REPURPOSING ALPHAFOLD3-LIKE PROTEIN FOLDING MODELS FOR ANTIBODY SEQUENCE AND STRUCTURE CO-DESIGN

Nianzu Yang[1†], **Songlin Jiang**[1†], **Jian Ma**[1†], **Huaijin Wu**[1], **Shuangjia Zheng**[1], **Wengong Jin**[2*], **Junchi Yan**[1*]
[1] Shanghai Jiao Tong University, Shanghai, China
[2] Northeastern University, Boston MA, USA
Code: https://github.com/yangnianzu0515/MFDesign
Model Weights: https://huggingface.co/yangnianzu/MFDesign

ABSTRACT

Diffusion models hold great potential for accelerating antibody design, but their performance is so far limited by the number of antibody-antigen complexes used for model training. Meanwhile, AlphaFold3-like protein folding models, pre-trained on a large corpus of crystal structures, have acquired a broad understanding of biomolecular interaction. Based on this insight, we develop a new antigen-conditioned antibody design model by adapting the diffusion module of AlphaFold3-like models for sequence-structure co-diffusion. Specifically, we extend their structure diffusion module with a sequence diffusion head and fine-tune the entire protein folding model for antibody sequence-structure co-design. Our benchmark results show that sequence-structure co-diffusion models not only surpass state-of-the-art antibody design methods in performance but also maintain structure prediction accuracy comparable to the original folding model. Notably, in the antibody co-design task, our method achieves a CDR-H3 recovery rate of 65% for typical antibodies, outperforming the baselines by 87%, and attains a remarkable 63% recovery rate for nanobodies.

1 INTRODUCTION

Monoclonal antibodies are Y-shaped proteins that specifically recognize and neutralize the pathogens (commonly known as antigens) during immune responses (Janeway et al., 2001). The binding specificity is determined by their complementarity-determining regions (CDRs), whose sequences and structures exhibit significant variability. Recent work on generative models have shown great potential in designing CDRs that bind to an antigen (Jin et al., 2022b; Luo et al., 2022; Kong et al., 2023a; Martinkus et al., 2024; Zhu et al., 2024). However, their performance is limited by the scarcity of antibody-antigen complex data, as highlighted by Zhu et al. (2024).

To address this issue, we seek to leverage foundation models, such as protein folding models, that have learned general knowledge of biomolecular interaction from a large corpus of general protein interaction data. Our hypothesis is that the general knowledge of protein-protein interaction learned by protein folding models, such as AlphaFold3 (AF3) (Abramson et al., 2024), Chai-1 (Discovery et al., 2024), Boltz-1 (Wohlwend et al., 2024), and Protenix (Chen et al., 2025), are transferrable to the task of modeling of antibody-antigen complexes, a specialized type of protein interaction.

To validate this hypothesis, we need to effectively endow current protein folding models with the additional capability of generating both sequence and structure of an antibody or a protein. Our key observation is that AF3-like folding models already utilize diffusion modules for structure prediction. Indeed, diffusion models (Ho et al., 2020; Song et al., 2021; Karras et al., 2022) have shown various success in the co-design of antibody sequences and structures (Luo et al., 2022; Martinkus et al., 2024; Zhu et al., 2024). Therefore, we propose to integrate sequence diffusion into the diffusion modules

† Equal contribution. * Correspondence author.

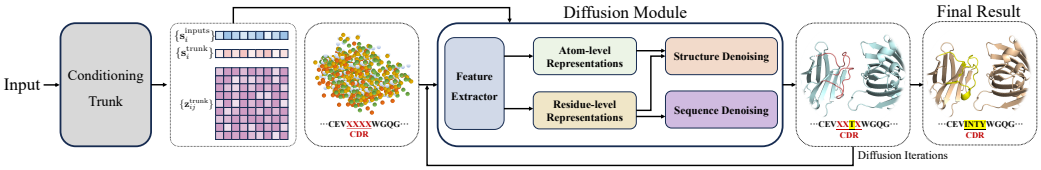

Figure 1: **Overview of our method.** We build upon AF3-like folding models by integrating sequence diffusion with the existing structure diffusion, enabling the co-diffusion of structure and sequence. The input for sequence denoising network directly reuses the residue token-level representations originally used for structure denoising in AF3-like models. Initially, CDR tokens are marked as **<X>**, which will be specified as one of the 20 amino acid tokens in the final output. Through *replacement* sampling, our method allows the CDR design to be conditioned on fixed co-crystal structures.

of AF3-like folding models for sequence-structure co-design. We demonstrate the feasibility of this idea using Boltz-1[1] (Wohlwend et al., 2024), an open-source implementation of AF3 provided with training scripts. Specifically, we augment the diffusion module of Boltz-1 with a sequence diffusion head. We implement two approaches for sequence diffusion: a discrete diffusion approach trained in the D3PM-absorbing (Austin et al., 2021) fashion, and a continuous diffusion approach trained using EDM (Karras et al., 2022) method. After properly aligning sequence and structure diffusion steps, we fine-tune the entire Boltz-1 network using a training set of antibody-antigen complexes. Results showcase that the modified AF3-like model outperforms existing baseline methods in antibody sequence and structure co-design tasks, while retaining Boltz-1's structure prediction capability.

2 Methodology

We first present a description of the task we investigate in this paper in Sec. 2.1. Sec. 2.2 details how straightforward modifications to AF3-like models enable support for antibody co-design, with supplementary methodological details provided in Appendix C. Before proceeding to our method, we refer readers to Appendix B for a quick review of AF3's workflow and the key notations involved.

2.1 Problem Description

We present a diagram of an antibody-antigen complex in Fig. S1. In this work, we follow the widely adopted setting for antibody design: re-designing CDRs in experimentally resolved co-crystal structures, where all non-CDR regions (antibody framework sequences/structures, antigen sequences/structures, and antibody-antigen binding pose) are given. The objective is to co-design the sequence and structure of CDRs within this predefined context.

Particularly, rather than designing a single CDR at a time, we aim to simultaneously design all CDRs at once, which is a more challenging task (Martinkus et al., 2024). Within this study, we use the notation system from AF3 to characterize protein complexes, employing $\{\mathbf{k}_i\}$ and $\{\mathbf{x}_l\}$ to specify the amino acid (*a.k.a.* residue) composition and structure of the complex, respectively. $\mathbf{k}_i$ is a one-hot encoding that represents the type of each amino acid token (20 standard amino acid types + **<X>**). The **<X>** token is originally used in AF3-like folding models to indicate positions where the residue type is unknown, for which only the backbone atom coordinates will be predicted. Each element in $\{\mathbf{x}_l\}$ represents the three-dimensional coordinates of an atom.

We use the **<X>** token to represent residues within the CDRs that need to be designed. In the final output, each **<X>** token is replaced with a specific residue from the 20 standard types, accompanied by their corresponding atomic coordinates, thus achieving the goal of the antibody co-design.

2.2 Our Method

The input includes residue tokens from antibodies and antigens. For the antibody CDRs that need to be designed, these positions are filled with **<X>** tokens. Considering the discrete nature of residue tokens as categorical variables, along with the resemblance of the input format to masked language models (MLMs) like BERT (Devlin et al., 2019) — where **<X>** is analogous to the **<MASK>** tokens

[1] https://github.com/jwohlwend/boltz

used in MLMs — the D3PM-absorbing (Austin et al., 2021) method, inspired by MLMs, intuitively aligns well with this task. Therefore, we choose to use D3PM-absorbing for sequence diffusion.

During the training of diffusion-based co-design methods, it is required to corrupt the sequence and structure to a matching degree, which can be achieved by sampling them at the same timestep. However, as noted in Appendix B, AF3-like models perform structure diffusion training by directly sampling noise levels, unlike D3PM-absorbing, which uses timestep sampling. Furthermore, to the best of our knowledge, there is no established theory for training discrete diffusion without sampling timesteps during so far. Aligning sequence and structure diffusion is therefore challenging.

Continuous diffusion as an alternative for sequence. It is worth noting that Hoogeboom et al. propose a method for applying continuous diffusion to discrete data by adding Gaussian noise to their one-hot encodings in training. This allows sequence to be trained also using the EDM method. In this way, sequence diffusion can readily align with structure diffusion training in AF3. For the generation of discrete data, this method tends to be less effective than discrete diffusion (Austin et al., 2021; Gruver et al., 2023), as supported by results in Sec. 3, which we have also implemented. However, our results also reveal its unique advantage for structure prediction. Since this paper primarily focuses on the additional implementation of sequence design compared to AF3, which involves discrete data, we have concentrated on discussing the method using D3PM-absorbing for sequence diffusion in the following main paper. The implementation of the continuous version, being relatively more straightforward and simpler, is provided in Appendix C.7.

Structure & sequence diffusion alignment. Although the training method for structure diffusion directly samples noise levels, we notice that AF3 uses a predefined sequence of 200 decreasing noise levels during sampling. This approach closely resembles the concept of discrete timesteps. Thus, we consider modifying structure diffusion training to also sample noise levels only from these 200 noise levels. This allows us to create a predefined noise scheduler for training, where we sample a timestep and retrieve the corresponding noise level according to this scheduler. Accordingly, we set the number of diffusion steps for sequence to 200, thus enabling easy alignment. Since these same 200 levels are used during both training and inference of the structure diffusion, the approach shifts towards a style resembling the classic DDPM's discrete time training (Ho et al., 2020). Empirical results suggest that fine-tuning with this new structure diffusion approach preserves the model's ability to accurately predict complex structures.

Instantiation of discrete sequence diffusion. Diffusion is applied specifically to CDRs. **<X>** is naturally suited as the absorbing state for residues in CDRs. The forward process is characterized by a discrete transition matrix that determines the probability of a token mutating into **<X>**. The corresponding prior is a point mass on the sequence of which the CDRs are made up entirely of **<X>**. On top of AF3, we introduce an additional module named **TokenDenoiser**. Since we need to make predictions at token-level, token-level representations are required for **TokenDenoiser**. As described in Appendix B, the AF3's structure diffusion process already drives token-level representations $\{\mathbf{a}_i\}$, which are obtained via full self-attention on token level and serve the purpose of structure denoising. Here we choose to directly reuse $\{\mathbf{a}_i\}$ as the input to **TokenDenoiser**. The **TokenDenoiser** is implemented simply as a MLP, consisting of LINEAR layers connected by GELU activation function.

Next, we detail how to input the corresponding noisy sequence, denoted as $\{\mathbf{k}_i^{\text{noisy}}\}$, into the diffusion module at each sampled timestep during training. As noted in Appendix B, $\{\mathbf{s}_i^{\text{inputs}}\}$ from the trunk serves as the a basic encoding of the raw input, where part of its continuous dimensions is actually the one-hot encoding to the tokens. Thus, we can easily achieve the input of the noisy sequence at sampled timestep by directly replacing these dimensions with $\{\mathbf{k}_i^{\text{noisy}}\}$, resulting in $\{\mathbf{s}_i^{\text{replaced}}\}$. Then, $\{\mathbf{s}_i^{\text{replaced}}\}$ is fed into our structure and sequence co-diffusion module and its subsequent usage remains consistent with that of $\{\mathbf{s}_i^{\text{inputs}}\}$. We denote the restored sequence as $\{\mathbf{k}_i^{\text{denoised}}\}$. Note that except for $\{\mathbf{s}_i^{\text{inputs}}\}$, all these notations are associated with a specific timestep t. However, for simplicity, we omit t in the notations presented here.

We fully fine-tune the AF3-like models by combining the original objective of AF3 for structure prediction with an additional objective specifically for sequence diffusion. This additional objective is to maximize the likelihood of the denoising process on the ground truth CDRs sequences, thereby learning the optimal parameters for the **TokenDenoiser**. Our sequence sampling follows the D3PM-absorbing standard method, as used by Gruver et al.. To allow our sampling to be conditioned on

Table 1: Comparison of simultaneously designed CDRs for typical antibodies, with the **best** and the second-best highlighted.

CDR	Method	AAR ↑	RMSD ↓	CDR	Method	AAR ↑	RMSD ↓
H1	DiffAb	60.92%	1.52Å	L1	DiffAb	56.72%	1.43Å
	dyMEAN	70.31%	1.65Å		dyMEAN	73.06%	1.58Å
	MFDesign-C	74.89%	**1.48Å**		MFDesign-C	**82.98%**	**1.41Å**
	MFDesign-D	**74.95%**	1.61Å		MFDesign-D	**82.98%**	1.65Å
H2	DiffAb	33.53%	1.44Å	L2	DiffAb	55.08%	1.21Å
	dyMEAN	66.38%	1.47Å		dyMEAN	76.04%	1.23Å
	MFDesign-C	65.59%	**1.26Å**		MFDesign-C	**88.84%**	**1.00Å**
	MFDesign-D	**67.54%**	1.44Å		MFDesign-D	87.81%	1.15Å
H3	DiffAb	22.26%	4.29Å	L3	DiffAb	43.03%	1.80Å
	dyMEAN	34.69%	6.15Å		dyMEAN	52.69%	1.59Å
	MFDesign-C	63.13%	**3.54Å**		MFDesign-C	78.93%	**1.55Å**
	MFDesign-D	**65.04%**	3.71Å		MFDesign-D	**80.15%**	1.69Å

Table 2: Results on typical antibodies with simultaneously designed CDRs, highlighting the **best** and the second-best results.

Method	Loop-AAR ↑	Loop-RMSD ↓	IMP ↑
DiffAb	15.86%	5.03Å	56.77%
dyMEAN	20.80%	7.84Å	10.56%
MFDesign-C	61.71%	**4.13Å**	**66.24%**
MFDesign-D	**63.38%**	4.28Å	59.66%

the given co-crystal structures and redesign CDRs, we employ the *replacement* sampling technique. Orginally from image inpainting domain (Lugmayr et al., 2022), this technique has been applied in molecule and protein generation tasks to condition the sampling on certain motifs (Schneuing et al., 2024; Trippe et al., 2023). Refer to Appendix C for more details on our training and sampling.

Our method is outlined in Fig. 1. We extend the AF3's **DiffusionModule** with sequence diffusion to enable structure and sequence co-design, which we call **CoDiffusionModule**, formalized in Alg. S1. It follows a feature extraction process akin to the **DiffusionModule**, but reuses the final obtained token-level representations for sequence design. AF3-like folding models inherently support the input of **<X>** tokens. For these unknown residues, they only sample and denoise the backbone atom coordinates. In line with this design, we exclusively generate the backbone atom coordinates for the CDRs. Subsequently, following Luo et al. and Zhu et al., we perform side-chain packing and structure relaxation using PyRosetta (Chaudhury et al., 2010) to obtain final all-atom structure.

3 Results on the Antibody Sequence and Structure Co-design

Due to the unavailability of AF3's training scripts, we opt to make modifications based on Boltz-1 in this work, which is a faithful reimplementation by Wohlwend et al. that follows the same variable and function definitions as AF3. Our method is termed **MFDesign** in subsequent sections, which stands for the **M**odified **F**olding model for the co-**D**esign of antibody sequence and structure. In this section, we focus on the antibody sequence and structure co-design task. Additional results are presented in Appendix G, with a focus on the structure prediction task detailed in Appendix G.1.

Dataset. The data for evaluation comes from the Structural Antibody Database (SAbDab) (Dunbar et al., 2014). Boltz-1 itself is pre-trained on PDB structures released before the same cut-off date as AlphaFold3, *i.e.*, September 30, 2021. In previous studies (Luo et al., 2022; Kong et al., 2023a; Martinkus et al., 2024; Zhu et al., 2024), their experimental setups do not consider the release date when splitting data. To prevent data leakage, meaning the test data should not have been seen during Boltz-1's pre-training phase, we cannot directly adopt the splits used by previous works. Instead, we need to consider the release date when partitioning the data. We first follow the approach of DiffAb (Luo et al., 2022) and AbDiffuser (Martinkus et al., 2024) by using MMSeq2 (Steinegger & Söding, 2017) to cluster antibodies according to CDR-H3 sequences at 50% sequence identity. These clusters are then divided into training, validation, and test sets in an 9:0.5:0.5 ratio, ensuring that all samples released before the cut-off date are included in the training set. This results in 5,843 training samples, 187 validation samples, and 204 test samples, with the test set comprising 161 regular antibodies and 43 nanobodies. Our test set is much larger than those in previous studies, offering a more robust demonstration of the models' ability to generalize. Moreover, it's worth noting that some prior works (Jin et al., 2022a; Kong et al., 2023a; Zhu et al., 2024) employ the RAbD Benchmark (Adolf-Bryfogle et al., 2018), which includes 60 diverse complexes, to test antibody generation performance. However, since these data are released in the PDB before the cut-off date and Boltz-1 has already seen them, we do not adopt the RAbD Benchmark for evaluation to ensure fair comparisons. We provide our detailed data processing procedure in Appendix D to establish a standardized pipeline, which aims to assist future studies in replicating or building upon our work.

Table 3: Comparison of generated nanobodies with simultaneously designed CDRs, with the **best** and the second-best results highlighted. Note that we do not compare against dyMEAN here, as its source code does not support processing nanobodies.

Method	CDR-H1		CDR-H2		CDR-H3		Loop-AAR	Loop-RMSD	IMP
	AAR	RMSD	AAR	RMSD	AAR	RMSD			
DiffAb	45.17%	2.52Å	31.11%	**1.54Å**	15.39%	4.71Å	10.51%	5.50Å	30.93%
MFDesign-C	65.71%	**2.48Å**	55.60%	1.62Å	61.40%	**3.71Å**	**61.01%**	**4.26Å**	**42.67%**
MFDesign-D	**67.34%**	2.62Å	**58.93%**	1.78Å	**63.51%**	3.84Å	60.80%	4.41Å	38.26%

Baselines. We focus on the more challenging setting of designing all CDRs at once (Martinkus et al., 2024), rather than designing a single CDR at a time. Consequently, we do not compare our approach with those methods that are limited to designing a single CDR at a time. Although both AbDiffuser (Martinkus et al., 2024) and AbX (Zhu et al., 2024) are capable of designing all CDRs simultaneously, we exclude these methods from our comparison due to the unavailability of available source code for AbDiffuser and the absence of training scripts for AbX (Zhu et al., 2024). Ultimately, we choose **DiffAb** (Luo et al., 2022) and **dyMEAN** (Kong et al., 2023a) as our baselines because they serve as representative methods for the generative and discriminative categories of antibody co-design methods, respectively. For our method, we provide results for sequence diffusion implemented with both discrete and continuous diffusion. We denote the discrete version of the model as **MFDesign-D** and the continuous version as **MFDesign-C**.

Metrics. We adopt the following metrics to evaluate all methods: (1) **AAR** (%): measures the accuracy of the generated sequences of CDRs by comparing amino acid identity with native sequences; (2) **RMSD** (Å): calculates the root-mean-square deviation of C_α atom coordinates between generated and native CDRs. Following Zhu et al., we introduce additional metrics for the middle loop residues within CDR-H3, which primarily contribute to antigen binding: (3) **Loop-AAR** (%): applies AAR specifically to the middle loop residues of CDR-H3, assessing sequence recovery in this key region; (4) **Loop-RMSD** (Å): uses RMSD to evaluate the structural accuracy of the middle loop residues within CDR-H3. Furthermore, we use: (5) **IMP** (%): is the percentage of designed antibodies with enhanced binding energy (ΔG) compared to the original, assessed with Rosetta's *InterfaceAnalyzer* tool (Alford et al., 2017). For generative-based methods, DiffAb and our method, we generate 20 candidates per sample and report the averaged results.

For a fair comparison, we follow the same experimental setting as previous works (Luo et al., 2022; Kong et al., 2023a; Martinkus et al., 2024; Zhu et al., 2024), which involves re-designing CDRs within given co-crystal structures. In this setup, the sequences and structures of all non-CDR regions, including the framework and antigen, along with the antibody-antigen binding conformation, are provided. By using the *replacement* sampling technique mentioned in Sec. 2.2, our method enables CDRs re-design based on the given co-crystal structures. The results are presented below.

Results on Typical Antibodies. Table 1 and Table 2 present the results of simultaneously designed CDRs for typical antibodies, fully demonstrating the superiority of our method. From Table 1, we see that MFDesign-C achieves the best results in RMSD across all CDRs. Its performance in AAR is also superior to the baselines, except for CDR-H2. MFDesign-D, while not surpassing our own continuous version on CDR-L2, outperforms the baselines in AAR across the remaining CDRs by a notable margin. This reflects the suitability of each model for different data types: coordinates are continuous, so the continuous version generally performs better in RMSD; sequences are discrete, so the discrete version generally performs better in AAR. Notably, for the more challenging CDR-H3, our method improves AAR by 87.49% and reduces RMSD by 17.48% compared to the best baseline. Table 2 shows the results when further ignoring the relatively conserved regions at both ends of CDR-H3 and focusing only on the highly variable middle loop, as these residues are the primary contributors to antigen binding (Kong et al., 2023a; Zhu et al., 2024). In this case, the best performance in Loop-AAR among the baselines, achieved by dyMEAN, is only 20.80%, while MFDesign-D achieves an accuracy of 63.38%, representing a 204.71% improvement. In terms of Loop-RMSD, MFDesign-D attains a 17.89% reduction compared to the best baseline DiffAb. Besides, our method reaches 66.24% in the IMP metric. We attribute the superior performance of our method to the broad knowledge of protein interactions originally acquired by AF3-like model during pre-training. Moreover, the AF3-like model themselves already have good structure prediction capabilities, which may also explain the effectiveness of MFDesign in RMSD-related metrics.

Results on Nanobodies. Nanobodies, *a.k.a.* single-domain antibodies, consist of only a single heavy chain and are gaining increasing attention due to thier favorable properties, such as small high solubility and stability (Bannas et al., 2017). Their structure is simpler, but because they lack a light chain to assist in antigen binding, their CDR-H3 region tends to be longer compared to typical antibodies and nanobody-antigen complex complex data is scarcer than typical antibodies. This longer CDR-H3 aids in antigen binding but also makes it more challenging to design. In this study, we validate the effectiveness of our method for designing nanobodies, where the labeled data is even more less, and summarize the results in Table 3. We observe that for all methods, performance decreases compared to typical antibodies, which may be due to the limited data availability. The baseline DiffAb shows a significant drop, especially in nanobody CDR-H3, where its AAR is only 15.39%, an 30.86% decrease. In contrast, both the continuous and discrete versions of our method maintain an AAR above 60% for CDR-H3, with Loop-AARs also above 60%.

4 Conclusion

This paper proposes an antigen-conditioned antibody design model by extending AlphaFold3-like models with sequence-structure co-diffusion. By utilizing their acquired understanding of biomolecular interactions during pre-training, our method effectively overcomes the challenge posed by the limited availability of labeled antibody-antigen complex data. Our modified AF3-like model achieves superior performance over existing baselines in the co-design of antibody sequence and structure, while maintaining structure prediction accuracy comparable to the original AF3-like model.

References

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024. (Cited on pages 1, 11, 15 and 20)

Jared Adolf-Bryfogle, Oleks Kalyuzhniy, Michael Kubitz, Brian D Weitzner, Xiaozhen Hu, Yumiko Adachi, William R Schief, and Roland L Dunbrack Jr. Rosettaantibodydesign (rabd): A general framework for computational antibody design. *PLoS Computational Biology*, 14(4):e1006112, 2018. (Cited on pages 4 and 11)

Rebecca F Alford, Andrew Leaver-Fay, Jeliazko R Jeliazkov, Matthew J O'Meara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of Chemical Theory and Computation*, 13(6):3031–3048, 2017. (Cited on page 5)

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. (Cited on pages 2, 3 and 15)

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023. (Cited on page 11)

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021. (Cited on page 11)

Minkyung Baek, Ivan Anishchenko, Ian R Humphreys, Qian Cong, David Baker, and Frank DiMaio. Efficient and accurate prediction of protein structure using rosettafold2. *bioRxiv*, pp. 2023–05, 2023. (Cited on page 11)

Minkyung Baek, Ryan McHugh, Ivan Anishchenko, Hanlun Jiang, David Baker, and Frank DiMaio. Accurate prediction of protein–nucleic acid complexes using rosettafoldna. *Nature Methods*, 21(1): 117–121, 2024. (Cited on page 11)

Peter Bannas, Julia Hambach, and Friedrich Koch-Nolte. Nanobodies and nanobody-based human heavy chain antibodies as antitumor therapeutics. *Frontiers in Immunology*, 8:1603, 2017. (Cited on page 6)

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic Acids Research*, 28(1): 235–242, 2000. (Cited on page 11)

Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. Pyrosetta: a script-based interface for implementing molecular modeling algorithms using rosetta. *Bioinformatics*, 26(5):689–691, 2010. (Cited on pages 4 and 21)

Xinshi Chen, Yuxuan Zhang, Chan Lu, Wenzhi Ma, Jiaqi Guan, Chengyue Gong, Jincai Yang, Hanyu Zhang, Ke Zhang, Shenghao Wu, Kuangqi Zhou, Yanping Yang, Zhenyu Liu, Lan Wang, Bo Shi, Shaochen Shi, and Wenzhi Xiao. Protenix - advancing structure prediction through a comprehensive alphafold3 reproduction. *bioRxiv*, pp. 2025–01, 2025. (Cited on page 1)

Cyrus Chothia and Arthur M Lesk. Canonical structures for the hypervariable regions of immunoglobulins. *Journal of Molecular Biology*, 196(4):901–917, 1987. (Cited on pages 17 and 20)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019. (Cited on page 2)

Chai Discovery, Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhnikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. *bioRxiv*, pp. 2024–10, 2024. (Cited on page 1)

James Dunbar and Charlotte M Deane. Anarci: antigen receptor numbering and receptor classification. *Bioinformatics*, 32(2):298–300, 2016. (Cited on page 17)

James Dunbar, Konrad Krawczyk, Jinwoo Leem, Terry Baker, Angelika Fuchs, Guy Georges, Jiye Shi, and Charlotte M Deane. Sabdab: the structural antibody database. *Nucleic Acids Research*, 42 (D1):D1140–D1146, 2014. (Cited on pages 4 and 17)

Dave Epstein, Allan Jabri, Ben Poole, Alexei Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *Advances in Neural Information Processing Systems*, 36:16222–16239, 2023. (Cited on page 11)

Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *bioRxiv*, pp. 2021–10, 2021. (Cited on page 11)

Nate Gruver, Samuel Don Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. (Cited on pages 3 and 11)

Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations*, 2023. (Cited on page 11)

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. (Cited on pages 1, 3, 11 and 12)

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022. (Cited on pages 3 and 15)

Charles Janeway, Paul Travers, Mark Walport, Mark Shlomchik, et al. *Immunobiology: the immune system in health and disease*, volume 2. Garland Pub. New York, 2001. (Cited on page 1)

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Antibody-antigen docking and design via hierarchical structure refinement. In *International Conference on Machine Learning*, pp. 10217–10227. PMLR, 2022a. (Cited on pages 4 and 11)

Wengong Jin, Jeremy Wohlwend, Regina Barzilay, and Tommi S. Jaakkola. Iterative refinement graph neural network for antibody sequence-structure co-design. In *International Conference on Learning Representations*, 2022b. (Cited on pages 1 and 11)

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. (Cited on page 11)

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. (Cited on pages 1, 2, 12, 15 and 16)

Xiangzhe Kong, Wenbing Huang, and Yang Liu. End-to-end full-atom antibody design. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 17409–17429, 2023a. (Cited on pages 1, 4, 5, 11 and 21)

Xiangzhe Kong, Wenbing Huang, and Yang Liu. Conditional antibody design as 3d equivariant graph translation. In *The Eleventh International Conference on Learning Representations*, 2023b. (Cited on page 11)

Gideon D Lapidoth, Dror Baran, Gabriele M Pszolla, Christoffer Norn, Assaf Alon, Michael D Tyka, and Sarel J Fleishman. Abdesign: A n algorithm for combinatorial backbone design guided by natural conformations and sequences. *Proteins: Structure, Function, and Bioinformatics*, 83(8): 1385–1406, 2015. (Cited on page 11)

Marie-Paule Lefranc, Christelle Pommié, Manuel Ruiz, Véronique Giudicelli, Elodie Foulquier, Lisa Truong, Valérie Thouvenin-Contet, and Gérard Lefranc. Imgt unique numbering for immunoglobulin and t cell receptor variable domains and ig superfamily v-like domains. *Developmental & Comparative Immunology*, 27(1):55–77, 2003. (Cited on pages 20 and 21)

VI Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet Physics Doklady*, 1966. (Cited on page 18)

Tong Li, Robert J Pantazes, and Costas D Maranas. Optmaven–a new framework for the de novo design of antibody variable region models targeting specific antigen epitopes. *PLoS ONE*, 9(8): e105954, 2014. (Cited on page 11)

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022. (Cited on pages 4, 15 and 16)

Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. (Cited on pages 1, 4, 5, 11, 17, 20, 21 and 22)

Karolis Martinkus, Jan Ludwiczak, Wei-Ching Liang, Julien Lafrance-Vanasse, Isidro Hotzel, Arvind Rajpal, Yan Wu, Kyunghyun Cho, Richard Bonneau, Vladimir Gligorijevic, et al. Abdiffuser: full-atom generation of in-vitro functioning antibodies. *Advances in Neural Information Processing Systems*, 36, 2024. (Cited on pages 1, 2, 4, 5 and 11)

Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, 2022. (Cited on page 20)

Bo Qiang, Yuxuan Song, Minkai Xu, Jingjing Gong, Bowen Gao, Hao Zhou, Wei-Ying Ma, and Yanyan Lan. Coarse-to-fine: a hierarchical diffusion model for molecule generation in 3d. In *International Conference on Machine Learning*, pp. 28277–28299. PMLR, 2023. (Cited on page 11)

Arne Schneuing, Charles Harris, Yuanqi Du, Kieran Didi, Arian Jamasb, Ilia Igashov, Weitao Du, Carla Gomes, Tom L Blundell, Pietro Lio, et al. Structure-based drug design with equivariant diffusion models. *Nature Computational Science*, 4(12):899–909, 2024. (Cited on pages 4, 15 and 16)

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. (Cited on pages 1, 11 and 12)

Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017. (Cited on pages 4 and 19)

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020. (Cited on page 12)

Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. (Cited on pages 4, 11 and 15)

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. (Cited on page 12)

Zhen Wang, Ziqi Liu, Wei Zhang, Yanjun Li, Yizhen Feng, Shaokang Lv, Han Diao, Zhaofeng Luo, Pengju Yan, Min He, et al. Aptadiff: de novo design and optimization of aptamers based on diffusion models. *Briefings in Bioinformatics*, 25(6):bbae517, 2024. (Cited on page 11)

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023. (Cited on page 11)

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, et al. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, pp. 2024–11, 2024. (Cited on pages 1, 2, 4, 14, 19 and 21)

Jianzong Wu, Xiangtai Li, Yanhong Zeng, Jiangning Zhang, Qianyu Zhou, Yining Li, Yunhai Tong, and Kai Chen. Motionbooth: Motion-aware customized text-to-video generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. (Cited on page 11)

Tian Zhu, Milong Ren, and Haicang Zhang. Antibody design using a score-based diffusion model guided by evolutionary, physical and geometric constraints. In *Forty-first International Conference on Machine Learning*, 2024. (Cited on pages 1, 4, 5, 11, 20 and 21)

Appendix

CONTENTS

A Related Works

Diffusion Model. Diffusion-based generative methods (Song et al., 2021; Ho et al., 2020) have demonstrated remarkable effectiveness across various domains, such as image and video generation (Epstein et al., 2023; Wu et al., 2024). These models can be classified by the type of data they target: continuous or discrete, with the theoretical foundations of continuous diffusion being more developed and mature. Recently, diffusion models have also shed light on new possibilities in the scientific domain. In particular, continuous diffusion methods have excelled in tasks like molecular generation (Guan et al., 2023; Qiang et al., 2023) and protein structure design (Trippe et al., 2023; Watson et al., 2023), due to their suitability for continuous spatial coordinate data. Meanwhile, discrete diffusion methods have shown promise in handling sequential data, proving effective in designing protein sequences (Gruver et al., 2023) and DNA (Avdeyev et al., 2023; Wang et al., 2024), which consist of discrete elements like amino acids and nucleotides.

Protein Folding Model. Determining protein structures directly from sequences is a core challenge in biology, known as the protein folding problem, and is vital for understanding protein functions. AI has become a key tool in tackling this challenge. AlphaFold2 (Jumper et al., 2021) is a groundbreaking development in this domain, achieving unprecedented accuracy in predicting protein structures through training on extensive datasets from the Protein Data Bank (PDB) (Berman et al., 2000), setting new benchmarks in structural biology. Building on this success, AlphaFold-Multimer extends the approach to model interactions between multiple proteins (Evans et al., 2021). Further advancing this field, DeepMind introduce AlphaFold3 (Abramson et al., 2024), a diffusion-based method that not only improves prediction accuracy but also expands support for a wider range of interactions, including those involving nucleic acids and ligands. In parallel, aside from the AlphaFold series, the RoseTTAFold series emerged as a powerful option for the folding task, including models such as RoseTTAFold (Baek et al., 2021), RoseTTAFold2 (Baek et al., 2023), RoseTTAFoldNA (Baek et al., 2024).

Antibody Design. Most antibody design efforts focus on the design of CDRs. Traditional methods (Li et al., 2014; Lapidoth et al., 2015; Adolf-Bryfogle et al., 2018) for antibody design primarily rely on energy-based optimization, which can be quite time-consuming. In contrast, recent deep learning-based methods have been proposed to enhance antibody design, broadly divided into discriminative and generative models. Discriminative models (Jin et al., 2022b;a; Kong et al., 2023b;a) typically utilize graph neural networks to extract features from the context, including antigens and antibody framework regions, to predict the most probable CDRs structure and sequence. Generative models (Luo et al., 2022; Martinkus et al., 2024; Zhu et al., 2024), which are mainly diffusion-based, excel at modeling the complex distributions of both structure and sequence. By accurately capturing these distributions, they can sample a more diverse set of antibody candidates from them.

B A Quick Review of AlphaFold3-like Folding Models

AF3-like methods are structured as conditional diffusion models, consisting mainly of two components: the conditioning trunk and the diffusion module. These models also feature a confidence module, but it is trained independently and is not the primary focus here, so we do not discuss it. Below, notations with subscript i/ij are token-level representations, while l/lm are atom-level. The modules introduced in the original AF3 paper are **highlighted** in the following discussion, and corresponding details can be referenced in the supplementary materials of AF3[2].

Conditioning Trunk. This trunk processes raw inputs denoted as $\{\mathbf{f}^*\}$, such as the protein sequences to be predicted, MSA results, and structural template information. Initially, these inputs pass through an **InputEmbedder**, which performs basic encoding to obtain input features $\{\mathbf{s}_i^{\text{inputs}}\}$. These initial input features are used to derive the preliminary single and pair representations. The single and pair representations are then updated iteratively through a stack of three modules: **TemplateModule**, **MSAModule**, and **PairFormerStack**. Finally, the input features $\{\mathbf{s}_i^{\text{inputs}}\}$ obtained at the beginning, along with the iteratively updated single and pair representations, denoted as $\{\mathbf{s}_i^{\text{trunk}}\}$ and $\{\mathbf{z}_{ij}^{\text{trunk}}\}$ respectively, are fed to the following diffusion module to condition the structural diffusion process.

[2] 👉 Click here to access the supplementary materials of AF3.

Structure Diffusion. AF3 adopts EDM (Karras et al., 2022) for structure diffusion training. Notably, EDM differs from typical diffusion models (Ho et al., 2020; Song et al., 2021) that sample a timestep to determine the noise level based on a pre-defined noise schedule during training. Instead, EDM directly samples the noise level by drawing the logarithm of the noise intensity from a defined Gaussian distribution.

Speicifically, during AF3's training, the noise level σ is sampled from the distribution $\sigma_{\text{data}} \cdot \exp(-1.2 + 1.5 \cdot \mathcal{N}(0, 1))$ and is fed, along with the three outputs from the previous trunk, into the **DiffusionConditioning** module. In this module, relative position encoding between residue tokens is incorporated into $\{\mathbf{z}_{ij}^{\text{trunk}}\}$ to facilitate both token-level and atom-level self-attention (Vaswani, 2017) used later, resulting in the output $\{\mathbf{z}_{ij}\}$. Besides, within this module, the sampled noise level is embedded using random Fourier features (Tancik et al., 2020) and then integrated with $\{\mathbf{s}_i^{\text{inputs}}\}$ and $\{\mathbf{s}_i^{\text{trunk}}\}$ to construct $\{\mathbf{s}_i\}$.

Next, following EDM, AF3 scales the noisy atom coordinates $\{\mathbf{x}_l^{\text{noisy}}\}$ to unit variance, which gives $\{\mathbf{r}_l^{\text{noisy}}\}$. $\{\mathbf{r}_l^{\text{noisy}}\}$, $\{\mathbf{s}_i^{\text{trunk}}\}$ and $\{\mathbf{z}_{ij}\}$ are further processed in the **AtomAttentionEncoder**. Initially, this module yields three atom-level representations: $\{\mathbf{q}_l\}$, $\{\mathbf{c}_l\}$, $\{\mathbf{p}_{lm}\}$. These representations are all based on atomic features from the raw inputs and incorporate additional information from $\{\mathbf{r}_l^{\text{noisy}}\}$, $\{\mathbf{s}_i^{\text{trunk}}\}$ and $\{\mathbf{z}_{ij}\}$, respectively. Then, sequence-local atom attention is applied to update $\{\mathbf{q}_l\}$ using $\{\mathbf{c}_l\}$ and $\{\mathbf{p}_{lm}\}$. The updated $\{\mathbf{q}_l\}$ are further aggregated to form the token-level representations $\{\mathbf{a}_i\}$.

Subsequently, $\{\mathbf{s}_i\}$ is used to update $\{\mathbf{a}_i\}$. The updated $\{\mathbf{a}_i\}$, along with outputs $\{\mathbf{a}_i\}$ and $\{\mathbf{z}_{ij}\}$ from the previous **DiffusionConditioning** module, is fed into the **DiffusionTransformer** module to perform full self-attention on the token level. This module further updates $\{\mathbf{a}_i\}$ and layer normalization is applied to $\{\mathbf{a}_i\}$.

The residue-level representations $\{\mathbf{a}_i\}$, together with the atom-level representations $\{\mathbf{q}_l\}$, $\{\mathbf{c}_l\}$ and $\{\mathbf{p}_{lm}\}$ are then inputted into the **AtomAttentionDecoder** module, which yields $\{\mathbf{r}_l^{\text{update}}\}$. $\{\mathbf{r}_l^{\text{update}}\}$ is rescaled back and combined with the input noisy structure $\{\mathbf{r}_l^{\text{noisy}}\}$, in line with EDM, to obtain the final predicted denoised structure $\{\mathbf{x}_l^{\text{denoised}}\}$.

The structure diffusion training process outlined above is encapsulated within the **DiffusionModule** of AF3, as formalized in Algorithm 20 of AF3's supplementary materials. In summary, this process uses the outputs from the trunk, in conjunction with the raw inputs, to further construct atom-level and token-level representations. These representations are then fed to the structure denoising network for recovery.

For inference, AF3 adopts the EDM stochastic sampler and from a sequence of 200 decreasing levels defined by:

$$\left\{ \sigma_{\text{data}} \cdot \left(\sigma_{\text{max}}^{1/\rho} + \frac{t}{200 - 1} \cdot \left(\sigma_{\text{min}}^{1/\rho} - \sigma_{\text{max}}^{1/\rho} \right) \right)^{\rho} \;\middle|\; t = 0, 1, \ldots, 199 \right\},$$

where σ_{data}, σ_{max}, σ_{min} and ρ are hyper-parameters.

C Supplementary Details about the Our Method

Due to space limitations in the main text, this section presents supplementray details on the methodological aspects of MFDesign. For implementation specifics, we defer to Appendix F.2.

C.1 Algorithmic Formulation of CoDiffusionModule

For a clearer and more intuitive understanding, we formalize the proposed **CoDiffusionModule** in Algorithm S1.

C.2 Avoiding Data Leakage

Even when the CDRs in the antibody query sequences for MSA are filled with the unknown token **<X>**, the MSA results contain a significant number of sequences that, while not exactly identical

Algorithm S1 Sequence and Structure Co-diffusion Module: **CoDiffusionModule**

Input: noisy structure $\{\mathbf{x}_l^{\text{noisy}}\}$, noisy sequence $\{\mathbf{k}_i^{\text{noisy}}\}$, noise level for structure σ, raw inputs $\{\mathbf{f}^*\}$, trunk's outputs $\left\{\{\mathbf{s}_i^{\text{inputs}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}^{\text{trunk}}\}\right\}$, hyper-parameter $\{\sigma_{\text{data}}\}$;

$\{\mathbf{s}_i^{\text{replaced}}\} \leftarrow$ substitute the specified dimensions within $\{\mathbf{s}_i^{\text{inputs}}\}$ with $\{\mathbf{k}_i^{\text{noisy}}\}$

 # Pre-conditioning for diffusion

$\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\} =$ **DiffusionConditioning** $\left(\sigma, \{\mathbf{f}^*\}, \{\mathbf{s}_i^{\text{replaced}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}^{\text{trunk}}\}, \sigma_{\text{data}}\right)$ ▶ Refer to Alg. 21 in AF3

 # Scale coordinates with approximately unit variance

$\mathbf{r}_l^{\text{noisy}} = \mathbf{x}_l^{\text{noisy}} / \sqrt{\sigma^2 + \sigma_{\text{data}}^2}$

 # Sequence-local atom attention and aggregation to coarse-grained tokens

$\{\mathbf{a}_i\}, \{\mathbf{q}_l\}, \{\mathbf{c}_l\}, \{\mathbf{p}_{lm}\} =$ **AtomAttentionEncoder** $\left(\{\mathbf{f}^*\}, \{\mathbf{r}_l^{\text{noisy}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}\}\right)$ ▶ Refer to Alg. 5 in AF3

 # Full self-attention on token level

$\mathbf{a}_i \mathrel{+}=$ LinearNoBias(LayerNorm($\mathbf{s}_i$))

$\{\mathbf{a}_i\} =$ **DiffusionTransformer** $(\{\mathbf{a}_i\}, \{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\})$ ▶ Refer to Alg. 23 in AF3

$\mathbf{a}_i =$ LayerNorm $(\mathbf{a}_i)$

 # Structure denosing

$\left\{\mathbf{r}_l^{\text{update}}\right\} =$ **AtomAttentionDecoder** $(\{\mathbf{a}_i\}, \{\mathbf{q}_l\}, \{\mathbf{c}_l\}, \{\mathbf{p}_{lm}\})$ ▶ Refer to Alg. 6 in AF3

$\mathbf{x}_l^{\text{denoised}} = \sigma_{\text{data}}^2 / \left(\sigma_{\text{data}}^2 + \sigma^2\right) \cdot \mathbf{x}_l^{\text{noisy}} + \sigma_{\text{data}} \cdot \sigma / \sqrt{\sigma_{\text{data}}^2 + \sigma^2} \cdot \mathbf{r}_l^{\text{update}}$

 # Sequence denoising

$\{\mathbf{k}_i^{\text{denoised}}\} =$ **TokenDenoiser** $(\{\mathbf{a}_i\})$

Output: denoised structure $\{\mathbf{x}_l^{\text{denoised}}\}$, denoised sequence $\{\mathbf{k}_i^{\text{denoised}}\}$;

to the antibody sequences, have sequences in positions corresponding to the antibody CDRs highly consistent with the CDRs ground-truth sequences. This can cause the model to learn to predict CDR sequences directly from the MSA results, leading to data leakage. To address this, in both our training and inference, we exclude sequences where the regions corresponding to the CDRs in the query antibody sequence have a similarity to the CDRs that exceeds a threshold, as detailed in Appendix D.4. This processing is to ensure fair benchmarking. In practical antibody design, filtering is not required.

C.3 Training Strategy

Similar to AF3 models that use token cropping for training, we also implement this technique. Our training process is divided into four stages as follows:

- **1st-stage:** We only input antibody sequences, using a maximum token size of 256 to accommodate the VH and VL sequences. This allows for a larger batch size, enabling the model to quickly learn to design CDRs and predict antibody structures based only on antibody sequences.

- **2nd-stage & 3rd-stage:** We continue to input full VH and VL sequences while randomly selecting tokens from the antigen epitope and the nearby regions to introduce the antigen context. Both stages follow the same cropping method detailed in Alg. S2, differing only in maximum token size: 384 in the second stage and 512 in the third.

- **4th-stage:** We perform full-parameter fine-tuning, which may affect the model's ability to predict structure learned during pre-training. Besides, considering the first three stages focus on predicting structures around the antibody and nearby antigen regions, the model might underperform when predicting structures of antigen regions far from the antibody. To address this, we implement a sampling strategy where there is a 50% probability of selecting tokens from regions around the antibody and a 50% probability of selecting tokens from any area within the complex. This approach helps the model retains its newly acquired antibody co-design capability while also preserving its pre-trained ability to predict complex structures. The maximum token size for this stage is kept at 512.

Algorithm S2 Cropping Strategy Adopted in Our 2nd & 3rd & 4th Training Phases

Input: tokens, max token number $N_{\text{token}}^{\text{max}}$, max atom number $N_{\text{atom}}^{\text{max}}$, **neighborhood_sizes** = [0, 2, 4, . . ., 40];
cropped_tokens = []
Add all tokens in VH and VL sequence of the antibody to **cropped_tokens**
Sample **neighborhood_size** uniformly at random from **neighborhood_sizes**
Sample **center_token** uniformly within the tokens in the 6 CDRs of VH and VL
sorted_epitope_tokens ← sort tokens in the epitope of antigen by ascending their distance to the selected **center_token**
for token in sorted_epitope_tokens do
  Let **chain_tokens** be the entries in the same chain as **token** (including **token**)
  if len(**chain_tokens**) ≤ **neighborhood_size then**
    selected_tokens = **chain_tokens**
  else
    selected_tokens = [**token**]
    min_idx ← index of **token** in **chain_tokens**
    max_idx ← index of **token** in **chain_tokens**
    while len(**selected_tokens**) < **neighborhood_size do**
      min_idx = **min_idx** - 1
      max_idx = **max_idx** + 1
      Let **selected_tokens** be the entries ∈ **chain_tokens** whose index in **chain_tokens** ∈ [**min_idx**, **max_idx**]
    end while
  end if
  Let **new_tokens** be the entries in **selected_tokens** that are not present in **cropped_tokens**
  if adding **new_tokens** to **cropped_tokens** would exceed max token number $N_{\text{token}}^{\text{max}}$ or max atom number $N_{\text{atom}}^{\text{max}}$ **then**
    Break the **for** loop
  else
    Add **new_tokens** to **cropped_tokens**
  end if
end for
Output: cropped_tokens;

C.4 Token Cropping

As discussed in the Sec. 2.2, our model training consists of four stages, involving three token cropping strategies. The first strategy, employed in the first stage, is straightforward, involving inputting only the VH and VL tokens of the antibodies.

The other strategies are used in later stages. One of them, used specifically in the fourth stage, is a random token sampling method. For this, we directly adopt the cropping technique from the Boltz-1 (Wohlwend et al., 2024) paper, specifically utilizing the approach detailed in their Algorithm 2. This cropping method effectively handles complex of varying size by interpolating between *spatial* and *contiguous* cropping strategies. It defines "neighborhoods" around specific tokens, which are incrementally added based on their distance from a randomly selected crop center. By varying the size of neighborhoods, the algorithm can seamlessly transition from *spatial* to *contiguous* cropping.

The last strategy, used in the second, third, and fourth stages, focuses on cropping tokens around the antibody. This method, based on the above random token sampling method (Algorithm 2 from the Boltz-1 paper), is formalized in Algorithm S2 of our work, where we provide a comprehensive description of its tailored implementation for our specific need. In brief, this cropping approach first includes the VH and VL sequences of the antibody. Then, it randomly selects a token from the six CDRs as the center token. Tokens located at the antigen epitopes are then sorted in ascending order based on their distance to this selected center token. In this work, we define a residue on the antigen surface as an epitope residue if any internal atom is within a distance of less than a threshold of 10 Åfrom the atoms of the antibody CDRs. The remaining process follows Algorithm 2 from Boltz-1.

C.5 Training Objective

In this work, we aim to tackle the task of co-designing antibody sequences and structures. For this purpose, we define two separate loss functions: one for the sequence diffusion training and another for the structure diffusion training. The structure diffusion loss, $\mathcal{L}_{\text{structure}}$, is directly taken from AF3, as defined by Equation 6 in AF3's supplementary materials. The sequence diffusion loss function is denoted as $\mathcal{L}_{\text{sequence}}$. Our final loss $\mathcal{L}$ is simply the sum of these two components:

$$\mathcal{L} = \mathcal{L}_{\text{sequence}} + \mathcal{L}_{\text{structure}},$$

Next, we give the detailed definition of $\mathcal{L}_{\text{sequence}}$ for the version using discrete diffusion for sequence. We adopt the D3PM-absorbing (Austin et al., 2021), a discrete diffusion framework, for our sequence diffusion here. We denote the clean sample as $\{\mathbf{k}_i\}_0$ and we perform sequence diffusion only the CDRs within $\{\mathbf{k}_i\}_0$. The forward diffusion process $p\left(\{\mathbf{k}_i\}_t \mid \{\mathbf{k}_i\}_0\right)$ gradually corrups CDRs tokens by replacing them with special token **<X>** at each timestep t, via a discrete transition matrix that defines the probability of mutating a residue into **<X>**. The prior distribution at $t = T$ is a sequence where residues within CDRs are all **<X>** tokens. Our denoising model, **TokenDenoiser**, parameterized by θ, $p_\theta\left(\{\hat{\mathbf{k}}_i\}_0 \mid \{\mathbf{k}_i\}_t, t)\right)$, is trained to recover the original sequence from the absorbing version $\{\mathbf{k}_i\}_t$. To optimize the parameter set θ, we maximize the likelihood of the denoising process on ground-truth tokens:

$$\mathcal{L}_{\text{sequence}} = \mathbb{E}_{\{\mathbf{k}_i\}_0, t}\left[-\log p_\theta\left(\{\mathbf{k}_i\}_0 \mid \{\mathbf{k}_i\}_t\right)\right], \text{where } \{\mathbf{k}_i\}_t \sim p\left(\{\mathbf{k}_i\}_t \mid \{\mathbf{k}_i\}_0\right).$$

C.6 Sampling

In this paper, two sampling methods are employed: one for the antibody sequence and structure co-design task, and the other for structure prediction. We present our two sampling algorithms for the method employing discrete diffusion for sequence in this section.

C.6.1 Sampling w/ *Replacement* Technique

The structure follows the sampling method from AF3 (Abramson et al., 2024) (refer to Algorithm 18 in AF3), which adopts the EDM (Karras et al., 2022) sampling method. For sequence sampling, we use the standard D3PM-absorbing (Austin et al., 2021) sampling method.

In particular, when designing antibody CDRs, experimentally resolved co-crystal structures are provided, where all non-CDR regions (antibody framework sequences/structures, antigen sequences/structures, and the antibody-antigen binding pose) are given. This means our sampling must be conditioned on these fixed components.

In this work, we utilize a *replacement* sampling technique to allow the design of our CDRs to be conditioned on the given co-crystal structures. This technique is initially proposed and widely adopted in the image inpainting task (Lugmayr et al., 2022). It has also been used in molecule and protein generation tasks (Schneuing et al., 2024; Trippe et al., 2023) to condition the sampling on given motifs. In simple terms, it involves replacing the generated contents corresponding to the fixed parts with their forward noised counterparts. In our sampling, we need to align the given ground-truth co-crystal structure to the denoised structure and then perform *replacement* operation.

Assuming the input has already been processed by the conditioning trunk to yield $\{\mathbf{s}_i^{\text{inputs}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}^{\text{trunk}}\}$, we then focus on the subsequent sampling process. We formalize our sampling method for antibody sequence and structure co-design in Algorithm S3. In fact, the raw inputs $\{\mathbf{f}^*\}$ include $\{\mathbf{k}_i^{\text{input}}\}$, but we choose to highlight them separately.

C.6.2 Sampling w/o *Replacement* Technique

For the structure prediction task, we sample from scratch. Therefore, we do not need to input a given structure as in Algorithm S3, nor do we use the *replacement* sampling technique. As a result, the sampling algorithm used here omits a given structure as input and also skips Lines 14-15 from Algorithm S3, which correspond to the *replacement* technique, while the rest remains consistent with Algorithm S3. If the input antibody sequence contains the **<X>** token, we will simultaneously predict its amino acid type while predicting the structure.

C.7 Sequence Diffusion in the Continuous Space

In the main body of this paper, we present an implementation utilizing D3PM-absorbing (Austin et al., 2021), a discrete diffusion method, for sequence diffusion. Additionally, we also implement a continuous version. Following the methodology of prior research (Hoogeboom et al., 2022), we introduce Gaussian noise directly to the one-hot encoding of residue types to handle discrete data through continuous diffusion.

Algorithm S3 Sampling with *Replacement* Method

Input: raw inputs $\{\mathbf{f}^*\}$, trunk's outputs $\left\{\{\mathbf{s}_i^{\text{inputs}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}^{\text{trunk}}\}\right\}$,
given ground-truth co-crystal structure $\{\mathbf{x}_l^{\text{GT}}\}$, input sequence $\{\mathbf{k}_i^{\text{input}}\}$ with CDRs marked in **<X>**,
hyper-parameters required by EDM (Karras et al., 2022) sampling method $\{\sigma_{\text{data}}, \sigma_{\max}, \sigma_{\min}, \rho, \gamma_0, \gamma_{\min}, \lambda, \eta\}$;

for $t \in [0, 1, 2, \ldots, 199]$ **do**
  # Define noise schedule for structure sampling
  $c_t = \sigma_{\text{data}} \cdot \left(\sigma_{\max}^{1/\rho} + \frac{t}{200-1} \cdot \left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho}$
  # Define noise schedule for sequence sampling
  $m_t = 1 - \frac{t}{200-1}$
end for
 # Initialize structure via sampling from a Gaussian noise
$\mathbf{x}_l \sim c_0 \cdot \mathcal{N}\left(\vec{\mathbf{0}}, \mathbf{I}_3\right)$
 # Input sequences naturally serve as initialization because CDRs are all already in the absorbing state
$\{\mathbf{k}_i^{\text{noisy}}\} = \{\mathbf{k}_i^{\text{input}}\}$
for $t \in [1, 2, \ldots, 199]$ **do**
  $\{\mathbf{x}_l\} = \textbf{CentreRandomAugmentation}(\{\mathbf{x}_l\})$ ▶ Refer to Algorithm 19 in AF3
  $\gamma = \gamma_0$ **if** $c_t > \gamma_{\min}$ **else** 0
  $\sigma = c_{t-1}(\gamma + 1)$
  $\xi_l = \lambda\sqrt{\sigma_t^2 - c_{t-1}^2} \cdot \mathcal{N}\left(\vec{\mathbf{0}}, \mathbf{I}_3\right)$
  $\mathbf{x}_l^{\text{noisy}} = \mathbf{x}_l + \xi_l$
  Same as AF3, which follows EDM sampling method
  $\{\mathbf{x}_l^{\text{denoised}}\}, \{\mathbf{k}_i^{\text{denoised}}\} = \textbf{CoDiffusionModule}\left(\{\mathbf{x}_l^{\text{noisy}}\}, \{\mathbf{k}_i^{\text{noisy}}\}, \sigma, \{\mathbf{f}^*\}, \{\mathbf{s}_i^{\text{inputs}}\}, \{\mathbf{s}_i^{\text{trunk}}\}, \{\mathbf{z}_{ij}^{\text{trunk}}\}\right)$ ▶ Alg. S1
  # Replacement sampling technique (Lines 14-15) (Lugmayr et al., 2022; Schneuing et al., 2024)
  $\mathbf{x}_l^{\text{GT-aligned}} = \textbf{RigidAlign}(\{\mathbf{x}_l^{\text{GT}}\}, \{\mathbf{x}_l^{\text{denoised}}\})$ ▶ Refer to Algorithm 28 in AF3
  $\mathbf{x}_l^{\text{denoised}} = \mathbf{x}_l^{\text{denoised}}$ **if** atom l is within CDRs **else** $\mathbf{x}_l^{\text{GT-aligned}}$
  $\delta_l = (\mathbf{x}_l - \mathbf{x}_l^{\text{denoised}})/\sigma$
  $dt = c_t - \sigma$
  $\mathbf{x}_l \leftarrow \mathbf{x}_l^{\text{noisy}} + \eta \cdot dt \cdot \delta_l$
  Same as AF3, which follows EDM sampling method
  # Randomly sample a probability from a uniform distribution between 0 and 1 for each token
  $p_i \sim \mathcal{U}(0, 1)$
  $\mathbf{k}_i^{\text{denoised}} = \mathbf{k}_i^{\text{denoised}}$ **if** residue i is within CDRs **else** $\mathbf{k}_i^{\text{input}}$
  $\mathbf{k}_i^{\text{noisy}} =$ one-hot encoding of **<X>** **if** $p_i < m_t$ **and** residue i is within CDRs **else** $\mathbf{k}_i^{\text{denoised}}$
  Follow D3PM-absorbing sampling method
end for
$\mathbf{x}_l^{\text{GT-aligned}} = \textbf{RigidAlign}(\{\mathbf{x}_l^{\text{GT}}\}, \{\mathbf{x}_l\})$ ▶ Refer to Algorithm 28 in AF3
$\mathbf{x}_l = \mathbf{x}_l$ **if** atom l is within CDRs **else** $\mathbf{x}_l^{\text{GT-aligned}}$
$\{\mathbf{k}_i\} = \{\mathbf{k}_i^{\text{denoised}}\}$
Output: predicted final sequence $\{\mathbf{k}_i\}$, predicted final structure $\{\mathbf{x}_l\}$;

As noted in Appendix B, the noise level for AF3's structure diffusion training σ is sampled from $\sigma_{\text{data}} \cdot \exp(-1.2 + 1.5 \cdot \mathcal{N}(0, 1))$. For inference, AF3 defines a sequence of decreasing noise levels: $\left\{\sigma_{\text{data}} \cdot \left(\sigma_{\max}^{1/\rho} + \frac{t}{200-1} \cdot \left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho} \mid t = 0, 1, \ldots, 199\right\}$. σ_{data}, $\sigma_{\max}$, $\sigma_{\min}$ and ρ are hyper-parameters. For structure diffusion, we adhere to the default settings in AF3 for these parameters.

Since our diffusion now is also based on continuous diffusion, we can also apply the EDM (Karras et al., 2022) training method to sequence diffusion. Similar to structure diffusion, we directly sample the noise level instead of first sampling a time step and then using a predefined noise scheduler to obtain the corresponding noise level. By sharing the noise level between structure and sequence, we can easily align their training. We define a scaling factor ω and set it to $1/4$ in our experiments. The noise level for the structure is sampled from $\sigma_{\text{data}} \cdot \exp(-1.2 + 1.5 \cdot \mathcal{N}(0, 1))$, and then this sampled noise level is multiplied by ω to serve as the noise level for the sequence. This ensures that the noise levels for the sequences are proportional to those for the structure. Once the noise level for the structure is determined, the noise level for the sequence diffusion can be obtained using ω, and noise is then added to the one-hot encodings of tokens to obtain $\{\mathbf{k}_i^{\text{noisy}}\}$. Diffusion is still applied only to the CDRs.

In this following, we present how to input the corresponding noisy sequence into the **CoDiffusion-Module** at each sampled timestep. During training, we add noise to the one-hot encodings of tokens to obtain $\{\mathbf{k}_i^{\text{noisy}}\}$. Similar to the method used in discrete sequence diffusion, we then replace the specified dimension in $\{\mathbf{s}_i^{\text{inputs}}\}$ with $\{\mathbf{k}_i^{\text{noisy}}\}$. For each denoising step in inference, we directly add

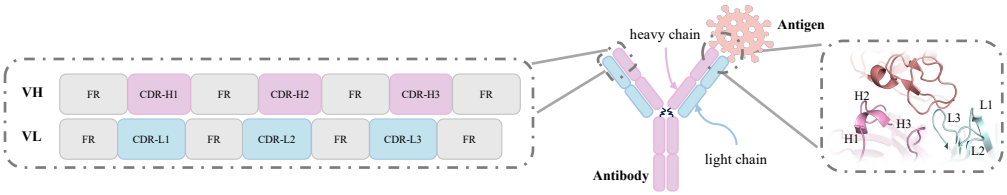

Figure S1: **Illustration of an antibody-antigen complex.** Typical antibodies consist of heavy and light chains, each with a variable domain (VH and VL, respectively) composed of three CDRs and framework regions. CDRs are highly variable and mainly determine the antibody specificity for antigens. In contrast, nanobodies consist of only a heavy chain.

noise to the token-level probability distributions, $\{\mathbf{h}_i\}$, which represents the probability distribution of each amino acid type for each token. This is obtained by inputting previously predicted $\{\mathbf{k}_i^{\text{denoised}}\}$ after last step into the SOFTMAX function. Then, we replace the specified dimensions in $\{\mathbf{s}_i^{\text{inputs}}\}$ with $\{\mathbf{h}_i\}$.

C.8 LIMITATIONS AND FUTURE WORK

One limitation of our current approach is that we only generate backbone atoms for CDRs, requiring an additional side-chain packing step to obtain full-atom structures. Accordingly, we aim to develop an end-to-end model capable of direct all-atom structure prediction in the future. Besides, we currently use full-parameter fine-tuning and plan to explore more efficient strategies to reduce computational costs while maintaining result quality.

D DATA PREPROCESS

Here, we present a detailed description of our data processing pipeline for the Structural Antibody Database (**SAbDab**) (Dunbar et al., 2014), with the goal of establishing a standardized approach that facilitates reproducibility and enables future research endeavors in this field. As a comprehensive repository of antibody structures that undergoes weekly updates, SAbDab provides an invaluable resource for structural antibody research. In Fig. S1, we illustrate a typical antibody-antigen complex structure. The data used in this work are collected from SAbDab up to November 27, 2024. The full data processing scripts and processed data will be released upon acceptance.

D.1 PRELIMINARY DATA FILTERING

Based on the summary file in TSV format provided by SAbDab, we only include samples whose annotated `antigen_type` field falls into one of the following five categories:

- `protein`
- `protein | protein`
- `protein | protein | protein`
- `protein | protein | protein | protein`
- `protein | protein | protein | protein | protein`

Furthermore, we only consider antibodies that either contain both heavy and light chains or consist of a single heavy chain (*i.e.*, nanobody), with a resolution no worse than 4.5 Å. Following DiffAb (Luo et al., 2022), we also utilize the Chothia scheme (Chothia & Lesk, 1987) for numbering the CDRs. After obtaining antibody sequences through parsing SAbDab's Chothia-scheme numbered antibodies, we further validate the CDR positions using the AbNumber[3] toolkit, which is based on ANARCI (Dunbar & Deane, 2016). Moreover, we filter out antibody sequences that can not be correctly identified by AbNumber. These problematic sequences are found to either have incorrect lengths, missing regions among FR1/CDR1/FR2/CDR2/FR3/CDR3/FR4, or notably long CDR3 regions (*e.g.* chain I in PDB ID: 4k3e, which is an antibody heavy chain). Note that DiffAb also filters out sequences with abnormally long CDR3 regions in their preprocessing.

[3] https://github.com/prihoda/AbNumber

D.2 DEDUPLICATION

In the TSV format summary file provided by SAbDab, there may be multiple antibody-antigen complex samples that correspond to the same PDB structure. We closely examine the variable domain sequences of the antibody heavy chains from these samples derived from the same PDB structure. After parsing the PDB structures and using the AbNumber tool to calibrate the CDRs regions, we employ AbNumber's alignment tool for pairwise comparisons and obtain some observations. Based on these observations, we develop a deduplication algorithm for these samples. Below, we present several typical examples of comparisons found in this process, focusing only on the CDRs sequences. First, consider the following pair of examples:

- VH sequence of Chain **H** in PDB **1bj1**:

  —FR1— GYTFTNY —FR2— NTYTGE —FR3— YPHYYGSSHWYFDV —FR4—

- VH sequence of Chain **K** in PDB **1bj1**:

  —FR1— GYTFTNY —FR2— NTYTGE —FR3— YPHYYGSSHWYFDV —FR4—

Both samples are associated with the same PDB structure, and their CDRs sequences are completely identical. For such samples, we believe that retaining just one is sufficient. We choose to retain the one with the longer VH sequence, as it provides more complete information. Now, consider the next set of examples:

- VH sequence of Chain **J** in PDB **6tyl**:

  —FR1— GGIVHIS —FR2— PSNGD —FR3— FLRHTASASYNNY —FR4—

- VH sequence of Chain **E** in PDB **6tyl**:

  —FR1— GGIVHIS —FR2— PSNGD —FR3— FSYNNY —FR4—

In the second set, the CDR-H1 and CDR-H2 regions are identical, but differences exist in the CDR-H3 sequence, where the lower sequence is noticeably shorter than the upper one. Upon closer inspection, it can be observed that the lower sequence lacks some amino acids present in the upper sequence. This situation may arise due to some unexpected uncertainties during the data collection process of the PDB structure. In our study, we regard these two sequences as essentially the same, and it is evident that the upper sequence provides more complete information, so we discard the lower sample. Now, consider the third set of examples:

- VH sequence of Chain **K** in PDB **7xj6**:

  —FR1— GFTFSSS —FR2— VVGSGN —FR3— PNCNSTTCHDGFDI —FR4—

- VH sequence of Chain **E** in PDB **7xj6**:

  —FR1— GGTFSSY —FR2— IPILGI —FR3— GTEYGDYDVSHD —FR4—

In the third set of examples, the sequences of CDR-H1 and CDR-H2 have the same length, but the amino acid types differ at certain positions. In the CDR-H3 sequence, although the lower one is shorter than the upper, this is not a case of missing parts as in the second example but rather indicates significant sequence differences. Thus, we consider these two chains as fundamentally different antibody sequences, and we retain both.

Based on these findings, we design the following algorithm to deduplicate the data obtained after processing in Section D.1. First, we use the Levenshtein distance to measure the differences between the CDRs sequences of two chains. The Levenshtein distance (Levenshtein, 1966) between two strings is defined as the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other. In the examples above, the Levenshtein distance values for the three sets of sequences are 0, 7, and 20, respectively, illustrating varying degrees of similarity and difference.

We concatenate the CDRs sequences of each sample's VH sequence into a single string and then calculate the Levenshtein distance between these concatenated strings for each pair of samples when assessing differences. In our study, we set a threshold of 9 for the Levenshtein distance; if the distance

Algorithm S4 SAbDab Deduplication

```
Input: a dataframe DF gathering samples sampled from Sec. D.1, distance_threshold;
all_reserved = []
# Group samples according to related PDB_ID and then iterate over each group
for group in DF.groupby('PDB_ID') do
    # Sort the samples in descending order within current group based on the length of the VH sequence
    sorted_group = group.sort_values(by='VH_SEQUENCE', key=lambda x:x.str.len(), ascending=False)
    # Reserve the sample with the longest VH sequence within the group
    reserved=[sorted_group.iloc[0]]
    # Iterate over every sample after the first one within the group
    for entry in sorted_group.iloc[1:] do
        should_add_flag = True
        for reserved_entry in reserved do
            distance=levenshtein_distance(entry['VH_SEQUENCE'], reserved_entry['VH_SEQUENCE'])
            if distance ≤ distance_threshold then
                should_add_flag = False
                break
            end if
        end for
        if should_add_flag then
            reserved.append(entry)
        end if
    end for
    # Merge reserved samples
    all_reserved += reserved
end for
filtered_DF = DataFrame(all_reserved)
Output: a dataframe filtered_DF composed of remaining data after Sec. D.2;
```

does not exceed 9, the sequences are considered not significantly different and likely represent the same sequence, in which case one should be discarded. As some samples are nanobodies without a light chain, we only compare the CDRs of the VH sequence for all samples.

The algorithm is as follows: we group the samples by their associated PDB ID. For each group, we maintain a list to store the samples that are ultimately retained within the group. We first sort the samples in descending order based on the length of the VH sequence and add the longest sequence to the list, as it is considered to provide the most comprehensive information. Then, we iterate through the remaining sequences: if the Levenshtein distance to any sample in the list does not exceed the threshold, it is considered a duplicate; if the Levenshtein distance to all samples in the list exceeds the threshold, it is retained and added to the list. This process is detailed in Algorithm S4.

Following the processing detailed in Section D.1, we initially have 11,009 samples associated with 5,762 PDB structures. Upon completing the deduplication step, we retain 6,347 samples.

D.3 Train & Validation & Test Sets Curation

As discussed in Section 3 of the main text, it is essential to consider the release date of the data to prevent data leakage when partitioning it into training, validation, and test sets.

We begin by applying MMSeqs2 (Steinegger & Söding, 2017) to cluster all antibodies based on their CDR-H3 sequences, using a 50% sequence identity threshold. These clusters form the basis for subsequent data partitioning and filtering.

Clusters that contain any structures with release dates prior to the cut-off date of September 30, 2021, are entirely included in the training set. This ensures that all pre-cut-off date data is used for training and avoids any risk of leakage into validation or test phases. For those clusters with data released on or after the cut-off date, we further divide them into validation and test sets to ensure these sets contain completely unseen data. The overall target ratio for the training, validation, and test clusters is approximately 9:0.5:0.5.

Due to Boltz-1's (Wohlwend et al., 2024) memory limitations, which do not support structures with more than 2000 residues/tokens on standard GPUs with 80GB memory (refer to this GitHub issue:

https://github.com/jwohlwend/boltz/issues/17), we also filter out samples in the validation and test sets where the combined antibody-antigen complex exceeds 2000 residues.

Moreover, we incorporate two specific filters inspired by AlphaFold3 (Abramson et al., 2024) to improve data quality:

- Remove any sample where a complex contains a single chain with all residues marked as unknown.
- Exclude protein chains with consecutive C_α atoms that are more than 10 Åapart. This filter is applied only to the antibody chains and not to the antigen chains.

As a result of this process, we obtained 5,843 training samples from 2,570 clusters, 187 validation samples from 142 clusters, and 204 test samples from 144 clusters.

D.4 MSA Construction

The Boltz-1 repository has already provided a way to obtain MSA results through the ColabFold (Mirdita et al., 2022) API. However, with limited computational resources, such a public remote server needs to queue incoming requests when handling multiple concurrent users, resulting in unstable processing speeds. To address this issue, we choose to build our own local server that exclusively processes our requests. Consistent with Boltz-1, we adopt these two databases for MSA search: `uniref30_2302`, `colabfold_envdb_202108`. After the setup, we can simply replace the default `msa_server_url` parameter in Boltz-1 with our local server URL to obtain paired and unpaired MSA results from the local MSA server. The server is built following the local setup guide provided in the ColabFold repository, available at: https://github.com/sokrypton/ColabFold/tree/main/MsaServer.

Filtering. We have already discussed in the main text the need to filter MSAs to prevent data leakage during both the training and inference stages. The filtering is done as follows: The MSA results provide an amino acid-by-amino acid alignment of the retrieved sequences with the query sequence. For each VH or VL query sequence, we examine each sequence retrieved one by one. We extract the segments that correspond to the three CDRs of the query sequence. If any of these segments have a sequence identity greater than or equal to a specified threshold with the corresponding CDR sequence, we remove the entire sequence from the final MSA results. In both the training and inference stages discussed in the main text, this threshold is set to 0.2.

E Details about the Adopted Metrics

In this section, we provide more details about the metrics we used in this paper for evaluation.

E.1 Metrics Adopted in Sec. 3

AAR. It measures the accuracy of the generated CDRs sequences by comparing the amino acid identity with native sequences. It calculates the percentage of amino acids in the generated sequence that match the reference sequence.

RMSD. RMSD measures the deviation of the C_α atom coordinates between generated and native CDRs. Following DiffAb (Luo et al., 2022), the antibody frameworks are aligned prior to RMSD calculation. In this work, this alignment is specifically based on the C_α atoms.

Loop-AAR. This metric evaluates the accuracy of amino acid recovery specifically for the central loop residues of CDR-H3, the region most critical for antigen binding. Consistent with (Zhu et al., 2024), we exclude specific residues from our analysis. For the Chothia numbering scheme (Chothia & Lesk, 1987), we exclude the initial and terminal two residues of CDR-H3. In contrast, for the IMGT scheme (Lefranc et al., 2003), we exclude the first four and the last two residues.

Loop-RMSD. This metric assesses the structural deviation of the central loop residues within CDR-H3 between generated and native structures, focusing on the region critical for antigen binding. Consistent with Loop-AAR, we follow the loop region definitions specific to each numbering scheme. Similar to the RMSD calculation, the antibody frameworks are aligned before computing Loop-RMSD.

IMP. The Improvement Percentage (IMP) is determined by evaluating the binding energies of antibody-antigen complexes, calculated using the *InterfaceAnalyzer* with the *ref2015* score function in PyRosetta toolkit (Chaudhury et al., 2010). IMP represents the fraction of designed complexes that achieve lower binding energies compared to their natural counterparts.

Metrics calculated from structural data are computed after we perform side-chain packing and relaxation.

E.2 Metrics Adopted in Appendix G.1

RMSD. Here, RMSD measures the root-mean-square deviation of all C_α atom coordinates between generated and native protein structures. For this calculation, the entire protein structure is aligned, focusing on C_α atoms to provide an overall measure of structural accuracy.

Loop-RMSD. It evaluates the structural deviation of C_α atoms specifically within the central loop of the CDR-H3 region. This metric is calculated after aligning the entire protein structure and focuses on prediction accuracy in this highly variable and critical region for antigen binding.

TM-score. TM-score assesses the similarity between predicted and native protein structures, with scores ranging from 0 to 1. A higher TM-score indicates better global similarity and is particularly sensitive to the overall fold of the protein, rather than local variations.

Given that these metrics only require backbone predictions, and as noted by AlphaFold 3 that that the relax postprocessing step is rarely needed when using AF3, we do not perform side-chain packing or relaxation here. The models' prediction results are used directly for evaluation.

F More Details about the Implementations

Due to our use of different data splits from those in the original papers of the baselines, it is necessary to retrain these baseline models to ensure a fair comparison. In this section, we provide detailed information on the implementation of both the baseline models and MFDesign. All experiments run on a single node consisting of 8 $\times$ H100 GPUs with 80 GB HBM3 each (aggregated GPU memory of 640 GB), 2 $\times$ Intel Xeon Platinum 8468 processors comprised of 48 CPUs each (total 96 cores, 192 threads).

F.1 Details about the Implementations of Baselines

DiffAb[4] (Luo et al., 2022) We implement the baseline using the code provided by the authors, and we adhered to the hyper-parameter settings detailed in the *codesign_multicdrs.yml* file from the repository for our training configuration. DiffAb employs the Chothia numbering scheme, which is consistent with our approach.

dyMEAN[5] (Kong et al., 2023a) We utilize the code provided by the authors and set our training parameters according to the hyper-parameters specified in the *multi_cdr_design.json* file in the repository. dyMEAN employs the IMGT scheme (Lefranc et al., 2003) for numbering antibodies. We choose to adhere to its original selection regarding the scheme as Zhu et al. do.

Remark. The official dyMEAN code does not support nanobody processing, which results in a reduced number of samples for evaluation in dyMEAN as compared to the full test set. In contrast, both DiffAb and our method are capable of effectively handling nanobodies, allowing for evaluation on the complete test set.

F.2 Details about the Implementations of Our Method

Since AlphaFold3 has not open-sourced its training scripts, in this work, we base our approach on Boltz-1, a reproduced version of AF3 by Wohlwend et al., which provides these scripts. Additionally, Boltz-1 maintains consistency with AlphaFold3 in terms of variable and function definitions, facilitating a clearer understanding of the implementation alongside the original AlphaFold3 paper.

[4] https://github.com/luost26/diffab
[5] https://github.com/THUNLP-MT/dyMEAN

Table S1: Comparison of antigen structure prediction performance, with and without the fourth training stage, evaluated using RMSD and TM-score, with **best results** highlighted.

w/ 4-th training stage?	RMSD ↓	TM-score ↑
✗	9.87Å	74.91%
✓	**9.59Å**	**76.72%**

Our primary modification compared to AF3 is the addition of a denoising network for sequences, which we implement using a simple MLP composed of several LINEAR layers connected by GELU activation functions. We observe that in the implementation of DiffAb (Luo et al., 2022), an additional feature is encoded for each amino acid to indicate its region. We adopt this approach in our implementation and find it beneficial for accelerating early convergence of the model. Specifically, we include a feature indicating which chain a residue belongs to, specifying whether it is the heavy chain, light chain, or antigen. For antibody sequences (VH and VL), another feature specifies the region within the chain, such as FR1, CDR1, FR2, CDR2, FR3, CDR3, or FR4. For antigen residues in the antibody co-design task, where the baseline provides epitope information, we include a feature to identify if an antigen residue is part of the epitope region. However, in complex structure prediction, we do not differentiate based on epitopes, as incorporating such information does not align with the bind-docking setting. These additional features, along with the previously obtained token-level representations, are fed into the sequence denoising network.

For our models used in the co-design of antibody sequences and structures, the implementation of both discrete and continuous diffusion for sequence follows the same training approach:

- **1st-stage:** A warm-up strategy is employed, where the learning rate increases linearly from 0 to a predefined value 5×10^{-4} over 10 steps. Subsequent to this warm-up phase, the learning rate decays by a factor of 0.999 every 100 steps. The max token number is set to 256. The batch size for a single GPU is 6, thus amounting to 48 samples per training step. This stage consists of a total of $5,000$ training steps.
- **2nd-stage:** Training continues for a total of $2,000$ steps, starting with an initial learning rate that is set directly. The learning rate then decays by a factor of 0.999 every 100 steps. The max token number is set to 384, and the batch size for a single GPU is set to 3, resulting in 24 samples per training step.
- **3rd-stage:** Training continues for a total of $5,000$ steps, starting with an initial learning rate that is set directly. The learning rate then decays by a factor of 0.999 every 100 steps. The max token number is set to 512, and the batch size for a single GPU is set to 1, resulting in 8 samples per training step.
- **4th-stage:** Training continues for a total of $9,000$ steps, starting with an initial learning rate that is set directly. The learning rate then decays by a factor of 0.999 every 100 steps. The max token number is set to 512, and the batch size for a single GPU is set to 1, resulting in 8 samples per training step.

G MORE EXPERIMENTAL RESULTS

G.1 STRUCTURE PREDICTION

In this section, we evaluate our method in the structure prediction task, *a.k.a.* protein folding problem. Unlike in Sec. 3, we do not use impainting for sampling for this task because revealing any ground-truth structure information is not allowed here. We observe that the origin Boltz-1 struggles with accurately predicting antibody-antigen binding sites, a common issue among existing structure prediction models. This limitation becomes evident when Boltz-1 is tasked with predicting complex structures for antibody-antigen pairs from the test set in Sec. 3. For each sample, the model generates five predicted structures, and the average TM-score across these predictions is only 0.62. As our model is fine-tuned from Boltz-1, it inherits this limitation.

Table S2: Performance comparison on the structure prediction task, evaluated by RMSD, Loop-RMSD, and TM-score.

Given CDR?	Fine-tuned?	RMSD ↓	Loop-RMSD ↓	TM-score ↑
✗	✗	2.21Å	5.57Å	91.84%
✗	✓	1.73Å	4.90Å	93.88%
✓	✗	1.39Å	3.53Å	95.45%
✓	✓	1.65Å	4.55Å	94.25%

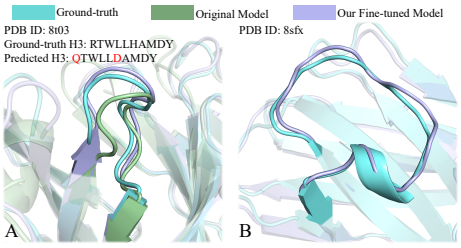

Figure S2: **Examples of predicted structures.**

Thus, we choose to predict antibody structures alone, a relatively simpler task, yet still challenging, particularly in accurately predicting the CDR-H3 region. We experiment with two input types: one where the CDRs are represented as **<X>**, and another where the CDRs are provided. We compare our fine-tuned Boltz-1 model with the original, non-fine-tuned model. For the first input type, we apply filtering to the MSA results. For the second, we do not filter the MSA results. We adopt the test set used in Sec. 3. We use the model implemented in discrete sequence diffusion here for evaluation.

Metrics. We utilize the following set of metrics for detailed evaluation: (1) **RMSD** (Å): measures the root-mean-square deviation of all C_α atom coordinates after aligning the entire protein structure, providing an overall measure of structural accuracy; (2) **Loop-RMSD** (Å): calculates the RMSD for the C_α atoms specifically in the middle loop of the CDR-H3 region after whole-structure alignment, which is particularly challenging to predict; (3) **TM-score**: ranges from 0 to 1, with higher scores indicating better similarity between the predicted and reference structures. We generate 20 predictions for each sample and report the averaged results.

Table S2 summarizes the results on antibody structure prediction task. In comparisons where the input antibody's CDRs are unknown, our fine-tuned model consistently surpasses the original across all three metrics. In Fig. S2A, we demonstrate a case where CDRs are not present in the input. We see that the original Boltz-1 and our fine-tuned model can both accurately predict the framework regions of this sample. However, for the CDRs, the original model's prediction is less precise. Our model, despite starting with unknown residues, can predict CDRs sequences during the sampling process. As the sequence becomes clearer, this allows for more accurate CDRs structure predictions. This suggests that our method could provide valuable insights into what the functional structure of an antibody might look like, when only the framework regions sequences are given.

With the CDRs provided, we observe that the results are improved for both compared to when CDRs are absent. Our fine-tuned model performs worse than the original model on three metrics. However, the gap is slight, with the TM-score decreasing by only 1.26%. This indicates that our fine-tuning retains a structure prediction capability comparable to that of the original model. This slight decline in performance might be due to changes in the training method for structural diffusion. Additionally, because we apply full parameter fine-tuning, it might change the feature space learned during pre-training that was more beneficial for structure prediction. The added sequence prediction task may introduce gradients that conflict with structure prediction, leading to decreased performance in the latter. In Fig. S2B, we present the prediction of the fine-tuned model for a nanobody, whose CDR-H3 is longer and thus more challenging to predict accurately than that of a typical antibody. However, our fine-tuned model successfully predicts the structure of the CDR-H3 for this sample, further exemplifying our method's good structure prediction ability.

G.2 Ablation of the 4th Training Stage

Recall that we adopt a four-stage training approach. The first three stages primarily focus on selecting tokens from regions around the antibody and its nearby areas, which could result in inadequate training for antigen regions far from the antibody. To address this limitation, we introduce a fourth training stage specifically designed to ensure adequate learning of these distant antigen regions.

Here, we use our model to predict the structures of antigens, comparing the performance with and without the fourth stage of training. For each antigen in the test set from Sec. 3, we generate 5 predictions. We employ RMSD and TM-score as evaluation metrics and summarize the results in Table S1. We use the MF-Design with discrete sequence diffusion for this study. We find that, after

undergoing the fourth training stage, the model shows improvements in both metrics compared to the model trained with only the first three stages.

G.3 Visualization of More Examples

In this section, we provide more visualizations of our results. In Fig. S3, we show the outcomes from simultaneously designing all CDRs of antibodies conditioned on co-crystal structures, as discussed in Sec. 3, with a focus on our designed CDR-H3. In Fig. S4, we present additional cases from Appendix G.1, where sequences with unknown amino acid types in the CDRs of nanobodies are used as inputs, and our model is able to predict the structure while also predicting the sequences of the nanobody CDRs. Additionally, in Fig. S5, we illustrate some predicted complex structures by our model when given the complete antibody-antigen complex sequence as input. These structures here are all predicted using the model with a discrete implementation of sequence diffusion.

H Correspondence of Modules Referenced in This Paper to AlphaFold 3 Algorithms

For direct technical cross-referencing, This section explicitly maps the AlphaFold3's modules referenced in this paper to their original algorithm numbers in the AlphaFold3's supplementary materials in Table S4.

I Notations

We summarize the notations used in this paper in Table S3 to facilitate reading.

Table S3: Notations.

Notation	Description
<X>	a token which means the corresponding residue's type is unknown and requires to be speficied
$\mathbf{k}_i$	a token, which is an one-hot encoding of its type
$\{\mathbf{k}_i\}$	a token set
$\mathbf{x}_l$	three-dimensional coordinates of a atom
$\{\mathbf{x}_l\}$	a structure characterized by a set of atoms' coordinates
$\{\mathbf{f}^*\}$	raw inputs
$\{\mathbf{s}_i^{\text{inputs}}\}$	basic encodings for the raw inputs, generated by **InputEmbedder** in the conditioning trunk
$\{\mathbf{s}_i^{\text{trunk}}\}$	token-level single representations, the output of the conditioning trunk
$\{\mathbf{z}_{ij}^{\text{trunk}}\}$	token-level pair representations, the output of the conditioning trunk
$\{\mathbf{s}_i\}$	token-level single representations generated by **DiffusionConditioning**, which embed information from the noise level, $\{\mathbf{s}_i^{\text{inputs}}\}$ and $\{\mathbf{s}_i^{\text{trunk}}\}$
$\{\mathbf{z}_{ij}\}$	token-level pair representations generated by **DiffusionConditioning**, which embed information from relative position encoding and $\{\mathbf{z}_{ij}^{\text{trunk}}\}$
$\{\mathbf{x}_l^{\text{noisy}}\}$	noisy structure
$\{\mathbf{r}_l^{\text{noisy}}\}$	scaled $\{\mathbf{x}_l^{\text{noisy}}\}$ with approximately unit variance
$\{\mathbf{q}_l\}$	atom-level single representations generated by **AtomAttentionEncoder**, which encode basic atomic features and incorporate information from $\{\mathbf{r}_l^{\text{noisy}}\}$
$\{\mathbf{c}_l\}$	atom-level single representations generated by **AtomAttentionEncoder**, which encode basic atomic features and incorporate information from $\{\mathbf{s}_i^{\text{trunk}}\}$
$\{\mathbf{p}_{lm}\}$	atom-level pair representations generated by **AtomAttentionEncoder**, which encode basic atomic features and incorporate information from $\{\mathbf{z}_{ij}\}$
$\{\mathbf{a}_i\}$	token-level single representations generated by **AtomAttentionEncoder** and updated by **DiffusionTransformer**
$\{\mathbf{x}_l^{\text{denoised}}\}$	denoised structure
σ_{data}	hyper-parameters for structure sampling
σ_{min}	hyper-parameters for structure sampling
σ_{max}	hyper-parameters for structure sampling
ρ	hyper-parameters for structure sampling
γ_0	hyper-parameters for structure sampling
γ_{min}	hyper-parameters for structure sampling
λ	hyper-parameters for structure sampling
η	hyper-parameters for structure sampling
t	a sampled timestep
T	the total number of timesteps for our sequence and structure diffusion training
m_t	the mask rate for sequence diffusion at timstep t
$\{\mathbf{k}_i^{\text{noisy}}\}$	noisy sequence
$\{\mathbf{k}_i^{\text{denoised}}\}$	denoised sequence
$\{\mathbf{x}_l^{\text{GT}}\}$	given ground-truth co-crystal structure

Table S4: Mapping of referenced modules in this paper to AlphaFold3's algorithms.

Module	AF3 Algorithm
InputEmbedder	Algorithm 2
TemplateModule	Algorithm 16
MSAModule	Algorithm 8
PairFormerStack	Algorithm 17
DiffusionConditioning	Algorithm 21
AtomAttentionEncoder	Algorithm 5
DiffusionTransformer	Algorithm 23
AtomAttentionDecoder	Algorithm 6
DiffusionModule	Algorithm 20
CentreRandomAugmentation	Algorithm 19
RigidAlign	Algorithm 28

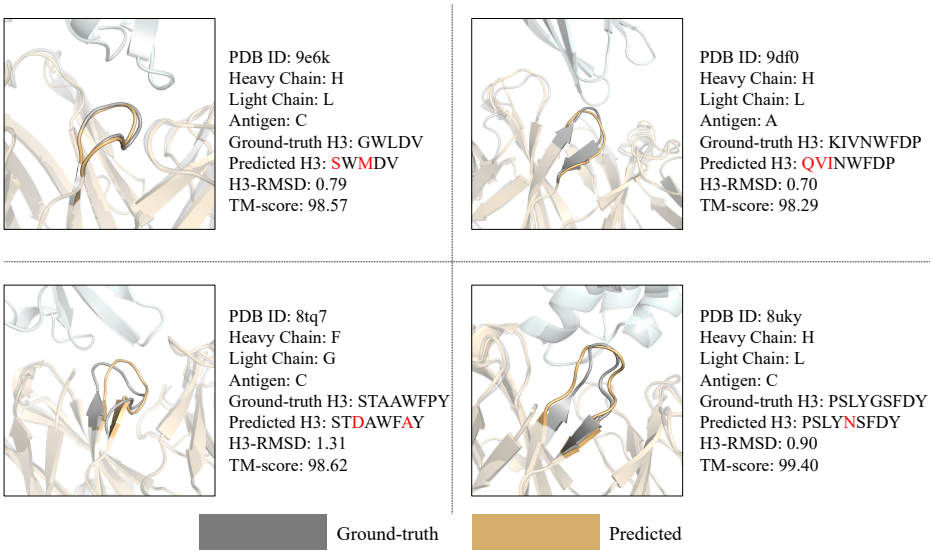

Figure S3: Examples of generated typical antibodies with simultaneously designed CDRs.

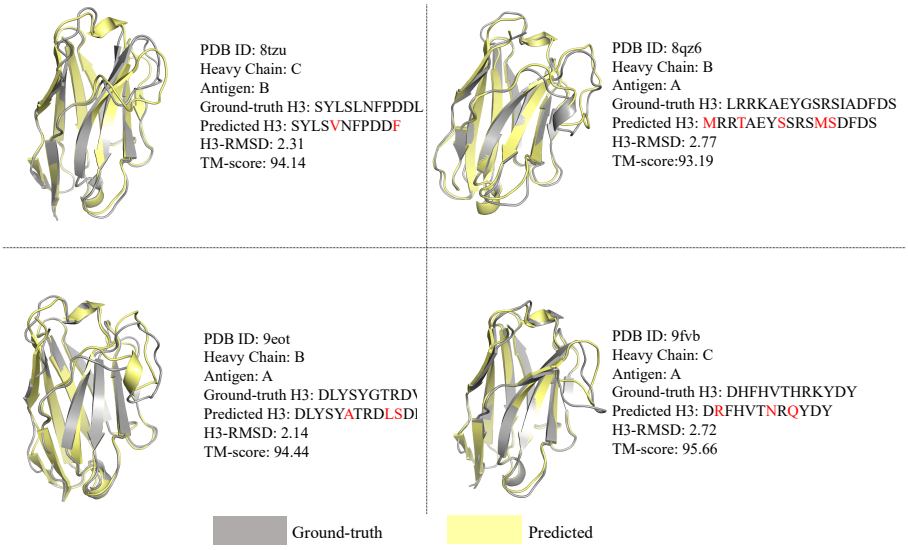

Figure S4: Examples of predicted nanobody structures using sequences with unknown CDRs as input.

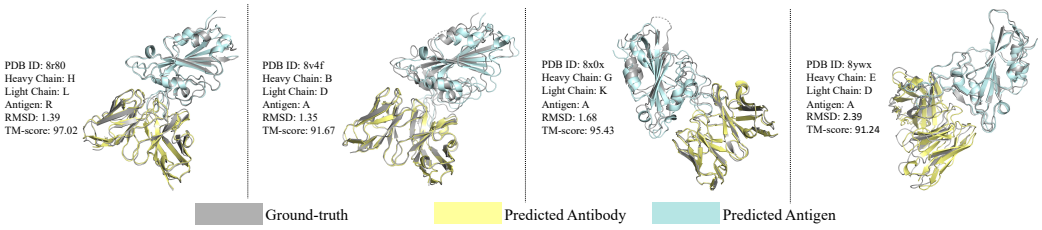

Figure S5: Examples of predicted antibody-antigen complex structure using complete sequences.