The Discovery of Binding Modes Requires Rethinking Docking Generalization

Gabriele Corso[*1], **Arthur Deng**[*2], **Nicholas Polizzi**[3], **Regina Barzilay**[1], **Tommi S. Jaakkola**[1]
[1] MIT, [2] University of California, Berkeley, [3] Dana-Farber Cancer Institute, HMS
Correspondence to gcorso@mit.edu

Abstract

Accurate blind docking has the potential to lead to new biological breakthroughs, but for this promise to be realized, it is critical that docking methods generalize well across the proteome. However, existing benchmarks fail to rigorously assess generalizability. Therefore, we develop DOCKGEN, a new benchmark based on the ligand-binding domains of proteins, and we show that machine learning-based docking models have very weak generalization abilities even when combined with various data augmentation strategies. Instead, we propose CONFIDENCE BOOTSTRAPPING, a new training paradigm that solely relies on the interaction between a diffusion and a confidence model. Unlike previous self-training methods from other domains, we directly exploit the multi-resolution generation process of diffusion models using rollouts and confidence scores to reduce the generalization gap. We demonstrate that CONFIDENCE BOOTSTRAPPING significantly improves the ability of ML-based docking methods to dock to unseen protein classes.

1 Introduction

Understanding how small molecules and proteins interact, a task known as molecular docking, is at the heart of drug discovery. The conventional use of docking in the industry has led the field to focus on finding binding conformations when restricting the search to predefined pockets and evaluating these on a relatively limited set of protein families of commercial interest. However, solving the general blind docking task (i.e. without pocket knowledge) would have profound biological implications. For example, it would help us understand the mechanism of action of new drugs to accelerate their development [31], predict adverse side-effects of drugs before clinical trials [24], and discover the function of the vast number of enzymes and membrane proteins whose biology we do not yet know [38]. All these tasks critically require the docking methods to generalize beyond the relatively small class of well-studied proteins for which we have many available structures.

Existing docking benchmarks are largely built on collections of similar binding modes and fail to rigorously assess the ability of docking methods to generalize across the proteome. To address this need, we propose DOCKGEN, a new benchmark that aims to test a method's ability to generalize across protein domains. With DOCKGEN, we show that existing machine learning-based docking methods poorly predict binding poses on unseen binding pockets, and that standard data augmentation techniques do not help bridge this generalization gap.

To move beyond this challenge, we propose CONFIDENCE BOOTSTRAPPING, a novel self-training scheme inspired by Monte Carlo tree-search methods, where we train directly on protein-ligand complexes from unseen domains without access to their structural data. The training is enabled by the interaction between a diffusion model rolling out the sampling process and a confidence model assigning confidence scores to the final sampled poses. These confidence scores are then fed back into the early steps of the generation process (see Figure 1 for a visual representation). This process is iterated to improve the diffusion model's performance on unseen targets.

[*]Equal contribution.

NeurIPS 2023 Generative AI and Biology Workshop.

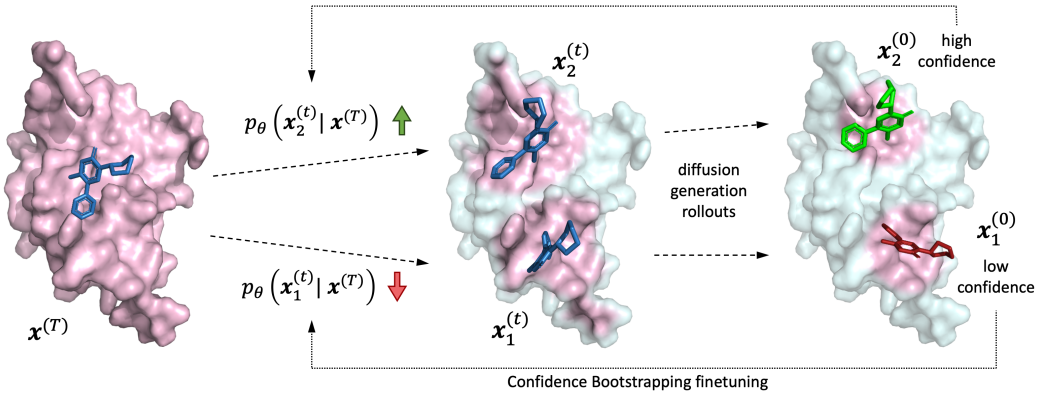

Figure 1: Visual representation of the CONFIDENCE BOOTSTRAPPING training scheme. The dashed lines represent the reverse diffusion generation rollouts that the model executes. The dotted lines illustrate the bootstrapping feedback from the confidence model that is used to update the likelihood of the early diffusion steps by changing the weights of the score model.

Unlike previous self-training methods that were applied to other fields, CONFIDENCE BOOTSTRAPPING directly takes advantage of the structure of the sampling process. In particular, we exploit the multi-resolution structure of diffusion models by identifying that the final confidence model generalizes significantly better than the diffusion model and using information from the confidence model to guide early stages of the reverse diffusion process. CONFIDENCE BOOTSTRAPPING, via its iterative feedback, is able to close the gap between the diffusion model and the confidence model.

We test CONFIDENCE BOOTSTRAPPING on the new DOCKGEN benchmark by fine-tuning the best version of DIFFDOCK on individual clusters of protein domains. In each of these clusters, within the first few iterations of bootstrapping, the diffusion model is pushed to generate docked poses with increasingly high confidence. This increased confidence also translates into significantly higher accuracy with the fine-tuned models improving from 9.8% to 25.6% success rate overall, and above 40% in half of the protein domains. Remarkably, it doubles the performance of search-based docking methods, which have been shown to generalize significantly better than ML methods.

2 The DockGen Benchmark

We argue that the existing approaches used to evaluate the ML docking methods fall short in analyzing their generalization capacity to different parts of the proteome. Binding pockets, due to their importance to many biological mechanisms, are often among the most well-conserved regions in proteins. Therefore, just looking at the UniProt-ID of a protein or its global sequence similarity often leads to proteins in the training and testing sets that have the same underlying pocket. Figure 3-A shows an example of such failures, where two proteins even with only 16% sequence similarity (30% is often used as cutoff) share very similar binding pockets.

To better detect these cases we delve one level deeper into the organization of proteins and look at protein domains. Protein domains are the structural and functional units that compose proteins. Very similar domains can appear in different sequences but have similar local structural characteristics. Therefore, by looking at the protein domains where ligands bind, we can form a more granular classification of the protein-ligand interactions. In practice, we achieve this by using the ECOD [4] domain classification and select complexes from the Binding MOAD dataset [14]. More details on the dataset generation strategy and its advantages can be found in Appendix B.

We then run a number of baselines considered to be the state-of-the-art open-source or open-access models: for search-based methods SMINA [19] and GNINA [27], while for ML methods EQUIBIND [33], TANKBIND [23] and DIFFDOCK [7]. As reported in Table 1 in the appendix, ML methods significantly underperform in this new benchmark and their performances are only a fraction of those that they have in the time-split complexes from the PDBBind test set, with regression methods having nearly no success. In Appendix D.1, we will further show that even increasing the training data or performing a range of data augmentation strategies is not able to considerably close this generalization gap, motivating the exploration of different training strategies.

3 Confidence Bootstrapping

Motivation For many decades, docking, along with other structural biology problems like protein folding, was studied and treated as an NP-hard combinatorial optimization problem [34, 29]. This NP perspective suggests a useful insight to the problem: it is easier to check that a pose is good than to generate a good pose. Combined with the necessity of alternative training methods to bridge the generalization gap, this insight points towards the exploration of new self-training-based strategies.

Existing self-training methods do not, however, directly exploit the structure of the generative process they optimize. Moreover, they often fail if the initial generator is not good enough because of the low signal-to-noise ratio causing an amplification of the errors of the original model [26]. We argue that diffusion models, because of their multi-resolution structure, offer a unique opportunity to more precisely target the effect of self-training and avoid error amplification.

Overview We introduce CONFIDENCE BOOTSTRAPPING, a training mechanism that refines a diffusion generator based on feedback from a confidence model. The diffusion model is used to "roll out" the reverse diffusion process, generating poses that are then scored with the confidence model. Taking advantage of the multi-resolution nature of diffusion models, these scores are used to inform how to update the parameters of the early steps of the diffusion process so that the model will generate more poses close to those with high confidence (see a graphical representation in Figure 1). This process is then repeated for several steps.

CONFIDENCE BOOTSTRAPPING is particularly well suited for the molecular docking task. First, the limited amount of training data and its lack of diversity make alternative training methods critical. Furthermore, CONFIDENCE BOOTSTRAPPING can leverage information from affinity datasets such as BindingDB [21], which are orders of magnitude larger than the number of known structures. Finally, docking screens are usually run on a very large number of complexes all on a restricted set of proteins. Therefore, any time spent fine-tuning the docking algorithm for the restricted set is largely amortized.

Formalization We now formalize the CONFIDENCE BOOTSTRAPPING training routine. Let $p_\theta(\mathbf{x}; d)$ be the probability distribution of poses learned by the diffusion model with score $s_\theta(\mathbf{x}^{(t)}, t; d)$ where d is the known information about the complex (e.g protein and molecule identity). Let $c_\phi(\mathbf{x}, d)$ be the output of the confidence model, and let $D = \{d_1, d_2, ...\}$ be a set of known binders (e.g. from BindingDB) for the target distribution of interest.

CONFIDENCE BOOTSTRAPPING consists of K iterations where at each iteration i the score model weights θ are updated based as following optimization:

$$\theta^{i+1} =_\theta \Bigg[\mathbb{E}_{t\sim U[0,T]} \{\lambda(t) \mathbb{E}_{\mathbf{x}^{(0)}, d\sim p_{\text{train}}} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} [\| s_\theta(\mathbf{x}(t), t; d) - \nabla_{\mathbf{x}^{(t)}} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)\|_2^2]\}$$

$$+ \mathbb{E}_{t\sim U[t_{\min},T]} \{\lambda(t) \mathbb{E}_{\mathbf{x}^{(0)}, d\sim p_{\theta^i,\phi}} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} [\| s_\theta(\mathbf{x}(t), t; d) - \nabla_{\mathbf{x}^{(t)}} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)\|_2^2]\} \Bigg]$$

where θ^0 are the weights of the pretrained diffusion model (if not trained from scratch), $t_{\min} \in [0, T)$ and $p_{\theta,\phi}(\mathbf{x}, d) \propto p_\theta(\mathbf{x}; d)\ \exp[c_\phi(\mathbf{x}, d)]$. $t_{\min} > 0$ ensures that the bootstrapping feedback is only used to update the initial steps of the reverse diffusion. This is used because the samples taken from the combination of diffusion and confidence models are likely to be too noisy to provide fine-grained guidance for small t.

Each of these iterations $i \in [0, K)$ is achieved by performing a rollout stage, followed by an update stage. During the rollout stage, we first sample d from D, then sample points from $p_{\theta^i}(\cdot, d)$, forming a buffer $B = [(x_1, d_1), ...]$. Then during the update stage, a fixed number of stochastic gradient descent steps are performed where half of the batch elements are taken from the training dataset (first half of optimization objective) and half are taken from B (second half). In particular to approximate samples from $p_{\theta^i,\phi}(\mathbf{x}, d)$, the elements $(\mathbf{x}, d)$ of B are sampled with probabilities proportional to $\exp[c_\phi(\mathbf{x}, d)]$. Further details on the implementation and optimization of this routine can be found in Appendix F.

Model Architecture We test the effectiveness of CONFIDENCE BOOTSTRAPPING in improving molecular docking generalization by fine-tuning DIFFDOCK on individual classes of protein domains. However, to enable an efficient execution of this iterative training routine, we make a number of changes to the architecture of DIFFDOCK's score and confidence models that make them significantly faster and better suited for bootstrapping. We details these architecture improvements in Appendix E.

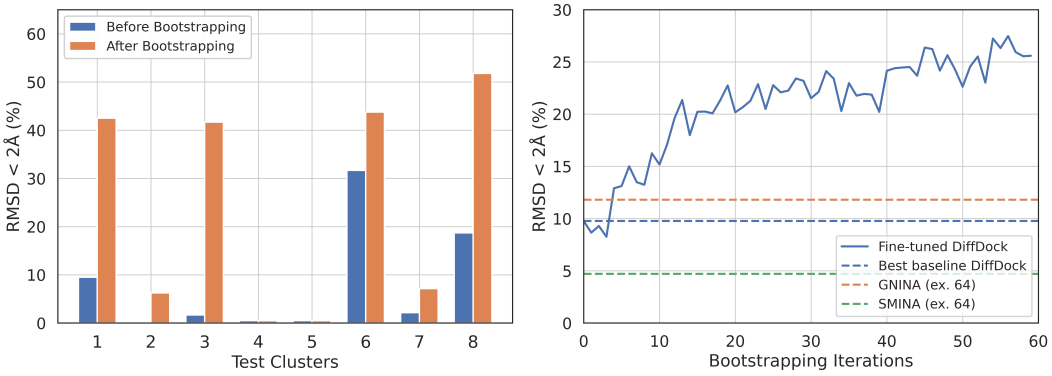

Figure 2: Empirical performance of CONFIDENCE BOOTSTRAPPING across the 8 protein domain clusters within DOCKGEN-cluster. *Left:* performance for each cluster before the fine-tuning and after the K=60 steps of CONFIDENCE BOOTSTRAPPING. *Right:* aggregated performance along the fine-tuning for all the clusters weighted by their count with, as references, the performance of some of the baselines on the same set.

4 Experiments

We test CONFIDENCE BOOTSTRAPPING on the new DOCKGEN benchmark, where we fine-tune a model on each protein domain cluster[2]. We use clusters with at least 6 complexes and restrict the test set to 8 separate clusters (5 for validation) for a total of 85 complexes, which compose the DOCKGEN-clusters subset (see Table 1 for the performance of baselines in this subset).

In Figure 2-right, we plot the average performance across all clusters in comparison to that of the baselines. From this, we see that, in DOCKGEN-clusters, CONFIDENCE BOOTSTRAPPING considerably raises the baseline DIFFDOCK's performance going from 9.8% to 25.6% and doubles that of the traditional search-based methods even when run with high exhaustiveness.

The analysis becomes even more interesting when looking into the evolution of the performance in the individual clusters (Figure 2-left). In half of the clusters, the model is able to reach top-1 RMSD < 2Å performance above 40%. These clusters mostly constitute those in which the original model has non-zero accuracy with an initial performance varying from around 5% to 25%. Then we have two clusters where the accuracy is improved to only 5-10% and two clusters where the model never selects good poses neither before nor after the bootstrapping.

5 Conclusion

Given the potential of blind docking in biology and drug discovery, it is important to track the progress of ML-based methods to generalize to to unseen pockets. To this end, we have proposed DOCKGEN, a new benchmark for blind docking generalization based on the classification of binding protein domains. The performance analysis on DOCKGEN highlights that none of the existing ML methods approach a satisfactory generalization performance, and data augmentation techniques cannot meaningfully reduce the generalization gap.

To tackle this, we proposed CONFIDENCE BOOTSTRAPPING, a self-training method that only relies on the interaction between a diffusion and a confidence model and exploits the multi-resolution structure of the sampling process. This allows the direct training of the docking model on classes of proteins where binding structural data is not available. Empirically, the method shows significant improvements on the DOCKGEN benchmark, going from 10% to 25% success rate and surpassing that of search-based methods. Finally, we believe this opens up the possibility of training large-scale docking models that have so far been obstructed by the size and diversity of the available data, bringing us one step closer to a generalizable solution to the docking challenge.

[2]Code and data for the experiments can be found at this URL: https://anonymous.4open.science/r/confiboot-3118

References

[1] A. Alhossary, S. D. Handoko, Y. Mu, and C.-K. Kwoh. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*, 31(13):2214–2216, 02 2015.

[2] J. Bazan, L. Weaver, S. Roderick, R. Huber, and B. Matthews. Sequence and structure comparison suggest that methionine aminopeptidase, prolidase, aminopeptidase p, and creatinase share a common fold. *Proceedings of the National Academy of Sciences*, 91(7):2473–2477, 1994.

[3] H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide Protein Data Bank. *Nat Struct Biol*, 10(12):980, Dec 2003.

[4] H. Cheng, R. D. Schaeffer, Y. Liao, L. N. Kinch, J. Pei, S. Shi, B.-H. Kim, and N. V. Grishin. Ecod: an evolutionary classification of protein domains. *PLoS computational biology*, 10(12):e1003926, 2014.

[5] U. Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2015.

[6] G. Corso. Modeling molecular structures with intrinsic diffusion models. *arXiv preprint arXiv:2302.12255*, 2023.

[7] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*, 2022.

[8] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. Wicky, A. Courbet, R. J. de Haas, N. Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[9] V. De Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh, and A. Doucet. Riemannian score-based generative modeling. *arXiv preprint arXiv:2202.02763*, 2022.

[10] M. Geiger and T. Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[12] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks. Glide: a new approach for rapid, accurate docking and scoring. 2. enrichment factors in database screening. *Journal of medicinal chemistry*, 2004.

[13] N. M. Hassan, A. A. Alhossary, Y. Mu, and C.-K. Kwoh. Protein-ligand blind docking using quickvina-w with inter-process spatio-temporal integration. *Scientific Reports*, 7(1):15451, Nov 2017.

[14] L. Hu, M. L. Benson, R. D. Smith, M. G. Lerner, and H. A. Carlson. Binding moad (mother of all databases). *Proteins: Structure, Function, and Bioinformatics*, 60(3):333–340, 2005.

[15] A. N. Jain. Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *Journal of medicinal chemistry*, 46(4):499–511, 2003.

[16] W. Jin, J. Wohlwend, R. Barzilay, and T. Jaakkola. Iterative refinement graph neural network for antibody sequence-structure co-design. *arXiv preprint arXiv:2110.04624*, 2021.

[17] B. Jing, G. Corso, J. Chang, R. Barzilay, and T. Jaakkola. Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*, 2022.

[18] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[19] D. R. Koes, M. P. Baumgartner, and C. J. Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53(8):1893–1904, 2013.

[20] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, A. d. Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *arXiv*, 2022.

[21] T. Liu, Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2007.

[22] Z. Liu, M. Su, L. Han, J. Liu, Q. Yang, Y. Li, and R. Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of Chemical Research*, 50(2):302–309, 2017.

[23] W. Lu, Q. Wu, J. Zhang, J. Rao, C. Li, and S. Zheng. Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction. *Advances in neural information processing systems*, 2022.

[24] H. Luo, A. Fokoue-Nkoutche, N. Singh, L. Yang, J. Hu, and P. Zhang. Molecular docking for prediction and interpretation of adverse drug reactions. *Combinatorial Chemistry & High Throughput Screening*, 21(5):314–322, 2018.

[25] M. R. Masters, A. H. Mahmoud, and M. A. Lill. Fusiondock: Physics-informed diffusion model for molecular docking.

[26] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

[27] A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, R. Meli, M. Ragoza, J. Sunseri, and D. R. Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):1–20, 2021.

[28] R. Meli and P. C. Biggin. spyrmsd: symmetry-corrected rmsd calculations in python. *Journal of Cheminformatics*, 12(1):49, 2020.

[29] M. Mukhopadhyay. A brief survey on bio inspired optimization algorithms for molecular docking. *International Journal of Advances in Engineering & Technology*, 7(3):868, 2014.

[30] N. F. Polizzi and W. F. DeGrado. A defined structural unit enables de novo design of small-molecule–binding proteins. *Science*, 369(6508):1227–1233, 2020.

[31] G. Schottlender, J. M. Prieto, M. C. Palumbo, F. A. Castello, F. Serral, E. J. Sosa, A. G. Turjanski, M. A. Martì, and D. Fernández Do Porto. From drugs to targets: Reverse engineering the virtual screening process on a proteomic scale. *Frontiers in Drug Discovery*, 2:969983, 2022.

[32] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[33] H. Stärk, O. Ganea, L. Pattanaik, R. Barzilay, and T. Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *International Conference on Machine Learning*, pages 20503–20521. PMLR, 2022.

[34] R. Thomsen. Protein–ligand docking with evolutionary algorithms. *Computational Intelligence in Bioinformatics*, pages 167–195, 2007.

[35] R. Thomsen and M. H. Christensen. Moldock: a new technique for high-accuracy molecular docking. *Journal of medicinal chemistry*, 49(11):3315–3321, 2006.

[36] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, pages 1–3, 2023.

[37] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[38] X. Yi, Y. Zhang, P. Wang, J. Qi, M. Hu, and G. Zhong. Ligands binding and molecular simulation: the potential investigation of a biosensor based on an insect odorant binding protein. *International journal of biological sciences*, 11(1):75, 2015.

A Related work

Search-based docking Due to its importance in biological research and drug discovery, molecular docking has for decades been a central challenge for the computational science community [12, 15, 35]. Originally, most techniques followed the search-based paradigm, which is still prevalent today. These methods consist of a scoring function and an optimization algorithm. The latter searches over thousands or millions of different conformations, which are passed to the scoring function that determines their likelihood/goodness. While these methods tend to show relatively robust generalization across protein classes, they are significantly affected by the size of the search space, which grows exponentially as the ligand gets larger or as assumptions are removed (e.g. receptor rigidity).

ML-based docking Researchers have recently tried to move beyond the search-based paradigm and directly generate poses with deep learning models. The first attempts [33, 23] framed the docking task as a regression problem; this showed significant improvements in runtime but did not reach the accuracy of search-based methods. Corso et al. [7] proposed DIFFDOCK, a generative model based on the diffusion modeling framework that is trained to sample docked ligand poses. In particular, DIFFDOCK uses a diffusion model to sample a small number of possible poses which are then passed to a confidence model that ranks and assigns a confidence score to each.

Blind docking benchmarks The majority of previous ML-based methods used the PDBBind dataset [22], a curated set of protein-ligand crystallographic structures from PDB [3], to train and test models. In particular, they adopted a time-based split of the dataset where structures that were resolved before 2019 went into training and validation, and those from 2019 formed the test set. Stark et al. [33] and others also evaluate the performance on a reduced set of proteins with different UniProt-ID [5] compared to those in the training set. Here, while ML methods show a drop in performance, they remain in line with search-based techniques [7]. In terms of evaluation metric, most docking works measure the proportion of approximately correct poses, which are commonly considered to be those with a ligand RMSD under 2Å [1, 7].

Diffusion models Let $p(\mathbf{x})$ be some data distribution of interest. Score-based diffusion generative models [32] are trained to sample from p by defining a continuous diffusion process $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}$, where $\mathbf{w}$ represents the Wiener process, that transforms the data distribution in a simple prior and learning to reverse such process. This is enabled by the existence of a corresponding reverse diffusion process which can be expressed by the SDE $d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\mathbf{w}$ where $p_t(\mathbf{x})$ is the likelihood of the evolving data distribution. To run this reverse diffusion equation, we need to learn to approximate the score of the evolving data distribution $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, this is achieved by optimizing the parameters θ via the denoising score matching loss:

$$\theta^* =_\theta \left[\mathbb{E}_{t\sim U[0,T]}\{\lambda(t)\mathbb{E}_{\mathbf{x}^{(0)}\sim p_{\text{train}}}\mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)}[\|s_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}^{(t)}} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)\|_2^2]\}\right]$$

where $\lambda(t)$ is a positive weighting function. Diffusion models were then generalized to compact Riemannian manifolds [9], a formulation that is particularly well suited for scientific applications where the main degrees of freedom can be well represented by actions on a low-dimensional Riemannian manifold [6]. This idea underlies DIFFDOCK and other recent advances in computational sciences [17, 36].

Self-training methods Self-training refers to a range of techniques that have been employed in several different ML application domains where labels predicted by some model on unlabelled data are used for training. For example, in the setting of image classification, Xie et al. [37] used unlabelled images to improve a classifier by first generating labels from the clean image with the current classifier version, and then training the classifier to make the same prediction on noisy versions of the same image. This method was taken as inspiration for the self-distillation technique used by AlphaFold2 [18], where after a first training iteration, predicted structures satisfying a certain residue pairwise distance distribution were used for a second iteration of model training.

In the realm of generative models, McClosky et al. [26] used the labels predicted by a discriminative reranker to select the best parses generated by a generative parser and add them to the training set. Jin et al. [16] took a similar approach for antibody optimization via the feedback of a neutralization predictor. Finally, Generative Adversarial Networks (GANs) [11] also use the feedback from a discriminator to train a generative model. However, in GANs one relies on having in-distribution data to jointly train the discriminator.

B DockGen Benchmark Details

B.1 Overview

To classify each domain, we used the ECOD [4] classification. This clustering divides the 17k complexes from PDBBind before 2019, which have been used for training and validation of previous ML models, into 487 clusters. The remaining data from 2019, from which the test set was generated, presents only 8 additional clusters composed of a total of 15 complexes. This clustering approach is very different from that taken by other methods based on global sequence similarity. In fact, in Figure 3-B we present the histogram of the sequence similarity between proteins within ECOD clusters of PDBBind where the median similarity is only 23.5%.

To obtain a more sizable test set without retraining the models on a reduced set, we turn to the Binding MOAD dataset [14]. Similar to PDBBind, Binding MOAD is a curated collection of protein-ligand complexes from the PDB. However, due to its different filtering and requirements (e.g. no requirement for known affinity), it contains a set of 41k complexes partially disjoint from PDBBind. These come from 525 ECOD clusters, 346 of which are in common with PDBBind, and 179 of which are not present in PDBBind.

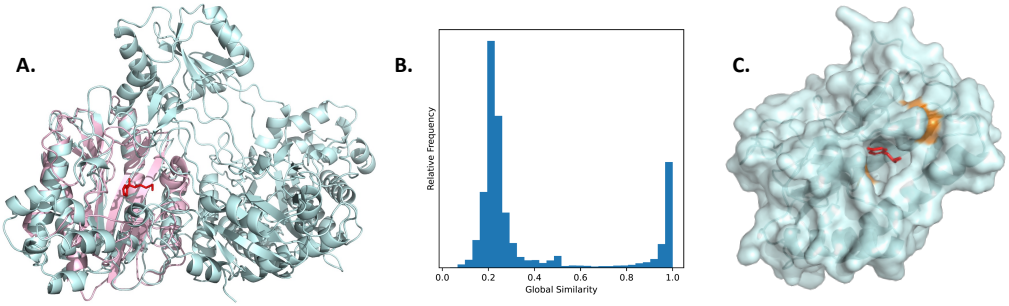

Figure 3: **A.** An example of two proteins in PDBBind, 1QXZ in pink and 5M4Q in cyan, that share a very similar binding pocket structure (the ligand is in red), but have only 16% sequence similarity (maximum of 20% on individual chains). While sequence similarity splits would classify them in separate clusters, our approach correctly identifies that the binding domain of these two proteins is the same ("Creatinase/aminopeptidase" sometimes referred to as "pita-bread" fold [2]). **B.** Histogram of the global pairwise sequence similarity between pairs of proteins within an ECOD cluster in PDBBind. This highlights how common global sequence similarity splits (that often use 30% similarity cutoff) are inadequate in this setting. **C.** Visualization of the van der Mer-inspired synthetically generated docked poses. In this case, a tyrosine (in red) is taken to be the ligand, and the amino acids that are nearby in the primary sequence are removed from the protein structure of 1QXZ (the created chain breaks are highlighted in orange).

B.2 Construction details

In this section, we specify the details of how the Binding MOAD dataset was parsed and filtered to obtain the DOCKGEN benchmark validation and test sets. The benchmark was created following these steps:

1. We perform the ECOD-based clustering. Each ligand in a protein is classified by the ECOD domain (t group) of the chain that is making the most contacts with the ligand. The t group of the ligand is assigned using the consensus of per-residue t-group assignments of each contacting amino acid (5Å heavy atom distance) in the dominant chain. We use this approach to classify both the complexes in PDBBind and MOAD. There are 179 clusters in MOAD that do not appear in PDBBind. These 179 are randomly and equally divided between validation and test datasets.
2. Within these clusters, since most docking tools do not support the simultaneous docking of multiple ligands, we discard ligands that are within 4.8Å from the heavy atoms of another ligand.
3. We discard the ligands labeled by MOAD as "part of proteins" as these are constituted mostly by metals and covalent binders.

4. We discard very large ligands formed by more than 60 heavy atoms.
5. Ligands with equal chemical composition that are bound to the same protein are also clustered together forming a single ligand with multiple correct poses.
6. To ensure diversity in the benchmark, if the same ligand appears bound to different biological assemblies of the same PDB entry we only select one of these at random.
7. To avoid overrepresentation of certain ligand classes, we limit the maximum number of ligands to 5 of the same molecule separately in the validation and test sets. This avoids, for example, the presence of a large number of NAD ligands when considering the "NAD-binding domain" cluster.
8. To follow the convention that has been used to train previous ML-based blind docking algorithms, we select only the protein chains that have a C-alpha atom within 10Å of a ligand heavy atom. In the cases of multiple equal ligands bound to the same complex, one was selected as reference for the filtering.

After these steps we are left with a validation set composed by 141 unique complexes from 70 different clusters and a test set formed by 189 complexes from 63 clusters.

C Data Augmentation Strategies Details

C.1 Increasing the training data

To increase the training data using the clusters of Binding MOAD that did not go into the validation or test set, we first take the clusters that remained after applying step 1 above. We also remove all complexes resolved in 2019 or later to maintain the validity of the PDBBind test set. Then, we again discard all ligands that are close to other ligands and those with a single heavy atom.

We take the remaining complexes, forming a dataset with 20,012 ligand poses bound to 14,214 different biological units, and use them for training. For ligands that are bound to the same biological unit, at every epoch we select a single ligand at random for training. At training time, we simply concatenate this dataset to PDBBind's training set. One should note that the two datasets are not disjoint and many complexes will appear in both. We avoid removing the redundancy to give more weight to complexes that passed both filtering processes of MOAD and PDBBind and are therefore more likely to be of higher quality.

C.2 van der Mer-inspired training

To extract van der Mer-like amino-acid ligands, we start from a large collection of protein structures that comprise the ProteinMPNN [8] training set. At preprocessing time, for every sidechain s in the protein, we compute n_s the number of other amino acids that have heavy atoms within 5Å and are not local in primary sequence (within 10 residues). This determines the extent to which the remaining part of the protein forms a pocket around the selected sidechain, once the adjacent sequence on each side is removed.

At training time, we iterate over the clusters of proteins in the dataset. We use the same clusters adopted by ProteinMPNN that were generated with a sequence similarity cutoff of 30%. For each cluster, we take one protein at random, and within this, we select a sidechain s at random with probability proportional to $\max(0, n_s - 5)$. Effectively, sampling buried residues with a higher likelihood. If no sidechain has more than 5 contacts then the protein is skipped.

Once a sidechain is selected, the sequence-local segment of the protein chain, consisting of 21 residues centered at the selected sidechain (i.e. 10 residues on either side), is removed from the protein structure. The sidechain and its backbone atoms are then used as the ligand that the model is trained to dock.

Note that for computational efficiency, instead of re-computing the embedding each time we truncate a protein structure, we compute the ESM embeddings [20] used by DiffDock only once with the full sequence and then simply crop the removed sequence out (even though the embedding of the remaining residues might be affected by the residues that were removed).

D Performances on DockGen Benchmark

D.1 Data augmentation strategies

Before looking at the empirical performance of CONFIDENCE BOOTSTRAPPING, we explore whether data augmentation strategies are sufficient to improve the generalization capacity of an ML docking model like DIFFDOCK.

Increasing the training data First we investigate how much the addition of more data, within the protein domains that the model already sees, can help with generalization. This is an important question because in recent years there has been a call from the research community for pharmaceutical companies to release the large number of crystallography structures they possess. We test this question by including in the training data all MOAD complexes from the same binding protein domains as those seen in PDBBind training and validation sets and released before 2019 (to maintain the validity of the PDBBind test set). After filtering out duplicates and non-conformant ligands, this increases the number of training data points by approximately 50%.

Van der Mer-inspired docking A problem with increasing the amount of training data, as discussed above, is the fact that it is limited to the same pocket diversity as the extra data points lie within the same protein domain clusters. To try to move beyond this, we design a novel auxiliary training task based on the generation of synthetic docked poses using protein sidechains as surrogate ligands. We take inspiration from the concept of a van der Mer, which has been used successfully in the design of proteins that bind target ligands [30]. A van der Mer is an amino acid that interacts with another amino acid that is distant in the 1D protein sequence, which can closely approximate a noncovalent protein-ligand interaction. In a given protein crystal structure, we identify a sidechain with a large number of protein contacts distant in sequence. The interacting amino acids are assigned as the "binding pocket" for the chosen sidechain (see Figure 3-C). We remove the coordinates of the "ligand" residue and its sequence-local neighbors from the protein to generate the new target protein (more details on how exactly these are chosen can be found in Appendix C.2).

The advantage of these synthetic data points is that they are myriad and easy to compute, since any (even unliganded) protein structure can be used to generate such examples. Thus, we can potentially dramatically increase the structural and chemical diversity of binding domains and pockets. This diversity could help the model understand the chemical and geometrical environments of different pockets. The drawbacks are that these synthetic complexes are of unknown affinity (many could be weak binders), and the chemical diversity of ligands is limited to the 20 amino acids.

D.2 Experimental results

The results of these additions, reported at the bottom of Table 1 show that, on one hand, increasing the training data improved the generalization of DIFFDOCK. On the other, the van der Mer-inspired sidechain-docking strategy exhibits some improvements on the validation set, but does not make significant accuracy improvements on the held-out test set. Still, ML-based docking models are far from the accuracy of search-based methods and even further from a performance that would be satisfactory to fulfill the promise of blind docking. Therefore, we hypothesize that, due to the plateauing of the increase in available crystallographic structures, ML-based docking methods must utilize training techniques that go beyond the data augmentation strategies to achieve a combination of high expressivity and strong generalizability.

E Improvements on Model Architecture

Score model We speed up the architecture and execution of DIFFDOCK's score model in a number of ways. First, we add a number of embedding message-passing layers which, unlike the cross-attentional interaction layers of DIFFDOCK, independently process the protein and ligand structures. This allows us to increase the depth of the architecture with very little added runtime. In fact, due to the significantly higher number of nodes, the main complexity of the embedding layer lies in the protein component. However, under the rigid protein assumption, the structure is the same across all the different samples and diffusion steps. Therefore, the protein embedding can be computed only once resulting in minor computational overhead.

Table 1: Performance of different methods on the PDBBind and DOCKGEN benchmarks. Baseline performances on PDBBind are taken from Stark et al. [33] and Corso et al. [7]. DIFFDOCK† indicated the series of changes made to DIFFDOCK score and confidence model to make it more efficient reported in Appendix E. Runtimes were computed as averages over the PDBBind test set. * indicates that the method was only run on CPU as the code did not support GPU acceleration.

	PDBBind		DOCKGEN-full		DOCKGEN-clusters		Average
Method	%<2Å	%<5Å	%<2Å	%<5Å	%<2Å	%<5Å	Runtime (s)
SMINA	18.7	38.0	7.9	22.8	2.4	15.3	126*
SMINA (EX. 64)	25.4	47.8	10.6	22.2	4.7	14.1	347*
P2RANK+SMINA	20.4	43.0	7.9	22.7	1.2	14.1	126*
GNINA	22.9	40.8	14.3	28.0	9.4	28.2	127
GNINA (EX. 64)	32.1	53.6	**17.5**	39.2	**11.8**	43.5	348
P2RANK+GNINA	28.8	47.8	13.8	27.0	4.7	22.4	127
EQUIBIND	5.5	39.1	0.0	4.8	0.0	7.1	**0.04**
TANKBIND	20.4	59.0	0.5	16.9	0.0	15.9	0.7
DIFFDOCK (10)	35.0	61.7	7.1	33.7	6.1	40.2	10
DIFFDOCK (40)	**38.2**	63.2	6.0	32.1	3.7	35.4	40
DIFFDOCK† (10)							
+ increased training data	29.8	61.0	7.6	39.1	9.8	47.6	2.8
+ van der Mers training	33.0	**66.0**	7.1	**40.8**	8.5	**59.8**	2.8

Second, we limit the order of the spherical harmonics used to represent the edges to one (rather than two). This reduces the tensor product convolution used by DIFFDOCK's score and confidence models to a set of scalar, dot, and vector products, for which we can explicitly write a more efficient kernel instead of relying on expensive `e3nn` convolutions [10]. Finally, we batch and move the whole reverse diffusion process to GPU to improve parallelization and memory transfers. As shown in Table 1, this set of techniques provides us with a significantly faster sampling method.

Confidence model We also change a number of design choices in the confidence model to make it more efficient and better suited for CONFIDENCE BOOTSTRAPPING. The first choice is to force the model to reason about local interactions, as the affinity of a ligand to a particular pose is largely a local property. We achieve this by simply feeding to the model the receptor structures of only the amino acids whose C-alpha is within 20Å of any of the predicted ligand atoms' positions.

Further, during the binary classification training of the confidence model, we try to remove the bias of the model against hard targets by balancing the proportion of positive and negative examples the model is trained over. When not possible with sampled poses (because the model only samples negatives), we use the so-called conformer-matched ligand poses [17] as positive examples. Moreover, to make the transition smoother and reduce the perceived variance in the labels, we separate the positive (poses with RMSD<2Å) and negative classes (>4Å)[3]. Finally, we supervise the model not only to predict the label of the whole ligand but also that of each individual atom (supervised on their individual distance to the ground truth pose).

F Confidence Bootstrapping Details

In this section, we discuss the CONFIDENCE BOOTSTRAPPING procedure and our experimental setup. We demonstrate that CONFIDENCE BOOTSTRAPPING improves blind docking performance of the pretrained score model on a previously unseen cluster without knowing the ground-truth poses of that cluster. This is achieved by iteratively sampling from the score model (the rollout process), evaluating these samples with the confidence model, and updating the score model treating high-quality samples as ground-truth poses. Our procedure assumes a DIFFDOCK score model and a confidence model both pretrained. We use the procedure outlined in Corso et al. [7] to train and sample from the score model.

[3]We note that contemporary work Masters et al. [25] also applies this strategy.

F.1 Training Procedure

The training procedure takes as input ligand/receptor pairs $D = \{d_1, d_2, ...\}$ from a cluster of interest, a pretrained diffusion model parametrized by θ with learned distribution over poses $p_\theta(\mathbf{x}; d)$ and score $s_\theta(\mathbf{x}^{(t)}, t; d)$, and a pretrained confidence model $c_\phi(\mathbf{x}, d)$. At each step, we sample complexes from the score model through the reverse diffusion process, score these complexes with the confidence model, and use high-confidence samples to update the score model. More specifically, we start with the buffer B that is initially empty; at step i, we sample q candidate poses $(\mathbf{x}_j, d_i)$ for $j \in \{1, \cdots, q\}$ for each $d_i \in D$ from $p_{\theta^i_{ema}}(\mathbf{x}; d)$ and evalute these candidate poses with c_ϕ. We use the exponential moving averaged version of model weights θ^i_{ema} as in Corso et al. [7] to stabilize the inference over different training epochs. We add the sampled poses with a confidence score $c_\phi(\mathbf{x}, d) > k$, where k is the confidence cutoff, to the buffer: $B = B \cup \{(\mathbf{x}_j, d_i) \mid c_\phi(\mathbf{x}_j, d_i) > k\}$. Then, we update the score model θ^i by treating complexes in B as ground truth poses and performing a fixed number of SGD update steps on the diffusion objective to obtain θ^{i+1}.

There are a few nuances to building and using the buffer B. First, we resample a constant m complexes from B to train the score model at every step. Motivated by the intuition that higher-score poses should be sampled more frequently, the poses are sampled from the distribution $p(\mathbf{x}, d) \sim \frac{1}{Z} \exp[c_\phi(\mathbf{x}, d)]$ where Z is a normalization constant. Additionally, we enforce an upperbound on the number of samples per complex to encourage diversity across complexes in B by only keeping the n highest scoring poses per protein/ligand pair. Moreover, updating the score model with samples from the buffer may cause it to lose information about how to perform steps with $t < t_{\min}$ and/or learn to sample from pathological spaces of the confidence model, where the confidence model assigns high scores to poses distant from the ground truth. Therefore, at each training step, we also include p randomly sampled complexes from PDBBind in addition to complexes sampled from B. In our scheme, k, p, q, m, and n are hyperparameters selected through testing the method on the validation dataset.

F.2 Experimental Details

Setup. We tested CONFIDENCE BOOTSTRAPPING on DOCKGEN-clusters, a subset of DOCKGEN containing 8 clusters with at least 6 complexes each. For every cluster in DOCKGEN-clusters, we started with the same pretrained diffusion and confidence models, and ran 60 iterations of CONFIDENCE BOOTSTRAPPING, where each iteration contained a rollout step and an update step with 200 SGD updates. We did two evaluation runs with the same hyperparameters and reported the averaged results. To evaluate the generated complexes, we computed the symmetric-corrected RMSD of sPyRMSD [28] between the predicted and the crystal ligand atoms. We reported the percentage of top-ranked predictions that have an RMSD less than 2Å, a metric generally considered to be representative of getting the correct pose [1, 13, 27, 7].

Hyperparameters. In our experiments, we chose confidence cutoff $k = -4$, number of complexes sampled from PDBBind $p = 100$, number of complexes sampled from the buffer $m = 100$, number of inference samples $q = 32$, and maximum samples per protein/ligand pair $n = 20$. At the rollout step, we ran 4 inference steps each with 8 samples and compute the RMSD less than 2Å metric with the top-ranked pose from each inference step to reduce variance in the reported metric. Additionally, we set number of inference samples to 80 for the first bootstrapping inference step to fill the initially empty buffer. These parameters were selected by testing the method on the 5 DOCKGEN validation clusters.

F.3 Oracle Experiments

In a separate experiment, we tested CONFIDENCE BOOTSTRAPPING with "oracle" confidence predictions. The oracle confidence predictor is a monotonic transformation of the true RMSD between the ground truth pose and the predicted pose: $c_\phi(\mathbf{x}, d) = -4 \tanh\left(\frac{2}{3} RMSD(\mathbf{x}, \mathbf{x}^*) - 2\right)$. Note that this is not a practical setting as we would not have access to the ground truth poses $\mathbf{x}^*$ in real applications. However, this is an illustrative experiment establishing an upperbound on the performance gains achievable through CONFIDENCE BOOTSTRAPPING with a "perfect" confidence model. We used the same set of hyperparameters used on the test datasets and report the results at Figure C.1.

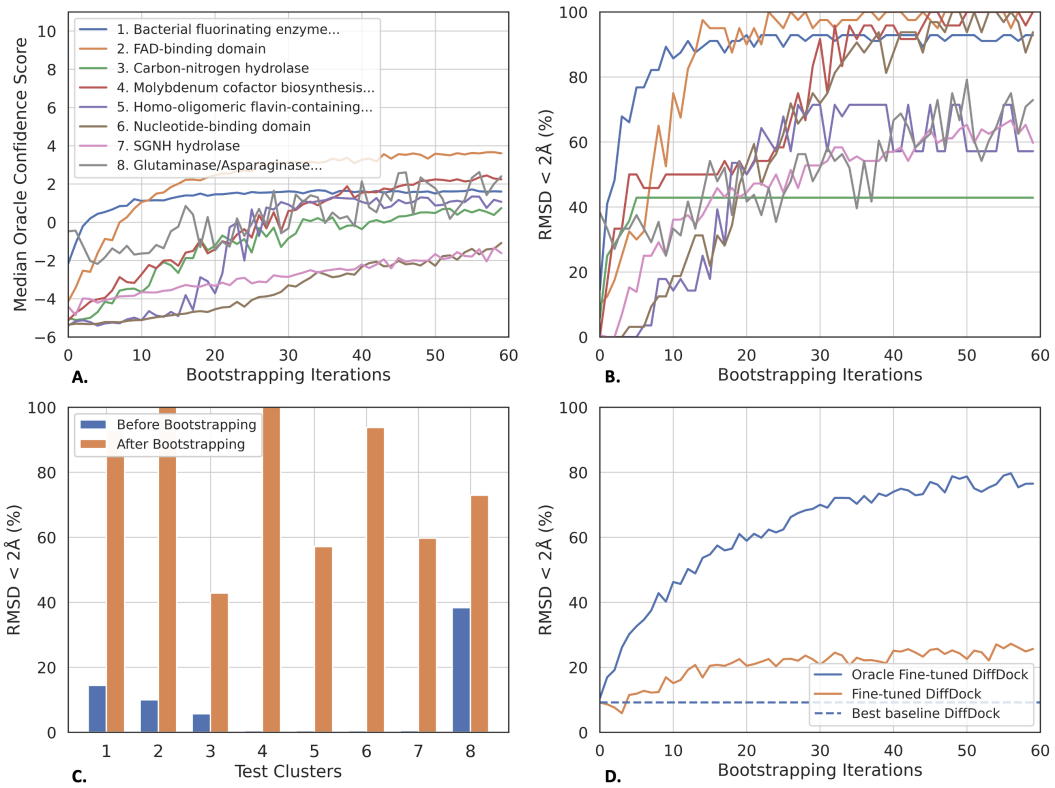

Figure C.1: Empirical performance of CONFIDENCE BOOTSTRAPPING with oracle confidence across the 8 protein domain clusters within DOCKGEN-cluster, the bootstrapping method is run twice for every cluster and we show the average results of the two runs. All performances are measured based on the top-1 pose when taking 8 inference samples with the fine-tuned models. **A.** Median confidences of sampled points at every iteration for each cluster. **B.** Proportion of top-1 predictions below 2Å over the course of the iterations for each cluster. **C.** Performance for each cluster before the fine-tuning and after the K=60 steps of CONFIDENCE BOOTSTRAPPING. **D.** Aggregated performance for all the clusters weighted by their number of complexes, showing results using the oracle confidence model and pretrained confidence model.