

Appendices

This supplementary material is organized as follows:

1. Section A includes a broader impact statement.
2. Section B includes model and implementation details.
3. Section C provides some additional results for antibody CDR redesign.
 - Section C.1 presents some additional in silico results
 - Section C.2 presents our in vitro results.
4. Section D explain how we inferred non canonical amino acids without OpenBabel.
5. Section E presents some additional results for macro-cyclic peptide generation
 - Section E.1 presents some additional results.
 - Section E.2 presents how we curated the dataset for training FuncBind.
 - Section E.3 presents the train / val / test splitting logic.

A Broader impacts

This work introduces FuncBind, a novel framework for all-atom, structure-conditioned molecular generation, whose primary positive impact lies in its potential to accelerate and enhance the discovery of new therapeutics across diverse modalities, from small molecules to complex biologics like peptides and antibodies. While the underlying principles could find applications in other scientific fields like materials science, its deployment in drug discovery requires addressing significant challenges, including the validation gap between in silico predictions and experimental testing (in vitro, in vivo, and clinical trials).

B Model details

B.1 Representation

FuncBind is based on a neural field representation that models an atomic density field, a smooth function taking values between 0 (far away from all atoms) and 1 (at the center of atoms). This field takes the following form [50, 51]:

$$\forall x \in \mathbb{R}^3, v_a(x) = 1 - \prod_{i=1}^{n_a} \left(1 - \exp\left(-\left(\frac{\|x - x_{a_i}\|}{.93r}\right)^2\right) \right), \quad (4)$$

where a_i is the i^{th} atom of type a (among n choices), for a total of n_a atoms and r is the atoms' radius set to $r = 1.0\text{\AA}$ for all atom types.

We consider $n = 8$ element types C, O, N, S, F, Cl, P, Br that cover all major atom types across small molecule, macrocyclic peptides and proteins. Note that protein-specific atom types (*e.g.* C_α, C_β etc.) are merged into a single element type (*e.g.* C). This helps transfer learning across modalities.

Finally, the field is defined over a volume of $(32\text{\AA})^3$. It is the continuous version of a voxel grid of spatial dimension 128 and resolution of 0.25\AA , in $\mathbb{R}^{8 \times 128^3}$.

B.2 Neural Field

The encoder E_ψ is a 3D CNN containing 4 residual blocks (number of hidden units 256, 512, 1024, 2048 for each block), where each block contains 3 convolutional layers followed by BatchNorm, ReLU and pooling layers (we use max pooling on the first three blocks). The encoder has 59M parameters. The input to the encoder is a low-resolution grid of spatial grid dimension $L = 16$ corresponding to a resolution of 2\AA . Before voxelizing the molecules, we first center the atoms around the tightest bounding box encapsulating the molecule, apply a random rotation to the atoms

(each Euler angle rotated randomly between $[0, 2\pi)$) and random translation between $[-1, 1]\text{\AA}$ then normalize their coordinates to the range of $[-1, 1]$.

The decoder D_ϕ is a conditional Multiplicative Filter Network (MFN) [57, 58] with Gabor filters and 6 FiLM-modulated layers, where each fully-connected layer has 2048 hidden units. The decoder has 59M parameters.

The auto-encoder is trained with Adam Optimizer [86] with learning rate 10^{-2} , $\beta_1 = 0.9$, $\beta_2 = 0.999$. We apply a KL regularization weight of $\lambda = 10^{-5}$. Batch size is 32 over 1 B200 GPU; we sample 15000 coordinates per batch.

B.3 Denoiser

The hyperparameters of the Karras et al. [62] UNet architecture are as follows: 5B model with 8 blocks with 512 channels, channel multipliers [1,2,3,4], attention resolutions [4, 2]. This model follows the XXL setting of EDM2 [62] and increased the number of channels from 448 to 512. For reference, Protéina [63], the largest protein backbone generative model, comprises 400M parameters and RFDiffusion [8] roughly 100M parameters. As [65], we apply preconditioning to learn the denoiser across noise levels. Moreover, we sample the noise levels from a log-normal distribution with mean 1.2 and standard deviation 0.8.

The target encoder $E_{\psi'}$ is a 3D CNN which takes as input a voxel grid of dimension $\mathbb{R}^{4 \times 32^3}$ of the target protein (resolution 1.0\AA), considering atom elements C, O, N, S , and consists of a magnitude preserving [62] CNN layer with kernel $3 \times 3 \times 3$ and output channels 64, then a downsampling layer to a spatial grid in $\mathbb{R}^{64 \times 16^3}$ then a magnitude preserving U-Net block [62] with output channels $C = 128$ leading to a voxel of size $\mathbb{R}^{C \times 16^3}$.

The parameters are optimized with Adam optimizer [86] with learning rate $\alpha_{\text{ref}} = 10^{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.95$ using an aggregated batch size of 768 over 8 B200 GPUs. We perform early stopping on the validation loss. We use the power function exponential moving average from EDM2 [62] with an EMA length of 5%. Moreover, we adopt the inverse square root decay schedule of [86], also used in [62] which sets $\alpha(t) = \frac{\alpha_{\text{ref}}}{\sqrt{\max(t/t_{\text{ref}}, 1)}}$, where we set $t_{\text{ref}} = 20040$. Finally, the networks are trained by randomly dropping the conditioning information 10% of the time.

B.4 Sampling

To improve uniqueness, we apply different rotations to the pocket on each MCMC chain, in a similar fashion to how rotation-based data augmentation is performed at training time.

B.4.1 Denoising diffusion

We set as follows the sampling parameters of EDM2 [62]:

- $N = 128$ steps
- $\sigma_{\min} = 0.01$, $\sigma_{\max} = 10$
- $S_{\min} = 5.0$, $S_{\max} = 7.0$
- $S_{\text{churn}} = 30.0$
- $S_{\text{noise}} = 1.003$
- $\rho = 7$

Moreover, we apply a temperature scaling with $\tau = 0.5$ on Crossdocked and $\tau = 0.33$ on MCP-protein complexes.

B.4.2 Walk-Jump sampling

We also implemented a conditional form of the Walk-Jump Sampling (WJS), a score-based generative model that is based on a probabilistic formulation of least-squares denoising [18]. The framework is based on the Tweedie-Miyasawa formula (TMF) [66, 67], which relates the least-squares denoiser

at a noise level σ with the score function at the noise level. Given $y = z + \sigma\varepsilon, \varepsilon \sim \mathcal{N}(0, I_d)$, the conditional extension of TMF was derived in [7]. In our notation, it takes the form:

$$\nabla_y \log p(y | z^{\text{tar}}, \sigma, c) \approx s_\theta(y | z^{\text{tar}}, \sigma, c) := (\hat{z}_\theta(y | z^{\text{tar}}, \sigma, c) - y)/\sigma^2, \quad (5)$$

where \hat{z}_θ is the minimizer of Equation (2); $s_\theta(y | z^{\text{tar}}, \sigma, c)$ is the learned conditional score function.

Walk-jump sampling. WJS [18] is composed of two stages: (i) (*walk*) samples the noisy latent variables conditioned on z^{tar}, c using Langevin Markov chain Monte Carlo (MCMC) via the learned score function (Equation (5)), (ii) (*jump*) estimates “clean” z by single-step denoising. There is a fundamental trade-off in this sampling strategy: for larger σ , sampling from the smoother density becomes easier, but the denoised samples move farther away from the distribution of interest [87].

Multimeasurement walk-jump sampling. The sampling trade-off in WJS is addressed in multimeasurement denoising models [88, 87], in which the problem is framed as sampling from the distribution $p_\sigma(y_{1:m})$ associated with $y_{1:m} := (y_1, \dots, y_m)$, where $y_k = z + \sigma\varepsilon_k, k \in \{1, \dots, m\}$, and $\varepsilon_k \sim \mathcal{N}(0, I_d)$ all independent of z . Saremi et al. [87] studied a sequential scheme for sampling from $p_\sigma(y_{1:m})$ and showed that the noise level effectively decreases (as far as the denoiser is concerned) at the rate σ/\sqrt{m} . Furthermore, it was shown that sampling becomes easier upon accumulation of measurements. The general sampling problem is therefore mapped to a sequence of sampling noisy data at a *fixed* noise scale, while the effective noise decreases via accumulation of measurements. We refer to this scheme as WJS- m , which involves two hyperparameters: the noise level σ , and the number of measurements m . In this construction, we only need to keep track of the empirical mean of noisy samples. In particular, we have (see [87, Eq. 4.9]):

$$\nabla_{y_m} \log p_\theta(y_m | y_{1:m-1}, z^{\text{tar}}, \sigma, c) = \frac{1}{m} s_\theta(\bar{y}_{1:m} | z^{\text{tar}}, \frac{\sigma}{\sqrt{m}}, c) + \frac{1}{\sigma^2} (\bar{y}_{1:m} - y_m),$$

where $\bar{y}_{1:m}$ is the empirical mean of the measurements (y_1, \dots, y_m) . The score function above is used in sampling (y_1, \dots, y_m) iteratively using Langevin MCMC [87, Algorithm 1]. Finally, the denoising “jump” in WJS- m is achieved via (single-measurement) TMF using the sufficient statistics $\bar{y}_{1:m}$ at the noise scale σ/\sqrt{m} . It is clear that the vanilla WJS discussed above reduces to WJS-1. Although there is a flavor of diffusion in this scheme due to its sequential strategy, WJS- m is arguably more “surgical” in that, by construction, we do not need to learn score functions over a continuum of noise levels, but only a finite one identified by m . This is especially appealing for applications where WJS-1 already shows reasonable performance and m is therefore taken to be small. This work contains the first experimental validation of WJS- m in generative modeling applications.

We report the results for WJS for CDR H3 inpainting in Section C.1.2. The parameters are set to $\sigma = 7.0$ and $m = 16$. We use underdamped Langevin MCMC from Sachs *et al.* [89] in the BAOAB scheme with $K = 50$ steps, friction $\gamma = 1.0$, discretization step $\delta = \sigma/2$ ([87, Algorithm 1]).

C CDR redesign

C.1 In silico evaluation

C.1.1 Uniqueness

Table 4 reports our CDR sampling results over unique sequences.

We observe that higher uniqueness usually leads to lower Amino Acid Recovery (AAR). In other words, repeated sequences tend to correlate more with the seed. We use this simple heuristic to select H3 designs for in vitro evaluation (see Section C.2).

C.1.2 Ablation with Walk Jump Sampling

We report the performance of multimeasurement WJS and diffusion. Overall the models are comparable with slightly higher uniqueness for diffusion.

C.1.3 Comparison to specialized model

We observe that the unified model has higher uniqueness than the specialized model, with slightly better CDR loop inpainting performance on unique samples. We observe this trend on the other data modalities as well.

Table 4: Impact of uniqueness on CDR inpainting performance. RMSD is in Å and AAR and Uniqueness are in %. * indicates designs with *unique* sequences.

Method	AAR \uparrow	RMSD \downarrow	Unique \uparrow	AAR \uparrow	RMSD \downarrow	Unique \uparrow
	H1			L1		
AbDiffuser	76.3	1.58	-	81.4	1.46	-
FuncBind	86.9	0.41	23.5	86.4	0.68	38.9
FuncBind*	75.9	0.45	100	79.3	0.85	100
	H2			L2		
AbDiffuser	65.7	1.45	-	83.2	1.40	-
FuncBind	78.2	0.52	20.6	86.2	0.83	19.4
FuncBind*	59.5	0.57	100	54.4	2.39	100
	H3			L3		
AbDiffuser	34.1	3.35	-	73.2	1.59	-
FuncBind	47.5	2.04	85.5	80.8	0.68	44.1
FuncBind*	44.1	2.16	100	68.9	0.94	100

Table 5: Diffusion vs WJS on H3 loop inpainting. * indicates designs with *unique* sequences.

Method	AAR \uparrow	RMSD \downarrow	Unique \uparrow
FuncBind _{diff}	47.5	2.04	85.5
FuncBind* _{diff}	44.1	2.16	100
FuncBind _{WJS-16}	51.0	1.89	73.8
FuncBind* _{WJS-16}	41.4	2.18	100

Table 6: Loop uniqueness comparison between Unified and Specialized models.

Loop	Uniqueness Unified	Uniqueness Specialized
H1	23.5	9.6
H2	20.6	12.6
H3	85.5	69.2
L1	38.9	10.6
L2	19.4	14.0
L3	44.1	22.3

Table 7: H3 loop performance on unique samples.

H3 loop	AAR	RMSD
Unified	0.441	2.16
Specialized	0.406	2.07

C.2 In vitro validation

To experimentally validate FuncBind’s capabilities, we performed wet-lab validation of H3 loop redesigns based on the co-crystal structure of an antibody bound to a rigid and a flexible epitope. We selected H3 loop redesign due to H3’s important contribution to the antibody’s functional properties. This study was conducted in a de novo setting: interfaces similar to the two complexes, identified using Ab-Ligity [83], were excluded from training.

From an initial pool of 10,000 unique generated H3 designs (all matching the original seed’s length), 190 (*i.e.* 2 SPR plates) were selected for experimental testing. This selection involved two steps:

1. The top 500 designs were shortlisted based on model confidence, as indicated by their repeated generation counts. Our in-silico validation showed that repeats is an useful proxy for high amino acid recovery of the seed.
2. These 500 designs were then clustered into 190 groups using weighted K-means based on sequence edit distance, where the weights were defined by the repeat generation count. The design with the highest repeat count (highest confidence) from each of these 190 clusters was chosen for synthesis and characterization.

The selected antibody designs were expressed and purified in the wet lab. Binding affinity was then determined using surface plasmon resonance (SPR) measurements.

C.2.1 Rigid epitope

An analysis of Amino Acid Recovery (AAR) and Root Mean Square Deviation (RMSD) for the selected designs are provided in Figure 6:

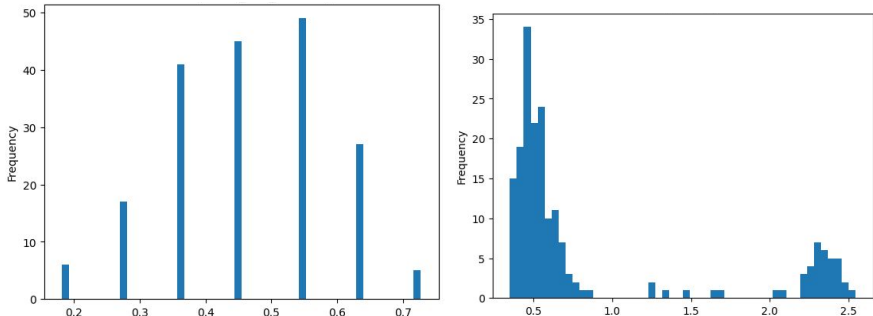


Figure 6: All generated CDR H3 designs on rigid epitope: AAR (left) and RMSD (right) histogram.

FuncBind successfully generated novel antibody binders in this de novo redesign problem; in fact 54% were binders (42% with pKD values). 94% of all 190 submitted designs were successfully expressed and purified. Experimental results confirmed that 42% of all submitted designs were binders, with pKD values in the range of [7.55, 11.29] ($K_D \in [5.08 \times 10^{-12}, 2.84 \times 10^{-8}]M$) and an average pKD of 9.56 ($K_D = 2.00 \times 10^{-9}M$). For comparison, the pKD of the parent antibody is 10.20 ($K_D = 2.63 \times 10^{-11}M$). 12% were binders with no pKD ("bad" binders). We identified a 5X binder in that set.

Table 8: Average RMSD and AAR for binders and non binders on rigid epitope

	Binders	Unassigned	Non-Binders	Global
RMSD	0.50	0.53	1.31	0.88
AAR	55.1	50.8	37.9	46.7

Table 9: Average RMSD and AAR for binders and non binders on flexible epitope

	Binders	Unassigned	Non-Binders	Global
RMSD	1.93	2.08	1.94	1.95
AAR	40.4	35.6	34.2	34.5

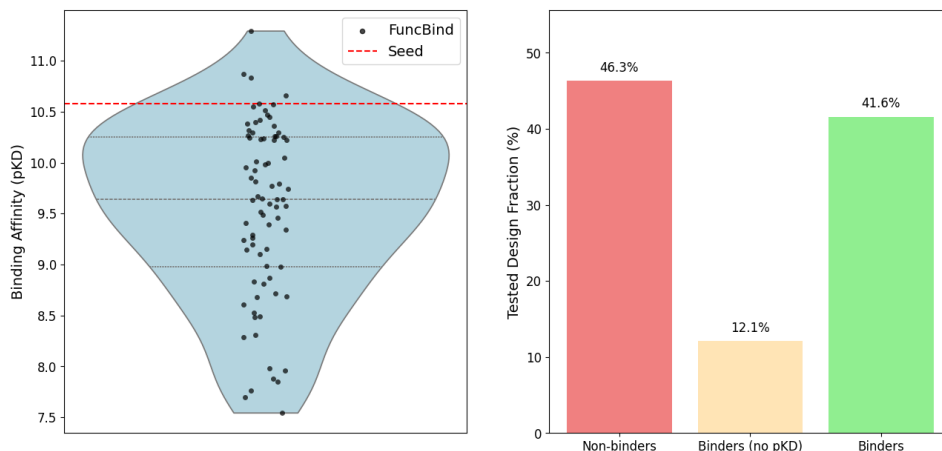


Figure 7: In vitro validation of FuncBind's designs against a rigid epitope; expression (left), binding affinity (center), binding rate (right). Expression rate is 93.68%.

C.3 Flexible epitope

An analysis of Amino Acid Recovery (AAR) and Root Mean Square Deviation (RMSD) for the selected designs are provided in Figure 8:

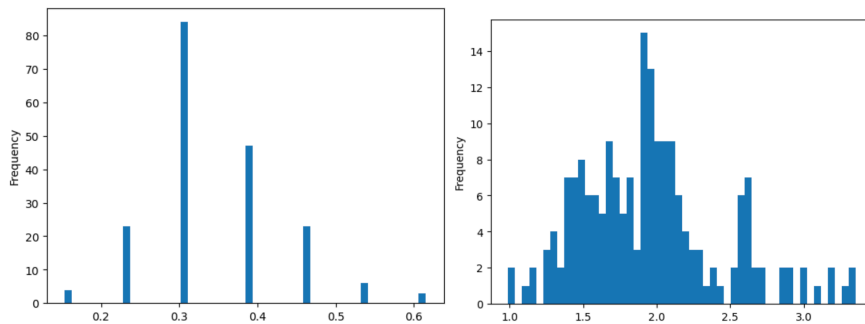


Figure 8: All generated CDR H3 designs against a flexible epitope; target: AAR (top left) and RMSD (top right) histogram. Bottom: Logo of generated designs.

Experimental results confirmed that 100% of the designs expressed and 2% of all submitted designs were binders, with pKDs [6.95, 7.44, 8.08, 8.47] ($K_D \in [3.38 \times 10^{-9}, 1.13 \times 10^{-7}]M$) and an average pKD of 7.74 ($K_D = 4.03 \times 10^{-8}M$). For comparison, the pKD of the parent antibody is 10.58 ($K_D = 6.32 \times 10^{-11}M$). 10% were binders with no pKD ("bad" binders); around 3 Unassigned binders had a very reasonable SPR curves.

Looking at Table 9, no specific correlation between binders and non binders were found based on RMSD on the limited set of binders we had. Though higher AAR seemed to be better (based on 4 binder samples only).

Table 10: Per-residue TS between the (a) seed MCP and (b) crystal MCP of the 20 sampled molecules in the pocket relaxed to 1vwe mutant 90.

(a)	CYS	HIS	A20	B67	PHE	CYS
1vwe_mut90	0.63 ± 0.24	0.24 ± 0.17	0.26 ± 0.07	0.14 ± 0.12	0.29 ± 0.14	0.53 ± 0.21
(b)	CYS	HIS	PRO	GLU	PHE	CYS
1vwe-CP	0.61 ± 0.23	0.24 ± 0.12	0.31 ± 0.14	0.28 ± 0.10	0.33 ± 0.08	0.57 ± 0.23

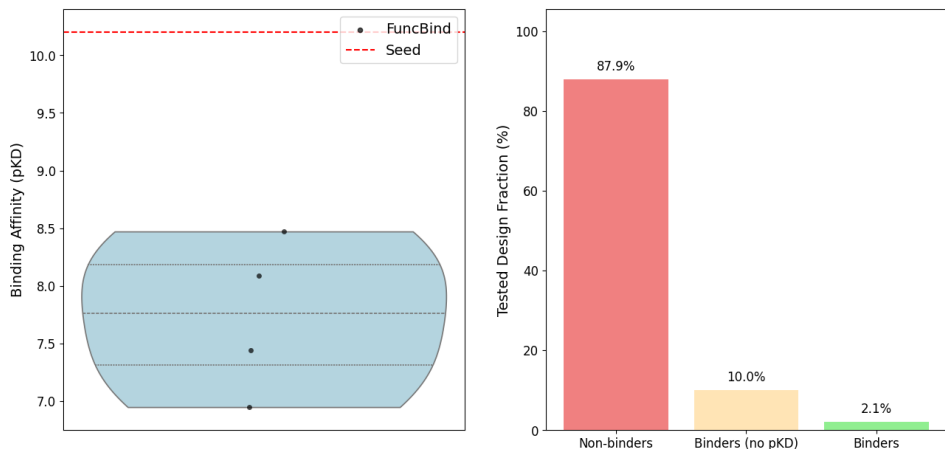


Figure 9: In vitro validation of FuncBind’s designs against a flexible epitope; expression (left), binding affinity (center), binding rate (right). Expression rate is 100%.

D Identifying non canonical amino acids

Unidentified amino acids were determined by recognizing repeated patterns of peptide backbone atoms around a chiral carbon. All atoms stemming from a C_α , including those in the side chain, were identified and labeled per canonical atom naming conventions. SMILES strings of unidentified amino acids were compared with a non-canonical amino acid library, assigning residue names when a match was found. If no match was found, the amino acid was labeled as unknown and added to the non-canonical library. In the output PDB file, a canonical residue name was assigned based on the closest alignment of atom naming patterns (*e.g.*, C_γ , C_{δ_1} , N_{ϵ_1}) to a known canonical residue.

E Macro cyclic peptides

E.1 Additional results

We perform the following studies:

- Figure 10: qualitative comparison between some MCP samples and the seed.
- Table 10: per residue tanimoto similarity (TS) for molecules samples with the seed mutant (a) and the crystal MCP (b).
- Figure 11 shows the categorization of the generated amino acids.
- Figure 12 shows some reasonable and novel generated non canonical amino acids.
- Figure 5 shows how a newly generated amino acid interacts with pocket residues that neither the seed nor a chemically similar amino acid at the same position engage.

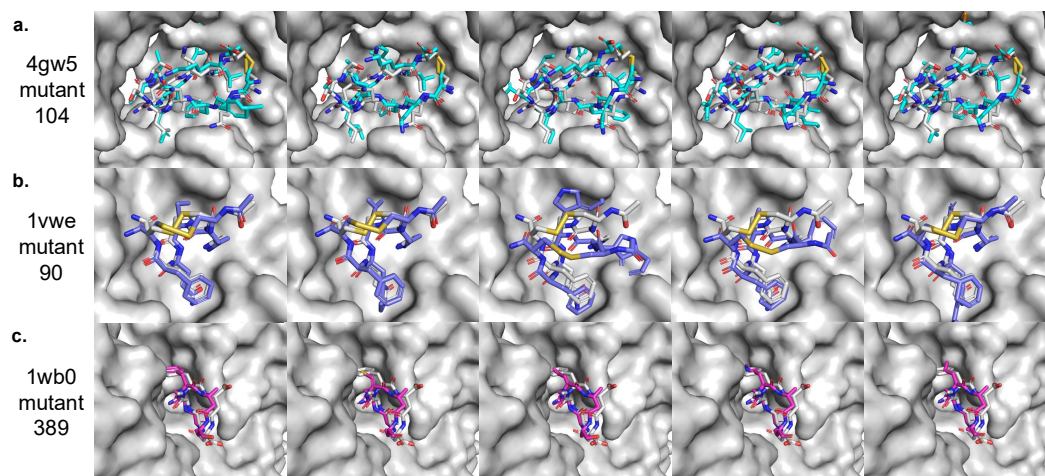


Figure 10: Results of sampling in pocket relaxed around MCP seed (grey) are in (a) blue in 4gw5 mutant 104 pocket, (b) purple for 1vwe mutant 90 pocket, and (c) 1wb0 mutant 389 pocket

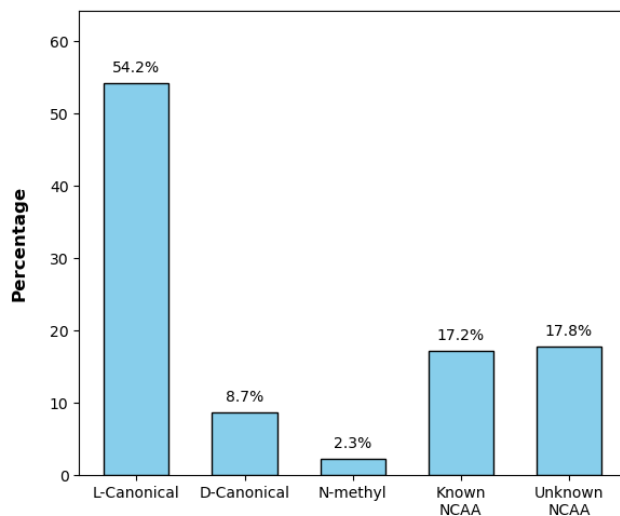


Figure 11: Proportion of amino acid types classified as L-canonical, D-canonical, N-methylated, other known non-canonical amino acids (as annotated in our library), and unknown non-canonical amino acids in the sampled set.

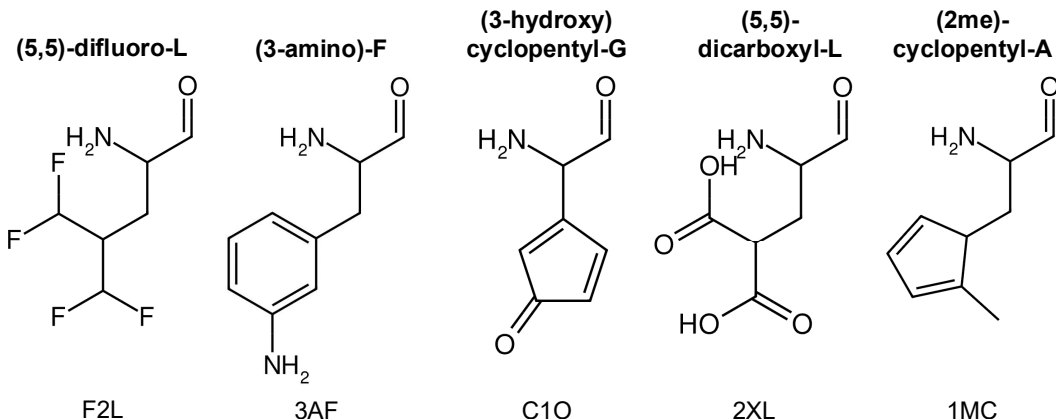


Figure 12: Examples of unknown or novel NCAAs that appeared in the sampled set but were not present in the initial test set library. Novel NCAAs are labeled with a formal name and an assigned 3-letter AA code.

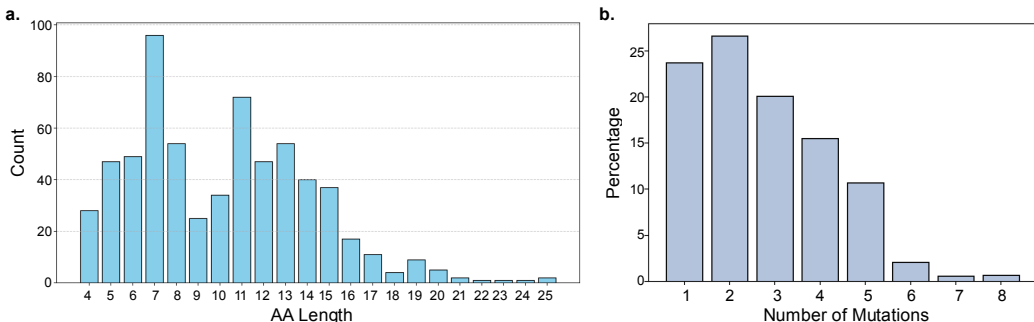


Figure 13: (a) Count of MCPs with each amino acid length in the source MCP-protein dataset. (b) Percentage of the number of mutations in the curated MCP-protein bound dataset.

E.2 Data curation

The MCP-protein pair dataset was curated by randomly mutating the MCPs from the source dataset at 1–8 different sites, using a list of 213 distinct amino acids (Figure 13b). The closure bonds in the source dataset consist of 55% N-to-C (head-to-tail), 28% disulfide (cysteine-cysteine), and 23% S-acetyl-cysteine. The remaining 4% contain other closure bonds, such as linkers, and were mainly avoided or modified in the curated MCP dataset so that mutations can be easily implemented in Rosetta. Mutations were avoided in amino acids involved in disulfide bonds and S-acetyl-cysteine linkage. The amino acid list used for random mutation of the MCPs included L-canonical, D-canonical, N-methylated, and other non-canonical types—such as alpha-modified, beta-modified, and peptoid amino acids. Many of these non-canonical residues were pre-parameterized and available in the Rosetta non-canonical rotamer libraries [90]. Following mutation, the MCPs were relaxed using the fast-relax protocol, which involves iterative cycles of side-chain packing and all-atom minimization [84]. The Rosetta interface energy scores—representing the binding energy of the protein-peptide complex at each position—were calculated using the `ref_2015_cart` energy function. From each source MCP-protein structure, over 2,000 mutated and relaxed complexes were generated, and approximately 500 with the lowest Rosetta interface scores were selected for the curated dataset. Therefore, we were able to expand the source dataset to 186,685 total MCP-protein structures in the curated dataset.

E.3 Clustering and Splitting

The pairwise similarity of protein sequences from the 641 protein targets in the curated dataset was evaluated using the Longest Common Subsequence (LCS) method [91]. Similarity between each

pair of sequences was calculated as the ratio of the length of their LCS to the length of the longer sequence. The target proteins were clustered together under a representative if their similarity score was greater than 0.5. If none of the similarity scores met the 0.5 threshold, a new cluster was created with that protein as the representative. 208 distinct protein clusters were used for training, validation, and testing. Clusters containing more than 100 MCP-protein pair structures in total were included in the training set. From the remaining clusters, 100 were randomly assigned to the test set, while the rest were added to the validation set.