Inpainting Protein Sequence and Structure with ProtFill

Elizaveta Kozlova[1,2] **Arthur Valentin**[1,2,3] **Daniel Nakhaee-Zadeh Gutierrez**[1]
[1]Adaptyv Biosystems [2]EPFL [3]Columbia University
{liza,arthur,daniel}@adaptyvbio.com

Abstract

Designing new proteins with specific binding capabilities is a challenging task that has the potential to revolutionize many fields, including medicine and material science. Here we introduce ProtFill, a novel method for the simultaneous design of protein structures and sequences. Employing an $SE(3)$ equivariant diffusion graph neural network, our method excels in both sequence prediction and structure recovery compared to SOTA models. We incorporate edge feature updates in GVP-GNN message passing layers to refine our design process. The model's applicability for the interface redesign task is showcased for antibodies as well as other proteins. The code is available at `https://github.com/adaptyvbio/ProtFill`.

1 Introduction

Proteins play a pivotal role in biological research, serving as the workhorses of the cellular machinery. Their functionality is intricately tied to their structure, as the arrangement of atoms in 3D space, and their sequence, as unique order of amino acids. The majority of recent computational protein design approaches have historically been tailored to translate between those two representations [4, 15]. However, many applications require simultaneous design of both sequence and structure to achieve desired protein functionality. This modality of protein design, known as co-design, aims to learn the relationship between sequence and structure and optimize both representations concurrently. In practice, co-design is often performed through training separate models for sequence and structure generation [25] [4], which are then stacked. However, recently, several models have been proposed that integrate co-design into a single architecture. These approaches are particularly well-suited for the task of protein inpainting, which involves filling in the missing regions of a known protein. This can be a useful set-up when trying to redesign the interface of a protein complex or improve the affinity of antibody to its target antigen. Despite the prevalence of antibody-specific inpainting research [12, 11, 6], there is a scarcity of frameworks for diverse protein interfaces. We propose ProtFill, a general-purpose $SE(3)$ equivariant graph neural network (GNN) for protein co-design. Protfill is based on a modification of GVP-GNN [14] that includes continuously updated residue pair representations. On an antibody dataset, the architecture outperforms existing alternatives in sequence design and achieves comparable results in structure prediction. We also merge ProtFill with a diffusion framework. In relation to other diffusion co-design models, it stands apart from multi-process approaches like Anand and Achim [1] and demonstrates broader applicability than models like diffAb [20] and ProteinGenerator [19].

2 Methods

2.1 Task formulation

ProtFill predicts the coordinates of the backbone atoms (C, C_α, N, O) and the amino acid identities jointly. Due to properties of peptide bonds, structure prediction of the backbone atoms can be closely

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

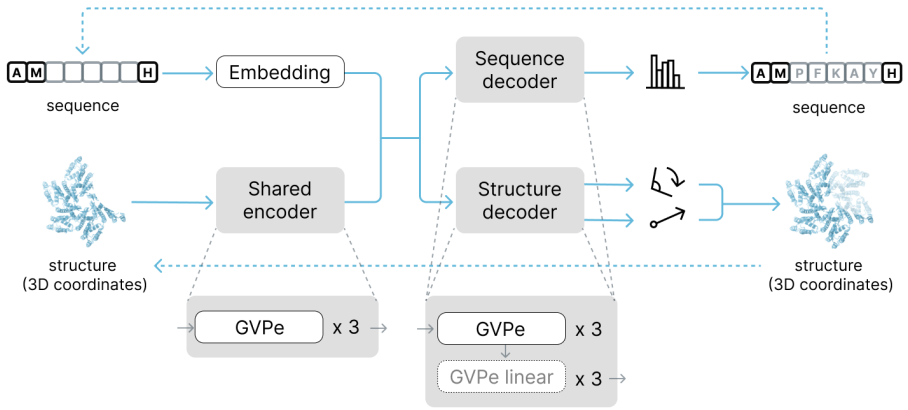

Figure 1: The architecture consists of a single encoder that creates a structure embedding and two decoders that generate structure and sequence. The encoder consists of three GVPe (GVP with edge updates) layers and the decoder is structured the same, with an optional addition of three linear graph GVPe layers at the end.

approximated with six degrees of freedom and divided into two tasks: predicting the coordinates of the C_α atom and the orientation of the C-C_α-N triangle. Overall, the co-design network has to return three different outputs for each masked amino acid: a set of three 3D rotation angles in a local coordinate system, a C_α translation vector relative to the model input positions, and a probability distribution over amino acid identities.

2.2 Network architecture

The model consists of one shared encoder and two decoders (one for sequence and one for structure outputs). Both the encoder and the decoders consist of three stacked message passing layers. The encoder generates a structure embedding which is concatenated with a simple sequence embedding (the output of a single embedding layer) and passed to the decoders. The network output is recursively fed into the same modules a total of three times, utilizing the idea of recycling [15].

Optionally the network can be used with linear update layers. They are message passing layers with the same architecture that operate on a linear graph, added at the end of the model. The edges in the linear graph are generated according to the primary structure of the protein, where each aminoacid is only connected to itself and to its direct neighbors in the sequence. The number of these layers is a hyperparameter that is set to zero for all experiments except for the model trained with diffusion and the co-design experiments on the diverse protein dataset with standard noising (see Section 2.4 for noising schemes). In those experiments the number of linear layers is three. See a schematic representation of the architecture at Figure 1.

Message passing The message passing layers used in this work are a modification over vector-gated GVP [13]. The main difference compared to the original architecture is the addition of edge feature updates. This is implemented by passing the shared hidden state to two copies of the original processing stream at every layer. The output of the first stream is aggregated over the neighborhoods, updating the node features, as in Jing et al. [14], while the output of the other stream is used directly to update the edge features (see Figure 2 and Algorithm 1). We refer to this new architecture as GVPe (GVP with edge updates). The rest of the processing is updated accordingly. See Appendix A for details.

Input features The edge features are an encoding of the distance between the two amino acids along the chain, radial basis function embeddings of all distances between N, C and C_α atoms, and a vector in the direction of $C_\alpha - C_\alpha$. The node features are vectors in the directions towards adjacent amino acids and a vector in the direction of $C_\alpha - C_\beta$ where C_β is imputed as in Dauparas et al. [4]. Note that since those features are translation invariant and the model is trained to predict translations

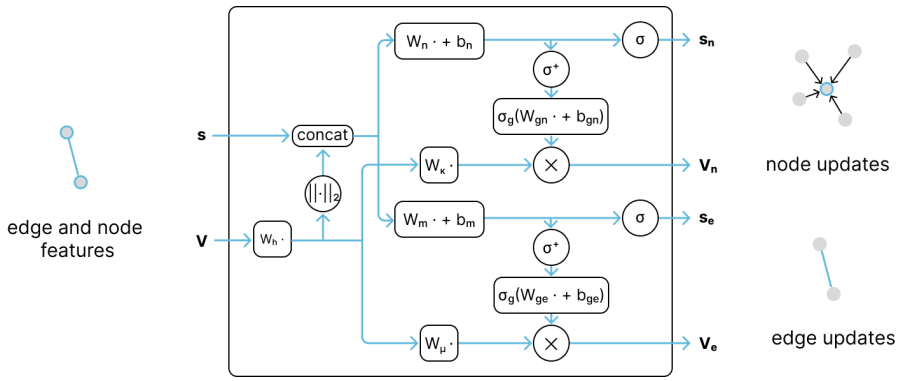

Figure 2: A scheme of the modified GVP message passing layer with edge updates (GVPe).

Algorithm 1 Geometric Vector Perceptron with edge update

Require: Scalar and vector features $(s, V) \in \mathbb{R}^n \times \mathbb{R}^{\nu \times 3}$

Ensure: Scalar and vector edge and node features $(s_e, V_e) \in \mathbb{R}^m \times \mathbb{R}^{\mu \times 3}$, $(s_n, V_n) \in \mathbb{R}^k \times \mathbb{R}^{\kappa \times 3}$

Set $h = \max(\nu, \mu, \kappa)$

GVPe:

$V_h \leftarrow W_h V$ with $V_h \in \mathbb{R}^{h \times 3}$

$V_\mu \leftarrow W_\mu V_h$ with $V_\mu \in \mathbb{R}^{\mu \times 3}$

$V_\kappa \leftarrow W_\kappa V_h$ with $V_\kappa \in \mathbb{R}^{\kappa \times 3}$

$s_h \leftarrow \|V_h\|_2$ (row-wise) with $s_h \in \mathbb{R}^h$

$s_{h+n} \leftarrow \text{concat}(s_h, s)$ with $s_{h+n} \in \mathbb{R}^{h+n}$

$s_m \leftarrow W_m s_{h+n} + b_m$ with $s_m \in \mathbb{R}^m$

$s_k \leftarrow W_n s_{h+n} + b_n$ with $s_k \in \mathbb{R}^n$

$s_e \leftarrow \sigma(s_m)$ with $s_e \in \mathbb{R}^m$

$s_n \leftarrow \sigma(s_k)$ with $s_e \in \mathbb{R}^k$

$V_e \leftarrow \sigma_+(W_{g_e} s_m + b_{g_e}) \odot V_\mu$ (row-wise multiplication) with $V_e \in \mathbb{R}^{\mu \times 3}$

$V_n \leftarrow \sigma_+(W_{g_n} s_k + b_{g_n}) \odot V_\kappa$ (row-wise multiplication) with $V_n \in \mathbb{R}^{\kappa \times 3}$

return $(s_e, V_e), (s_n, V_n)$

relative to the input coordinates, the architecture is equivariant to translations, despite the fact that GVP layers alone are not. A full proof is presented in Appendix A.1.

2.3 Data

We use two datasets generated with the ProteinFlow python library [17]: one contains antibodies and is split based on CDR (complementarity determining region) sequence similarity and the other is a diverse protein dataset split based on similarity of entire chain sequences. Both are publicly available, see Appendix B for the parameters and download instructions. The test set for the antibody dataset follows the logic in Luo et al. [20] and consists of all complexes containing certain antigens and antibodies whose CDR H3 sequences are similar to those of antibodies that bind to the specific set of antigens. Despite some data leakage on CDRs H1 and H2, it represents a realistic scenario of designing an antibody for a new target, as CDR H3 is the most important region for antibody specificity. We have also tested the models on a subset where all selected CDRs are sufficiently different from the training set, those results are presented in Appendix E. For the experiments done on the antibody dataset, the masked regions are CDRs. The identity of the CDR is masked one at a time during training. For the diverse protein dataset we mask residues on the complex interface. The detailed algorithm can be found in Appendix C.

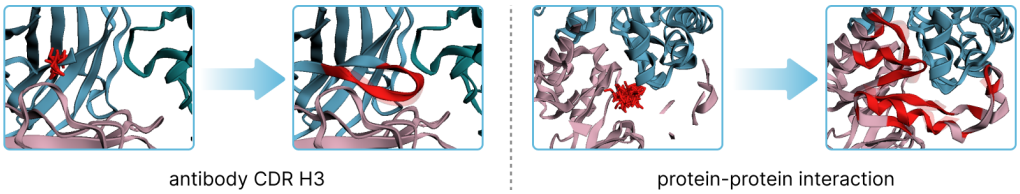

Figure 3: A visualization of structure recovery on random test set samples (from initialization to model prediction). The antibody sample (PDB ID: 2DD8) is on the left and the diverse protein sample (PDB ID: 3UWQ) is on the right. The solid red region denotes the redesigned coordinates and the transparent red is the ground truth structure.

2.4 Initialization

The model requires an initialization for the masked data. For amino acid identities, it is simply replaced with a mask token, and ground truth orientations are replaced with random rotation matrices (sampled from $\mathcal{U}_{SO(3)}$). For the C_α coordinates there are two options. The standard approach used, for instance, in Luo et al. [20] is replacing the masked coordinates with positions sampled from a Gaussian distribution. We apply this scheme for all experiments, unless specified otherwise. The Gaussian is centered at the mean of the unmasked residues that are the closest along the chain to the hidden region and has a standard deviation of 0.1 Å for non-diffusion experiments on antibodies and 1 Å for others. We also suggest an alternative noising method for the diverse protein dataset. In practice, it is reasonable to assume that when a binding surface is redesigned the amino acids should not move too far away from their original position, which means the ground truth information does not have to be destroyed completely. To model this, we also conduct experiments where instead of replacing the ground truth coordinates with a Gaussian distribution we add Gaussian noise (standard deviation 5 Å) to the data. This is referred to as *alternative noising* in the text.

2.5 Training

We use the predicted orientations to reconstruct the coordinates of all backbone atoms and then apply a mean squared error (MSE) loss on all atoms [25]. The final loss function is a sum of the MSE loss and cross-entropy on amino acid identity predictions. The models were trained for a fixed number of epochs and evaluated based on one random prediction per protein at the last epoch. All results on the antibody dataset are aggregated over five random seeds. More details can be found in Appendix F.

2.6 Diffusion

To train the model within a diffusion framework, we define noising and denoising schemes for the three data modalities. For translations, we follow the scheme proposed in Ho, Jain, and Abbeel [7] with the x_0 parameterization for noising and denoising the translation vectors. The orientations are diffused according to Leach et al. [18] and amino acid identities according to Austin et al. [2]. One difference introduced to the original formulations is not adding noise at every step at inference time, as in Mathis et al. [21], in order to account for the deterministic nature of inpainting in a small region. See the algorithms and other details in Appendix D. We refer to the model trained with diffusion as ProtFillDiff.

3 Results

3.1 Ablation

In this section, ablation experiments are conducted to assess the impact of two significant enhancements made to the baseline GVP-GNN structure: edge updates and recycling. The outcomes, presented in Table 1, confirm that both alterations notably enhance the model's performance, thereby justifying their integration.

Table 1: Sequence recovery rates and C_α RMSD (Å) on the antibody dataset for the model with and without recycling (Re) and edge updates (EU) on different CDRs. Bold text indicates the highest performing model.

Re	EU	H1	H2	H3
+	+	**0.71 (0.01)/1.19 (0.02)**	**0.57 (0.01)/1.10 (0.05)**	**0.30 (0.01)/2.93 (0.09)**
-	+	$0.69(0.01)/2.15(0.19)$	$0.51(0.01)/2.00(0.21)$	$0.27(0.01)/3.74(0.09)$
+	-	$0.59(0.02)/1.83(0.09)$	$0.41(0.02)/1.76(0.07)$	$0.27(0.01)/3.23(0.10)$
-	-	$0.56(0.03)/2.65(0.09)$	$0.36(0.01)/2.40(0.14)$	$0.27(0.01)/3.89(0.83)$

Table 2: Sequence recovery rates and C_α RMSD (Å) on the antibody dataset for our models and existing alternatives. Bold and underlined text indicates the highest performing models.

Model	H1	H2	H3
ProtFill	**0.71 (0.01)/1.19 (0.02)**	**0.57 (0.01)/1.10 (0.05)**	$0.30(0.01)/2.93(0.09)$
ProtFillDiff	$0.63(0.01)/1.38(0.07)$	$0.51(0.02)/1.21(0.05)$	**0.39 (0.01)**/$3.58(0.34)$
diffAb	$0.50(0.02)/2.31(0.14)$	$0.32(0.01)/2.18(0.14)$	$0.26(0.01)/4.59(0.25)$
MEAN	$0.54(0.02)/1.20(0.03)$	$0.36(0.01)/1.14(0.11)$	$0.26(0.03)$/**2.51 (0.11)**

3.2 Diverse protein dataset co-design

We investigate the performance of the model in the task of in-painting in a diverse protein dataset with the two noising schemes outlined in Section 2.4. With the standard (harder) noising the results on the test set are 27% sequence recovery rate and 2.22 Å C_α RMSD and with the alternative (easier) noising they are 33% and 1.11 Å. For visual representations of the predictions, refer to Figure 3.

3.3 Diffusion

Adding diffusion does not seem to impact the performance of the model. Overall, compared to existing alternatives, our models perform significantly better on sequence prediction and are comparable on structure prediction (see Table 2). The MEAN [16] and diffAb [20] models were re-trained on the same dataset following the same procedure as our models (trained for a fixed number of epochs on mixed CDRs and evaluated at a random prediction from the last checkpoint).

4 Discussion

This study introduces a unified model for designing protein structure and sequence simultaneously, within the framework of protein inpainting, with potential applications of interface design and binding optimization. Using $SE(3)$ equivariant graph neural networks with an updated version of GVP-GNN message passing layers results in superior sequence prediction and competitive structure recovery, demonstrating the potential of co-design for inpainting on diverse proteins. Incorporating the model into a diffusion framework, consistent with recent advances in protein design, reveals a promising avenue for further exploration. Ablation studies highlight the efficacy of edge feature updates and recycling in enhancing structure prediction accuracy. The results show promise for designing interfaces for protein-protein interactions, but further experimental validation is needed to confirm the utility of the method. Future work could explore combining the diffusion framework with various conditioning approaches and integrating variability of the length of the inpainted region.

References

[1] Namrata Anand and Tudor Achim. "Protein structure and sequence generation with equivariant denoising diffusion probabilistic models". In: *arXiv preprint arXiv:2205.15019* (2022).

[2] Jacob Austin et al. "Structured denoising diffusion models in discrete state-spaces". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17981–17993.

[3] H.M. Berman et al. "The Protein Data Bank". In: *Nucleic Acids Research* 28.1 (2000), pp. 235–242. DOI: 10.1093/nar/28.1.235.

[4] Justas Dauparas et al. "Robust deep learning–based protein sequence design using ProteinMPNN". In: *Science* 378.6615 (2022), pp. 49–56.

[5] James Dunbar et al. "SAbDab: the structural antibody database". In: *Nucleic acids research* 42.D1 (2014), pp. D1140–D1146.

[6] Kaiyuan Gao et al. "Incorporating Pre-training Paradigm for Antibody Sequence-Structure Co-design". In: *bioRxiv* (2022), pp. 2022–11.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[8] Emiel Hoogeboom et al. "Equivariant diffusion for molecule generation in 3d". In: *International conference on machine learning*. PMLR. 2022.

[9] Chloe Hsu et al. "Learning inverse folding from millions of predicted structures". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 8946–8970.

[10] John Ingraham et al. "Generative models for graph-based protein design". In: *Advances in neural information processing systems* 32 (2019).

[11] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Antibody-antigen docking and design via hierarchical structure refinement". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 10217–10227.

[12] Wengong Jin et al. "Iterative refinement graph neural network for antibody sequence-structure co-design". In: *arXiv preprint arXiv:2110.04624* (2021).

[13] Bowen Jing et al. "Equivariant graph neural networks for 3d macromolecular structure". In: *arXiv preprint arXiv:2106.03843* (2021).

[14] Bowen Jing et al. "Learning from protein structure with geometric vector perceptrons". In: *arXiv preprint arXiv:2009.01411* (2020).

[15] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[16] Xiangzhe Kong, Wenbing Huang, and Yang Liu. "Conditional antibody design as 3d equivariant graph translation". In: *arXiv preprint arXiv:2208.06073* (2022).

[17] Elizaveta Kozlova et al. "ProteinFlow: a Python Library to Pre-Process Protein Structure Data for Deep Learning Applications". In: *bioRxiv* (2023). DOI: 10.1101/2023.09.25.559346.

[18] Adam Leach et al. "Denoising diffusion probabilistic models on so (3) for rotational alignment". In: *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*. 2022.

[19] Sidney Lyayuga Lisanza et al. "Joint generation of protein sequence and structure with RoseTTAFold sequence space diffusion". In: *bioRxiv* (2023), pp. 2023–05.

[20] Shitong Luo et al. "Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9754–9767.

[21] Simon V. Mathis et al. "Normal Mode Diffusion: Towards Dynamics-Informed Protein Design". In: *The 2023 ICML Workshop on Computational Biology*. 2023.

[22] Tim Salimans and Jonathan Ho. "Progressive distillation for fast sampling of diffusion models". In: *arXiv preprint arXiv:2202.00512* (2022).

[23] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[24] Clement Vignac et al. "Digress: Discrete denoising diffusion for graph generation". In: *arXiv preprint arXiv:2209.14734* (2022).

[25] Joseph L Watson et al. "Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models". In: *BioRxiv* (2022), pp. 2022–12.

A Edge updates

Originally, GVP layers only update the node features, while the edge features remain constant. Following the example of ProteinMPNN [4], we wanted to explore the effect of updating the edge features as well.

In order to achieve that, within the message passing algorithm we generate node and edge features separately from the joint hidden state. See Algorithm 1 for details. In addition to the main modification in the message passing algorithm, updating the edges requires changes in the rest of the processing stream. GVPe performs graph propagation steps as follows.

$$s^{(i\to j)} = \text{concat}\left(s_n^{(i)}, s_n^{(j)}, s_e^{(i\to j)}\right), \tag{1}$$

$$V^{(i\to j)} = \text{concat}\left(V_n^{(i)}, V_n^{(j)}, V_e^{(i\to j)}\right), \tag{2}$$

$$(s'_e, V'_e), (s''_n, V''_n) = \text{GVPe}\left((s, V)\right), \tag{3}$$

$$s_n'^{(i)} = \frac{1}{K}\sum_{k\in\mathcal{N}(i)} s_n''^{(k\to i)}, \tag{4}$$

$$V_n'^{(i)} = \frac{1}{K}\sum_{k\in\mathcal{N}(i)} V_n''^{(k\to i)}, \tag{5}$$

$$s_n^{(i)} = \text{LayerNorm}\left(\text{GELU}\left(W_{s_n}\left[\text{concat}\left(s_n^{(i)}, \text{Dropout}\left(s_n'^{(i)}\right)\right)\right] + b_n\right)\right), \tag{6}$$

$$s_e^{(i\to j)} = \text{LayerNorm}\left(\text{GELU}\left(W_{s_e}\left[\text{concat}\left(s_n^{(i\to j)}, \text{Dropout}\left(s_e'^{(i\to j)}\right)\right)\right] + b_e\right)\right), \tag{7}$$

$$V_n^{(i)} = \gamma_n W_{v_n}\left[\text{concat}\left(V_n^{(i)}, \text{Dropout}\left(V_n'^{(i)}\right)\right)\right], \tag{8}$$

$$V_e^{(i\to j)} = \gamma_e W_{v_e}\left[\text{concat}\left(V_e^{(i\to j)}, \text{Dropout}\left(V_e'^{(i\to j)}\right)\right)\right]. \tag{9}$$

Other minor differences to the network introduced in Jing et al. [14] are replacing sum with concatenation and an additional linear layer at equations 6-9 and skipping vector normalization.

A.1 Equivariance

The original GVP layers are equivariant to rotations but not to translations. Since we do not introduce any new operations and simply create another copy of the same data stream, GVPe layers maintain the same qualities. Our network consists of stacked GVPe layers, so in order to prove that our update operation is equivariant to translations and rotations ($SE(3)$ equivariant) we need to show that it is equivariant to translations.

The model is trained to predict translations relative to the input data and uses input features that are invariant to translations. Therefore, it can be represented as

$$x' = f(x, h) = x + \text{ProtFill}\left(\text{feat}\left(x, h\right)\right), \tag{10}$$

where ProtFill is the network, f is the entire update operation and feat is the feature extraction operation, x is the coordinate array and h is the scalar feature array defined in space $\mathbb{R}^n$. We know that

$$\text{feat}\left(x, h\right) = \text{feat}(x + t, h)\ \forall t \in \mathbb{R}^3\ \forall x \in \mathbb{R}^3\ \forall h \in \mathbb{R}^n. \tag{11}$$

Therefore,

$$\text{ProtFill}\left(\text{feat}\left(x, h\right)\right) = \text{ProtFill}\left(\text{feat}\left(x + t, h\right)\right) \triangleq y(x, h). \tag{12}$$

This means that

$$\begin{aligned} f(x + t, h) &= x + t + \text{ProtFill}\left(\text{feat}\left(x + t, h\right)\right) = x + t + y(x, h) \\ &= x + \text{ProtFill}\left(\text{feat}\left(x, h\right)\right) + t = f(x, h) + t\ \forall t \in \mathbb{R}^3\ \forall x \in \mathbb{R}^3\ \forall h \in \mathbb{R}^n. \end{aligned} \tag{13}$$

Therefore, the update operation is equivariant to translations and thus $SE(3)$ equivariant.

Table 3: Parameters of the datasets used.

	Diverse protein	Antibody
PDB snapshot	20230102	live 26.06.23
Minimum resolution, Å	3.5	3.5
Minimum length, aminoacids	30	30
Maximum length, aminoacids	2'000	2'000
MMseqs threshold	0.3	0.4
Training set size, clusters	19'471	4'752
Validation set size, clusters	1'017	147
Test set size, clusters	977	550/32
Missing threshold (ends/middle)	0.3/0.1	0.5/0.2
Source	PDB	SAbDab
Remove redundancies	yes	no

B Data

In this paper, we used two datasets generated by ProteinFlow [17]. The diverse protein dataset includes all single chains and protein-protein interactions defined in the Protein Data Bank (PDB) [3] as biological units. This includes all antibody-antigen interactions, heteromers, monomers, as well as synthetic protein interactions. On the other hand, the antibody dataset only contains entries related to antibodies and nanobodies, defined within the SabDab database [5]. The filtering and processing parameters are listed in Table 3.

The antibody dataset has two test subsets. One is generated with the standard ProteinFlow pipeline and contains 32 clusters that do not belong to any of the interaction graph connected components used in training. The performance of the model on this data should be close to the lower boundary of completely new data. The results for this test set are presented in Appendix E. The second test set is created in accordance with Luo et al. [20]. It consists of all the complexes that include any of a list of antigens (identified in SAbDab as *'sars-cov-2 receptor binding domain', 'hiv-1 envelope glycoprotein gp160', 'mers s', 'influenza a virus', 'cd27 antigen'*, and other chains that are 70% or more similar to those) and others that share sequence similarity (greater or equal to 30%) to CDR-H3 of any of the antibodies binding to these antigens. This represents a realistic real-world scenario of re-designing a protein for a new therapeutic target but contains some data leakage. This subset contains 550 clusters. The results on this test set are presented in Section 3.2. The data is publicly available and can be accessed with ProteinFlow commands. It is then additionally filtered to only include entries where the total length of the chains does not exceed 2000 residues.

```
proteinflow download --tag 20230102_stable # diverse protein

proteinflow download --tag 20230626_sabdab --skip-splitting # antibody
proteinflow split --tag 20230626_sabdab --test_split 0.01 --valid_split
0.03 --exclude_clusters --exclude_based_on_cdr H3 -e 5xku-A -e 7chf-R -e
5tlj-X -e 5w9n-D -e 7che-R -e 5tlk-Y -e 5tlk-X -e 5w9h-G -e 7bwj-E -e
5tl5-A -e 7d6i-A -e 7chb-R -e 5w9h-D -e 5w9n-A -e 5w9h-A --min_seq_id 0.4
```

Additionally, all the input data is patched (cut) around the masked region in a manner similar to Luo et al. [20]. The patching is based on the *anchor points* coordinates, or the C_α coordinates of the unnoised amino acids that are the closest along the chain to the noised area. We select a number of residues that are spatially the closest to the set of the anchor points and discard the rest. For the diverse protein dataset, we select the closest residues, regargless of the chain they belong to. For the antibody, we select a union of 128 closest residues in all chains and 128 closest residues in the antigen.

B.1 Sampling

In order to avoid biasing the model towards more common proteins many works [10, 9] iterate over sequence identity clusters instead of over individual sequences at training and validation and use a

single representative from each cluster. We also follow the strategy of iterating over the MMSeqs2 clusters. However, instead of using a single representative for each cluster we sample a random one at each pass through the data, as in [4].

C Masking

In this paper some experiments are done on the antibody dataset and some on the diverse protein dataset. The two datasets require different masking algorithms.

C.1 Diverse protein

For the diverse protein protein dataset, we select parts of the sampled chain to be re-designed. In particular, one residue from the chain is chosen at random, followed by the addition of N spatially closest residues. Those residues are the masked region that the model needs to re-design. We set N to vary from 15 to 50 but never more than half of the chain length. In addition, we force the masked region to be close to an estimated binding site in multi-chain complexes 50% of the time during training and 100% of the time during inference. That means selecting the random center residue from the subset of amino acids that are within 10 angstroms from a different chain.

C.2 Antibody

For the experiments done on the antibody dataset, the masked regions are CDRs. CDRs are masked one at a time and the identity of the CDR is not known to the model. The unmasked CDRs are passed as input to the model.

D Diffusion

Since the model works with three data modalities (C_α coordinates, N-C-C_α orientation angles and amino acid identity distributions), defining the diffusion process means defining training and sampling algorithms for each of those modalities.

D.1 Variance schedule

For all the data modalities we use the cosine variance schedule, defined by the following equations.

$$\bar{\alpha}_t = \cos\left(\left(\frac{\pi}{2}\right)\frac{t/T + 0.01}{1 + 0.01}\right)^2, \tag{14}$$

$$\beta_t = \frac{\bar{\alpha}_{t+1}}{\bar{\alpha}_t}, \tag{15}$$

$$\alpha_t = 1 - \beta_t, \tag{16}$$

$$\sigma_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t. \tag{17}$$

Here T is the total number of diffusion steps, we set it to 50. In practice, we clamp $\beta_t = \min(0.99, \beta_t)$ and then recompute $\bar{\alpha}_t = \prod_{k \le t}(1 - \beta_t)$ and the other values in order to prevent the output from blowing up at inference time.

D.2 Coordinates

For the C_α coordinates, we use the standard diffusion formulation from [7] with the x_0 parameterization (see algorithms 2-3).

Note that the sampling algorithm here deviates from Ho, Jain, and Abbeel [7]. We have found that the best results for the inpainting task are achieved with no added noise during sampling, as in Mathis et al. [21]. Therefore we replace the $x_{t-1} = \tilde{\mu}_t\left(g_\theta\left(x_t\right)\right) + \sigma_t z,\ z \sim \mathcal{N}(0, I)$ update formula

Algorithm 2 Training $C\alpha$

repeat
  $x_0 \sim q(x_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\epsilon \sim \mathcal{N}(0, I)$
  Take gradient descent step on $\nabla_\theta \gamma_t ||g_\theta(x_t, t) - x_0||^2$
until converged

Algorithm 3 Sampling $C\alpha$

$x_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ **do**
  $x_{t-1} = \left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} g_\theta(x_t, t) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \right)$
end for
return x_0

with $x_{t-1} = \tilde{\mu}_t\left(g_\theta\left(x_t\right)\right)$, where $\tilde{\mu}_t\left(g_\theta\left(x_t\right)\right)$ is an estimation of the mean of x_{t-1}. Analogous modifications are introduced to the sampling algorithms for orientation and sequence as well.

The γ_t parameter in the loss function can be set to $\max\left(\frac{\alpha_t}{1-\alpha_t}, 1\right)$, as in [22], to account for different noise levels. However, we have found that this does not impact the results, so in all experiments we set it to 1.

Normalization For the diffusion process to stay $SE(3)$ equivariant, the data needs to be centered and normalized [8]. It is centered at the mean of the *anchor points*. The standard deviation of the ground truth data is 5.2 Å but we have found that setting the prior standard deviation to 1 Å leads to the best performance.

D.3 Orientation

For the diffusion of the orientation, we follow the algorithms suggested by Leach et al. [18] (4-5).

Algorithm 4 Training orientation

repeat
  $t \sim \text{Uniform}(\{1, ..., T\})$
  $x_0 \sim q(x_0)$
  $R \sim IG_{SO(3)}(I, \sqrt{1-\bar{\alpha}_t})$
  $S(v) = \frac{\log(R)}{\sqrt{1-\bar{\alpha}_t}}$
  $x_{\text{scale}} = \exp\left(\sqrt{\bar{\alpha}_t} \log x_0\right)$
  Take gradient descent step on: $\nabla_\theta \|v - g_\theta(R x_{\text{scale}}, t)\|^2$
until converge

Here $\tilde{\mu}_t$ has the meaning of an estimation for x_{t-1} but is computed on the SO(3) space.

$$\tilde{\mu}_t(x_t, \tilde{x}_0) = \lambda\left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}, \tilde{x}_0\right) \lambda\left(\frac{\sqrt{\alpha_{t-1}}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}, x_t\right), \tag{18}$$

$$\lambda(\gamma, x) = \exp(\gamma \log(x)). \tag{19}$$

The $\exp$ and $\log$ operations are redefined here to map between the Lie algebra $\mathfrak{so}(3)$ and $SO(3)$. In this space,

$$\log R = \frac{\theta}{2\sin\theta}(R^\top - R), \tag{20}$$

Algorithm 5 Sampling orientation

$x_T \sim U_{SO(3)}$
for $t = T, \ldots, 1$ **do**
  $v = g_\theta(x_t, t)$
  $\mathbf{a_1} = \exp\left(\sqrt{\frac{1}{\bar{\alpha}_t}}\log(x_t)\right)$
  $\mathbf{a_2} = \exp\left(S\left(\sqrt{\frac{1}{1-\bar{\alpha}_t}}v\right)\right)$
  $\tilde{x}_0 = \mathbf{a_1}\mathbf{a_2}^{-1}$
  $x_{t-1} = \tilde{\mu}_t(x_t, \tilde{x}_0)$
end for

where θ satisfies $1 + 2\cos\theta = \mathrm{tr}(R)$. Additionally, given the vector $v = (x, y, z)$, the skew-symmetric matrix $S(v)$ is defined as:

$$S(v) = \begin{pmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{pmatrix}. \tag{21}$$

D.4 Amino acid sequence

For diffusing the amino acid sequence, we follow the framework suggested in Austin et al. [2] and Vignac et al. [24] (algorithms 6-7).

Algorithm 6 Training sequence

repeat
  $x_0 \sim q(x_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\epsilon \sim \mathcal{N}(0, I)$
  $x_t \sim \mathrm{Cat}\left(x_t; p = x_0\bar{Q}_t\right)$
  Take gradient descent step on ∇_θ cross-entropy$(g_\theta(x_t, t), x_0)$
until converged

Algorithm 7 Sampling sequence

$x_T \sim \mathcal{N}(0, I)$
for $t = T, \ldots, 1$ **do**
  $\tilde{x}_0 \sim g_\theta(x_t, t)$
  $x_{t-1} = \mathrm{argmax}\left(\frac{x_t Q_t^\top \tilde{x}_0 \odot \bar{Q}_{t-1}}{\tilde{x}_0 \bar{Q}_t x_t^\top}\right)$
end for
return x_0

Here Q_t are state transition matrices defined by the variance schedule. Specifically, in this work we use the model of absorbing state. It is defined by the following matrix:

$$[Q_t]_{ij} = \begin{cases} 1 & \text{if } i = j = m, \\ \alpha_t & \text{if } i = j \neq m, \\ 1 - \alpha_t & \text{if } j = m, \text{ and } i \neq m, \end{cases} \tag{22}$$

where m is the index of the absorbing masked state. The α_t parameter follows the same variance schedule as for $C\alpha$ positions and orientations (section **??**). The $\bar{Q}_t$ matrices are cumulative products, with $\bar{Q}_t = Q_1 Q_2 ... Q_t$.

Table 4: Sequence recovery rates and C_α RMSD (Å) on the 'hard' antibody test set for the model with and without recycling (Re) and edge updates (EU) on different CDRs. Bold text indicates the highest performing model.

Re	EU	H1	H2	H3
+	+	$0.32(0.03)$/**1.90 (0.10)**	$0.25(0.05)$/**0.95 (0.03)**	$0.20(0.02)$/**2.61 (0.09)**
-	+	**0.34 (0.02)**/$2.56(0.18)$	**0.27 (0.02)**/$1.89(0.21)$	**0.27 (0.01)**/$3.74(0.10)$
+	-	$0.28(0.04)/2.42(0.04)$	$0.23(0.03)/1.48(0.10)$	$0.23(0.01)/3.05(0.15)$
-	-	$0.27(0.03)/3.06(0.11)$	$0.25(0.01)/2.30(0.25)$	$0.23(0.03)/3.9(0.4)$

Table 5: Sequence recovery rates and C_α RMSD (Å) on the 'hard' antibody test set for ProtFill and existing alternatives. Bold and underlined text indicates the highest performing models.

Model	H1	H2	H3
ProtFill	**0.32 (0.03)**/**1.90 (0.10)**	$0.25(0.05)$/**0.95 (0.03)**	$0.20(0.02)/2.61(0.09)$
ProtFillDiff	$\underline{0.31(0.02)}/2.18(0.12)$	**0.27 (0.02)**/$1.00(0.05)$	**0.29 (0.02)**/$2.56(0.19)$
diffAb	$0.27(0.03)/3.17(0.20)$	$\underline{0.26(0.03)}/2.07(0.22)$	$0.19(0.03)/\underline{4.23(0.14)}$
MEAN	$0.23(0.02)/\underline{1.95(0.06)}$	$0.23(0.02)/1.07(0.10)$	$0.17(0.01)$/**2.50 (0.16)**

E Lower boundary of performance

Here in Tables 4 and 5 we present the results of the experiments on the *hard test set* (described in Appendix B). They represent an approximation of the lower boundary of the model performance on completely new data.

F Training details

All models were trained for a fixed number of epochs, determined based on the flattening of metrics on the validation subset. For the ProtFill and ProtFillDiff models it is 250 epochs if recycling is used and 100 if it is not. For diffAb [20] it is 200'000 steps, as in the original configuration files. For MEAN [16] is it 100 epochs, since at the original 20 epochs the validation metrics were still improving. The original configuration files are used for all the other parameters. All models are trained on all CDRs mixed and evaluated at the last epoch, with no selection for the generated samples. The learning rate for our models is determined by the Noam optimizer [23].

The training time on the antibody dataset is approximately 23 hours for ProtFill and 29 for ProtFillDiff on one NVIDIA A10G Tensor Core GPU. On the larger diverse dataset ProtFill training took 118 hours. On the antibody dataset, each model was trained with five random seeds and the results were averaged.