{j \in \mathcal{N}(i)} \beta_{ij} \phi_h(\boldsymbol{\alpha}_{ij}[p] \cdot \boldsymbol{V}_j), \tag{8}$$

$$\boldsymbol{X}'_i[p] = \boldsymbol{X}_i[p] + \begin{cases} \sum_{j \in \mathcal{N}(i)} m_{ij,p} \cdot \boldsymbol{X}_{ij}[p], & \text{if } i \text{ belongs to ligand or pocket residues} \\ 0, & \text{if } i \text{ belongs to other protein residues} \end{cases} \tag{9}$$

where ϕ_h and ϕ_x are MLPs with concatenated representations as input (concatenation along the second dimension and $\boldsymbol{Q}_i[p]$ is repeated along rows). $\odot$ computes the element-wise multiplication. $\boldsymbol{H}'_i$ and $\boldsymbol{X}'_i$ denote the updated representation and coordinate matrices, and we can verify that the dimension size of $\boldsymbol{H}'_i$ and $\boldsymbol{X}'_i$ remains the same regardless of the neighboring block size n_j. Furthermore, as the attention coefficients $\boldsymbol{\alpha}_{ij}$ and β_{ij} are invariant under E(3) transformations, the modification of $\boldsymbol{X}'_i$ adheres to E(3)-equivariance. Additionally, the permutation of atoms within each block does not affect this update process.

Equivariant feed-forward network. We adapted the feed-forward network module (FFN) in the transformer model[77] to update $\boldsymbol{H}_i$ and $\boldsymbol{X}_i$. Specifically, the representation and coordinates of atoms are updated to consider the block's feature/geometric centroids (means). The centroids are denoted as:

$$\boldsymbol{h}_c = \mathrm{centroid}(\boldsymbol{H}_i), \qquad \boldsymbol{x}_c = \mathrm{centroid}(\boldsymbol{X}_i), \tag{10}$$

Then we obtain the relative coordinate $\Delta \boldsymbol{x}_p$ and the relative distance representation $\boldsymbol{r}_p$ based on the L2 norm of $\Delta \boldsymbol{x}_p$:

$$\Delta \boldsymbol{x}_p = \boldsymbol{X}_i[p] - \boldsymbol{x}_c, \qquad \boldsymbol{r}_p = \mathrm{RBF}(\|\Delta \boldsymbol{x}_p\|_2), \tag{11}$$

The representation and coordinates of atoms are updated with MLPs σ_h and σ_x. The centroids are integrated to inform of the context of the block:

$$\boldsymbol{H}'[p] = \boldsymbol{H}[p] + \sigma_h(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p), \tag{12}$$

$$\boldsymbol{X}'_i[p] = \boldsymbol{X}_i[p] + \Delta \boldsymbol{x}_p \sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p). \tag{13}$$

To stabilize and accelerate training, layer normalization[78] is appended at each layer of the equivariant bilevel graph transformer to normalize $\boldsymbol{H}$. The equivariant feed-forward network satisfies E(3)-equivariance. Thanks to each module's E(3)-equivariance, the whole proposed bilevel graph transformer has the desirable property of E(3)-equivariance (Theorem 1 in Supplementary Information shows the details).

Sequence refinement with pretrained protein language models and adapters

Protein language models (pLMs), such as the ESM family of models[40,41], have learned extensive evolutionary knowledge from the vast array of natural protein sequences, demonstrating a strong ability to design protein sequences. In PocketGen, we propose to leverage pLMs to help refine the designed protein pocket sequences. To infuse the pLMs with structural information, we implant lightweight structural adapters inspired by previous works[79,80]. In our default setting, only one structural adapter was placed after the last layer of pLM. Only the adapter layers are fine-tuned during training, and the other layers of PLMs are frozen to save computation costs. The structural adapter mainly has the following two parts.

Structure-sequence cross attention. The structural representation of the i-th residue $\boldsymbol{h}_i^{\text{struct}}$ is obtained by mean pooling of $\boldsymbol{H}_i$ from the bilevel graph transformer. In the input to the pLMs, the pocket residue types to be designed are assigned with the mask, and we denote the i-th residue representation from pLMs as $\boldsymbol{h}_i^{\text{seq}}$. In the structural adapter, we perform cross-attention between the structural representations $\boldsymbol{H}^{\text{struct}} = \{\boldsymbol{h}_1^{\text{struct}}, \boldsymbol{h}_2^{\text{struct}}, \cdots, \boldsymbol{h}_{N_s}^{\text{struct}}\}$ and sequence representations $\boldsymbol{H}^{\text{seq}} = \{\boldsymbol{h}_1^{\text{seq}}, \boldsymbol{h}_2^{\text{seq}}, \cdots, \boldsymbol{h}_{N_s}^{\text{seq}}\}$. The query, key, and value matrices are obtained as follows:

$$\boldsymbol{Q} = \boldsymbol{H}^{\text{seq}}\boldsymbol{W}_Q, \qquad \boldsymbol{K} = \boldsymbol{H}^{\text{struct}}\boldsymbol{W}_K, \qquad \boldsymbol{V} = \boldsymbol{H}^{\text{struct}}\boldsymbol{W}_V, \tag{14}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_h \times d_r}$ are trainable weight matrices. Rotary positional encoding[81] is applied to the representations, and we omit it in the equations for simplicity. The output of the cross attention is obtained as:

$$\text{CrossAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_r}}\right)\boldsymbol{V}. \tag{15}$$

Bottleneck feed-forward network. A bottleneck feed-forward network (FFN) is appended after the cross-attention to impose non-linearity and abstract representations, inspired by previous works such as Houlsby et al.[79]. The intermediate dimension of the bottleneck FFN is set to be half of the default representation dimension. Finally, the predicted pocket residue type $\boldsymbol{p}_i$ is obtained using an MLP on the output residue representation.

Training protocol

Inspired by AlphaFold2[39], we use a recycling strategy for model training. Recycling facilitates the training of deeper networks without incurring extra memory costs by executing multiple forward passes and computing gradients solely for the final pass. The training loss of PocketGen is the weighted sum of the following three losses:

$$\mathcal{L}_{\text{seq}} = \frac{1}{T}\sum_t \sum_i l_{\text{ce}}(\hat{\boldsymbol{p}}_i, \boldsymbol{p}_i^t); \tag{16}$$

$$\mathcal{L}_{\text{coord}} = \frac{1}{T}\sum_t \left[\sum_i l_{\text{huber}}(\hat{\boldsymbol{X}}_i, \boldsymbol{X}_i^t) + \sum_j l_{\text{huber}}(\hat{\boldsymbol{x}}(\boldsymbol{v}_j), \boldsymbol{x}^t(\boldsymbol{v}_j))\right]; \tag{17}$$

$$\mathcal{L}_{\text{struct}} = \frac{1}{T}\sum_t \left[\sum_{b\in\mathcal{B}} l_{\text{huber}}(\hat{b}, b^t) + \sum_{\theta\in\Theta} l_{\text{huber}}(\cos\hat{\theta}, \cos\theta^t)\right]; \tag{18}$$

$$\mathcal{L} = \mathcal{L}_{\text{seq}} + \lambda_{\text{coord}}\mathcal{L}_{\text{coord}} + \lambda_{\text{struct}}\mathcal{L}_{\text{struct}}, \tag{19}$$

where T is the total refinement rounds. $\hat{\boldsymbol{p}}_i, \hat{\boldsymbol{X}}_i, \hat{\boldsymbol{x}}(\boldsymbol{v}_j), \hat{b}$, and $\cos\hat{\theta}$ are the ground-truth residue types, residue coordinates, and ligand coordinates, bond lengths, and bond/dihedral angles; $\boldsymbol{p}_i^t, \boldsymbol{X}_i^t, \boldsymbol{x}^t(\boldsymbol{v}_j), b^t$, and $\cos\theta^t$ are the predicted ones at the t-th round by PocketGen. The sequence loss $\mathcal{L}_{\text{seq}}$ is the cross-entropy loss for pocket residue type prediction; the coordinate loss $\mathcal{L}_{\text{coord}}$ uses huber loss[82] for the training stability; the structure loss $\mathcal{L}_{\text{struct}}$ is added to supervised bond lengths and bond/dihedral angles for realistic local geometry. $\mathcal{B}$ and Θ denote all the bonds and angles in the protein pocket (including side chains). λ_{coord}, and λ_{struct} are hyperparameters balancing the three losses. We perform a grid search over $\{0.5, 1.0, 2.0, 3.0\}$ and choose these hyperparameters based on the validation performance to select the specific parameter values. In the default setting, we set λ_{coord} to 1.0 and λ_{struct} to 2.0.

Generation protocol

In the generation procedure, PocketGen initializes the sequence with uniform distributions over 20 amino acid types and the coordinates based on linear interpolations and extrapolations. Specifically, 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 $\mathcal{A} = \boldsymbol{a}_1 \cdots \boldsymbol{a}_{N_s}$, where N_s is the length of the sequence. Let $\boldsymbol{x}(\boldsymbol{a}_{i,1}) \in \mathbb{R}^3$ denote the C_α coordinate of the i-th residue. We take the following strategies to determine the C_α coordinate of the i-th residue: (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: $\boldsymbol{x}(\boldsymbol{a}_{i,1}) = \frac{1}{q-p}[(i-p)\boldsymbol{x}(\boldsymbol{a}_{q,1}) + (q-i)\boldsymbol{x}(\boldsymbol{a}_{p,1})]$. (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 $\boldsymbol{x}(\boldsymbol{a}_{i,1}) = \boldsymbol{x}(\boldsymbol{a}_{p,1}) + \frac{i-p}{p-q}(\boldsymbol{x}(\boldsymbol{a}_{p,1}) - \boldsymbol{x}(\boldsymbol{a}_{q,1}))$. Inspired by previous works[33,34], we initialize the other backbone atom coordinates according to their ideal local coordinates relative to the C_α coordinates. We initialize the side-chain atoms' coordinates with the coordinate of their corresponding C_alpha, added with Gaussian noise. We initialize the ligand molecular structure with the reference ligand structure from the dataset.

Since the number of pocket residue types and the number of side chain atoms are unknown at the beginning of generation, each pocket residue is assigned 14 atoms, the maximum number of atoms for residues. After rounds of refinement by PocketGen, the pocket residue types are predicted, and the full atom coordinates are determined by mapping the coordinates to the predicted residue types. For generation efficiency, we set the number of refinement rounds to 3.

Experimental setting

Datasets. We consider two widely used datasets for benchmark evaluation: **CrossDocked** dataset[43] contains 22.5 million protein-molecule pairs generated through cross-docking. Following previous works[24,58,83], 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[84] 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[44] contains around 41k experimentally determined protein-ligand complexes. Following previous work[42], we keep pockets with valid and moderately 'drug-like' ligands with QED score ≥ 0.3. We further filter the dataset to discard molecules containing atom types $\notin \{C, N, O, S, B, Br, Cl, P, I, F\}$ as well as binding pockets with non-standard amino acids. Then, we randomly sample and split the filtered dataset based on the Enzyme Commission Number (EC Number)[45] to ensure different sets do not contain proteins from the same EC Number main class. Finally, we have 40k protein-ligand pairs for training, 100 pairs for validation, and 100 pairs for testing. For all the benchmark tasks in this paper, PocketGen and all the other baseline methods are trained with the same data split for a fair comparison. In real-world pocket generation and optimization case studies, the protein structures were downloaded from PDB[85].

Implementation. Our PocketGen model is trained with Adam[86] optimizer for 5k iterations, where the learning rate is 0.0001, and the batch size is 64. We report the results corresponding to the checkpoint with the best validation loss. It takes around 48 hours to finish training on 1 Tesla A100 GPU from scratch. In PocketGen, the number of attention heads is set as 4; the hidden dimension d is set as 128; k is set to 8 to connect the k-nearest neighboring residues to build $\mathcal{E}$; k' is set as 3 to encourage sparsity in the attention matrix. For all the benchmark tasks of pocket generation and optimization, PocketGen and all the other baseline methods are trained with the same data split for a fair comparison. We follow the implementation codes provided by the authors to obtain the results of baseline methods. Algorithm 1 and 2 in the supplementary show the pseudo-codes of the training and generation process of PocketGen.

Baseline methods. PocketGen is compared with five state-of-the-art representative baseline methods. **PocketOptimizer**[18] is a physics-based method that optimizes energies such as packing and binding-related energies for ligand-binding protein design. Following the suggestion of the paper, we fixed the backbone structures. **DEPACT**[17] is a template-matching method that follows a two-step strategy[87] 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 scaffold to output the complete protein structure with PACMatch[17]. Both the backbone and the sidechain structures are changed in DEPACT. **RFDiffusion**[26], **RFDiffusionAA**[16], **FAIR**[24], and **dyMEAN**[25] are deep-learning-based models that for protein generation. RFDiffusion does not explicitly model protein-ligand interactions and is not directly applicable to small molecule-binding protein generation. Following the suggestions in RFDiffusion[26] and RFDiffusionAA[16], we use a heuristic attractive-repulsive potential to encourage the formation of pockets with shape complementarity to a target molecule. The residue sequence for the generated protein by RFDiffusion is derived with ProteinMPNN, and the side-chain conformation is decided with Rosetta[88] side-chain packing. RFDiffusionAA is the latest version of RFDiffusion, which can directly generate protein structures surrounding small molecules by combining residue-based representation of amino acids with atomic representation of small molecules. For RFDiffusion and RFDiffusionAA, we let them in paint the pocket area to obtain a consistent setting with other methods for comparison. We also note that RFDiffusion and RFDiffusionAA do not provide the training/finetuning scripts, so we use the provided pre-trained checkpoints for all the related experiments in our paper. FAIR[24] was specially designed for full-atom protein pocket design via iterative refinement. dyMEAN[25] was originally proposed for full atom antibody design, and we adapted it to our pocket design task with proper modifications. Detailed information on baselines is included in Supplementary Notes. The setting of the key hyperparameters is summarized in Table. S4. All the baselines are run on the same Telsa A100 GPU for a fair comparison with our PocketGen.

Data availability

This study's training and test data are available at Zenodo (https://zenodo.org/records/10125312). The project website for PocketGen is at https://zitniklab.hms.harvard.edu/projects/PocketGen.

Code availability

The source code of this study is freely available at GitHub (https://github.com/zaixizhang/PocketGen) to allow for replication of the results of this study.

References

1. Tinberg, C. E. *et al.* Computational design of ligand-binding proteins with high affinity and selectivity. *Nature* **501**, 212–216 (2013).
2. Kroll, A., Ranjan, S., Engqvist, M. K. & Lercher, M. J. A general model to predict small molecule substrates of enzymes based on machine and deep learning. *Nature Communications* **14**, 2787 (2023).
3. Lee, G. R. *et al.* Small-molecule binding and sensing with a designed protein family. *bioRxiv* 2023–11 (2023).
4. Qiao, Z., Nie, W., Vahdat, A., Miller III, T. F. & Anandkumar, A. State-specific protein–ligand complex structure prediction with a multiscale deep generative model. *Nature Machine Intelligence* 1–14 (2024).
5. Jiang, L. *et al.* De novo computational design of retro-aldol enzymes. *science* **319**, 1387–1391 (2008).
6. Röthlisberger, D. *et al.* Kemp elimination catalysts by computational enzyme design. *Nature* **453**, 190–195 (2008).
7. Dou, J. *et al.* De novo design of a fluorescence-activating β-barrel. *Nature* **561**, 485–491 (2018).
8. Yeh, A. H.-W. *et al.* De novo design of luciferases using deep learning. *Nature* **614**, 774–780 (2023).
9. Beltrán, J. *et al.* Rapid biosensor development using plant hormone receptors as reprogrammable scaffolds. *Nature Biotechnology* **40**, 1855–1861 (2022).
10. Bick, M. J. *et al.* Computational design of environmental sensors for the potent opioid fentanyl. *Elife* **6**, e28909 (2017).
11. Glasgow, A. A. *et al.* Computational design of a modular protein sense-response system. *Science* **366**, 1024–1028 (2019).
12. Herud-Sikimić, O. *et al.* A biosensor for the direct visualization of auxin. *Nature* **592**, 768–772 (2021).
13. Polizzi, N. F. & DeGrado, W. F. A defined structural unit enables de novo design of small-molecule–binding proteins. *Science* **369**, 1227–1233 (2020).
14. Basanta, B. *et al.* An enumerative algorithm for de novo design of proteins with diverse pocket structures. *Proceedings of the National Academy of Sciences* **117**, 22135–22145 (2020).
15. Dou, J. *et al.* Sampling and energy evaluation challenges in ligand binding protein design. *Protein Science* **26**, 2426–2437 (2017).
16. Krishna, R. *et al.* Generalized biomolecular modeling and design with rosettafold all-atom. *Science* eadl2528 (2024).
17. Chen, Y., Chen, Q. & Liu, H. Depact and pacmatch: A workflow of designing de novo protein pockets to bind small molecules. *Journal of Chemical Information and Modeling* **62**, 971–985 (2022).
18. Noske, J., Kynast, J. P., Lemm, D., Schmidt, S. & Höcker, B. Pocketoptimizer 2.0: A modular framework for computer-aided ligand-binding design. *Protein Science* **32**, e4516 (2023).
19. Malisi, C. *et al.* Binding pocket optimization by computational protein design. *PloS one* **7**, e52505 (2012).
20. Stiel, A. C., Nellen, M. & Höcker, B. Pocketoptimizer and the design of ligand binding sites. *Computational Design of Ligand Binding Proteins* 63–75 (2016).
21. Lu, L. *et al.* De novo design of drug-binding proteins with predictable binding energy and specificity. *Science* **384**, 106–112 (2024).
22. Byon, W., Garonzik, S., Boyd, R. A. & Frost, C. E. Apixaban: a clinical pharmacokinetic and pharmacodynamic review. *Clinical pharmacokinetics* **58**, 1265–1279 (2019).
23. Stark, H., Jing, B., Barzilay, R. & Jaakkola, T. Harmonic prior self-conditioned flow matching for multi-ligand docking and binding site design. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop* (2023).
24. Zhang, Z., Lu, Z., Hao, Z., Zitnik, M. & Liu, Q. Full-atom protein pocket design via iterative refinement. In *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

25. Kong, X., Huang, W. & Liu, Y. End-to-end full-atom antibody design. *ICML* (2023).

26. Watson, J. L. *et al.* De novo design of protein structure and function with rfdiffusion. *Nature* **620**, 1089–1100 (2023).

27. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* **33**, 6840–6851 (2020).

28. Baek, M. *et al.* Accurate prediction of protein structures and interactions using a three-track neural network. *Science* **373**, 871–876 (2021).

29. Dauparas, J. *et al.* Robust deep learning–based protein sequence design using proteinmpnn. *Science* **378**, 49–56 (2022).

30. Dauparas, J. *et al.* Atomic context-conditioned protein sequence design using ligandmpnn. *Biorxiv* 2023–12 (2023).

31. Jin, W., Wohlwend, J., Barzilay, R. & Jaakkola, T. Iterative refinement graph neural network for antibody sequence-structure co-design. *ICLR* (2022).

32. Jin, W., Barzilay, R. & Jaakkola, T. Antibody-antigen docking and design via hierarchical structure refinement. In *ICML*, 10217–10227 (PMLR, 2022).

33. Luo, S. *et al.* Antigen-specific antibody design and optimization with diffusion-based generative models. *NeurIPS* (2022).

34. Kong, X., Huang, W. & Liu, Y. Conditional antibody design as 3d equivariant graph translation. *ICLR* (2023).

35. Shi, C., Wang, C., Lu, J., Zhong, B. & Tang, J. Protein sequence and structure co-design with equivariant translation. *ICLR* (2023).

36. Anishchenko, I. *et al.* De novo protein design by deep network hallucination. *Nature* **600**, 547–552 (2021).

37. Yang, J. *et al.* Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences* **117**, 1496–1503 (2020).

38. Cao, L. *et al.* Design of protein-binding proteins from the target structure alone. *Nature* **605**, 551–560 (2022).

39. Jumper, J. *et al.* Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).

40. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS* (2019). URL https://www.biorxiv.org/content/10.1101/622803v4.

41. Lin, Z. *et al.* Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv* (2022).

42. Schneuing, A. *et al.* Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695* (2022).

43. Francoeur, P. G. *et al.* Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of chemical information and modeling* **60**, 4200–4215 (2020).

44. Hu, L., Benson, M. L., Smith, R. D., Lerner, M. G. & Carlson, H. A. Binding moad (mother of all databases). *Proteins: Structure, Function, and Bioinformatics* **60**, 333–340 (2005).

45. Bairoch, A. The enzyme database in 2000. *Nucleic acids research* **28**, 304–305 (2000).

46. Marcou, G. & Rognan, D. Optimizing fragment and scaffold docking by use of molecular interaction fingerprints. *Journal of chemical information and modeling* **47**, 195–207 (2007).

47. Trott, O. & Olson, A. J. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry* **31**, 455–461 (2010).

48. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).

49. Trippe, B. L. *et al.* Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations* (2023). URL https://openreview.net/forum?id=6TxBxqNME1Y.

50. Lin, Y. & AlQuraishi, M. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds. *ICML* (2023).

51. Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics* **57**, 702–710 (2004).

52. Yim, J. *et al.* Improved motif-scaffolding with se (3) flow matching. *arXiv preprint arXiv:2401.04082* (2024).

53. Yim, J. *et al.* Se (3) diffusion model with application to protein backbone generation. In *International Conference on Machine Learning*, 40001–40039 (PMLR, 2023).

54. Yoo, Y. J., Feng, Y., Kim, Y.-H. & Yagonia, C. F. J. Fundamentals of enzyme engineering (2017).

55. Traut, T. W. Protein engineering: Principles and practice. *American Scientist* **85**, 571–573 (1997).

56. Spencer, R. K. *et al.* Stereochemistry of polypeptoid chain configurations. *Biopolymers* **110**, e23266 (2019).

57. http://www.mlb.co.jp/linux/science/garlic/doc/commands/dihedrals.html .

58. Peng, X. *et al.* Pocket2mol: Efficient molecular sampling based on 3d protein pockets. *ICML* (2022).

59. Zhang, Z., Liu, Q., Lee, C.-K., Hsieh, C.-Y. & Chen, E. An equivariant generative framework for molecular graph-structure co-design. *Chemical Science* **14**, 8380–8392 (2023).

60. Kaplan, J. *et al.* Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

61. Alberts, B. *Molecular biology of the cell* (Garland science, 2017).

62. Shoichet, B. K. Virtual screening of chemical libraries. *Nature* **432**, 862–865 (2004).

63. Ertl, P. An algorithm to identify functional groups in organic molecules. *Journal of cheminformatics* **9**, 1–7 (2017).

64. Eronen, V. *et al.* Structural insight to elucidate the binding specificity of the anti-cortisol fab fragment with glucocorticoids. *Journal of Structural Biology* **215**, 107966 (2023).

65. Pinto, D. J. *et al.* Discovery of 1-(4-methoxyphenyl)-7-oxo-6-(4-(2-oxopiperidin-1-yl) phenyl)-4, 5, 6, 7-tetrahydro-1 h-pyrazolo [3, 4-c] pyridine-3-carboxamide (apixaban, bms-562247), a highly potent, selective, efficacious, and orally bioavailable inhibitor of blood coagulation factor xa. *Journal of medicinal chemistry* **50**, 5339–5356 (2007).

66. Hernandez, I., Zhang, Y. & Saba, S. Comparison of the effectiveness and safety of apixaban, dabigatran, rivaroxaban, and warfarin in newly diagnosed atrial fibrillation. *The American journal of cardiology* **120**, 1813–1819 (2017).

67. Stanley, T. H. The fentanyl story. *The Journal of Pain* **15**, 1215–1226 (2014).

68. Salentin, S., Schreiber, S., Haupt, V. J., Adasme, M. F. & Schroeder, M. Plip: fully automated protein–ligand interaction profiler. *Nucleic acids research* **43**, W443–W447 (2015).

69. Yang, J., Li, F.-Z. & Arnold, F. H. Opportunities and challenges for machine learning-assisted enzyme engineering. *ACS Central Science* (2024).

70. Zhou, Y., Pan, Q., Pires, D. E., Rodrigues, C. H. & Ascher, D. B. Ddmut: predicting effects of mutations on protein stability using deep learning. *Nucleic Acids Research* gkad472 (2023).

71. Wang, L. *et al.* Lingo3dmol: Generation of a pocket-based 3d molecule using a language model. *Nature Machine Intelligence* (2024).

72. Zhang, O. *et al.* Resgen is a pocket-aware 3d molecular generation model based on parallel multiscale modelling. *Nature Machine Intelligence* 1–11 (2023).

73. Jiang, Y. *et al.* Pocketflow is a data-and-knowledge-driven structure-based molecular generative model. *Nature Machine Intelligence* 1–12 (2024).

74. Zhang, Z. & Liu, Q. Learning subpocket prototypes for generalizable structure-based drug design. *ICML* (2023).

75. Kalliokoski, T., Olsson, T. S. & Vulpetti, A. Subpocket analysis method for fragment-based drug discovery. *Journal of chemical information and modeling* **53**, 131–141 (2013).

76. Kong, X., Huang, W. & Liu, Y. Generalist equivariant transformer towards 3d molecular interaction learning. *arXiv preprint arXiv:2306.01474* (2023).

77. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

78. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

79. Houlsby, N. *et al.* Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2790–2799 (PMLR, 2019).

80. Zheng, Z. *et al.* Structure-informed language models are protein designers. *bioRxiv* 2023–02 (2023).

81. Su, J. *et al.* Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864* (2021).

82. Huber, P. J. Robust estimation of a location parameter. *Breakthroughs in statistics: Methodology and distribution* 492–518 (1992).

83. Luo, S., Guan, J., Ma, J. & Peng, J. A 3d generative model for structure-based drug design. *NeurIPS* **34**, 6229–6239 (2021).

84. Steinegger, M. & Söding, J. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology* **35**, 1026–1028 (2017).

85. Sussman, J. L. *et al.* Protein data bank (pdb): database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica Section D: Biological Crystallography* **54**, 1078–1084 (1998).

86. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

87. Zanghellini, A. *et al.* New algorithms and an in silico benchmark for computational enzyme design. *Protein Science* **15**, 2785–2794 (2006).

88. Alford, R. F. *et al.* The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation* **13**, 3031–3048 (2017).

89. Maier, J. A. *et al.* ff14sb: improving the accuracy of protein side chain and backbone parameters from ff99sb. *Journal of chemical theory and computation* **11**, 3696–3713 (2015).

90. Shapovalov, M. V. & Dunbrack Jr, R. L. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure* **19**, 844–858 (2011).

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 thank Dr. Yaoxi Chen and Dr. Haiyan Liu from the University of Science and Technology of China for their constructive discussions on implementing and evaluating baseline methods, which greatly helped this research. M.Z. gratefully acknowledges the support of NIH R01-HD108794, NSF CAREER 2339524, US DoD FA8702-15-D-0001, awards from Harvard Data Science Initiative, Amazon Faculty Research, Google Research Scholar Program, AstraZeneca Research, Roche Alliance with Distinguished Scientists, Sanofi iDEA-iTECH Award, Pfizer Research, Chan Zuckerberg Initiative, John and Virginia Kaneb Fellowship award at Harvard Medical School, Aligning Science Across Parkinson's (ASAP) Initiative, Biswas Computational Biology Initiative in partnership with the Milken Institute, and Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University.

Author contributions statement

Z.X.Z., Q.L., and M.Z. designed the research, Z.X.Z. conducted the experiments, Z.X.Z., Q.L., and M.Z. analyzed the results, Z.X.Z., W.X.S., Q.L., and M.Z. wrote the manuscript. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Qi Liu and Marinka Zitnik.

Supplementary Information

Supplementary Notes

Training and Generation Algorithms

Algorithm 1 Training Algorithm of PocketGen

Input: protein sequences $\mathcal{A} = \boldsymbol{a}_1 \cdots \boldsymbol{a}_{N_s}$ and structures $\{\boldsymbol{x}(\boldsymbol{a}_i)\}_{i=1}^{N_s}$, ligand molecule $\mathcal{M}$, total iteration rounds T.
Initial: Initialize the coordinates of pocket residues $\{\boldsymbol{x}(\boldsymbol{b}_i)\}_{i=1}^{m}$. Initialize the sequences of pocket residues with uniform distributions over 20 residue categories. Initialize the residue/ligand representations $\{\boldsymbol{H}_i\}$.

while Model training is not converged **do**
  # Recycling training
  for $t = 1, \cdots T$ **do**
    Obtain the residue representations $\{\boldsymbol{H}_i\}$ and updated coordinates $\{\boldsymbol{X}_i\}$ with the equivariant bilevel graph transformer.
    Predict residue types $\boldsymbol{p}_i^t$ with structure refinement modules.
    if $t == T$ **then**
      $\mathcal{L}_{\text{seq}} = \sum_i l_{\text{ce}}(\hat{\boldsymbol{p}}_i, \boldsymbol{p}_i^t)$.
      $\mathcal{L}_{\text{coord}} = \sum_i l_{\text{huber}}(\hat{\boldsymbol{X}}_i, \boldsymbol{X}_i^t) + \sum_j l_{\text{huber}}(\hat{\boldsymbol{x}}(\boldsymbol{v}_j), \boldsymbol{x}^t(\boldsymbol{v}_j))$.
      $\mathcal{L}_{\text{struct}} = \sum_{b \in \mathcal{B}} l_{\text{huber}}(\hat{b}, b^t) + \sum_{\theta \in \Theta} l_{\text{huber}}(\cos\hat{\theta}, \cos\theta^t)$
    end if
  end for
  Minimize $\lambda_{\text{seq}}\mathcal{L}_{\text{seq}} + \lambda_{\text{coord}}\mathcal{L}_{\text{coord}} + \lambda_{\text{struct}}\mathcal{L}_{\text{struct}}$.
end while

Algorithm 2 Generation Algorithm of PocketGen

Input: protein sequences $\mathcal{A} = \boldsymbol{a}_1 \cdots \boldsymbol{a}_{N_s}$ and structures $\{\boldsymbol{x}(\boldsymbol{a}_i)\}_{i=1}^{N_s}$, ligand molecule $\mathcal{M}$, total iteration rounds T.
Initial: Initialize the coordinates of pocket residues $\{\boldsymbol{x}(\boldsymbol{b}_i)\}_{i=1}^{m}$. Initialize the sequences of pocket residues with uniform distributions over 20 residue categories. Initialize the residue/ligand representations $\{\boldsymbol{H}_i\}$.

for $t = 1, ..., T$ **do**
  Obtain the residue representations $\{\boldsymbol{H}_i\}$ and updated coordinates $\{\boldsymbol{X}_i\}$ with the equivariant bilevel graph transformer.
  Predict residue types $\boldsymbol{p}_i^t$ with structure refinement modules.
end for
Map $\{\boldsymbol{X}_i\}$ to the predicted residue types.
Output designed pockets.

Loss Functions

In Equation 19, we use Huber loss[82] in $\mathcal{L}_{\text{coord}}$ and $\mathcal{L}_{\text{struct}}$ for the stability of optimization, which is defined as follows:

$$l_{huber}(x, y) = \begin{cases} 0.5\ (x-y)^2,\ if\ |x-y| < \epsilon, \\ \epsilon \cdot (|x-y| - 0.5 \cdot \epsilon),\ else, \end{cases} \tag{20}$$

where x and y represent the predicted and ground-truth coordinates/ bond length/ angles. 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 applied for further finetuning. In practice, we find that directly using MSE loss sometimes leads to instability at the beginning of the training (e.g., very large gradient norm), while Huber loss makes the training procedure more stable. Following the suggestion of previous works[32,34], we set $\epsilon = 1$ in all our experiments.

Implementation Details of Baseline Methods and Hyperparameter Settings

PocketOptimizer[18] [1] is a physic-based computational protein design method that predicts mutations in the binding pockets of proteins to increase affinity for a specific ligand. We use the latest version, i.e., PocketOptimizer 2.0[18]. There are generally 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. Following the suggestions in the original paper, we use AMBER ff14S force field[89] for energy computation and the Dunbrack rotamer library[90] for rotamer sampling for PocketOptimizer in our implementation. We fixed the backbone structures following the suggestions of the original paper. As for the output design solutions, we select the top 100 designs identified by PocketOptimizer based on integer linear programming for downstream metric calculations.

DEPACT[17] [2] is a template-matching method that follows a two-step strategy for pocket design. Firstly, it searches the protein-ligand complexes in the template database with similar ligand fragments and constructs a cluster model (a set of pocket residues). The template databases are constructed based on the corresponding training datasets for fair comparisons. Secondly, it grafts the cluster model into the protein pocket with PACMatch. It works by placing residues from the cluster model on protein scaffolds by matching the atoms of residues with atoms of the protein scaffold. The backbone coordinates of the pocket residues are also modified in the process. 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 output of DEPACT+PACMatch is complete protein structures with redesigned pockets. In the paper, we only use DEPACT to represent the whole method of DEPACT+PACMatch for conciseness.

RFDiffusion[26] [3] is one of the state-of-the-art method for *de novo* protein backbone generation. It combines the RoseTTAFold structure prediction network with the diffusion probabilistic models (DDPMs) framework. To model the influence of ligand molecules, we use a heuristic attractive-repulsive potential to encourage the formation of pockets with shape complementarity to a target molecule following the suggestions of RFDiffusion[26] and RFDiffusionAA[16]. The residue sequence is further decided with ProteinMPNN[29], and the side-chain conformation is added with Rosetta[88] side-chain packing.

RFDiffusionAA[16] [4] is the latest version of RFDiffusion which combines a residue-based representation of amino acids and atomic representations of all other groups to model protein-small molecules/metals/nucleic acids/covalent modification complexes. Starting from random distributions of amino acid residues surrounding target small molecules, RFDiffusionAA can directly generate the small molecule binding protein backbone. Furthermore, with LigandMPNN[30], the latest version of ProteinMPNN[29], we can assign residue types and predict sidechain conformations considering the protein-ligand interactions. Experiments in RFDiffusionAA[16] show that the generated protein by RFDiffusionAA has better binding affinity than those obtained by RFDiffusion with auxiliary potential.

dyMEAN[25] [5] is an end-to-end full-atom model for E(3)-equivariant antibody design given the epitope and the incomplete sequence of the antibody. Its previous version, MEAN[34], only considers the backbone atoms, while dyMEAN considers the complete atom structure and performs better on downstream tasks. Generally, dyMEAN co-designs antibody sequence and structure via a multi-round progressive full-shot refinement manner, which is more efficient than auto-regressive or diffusion-based approaches. An adaptive multi-channel equivariant encoder is used in dyMEAN, which can process protein residues of variable sizes when considering full atoms. To adapt dyMEAN to our pocket design task, we replace the antigen with the target ligand molecule to provide the context information for pocket generation.

FAIR[24] [6] is our previous method for full atom pocket sequence-structure co-design. FAIR operates in two steps, proceeding in a coarse-to-fine manner (backbone refinement to full atoms refinement, including side chains) for full-atom generation. In FAIR, residue types and atom coordinates are updated using a hierarchical graph transformer composed of a residue-level and atom-level encoder.

[1] https://github.com/Hoecker-Lab/pocketoptimizer
[2] https://github.com/chenyaoxi/DEPACT_PACMatch
[3] https://github.com/RosettaCommons/RFdiffusion
[4] https://github.com/baker-laboratory/rf_diffusion_all_atom
[5] https://github.com/THUNLP-MT/dyMEAN
[6] https://github.com/zaixizhang/FAIR

Proof of E(3)-Equivariance of PocketGen

The E(3)-transformation on the euclidean coordinate $\boldsymbol{x} \in \mathbb{R}^3$ can be represented as: $T_g \cdot \boldsymbol{x} = \boldsymbol{O}\boldsymbol{x} + \boldsymbol{t}$, where $\boldsymbol{O} \in \mathbb{R}^3$ is the orthogonal transformation matrix, $\boldsymbol{t} \in \mathbb{R}^3$ is the translation vector. Implementing T_g on a coordinate matrix $\boldsymbol{X} \in \mathbb{R}^{n\times 3}$ means transforming each coordinate (i.e., each row) with T_g. PocketGen 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 PocketGen as $\{(\boldsymbol{b}_i, \boldsymbol{x}(\boldsymbol{b}_i))\}_{i=1}^m = p_\theta(\mathcal{A}\setminus\mathcal{B}, \mathcal{M})$, where $\{(\boldsymbol{b}_i, \boldsymbol{x}(\boldsymbol{b}_i))\}_{i=1}^m$ indicates the designed protein pocket seqeuce and structure. We have $\{(\boldsymbol{b}_i, T_g \cdot \boldsymbol{x}(\boldsymbol{b}_i))\}_{i=1}^m = p_\theta(T_g \cdot (\mathcal{A}\setminus\mathcal{B}), T_g \cdot \mathcal{M})$. Here, $T_g \cdot (\mathcal{A}\setminus\mathcal{B})$ and $T_g \cdot \mathcal{M}$ denote applying E(3)-transformation on the protein structures except for the pocket region and the molecular structure respectively.*

Then we prove the E(3)-equivariance of each module in PocketGen as follows.

Lemma 1. *Denote the bilevel attention module as $\{\boldsymbol{H}'_i, \boldsymbol{X}'_i\} = \mathrm{Att}(\{\boldsymbol{H}_i, \boldsymbol{X}_i\})$, then it is E(3)-equivariant. Namely, for any E(3)-transformation T_g, we have $\{\boldsymbol{H}'_i, T_g \cdot \boldsymbol{X}'_i\} = \mathrm{Att}(\{\boldsymbol{H}_i, T_g \cdot \boldsymbol{X}_i\})$.*

Proof. The key to the proof of Lemma 1 is to prove that the propagation in Eq. 2-9 is E(3)-invariant on $\boldsymbol{H}_i$ and E(3)-equivariant on $\boldsymbol{X}_i$. The correlation $\boldsymbol{R}_{ij}$ between block i and block j in Eq. 3 is E(3)-invariant because all the inputs, including the query $\boldsymbol{Q}_i$, the key $\boldsymbol{K}_j$, and the distance matrices $\boldsymbol{D}_{ij}$, are not influenced by the geometric transformation T_g. Therefore, we can immediately derive that the atom-level attention $\boldsymbol{\alpha}_{ij}$ in Eq. 4 is E(3)-invariant. Similarly, the residue/ligand-level attention β_{ij} in Eq. 6 is E(3)-invariant because it only operates on r_{ij} in Eq. 5 which aggregates $\boldsymbol{\alpha}_{ij}$. Finally, we can derive the E(3)-invariance on $\boldsymbol{H}$ and the E(3)-equivariance on $\boldsymbol{X}$ (ligand or pocket residues) as below:

$$
\begin{aligned}
\boldsymbol{H}'_i[p] &= \boldsymbol{H}_i[p] + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\phi_h(\boldsymbol{\alpha}_{ij}[p]\cdot\boldsymbol{V}_j),\\
T_g\cdot\boldsymbol{X}'_i[p] &= T_g\cdot(\boldsymbol{X}_i[p] + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\left(\boldsymbol{\alpha}_{ij}[p]\odot\phi_x(\boldsymbol{Q}_i[p]||\boldsymbol{K}_j||\mathrm{RBF}(\boldsymbol{D}_{ij}[p]))\right)\cdot\boldsymbol{X}_{ij}[p])\\
&= \boldsymbol{O}(\boldsymbol{X}_i[p] + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\left(\boldsymbol{\alpha}_{ij}[p]\odot\phi_x(\boldsymbol{Q}_i[p]||\boldsymbol{K}_j||\mathrm{RBF}(\boldsymbol{D}_{ij}[p]))\right)\cdot\boldsymbol{X}_{ij}[p]) + \boldsymbol{t}\\
&= (\boldsymbol{O}\boldsymbol{X}_i[p] + \boldsymbol{t}) + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\left(\boldsymbol{\alpha}_{ij}[p]\odot\phi_x(\boldsymbol{Q}_i[p]||\boldsymbol{K}_j||\mathrm{RBF}(\boldsymbol{D}_{ij}[p]))\right)\cdot\begin{bmatrix}\boldsymbol{O}(\boldsymbol{X}_i[p]-\boldsymbol{X}_j[1])\\ \vdots\\ \boldsymbol{O}(\boldsymbol{X}_i[p]-\boldsymbol{X}_j[n_j])\end{bmatrix}\\
&= (\boldsymbol{O}\boldsymbol{X}_i[p] + \boldsymbol{t}) + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\left(\boldsymbol{\alpha}_{ij}[p]\odot\phi_x(\boldsymbol{Q}_i[p]||\boldsymbol{K}_j||\mathrm{RBF}(\boldsymbol{D}_{ij}[p]))\right)\cdot\begin{bmatrix}\boldsymbol{O}\boldsymbol{X}_i[p]+\boldsymbol{t}-(\boldsymbol{O}\boldsymbol{X}_j[1]+\boldsymbol{t})\\ \vdots\\ \boldsymbol{O}\boldsymbol{X}_i[p]+\boldsymbol{t}-(\boldsymbol{O}\boldsymbol{X}_j[n_j]+\boldsymbol{t})\end{bmatrix}\\
&= T_g\cdot\boldsymbol{X}_i[p] + \sum_{j\in\mathcal{N}(i)} \beta_{ij}\left(\boldsymbol{\alpha}_{ij}[p]\odot\phi_x(\boldsymbol{Q}_i[p]||\boldsymbol{K}_j||\mathrm{RBF}(\boldsymbol{D}_{ij}[p]))\right)\cdot\begin{bmatrix}T_g\cdot\boldsymbol{X}_i[p]-T_g\cdot\boldsymbol{X}_j[1]\\ \vdots\\ T_g\cdot\boldsymbol{X}_i[p]-T_g\cdot\boldsymbol{X}_j[n_j]\end{bmatrix},
\end{aligned}
$$

which concludes the proof of Lemma 1. □

Lemma 2. *Denote the equivariant feed-forward network as as $\{\boldsymbol{H}'_i, \boldsymbol{X}'_i\} = \mathrm{FFN}(\{\boldsymbol{H}_i, \boldsymbol{X}_i\})$, then it is E(3)-equivariant. Namely, for any E(3)-transformation T_g, we have $\{\boldsymbol{H}'_i, T_g \cdot \boldsymbol{X}'_i\} = \mathrm{FFN}(\{\boldsymbol{H}_i, T_g \cdot \boldsymbol{X}_i\})$.*

Proof. The proof of Lemma 2 focuses on the single-atom updates in Eq. 10-13. First, it is easy to obtain the E(3)-equivariance of the centroid in Eq. 10:

$$T_g\cdot\boldsymbol{x}_c = T_g\cdot\mathrm{centroid}(\boldsymbol{X}_i) = \mathrm{centroid}(T_g\cdot\boldsymbol{X}_i).$$

Then we have the following equation on the relative coordinate $\Delta\boldsymbol{x}$ in Eq. 11:

$$\boldsymbol{O}\Delta\boldsymbol{x}_p = (\boldsymbol{O}\boldsymbol{X}_i[p]+\boldsymbol{t}) - (\boldsymbol{O}\boldsymbol{x}_c+\boldsymbol{t}) = T_g\cdot\boldsymbol{X}_i[p] - T_g\cdot\boldsymbol{x}_c.$$

Then we can obtain the E(3)-invariance of $\boldsymbol{r}_p$ in Eq. 11:

$$\boldsymbol{r}_p = \mathrm{RBF}(\|\Delta\boldsymbol{x}_p\|_2) = \mathrm{RBF}(\|\boldsymbol{X}_i[p]-\boldsymbol{x}_c\|_2) = \mathrm{RBF}(\|(\boldsymbol{O}\boldsymbol{X}_i[p]+\boldsymbol{t}) - (\boldsymbol{O}\boldsymbol{x}_c+\boldsymbol{t})\|_2) = \mathrm{RBF}(\|T_g\cdot\boldsymbol{X}_i[p]-T_g\cdot\boldsymbol{x}_c\|_2).$$

Finally we can derive the E(3)-invariance on $\boldsymbol{H}'[p]$ and the E(3)-equivariance on $\boldsymbol{X}_i'[p]$:

$$
\begin{aligned}
\boldsymbol{H}'[p] &= \boldsymbol{H}[p] + \sigma_h(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p), \\
T_g \cdot \boldsymbol{X}_i'[p] &= T_g \cdot (\boldsymbol{X}_i[p] + \Delta \boldsymbol{x}_p \sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p)) \\
&= \boldsymbol{O}(\boldsymbol{X}_i[p] + \Delta \boldsymbol{x}_p \sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p)) + \boldsymbol{t} \\
&= \boldsymbol{O}\boldsymbol{X}_i[p] + \boldsymbol{t} + \boldsymbol{O}\Delta \boldsymbol{x}_p \sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p) \\
&= T_g \cdot \boldsymbol{X}_i[p] + (T_g \cdot \boldsymbol{X}_i[p] - T_g \cdot \boldsymbol{x}_c)\sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p) \\
&= T_g \cdot \boldsymbol{X}_i[p] + (T_g \cdot \boldsymbol{X}_i[p] - \mathrm{centroid}(T_g \cdot \boldsymbol{X}_i))\sigma_x(\boldsymbol{H}_i[p], \boldsymbol{h}_c, \boldsymbol{r}_p),
\end{aligned}
$$

which concludes the proof of Lemma 2 □

The sequence refinement module with the pretrained protein language models and adapters operates on the scaler residue representations unaffected by the geometric transformation T_g. To sum up, with Lemma 1-2 at hand, it is obvious to deduce the E(3)-equivariance of the PocketGen.

Supplementary Figures

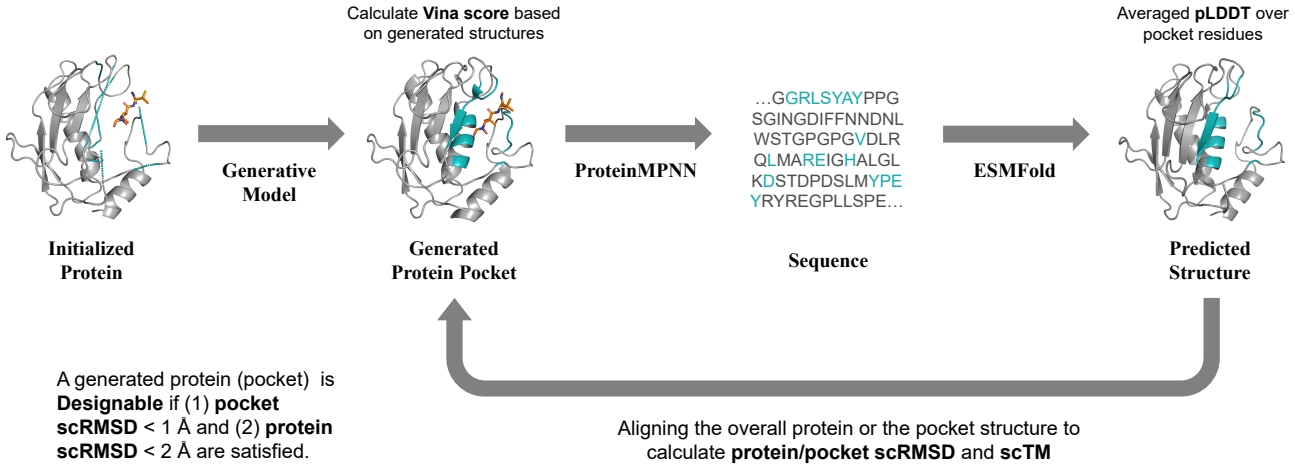


Figure S1. Schematic of computing metrics with respect to the generated pocket structures. black parts refer to the pocket, while the others are gray. The generative model refers to PocketGen or baseline methods leveraged to generate the protein pocket. The Vina score is calculated based on the generated protein-ligand complex. To calculate self-consistency scores, we first use ProteinMPNN to derive the residue sequence and then use ESMFold to predict the structure. By aligning the predicted structure with the generated structure, we can obtain the protein/pocket scRMSD. Similarly, we can obtain the scTM score of the protein structure. We also report the averaged pLDDT of the pocket residues from ESMFold.

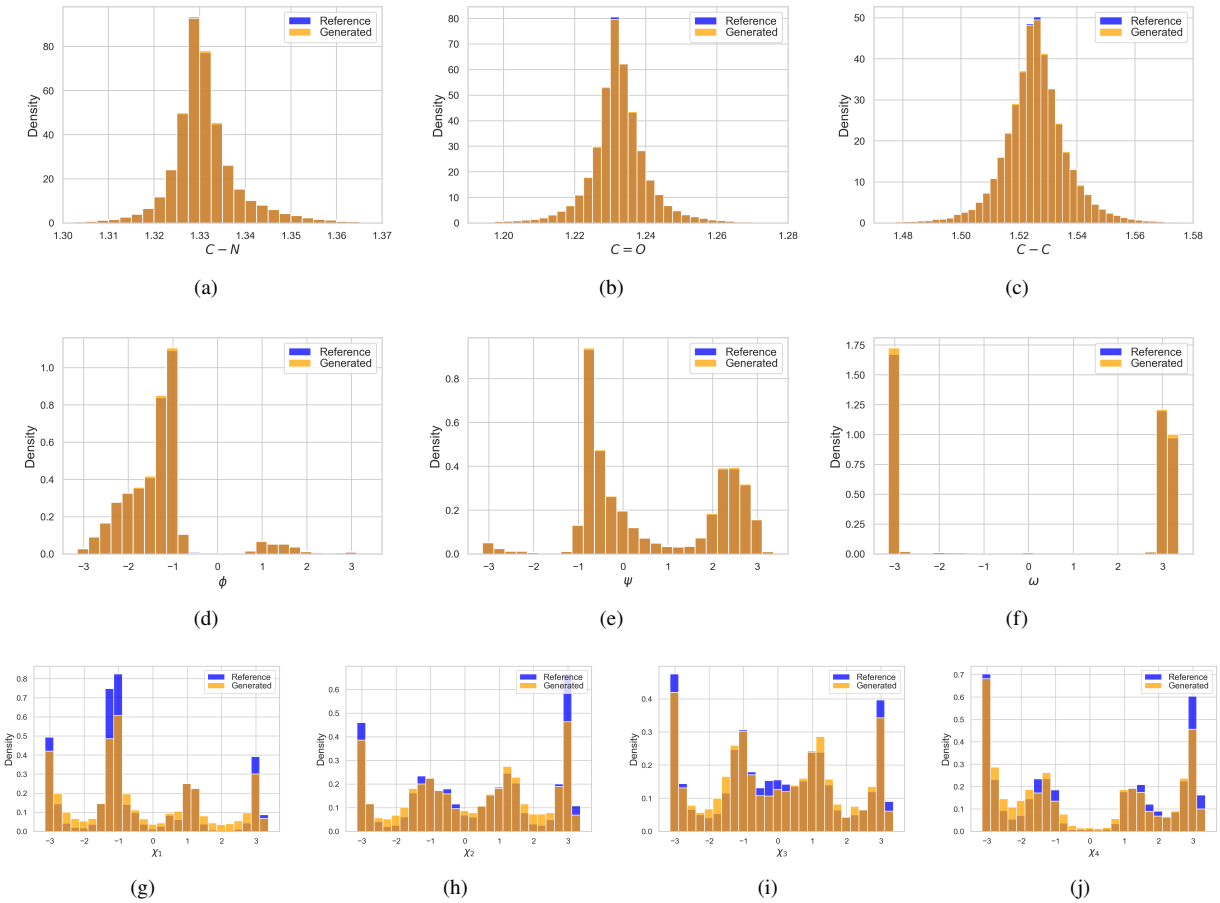
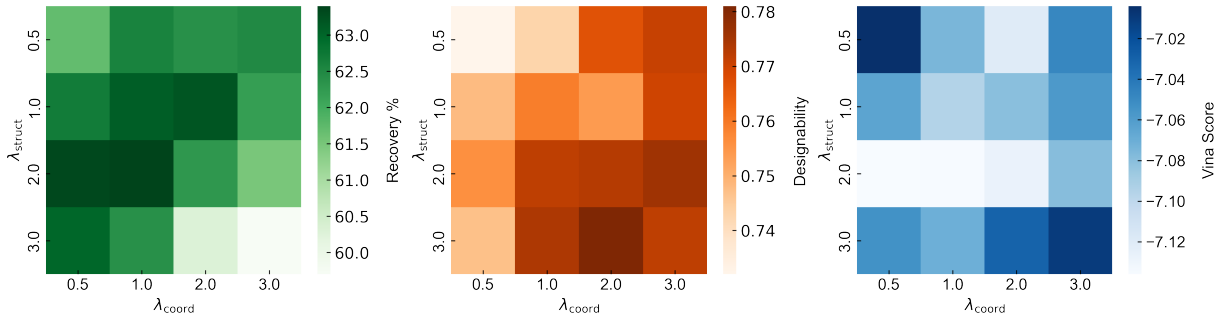


Figure S2. The bond length and dihedral angle distributions of the generated pockets and the reference (training set of CrossDocked dataset).



(a) AAR (b) Designability (c) Vina Score

Figure S3. Hyperparameter Analysis. Influence of hyperparameters (loss weights $\mathcal{L}_{\text{coord}}$ and $\mathcal{L}_{\text{struct}}$) on PocketGen's performance (AAR, Designability, and Vina Score) on the CrossDocked dataset. We perform a grid search over $\{0.5, 1.0, 2.0, 3.0\}$ and choose these hyperparameters based on the validation performance to select the specific parameter values. In the default setting, we set λ_{coord} to 1.0 and λ_{struct} to 2.0.

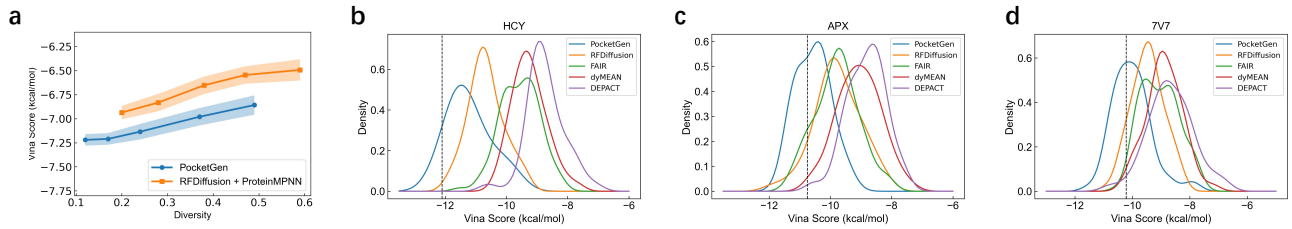


Figure S4. Additional results with RFDiffusion. a, The trade-off between quality (measured by Vina score) and diversity (1-average pairwise sequence similarity) of PocketGen. **b, c, d,** The pocket binding affinity distributions of PocketGen and baseline methods for three target molecules (HCY, APX, and 7V7, respectively).

Supplementary Tables

Table S1. The top 1/3/5/10 generated protein pocket (ranked by Vina score) with designability on the Binding MOAD dataset. The success rate measures the percentage of protein that the model can generate pockets with higher affinity than the reference ones in the datasets. We report the average plDDT of the predicted pocket, the scRMSD of the pocket backbone coordinates, and the scTM score of the whole protein. The plDDT, scRMSD, and scTM for PocketOpt are not reported, as PocketOpt keeps protein backbone structures fixed. We use to mark the results of affinity-related metrics, for pocket-structure related metrics, and for whole protein structure metrics.

Methods:	PocketOpt	DEPACT	dyMEAN	FAIR	RFDiffusion	RFDiffusionAA	PocketGen
Top-1 generated protein pocket							
Vina score (↓)	-9.828	-9.216	-9.352	-9.530	-9.901	-10.120	**-10.322**
Success Rate (↑)	0.89	0.78	0.77	0.84	0.90	0.92	**0.95**
pLDDT (↑)	-	83.1	83.2	83.5	85.0	**87.2**	87.1
scRMSD (↓)	-	0.642	0.622	0.620	0.581	**0.572**	0.575
scTM (↑)	-	0.908	0.911	0.922	0.937	**0.943**	0.942
Top-3 generated protein pocket							
Vina score (↓)	-9.403	-8.876	-8.971	-9.234	-9.589	-9.634	**-10.135**
pLDDT (↑)	-	82.7	82.1	82.8	84.9	85.9	**86.5**
scRMSD (↓)	-	0.645	0.620	0.596	0.580	0.574	**0.573**
scTM (↑)	-	0.903	0.908	0.916	0.932	0.937	**0.941**
Top-5 generated protein pocket							
Vina score (↓)	-9.260	-8.759	-8.842	-9.195	-9.478	-9.569	**-9.950**
pLDDT (↑)	-	81.9	81.9	82.3	85.1	85.8	**86.4**
scRMSD (↓)	-	0.639	0.628	0.612	0.587	0.583	**0.581**
scTM (↑)	-	0.905	0.910	0.915	0.936	0.936	**0.938**
Top-10 generated protein pocket							
Vina score (↓)	-8.981	-8.632	-8.690	-8.827	-9.268	-9.320	**-9.630**
pLDDT (↑)	-	81.6	81.7	82.7	84.5	85.3	**86.4**
scRMSD (↓)	-	0.644	0.626	0.592	0.589	0.586	**0.584**
scTM (↑)	-	0.909	0.906	0.918	0.924	0.930	**0.939**

Table S2. Substructure analysis of the generated molecules. We consider three covalent bonds in the backbone (C-N, C=O, and C-C), three conventional dihedral angles in the backbone (ϕ, ψ, ω[56]), and four dihedral angles in the side chains ($\chi_1, \chi_2, \chi_3, \chi_4$[57]). The KL divergence of the bond lengths and dihedral angles between the training set (CrossDocked) and the generated pockets are calculated following previous works[58,59]. Since PocketOpt sets the pocket backbone fixed, the corresponding backbone metrics are not calculated.

Methods	C-N	C=O	C-C	ϕ	ψ	ω	χ_1	χ_2	χ_3	χ_4
PocketOpt	-	-	-	-	-	-	**0.140**	0.098	0.071	0.104
DEPACT	0.010	0.011	0.009	0.033	0.032	0.097	0.190	0.126	0.109	0.148
dyMEAN	0.012	0.012	0.022	0.036	0.017	0.078	0.224	0.187	0.130	0.125
FAIR	0.014	0.011	0.020	0.025	0.024	0.080	0.235	0.067	0.110	0.167
RFDiffusion	0.009	0.006	0.006	0.027	0.016	0.066	0.160	0.092	0.073	**0.060**
RFDiffusionAA	0.008	0.007	**0.005**	**0.025**	0.014	**0.062**	0.159	0.087	0.071	0.064
PocketGen	**0.006**	**0.004**	**0.005**	0.029	**0.013**	0.077	0.158	**0.053**	**0.065**	0.097

Table S3. Ablations studies of PocketGen. **PocketGen w/o ligand update** indicates the ligand structure is fixed during the generation of pockets by PocketGen. **PocketGen w/o residue-level attention** denotes only the basic atom-level attention performed in the bilevel attention module. **PocketGen w/o pLM** means no pretrained protein language model is leveraged for sequence refinement. We report the means and standard deviations over three different runs (%). The best results are bolded.

Model	CrossDocked			Binding MOAD		
	AAR (↑)	Designability (↑)	Vina (↓)	AAR (↑)	Designability (↑)	Vina (↓)
PocketGen w/o ligand update	59.20±1.56%	0.75±0.03	-6.838±0.10	59.61±2.13%	0.77±0.05	-7.865±0.12
PocketGen w/o residue-level attention	58.19±1.78%	0.66±0.04	-6.924±0.08	57.30±2.28%	0.67±0.04	-7.805±0.10
PocketGen w/o pLM	42.70±1.45%	0.71±0.05	-6.894±0.07	44.23±1.31%	0.69±0.06	-7.787±0.12
PocketGen	**63.40±1.64%**	**0.77±0.02**	**-7.135±0.08**	**64.43±2.35%**	**0.80±0.04**	**-8.112±0.14**

Table S4. Hyperparameters for the baselines and our PocketGen.

hyperparameter	value	description
PocketOptimizer		
n_confs	50	The number of sampled ligand conformers.
n_poses	10000	The max number of ligand poses to sample.
ligand_scaling	1.0	The ligand scaling factor.
vdw_filter_thresh	100	The energy threshold for filtering rotamers (kcal/mol).
DEPACT		
interaction_threshold	-1.0	The threshold of ligand-subpocket interaction score for filtering.
mathcing_threshold	1.5 Å	The RMSD cutoff for finding geometrically consistent ways of placing the seeding residues.
num_CCSS	1000	The number of common chemical substructure (CCSS).
weight_seed	100	The weight of seeding residues in the scoring function.
FAIR		
hidden_size	128	Size of the hidden states in its hierarchical message passing network (MPN).
num_heads	4	Number of attention heads.
n_atom_layers	6	Number of atom-level layers in the MPN.
n_residue_layers	2	Number of residue-level layers in the MPN.
k_atom_neighbors	24	Number of neighbors for each node in the atom KNN graph.
k_residue_neighbors	8	Number of neighbors for each node in the residue KNN graph.
n_backbone_iter	5	Number of iterations in backbone refinement.
n_fullatom_iter	10	Number of iterations in full atom refinement.
RFDiffusion		
potential_scale	1.0	Scale of the auxiliary substrate contact potential.
temp_ProteinMPNN	0.1	The temperature in ProteinMPNN for sequence inference.
n_steps	50	Number of the diffusion steps.
RFDiffusionAA		
temp_ProteinMPNN	0.1	The temperature in LigandMPNN for sequence inference.
n_steps	50	Number of the diffusion steps.
dyMEAN		
embed_size	64	Size of the residue type embedding and the position number embedding.
hidden_size	128	Size of the hidden states in the MPN
n_layers	3	Number of layers in the MPN
n_iter	3	Number of iterations in the progressive full-shot decoding.
k_neighbors	9	Number of neighbors for each node in the KNN graph.
d	16	Size of the attribute vector of each channel (equal to the size of the atom type
PocketGen		
hidden_size	128	Size of the hidden states in the MPN
n_layers	4	Number of layers in the MPN
num_heads	4	Number of attention heads.
n_iter	3	Number of refinement rounds.
k_sparse	3	Number of elements to keep in the sparse attention.
k_neighbors	8	Number of neighbors for each node in the KNN graph.