Physics-informed deep neural network for rigid-body protein docking

Freyr Sverrisson, Jean Feydy, Joshua Southern, Michael M Bronstein, Bruno E Correia

PHYSICS-INFORMED DEEP NEURAL NETWORK FOR RIGID-BODY PROTEIN DOCKING

Freyr Sverrisson
École Polytechnique Fédérale de Lausanne
and Swiss Institute of Bioinformatics
freyr.sverrisson@epfl.ch

Jean Feydy
HeKA team, INRIA Paris
Centre de Recherche des Cordeliers,
Université de Paris Cité
jean.feydy@inria.fr

Joshua Southern
Imperial College London
joshua.southern17@imperial.ac.uk

Michael M. Bronstein
University of Oxford
michael.bronstein@cs.ox.ac.uk

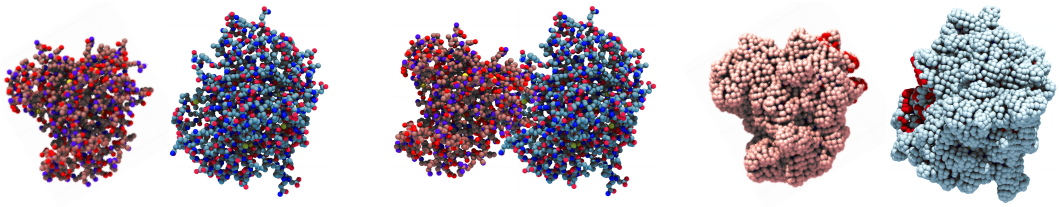
Bruno E. Correia
École Polytechnique Fédérale de Lausanne
and Swiss Institute of Bioinformatics
bruno.correia@epfl.ch

ABSTRACT

Proteins are biological macromolecules that perform many essential roles within all living organisms. Many protein functions arise from physical interactions between them and also with other biomolecules (e.g. DNA, metabolites). Being able to predict whether and how two proteins interact is an important problem in fundamental biological research and translational drug discovery. In this work, we present an energy-based model for generating ensembles of rigid-body transformations to predict the configuration of protein complexes. The method incorporates strong, interpretable physical priors. By construction, it is also $\mathrm{SE}(3)$ equivariant and fully-differentiable back to the atomic structure. We rely on the observation that bound protein-protein complexes show high geometric and chemical complementarity at the interface of the two proteins. Our method efficiently makes use of this prior by generating on-the-fly point cloud representations of the solvent-excluded surfaces of the proteins. Through learned point descriptors, we can infer regions of high complementarity between the two proteins and compute a proxy for the binding energy. By sampling transformations expected to adopt low energy states, we generate ensembles of bound poses where the two protein surfaces are brought into contact. We expect that the strong physical priors enforced by the network construction will aid in generalization to other related tasks and lead to a richer human understanding of the process behind the generation and scoring of the docked poses. The fact that the method is also fully differentiable allows for gradient-based modifications of the atomic structure which could be critical in tasks related to unbound docking or protein design which remain outstanding problems in protein modelling.

1 INTRODUCTION

Proteins are biological macromolecules built of sequences of amino acids. There are 20 different types of amino acids found in nature and the exact amino acid sequence of a protein determines the three-dimensional structure it will adopt. This structure can be determined experimentally with methods such as X-ray crystallography, Nuclear Magnetic Resonance and Cryo Electron Microscopy. Proteins serve an incredibly diverse set of functionalities within living organisms. The functionalities are defined by the protein structure and typically involve a sequence of binding events to other molecules.



(a) Binder. (b) Target. (c) Docking configuration. (d) Binder and target surfaces.

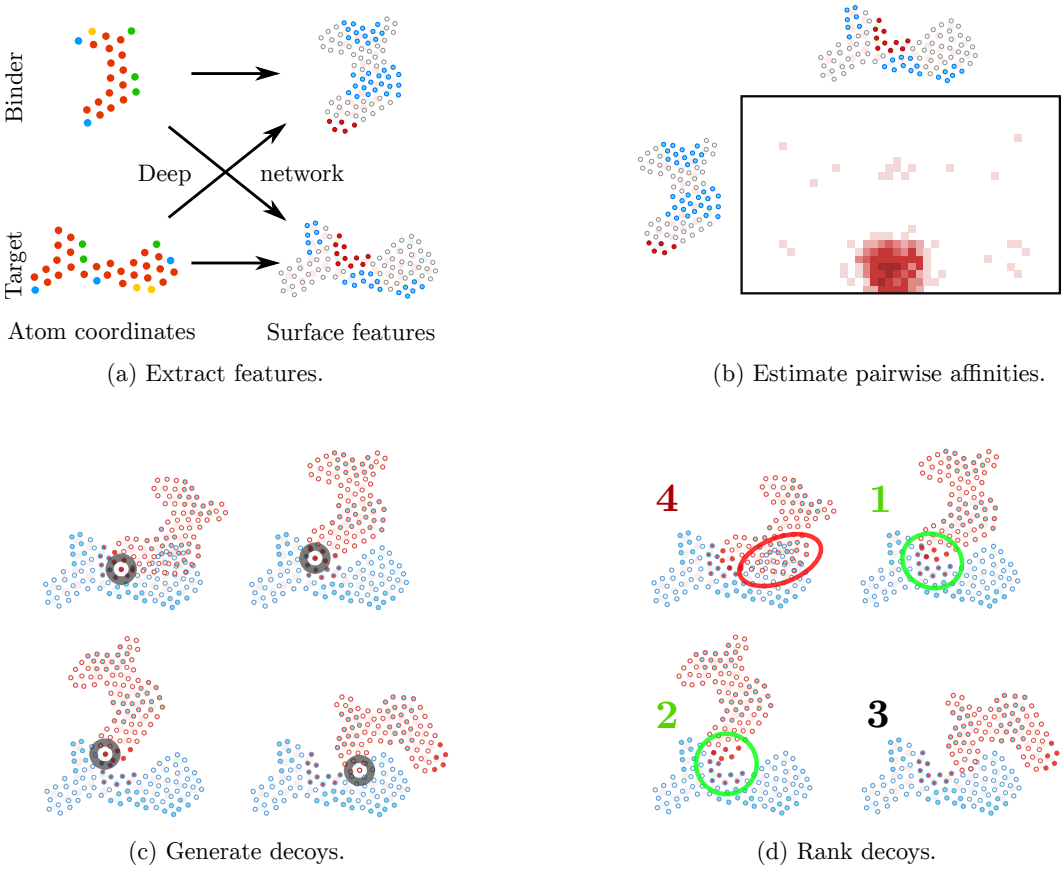
Figure 1: **The rigid docking problem.** Let us consider a pair of proteins, the moving binder (a) and the fixed target (b). Our goal is to predict a rigid-body transformation of the binder that corresponds to a docking configuration (c) that has been observed in the wild. To this end, we build upon our previous works on protein surface fingerprints Gainza et al. (2020); Sverrisson et al. (2021): we use a point neural network to predict the locations of the docking sites (highlighted in red) on both protein surfaces, before relying on a probabilistic method to generate and rank plausible 3D configurations.

Protein-protein docking is an important challenge within structural bioinformatics and revolves around the prediction of how two proteins interact with each other given their individual structures. The complexity of the task is increased dramatically by the fact that proteins are not structurally rigid. The rigidity varies over the structure with some parts highly flexible (such as in loop regions) and others (such as helices) being more stable. This non-rigidity of proteins allows them to undergo structural rearrangement, called induced-fit, when coming into contact with another molecule. However, there are proteins that undergo minimal structural changes upon binding to their partner. In such cases we can assume that the transformation that brings one protein into contact with another one is a rigid-body transformation.

In this paper, we focus on the problem of rigid-body docking and aim to address the problem of flexible docking in future work. As illustrated in Fig. 1, our framework focuses on the molecular surfaces of the interacting proteins. It is fully differentiable back to the atomic structure, is by construction equivariant to rigid transformations in the special Euclidean group $\mathrm{SE}(3)$, allows for the generation of multiple possible binding modes, and incorporates physical priors to aid in explainability.

2 Related work

The surface shape-complementarity of two proteins bound to each other is a well documented phenomena (Lawrence & Colman, 1993; Jones & Thornton, 1996) and has been widely used in classical methods for protein-protein docking (Gabb et al., 1997; Chen & Weng, 2002; Duhovny et al., 2002; Comeau et al., 2004). Rigid-body docking is typically the first step in traditional docking tools, followed by an iterative refinement which takes into account the flexibility of the molecules. Rigid-body docking was revolutionized by Katchalski-Katzir et al. (1992) by the usage of implicit representations of protein surfaces and by using fast Fourier transform of a correlation function to assesses the degree of shape complementarity and penetration upon rotation and translation of the molecules in three dimensions. We see our method as a natural development of the direction set by Katchalski-Katzir et al. (1992), taking full advantage of modern deep learning methods.
Deep learning has already made a big impact in structural biology (Laine et al., 2021). AlphaFold has provided a solution to the protein folding problem whereby the three-dimensional structure of a protein is determined from its amino acid sequence Jumper et al. (2021). Additionally, deep learning methods have started recently to be used for protein-protein docking. Evans et al. (2021) capitalized on the success of AlphaFold to develop AlphaFold-multimer which both folds and docks two protein structures. AlphaFold-multimer takes as inputs the proteins' sequences along with multiple sequence alignments (MSAs) of evolutionary related proteins. The MSAs can be used to detect co-evolving residues in the sequence and based on that infer about their proximity. Additionally, a pairwise-independent $\mathrm{SE}(3)$-equivariant graph matching network, termed EquiDock, has been proposed for end-to-end rigid protein docking and shows promising results for deep learning applied to this problem Ganea et al. (2021). A possible limitation to both these deep learning solutions, AlphaFold-multimer and EquiDock, is that they only predict a single binding mode for the two



(a) Extract features.

(b) Estimate pairwise affinities.

(c) Generate decoys.

(d) Rank decoys.

Figure 2: **The four main steps of our method.** (a) First, we apply a deep point neural network on both atomic point clouds to compute surface fingerprints. This step is detailed in Fig. 3 and relies on operations that were introduced in Sverrisson et al. (2021). We retrieve two 3D point clouds $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{\mathrm{N}}$ (binder) and $\boldsymbol{y}_1, \dots, \boldsymbol{y}_{\mathrm{M}}$ (target) sampled on the protein surfaces, with feature vectors $\boldsymbol{f}_1, \dots, \boldsymbol{f}_{\mathrm{N}}$ and $\boldsymbol{g}_1, \dots, \boldsymbol{g}_{\mathrm{M}}$ that describe local geometric and chemical properties. (b) We estimate a $\mathrm{N} \times \mathrm{M}$ matrix of binding energies $\Delta E(\boldsymbol{x}_i, \boldsymbol{y}_j)$ between any two points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ on the protein surfaces. We apply a softmax operator to turn this matrix of energies into a probability distribution over pairs of points $(\boldsymbol{x}_i, \boldsymbol{y}_j)$, displayed here as a heatmap. (c) We sample pairs of plausible contact points according to this probability distribution. We generate candidate docking configurations ("decoys") by aligning the corresponding tangent planes. (d) We use a global energy function to score and rank these docking configurations. Our energy function combines a Van der Waals term (to penalize collisions between the two atomic point clouds, highlighted in red) with a surface interaction term (to incentivize a full alignment of the docking sites, highlighted in green).

protein binding pairs. Proteins are however dynamical structures and global properties such as the binding energy between them relies on the whole ensemble of different binding configurations.

3 Method

3.1 Outline

Main physical priors. Our model makes use of several key physical priors. First and foremost, we rely on the observation that bound protein-protein complexes show high geometric and chemical complementarity at the interface of the two proteins. As a consequence, we work with surface representations of proteins and aim to learn informative surface point descriptors that can be used to match together corresponding interface regions. The model is SE(3)-equivariant and symmetric with respect to which one of the proteins is considered as the binder and which one is the target.

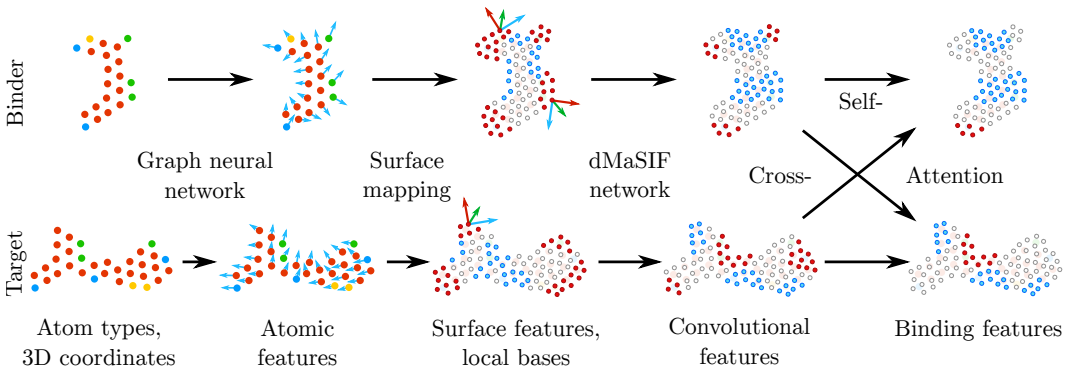


Figure 3: **Our feature extraction pipeline.** As detailed in Sec. 3.2, we apply a graph neural network, an atoms-to-surface mapping, a convolutional neural network and successive attention layers on our representations of the binder and target proteins. The networks on the two lines of this diagram share the same architectures and weights: our method produces the exact same results when the roles of the two proteins are exchanged with each other.

We also include more specific priors based on the structure of the energy function and explicitly construct surface features to behave like an electric field.

Our architecture. The overall proposed architecture consists of two main parts, a generator network and a discriminator network which are further divided into four steps: 1) we compute surface descriptors for both proteins; 2) we estimate the binding energy between their different surface regions; 3) we sample plausible transformations ("decoys") with low predicted binding energy; 4) we rank the generated transformations using a global energy function.
We rely on methods introduced in our previous works (Sverrisson et al., 2021; Gainza et al., 2020) to generate and perform operations on protein surfaces. In Sverrisson et al. (2021) we introduced dMaSIF, a framework that incorporates a fully-differentiable method to generate protein surfaces and a fast convolutional operator that operates in the quasi-geodesic space of the surface. In order to increase speed and reduce the memory footprint of our computations, we rely extensively on the KeOps library (Charlier et al., 2021; Feydy et al., 2020).

3.2 Computing surface fingerprints

The architecture for computing the point descriptors consists of three main parts: 1) a physics-informed graph neural network that passes information between atoms and from the atoms to the surface points; 2) a shallow convolutional neural network operating in the quasi-geodesic space of the surface point cloud and 3) a geometry-informed attention based network that passes information between the surfaces of the two binding partners.

Atomic features mapped to surface points In this method we aim to learn features which are both easily interpretable and have a strong connection to known physical laws. In our previous method, dMaSIF (Sverrisson et al., 2021), we mapped to the surfaces chemical features which were learned through graph neural networks that took as inputs the underlying atomic types and distances to atoms. In this work, we restrict the learnable feature space through global soft-constraints such that the chemical features mapped to the surface respect *Gauss' law*, which states that the total electric flux out of a closed surface is proportional to the total charge enclosed within that surface:

$$\oint \boldsymbol{E} \cdot d\boldsymbol{A} = \frac{q_{\text{enc}}}{\varepsilon_0}.$$

In our case, we do not know the exact charge of the protein enclosed within our surface, but can approximate it using the formal charge of the protein. The formal charge depends on the amino acid composition of the protein. Each amino acid is assigned a charge according to it's amino acid type and in order to get the total charge we sum over the protein sequence.

Our graph neural network has the following form: For atomic feature vectors $\boldsymbol{f}_i$ and the corresponding coordinates of that atom $\boldsymbol{x}_i$, one layer is given by

$$\boldsymbol{f}_i' = \sum_{j \in N(i)} (\boldsymbol{x}_i - \boldsymbol{x}_j)\text{MLP}(\boldsymbol{f}_i, \boldsymbol{f}_j, ||\boldsymbol{x}_i - \boldsymbol{x}_j||)$$

Where the output dimensionality of the MLP is 1. Note that if $\boldsymbol{f}_j$ includes for instance the atomic charges of the atoms, the MLP could learn a function that maps the electric field to $\boldsymbol{f}_i'$.

The next step maps the atomic features to the surface, for a surface point i and an atom j the atomic feature vector is first projected onto the distance vector between the surface point and the atom:

$$\boldsymbol{f}_j' = \frac{\boldsymbol{f}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)}{||\boldsymbol{x}_i - \boldsymbol{x}_j||}(\boldsymbol{x}_i - \boldsymbol{x}_j)$$

The chemical feature of the surface point then becomes

$$f_i = \sum_j \frac{\boldsymbol{f}_j' \cdot \hat{\boldsymbol{n}}_i}{\tau ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}$$

where $\hat{\boldsymbol{n}}_i$ is the surface normal at point i and τ is a learnable parameter.

In order to respect Gauss's law we then add as a loss term

$$l_{\text{Gauss}} = \left\| \frac{C_{\text{formal}}}{\epsilon} - \sum_i f_i \right\|$$

where C_{formal} is the formal charge of the protein and ϵ is a learnable parameter.

Surface convolution with dMaSIF. The input to our convolutional layers are the mean and Gaussian curvatures of the surface at different scales along with the chemical feature described above. For the convolution we follow the same methodology described in dMaSIF except for some small modifications: When performing the convolution

$$\boldsymbol{f}_i' = \sum_j w(d_{ij})\text{MLP}(\boldsymbol{p}_{ij})\boldsymbol{f}_j$$

we don't rely on a Gaussian window $w(d_{ij}) = \exp(-d_{ij}^2/2\sigma^2)$ with a fixed variance σ^2, rather we use linear combinations of multiple Gaussians

$$\boldsymbol{w}(d_{ij}) = \boldsymbol{W} \cdot \exp(-d_{ij}^2/2\boldsymbol{\sigma})$$

where $\boldsymbol{W}$ is a weight matrix and $\boldsymbol{\sigma}$ is a vector of learnable variances.

Interaction between both proteins with global attention. In order to pass information between the two binding partners and update their surface features based on the proposed binding protein they are passed through sequential blocks of cross- and self-attention:

$$\boldsymbol{F}' = \boldsymbol{F} + \text{CrossAttention}(\boldsymbol{F}\boldsymbol{W}_Q, \boldsymbol{G}\boldsymbol{W}_K, \boldsymbol{G}\boldsymbol{W}_V)$$
$$\boldsymbol{F}'' = \boldsymbol{F}' + \text{SelfAttention}(\boldsymbol{F}'\boldsymbol{W}_Q, \boldsymbol{F}'\boldsymbol{W}_K, \boldsymbol{F}'\boldsymbol{W}_V)$$

Where $\boldsymbol{F}$ is the matrix of feature vectors for the surface points of one protein and $\boldsymbol{G}$ is the matrix of feature vectors for it's corresponding binding partner. We pass each protein pair through 4 blocks of such attention-based feature updates where each attention layer has 2 heads.

The cross attention layers have the familiar form

$$\text{CrossAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$$

Where $\boldsymbol{Q}$ are the feature vectors of one protein after they have been linearly transformed and $\boldsymbol{Q}$ and $\boldsymbol{V}$ are the feature vectors of the binding partner after a linear transformation.

If two surface points are close in geodesic space and have normals and tangent vectors pointing in similar directions, their expected transformations should ideally be similar as well.

Guided by this intuition the self-attention layers in our architecture are constructed to be somewhat spatially aware, they take the form

$$\text{SelfAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}} - \frac{\boldsymbol{D}}{\tau})\boldsymbol{V}$$

Where $\boldsymbol{D}$ is a matrix of all pair-wise approximate geodesic distances and τ is a learned parameter. The feature vectors are moreover augmented by being concatenated with the surface normals and tangent vectors before the computation of $\boldsymbol{K}$ and $\boldsymbol{Q}$ (but not $\boldsymbol{V}$ in order not to break SE(3)-equivariance), that is $\boldsymbol{K} = [\boldsymbol{F}'|\boldsymbol{N}|\boldsymbol{U}|\boldsymbol{V}]\boldsymbol{W}_K$.

3.3 Generating and ranking decoys

Estimating the binding energy between any two points. When two separate proteins are freely floating in solution without interacting with each other, they have an "unbound" energy which associated to them. We assume that both proteins are rigid. If $S(\boldsymbol{x})$ (for "solvent") denotes the energy which is associated to the interaction between a surface point $\boldsymbol{x}$ and the solvent, we can write:

$$E_{\text{unbound}}(\text{P}_1, \text{P}_2) = \int_{\partial P_1} S(\boldsymbol{x})\, \text{d}\boldsymbol{x} + \int_{\partial P_2} S(\boldsymbol{x})\, \text{d}\boldsymbol{x}$$

Since the energy is defined up to an additive constant, we can use the convention that this unbound energy is equal to 0.

When the two proteins come in contact with each other, points at the interface get close to the atoms of the other protein and lose touch with water molecules. If $I(x, y)$ (for "interaction") is the energy that is associated to the contact between a point x in P_1 and a point y in P_2, we can write the change in energy (binding energy) as a result of that contact as:

$$\Delta E(P_1, P_2) = \int_{\partial P_1 \cap \partial P_2} I(\boldsymbol{x} \in P_1, \boldsymbol{y} \in P_2) - S(\boldsymbol{x} \in P_1) - S(\boldsymbol{y} \in P_2)\, \text{d}\boldsymbol{x}\text{d}\boldsymbol{y}$$

Our key modelling assumption is that the observed docking configurations correspond to low-energy states. We are going to learn surface-based functions I and S such that for any given protein pair (P_1, P_2) in our dataset, the ground truth binding transformation (R_{gt}, t_{gt}) is the transformation that results in the most favorable binding energy.

Let us assume that:

- The surface of the binder protein P_1 is sampled with N points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\text{N}}$ in 3D, with dMaSIF point descriptors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{\text{N}}$.
- The surface of the target protein P_2 is sampled with M points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\text{M}}$ in 3D, with dMaSIF point descriptors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{\text{M}}$.

Then, using small neural networks (Multi-Layer Perceptrons) that share the same parameters for both proteins and are applied to the point descriptors $\boldsymbol{f}_i$ and $\boldsymbol{g}_j$, we compute:

- Affinity vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ for all points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$.
- Complementarity vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ for all points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$.

- Unbound energies $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ for all points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$.

We then choose to compute the binding energy that is associated with bringing into contact the two points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ as:

$$\Delta E(\boldsymbol{x}_i, \boldsymbol{y}_j) = -\boldsymbol{a}_i^T \boldsymbol{a}_j + \boldsymbol{b}_i^T \boldsymbol{b}_j - \boldsymbol{s}_i - \boldsymbol{s}_j$$

Note that in our network, we approximate the interaction energy $I(\boldsymbol{x}_i, \boldsymbol{y}_j)$ using the general symmetric formula:

$$I(\boldsymbol{x}_i, \boldsymbol{y}_j) \simeq -\boldsymbol{a}_i^T \boldsymbol{a}_j + \boldsymbol{b}_i^T \boldsymbol{b}_j \ .$$

We stress that $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ (resp. $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$) are computed by applying the exact same networks (with the exact same weights) on both protein surfaces ∂P_1 and ∂P_2:

$$\boldsymbol{a}_i = \text{MLP}_A(\boldsymbol{f}_i) \ , \boldsymbol{a}_j = \text{MLP}_A(\boldsymbol{g}_j) \ , \boldsymbol{b}_i = \text{MLP}_B(f_i) \ , \boldsymbol{b}_j = \text{MLP}_B(g_j) \ .$$

This ensures that our network is fully symmetric with respect to a permutation of the proteins' roles.

In this context, decomposing the interaction between a "positive" part (associated to the affinity vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$) and a "negative" part (associated to the complementarity vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$) is necessary to represent the full range of possible interactions.

Generating docked decoys. Our generator network generates plausible transformations in the following way. First, we find all pair-wise transformations between surface points from the binder to the surface points of the target. For a given point i on the binder, we first flip the direction of its normal and one of its tangent vectors

$$[\hat{\boldsymbol{n}}_i' | \hat{\boldsymbol{u}}_i' | \hat{\boldsymbol{v}}_i'] = \text{diag}(-1, -1, 1) \times [\hat{\boldsymbol{n}}_i | \hat{\boldsymbol{u}}_i | \hat{\boldsymbol{v}}_i]$$

On the interface of the binder the directions of the normals are before this transformation by definition pointing in the opposite direction of the normals of the target points in contact, by flipping them we ensure that they point in the same directions. By flipping the direction of one of the tangent vectors we make sure the final transformation is a rotation and not a reflection.

We then find the transformation that brings the point to the origin of the coordinate system and rotates it such that its internal coordinate vectors align with the unit vectors along the axes:

$$x_i' = [\hat{\boldsymbol{n}}_i | \hat{\boldsymbol{u}}_i | \hat{\boldsymbol{v}}_i]^T (\boldsymbol{x}_i - \boldsymbol{x}_i)$$

For a given point j on the target, we then find the transformation that brings point i from the origin of the coordinate system to point j and aligns their normal and tangent vectors, that is

$$\boldsymbol{x}_i'' = [\hat{\boldsymbol{n}}_j | \hat{\boldsymbol{u}}_j | \hat{\boldsymbol{v}}_j] \boldsymbol{x}_i' + \boldsymbol{x}_j$$

The total transformation that brings point i into contact with point j consists then of a rotation and translation:

$$\boldsymbol{R}_{ij} = [\hat{\boldsymbol{n}}_j | \hat{\boldsymbol{u}}_j | \hat{\boldsymbol{v}}_j] \cdot [\hat{\boldsymbol{n}}_i | \hat{\boldsymbol{u}}_i | \hat{\boldsymbol{v}}_i]^T \ , \qquad \boldsymbol{t}_{ij} = \boldsymbol{x}_j - [\hat{\boldsymbol{n}}_j | \hat{\boldsymbol{u}}_j | \hat{\boldsymbol{v}}_j] \cdot [\hat{\boldsymbol{n}}_i | \hat{\boldsymbol{u}}_i | \hat{\boldsymbol{v}}_i]^T \boldsymbol{x}_i \ .$$

Given the binding energy between all pairs of points ΔE_{ij}, we can compute the expected binding energy of point i to the target as

$$\langle \Delta E_i \rangle = \frac{\sum_j \exp(-\Delta E_{ij}/\tau) \Delta E_{ij}}{\sum_j \exp(-\Delta E_{ij}/\tau)}$$

In the same way, we can get the expected value for the rotation, $\langle R_i \rangle$, and translation, $\langle t_i \rangle$ of point i. The rotations are parametrized as 6D vectors (Zhou et al., 2019) during the averaging.

In order to bias our generator network towards transformations close to the ground truth, we find the probability of a point i coming into contact with the target as:

$$P(i) = \sum_j P(i,j) = \frac{\sum_j \exp(-\Delta E_{ij}/\tau)}{\sum_{i,j} \exp(-\Delta E_{ij}/\tau)}$$

along with the interface-RMSD of the point's associated expected transformation as:

$$\text{RMSD}_i^{\text{pred}} = \sqrt{\sum_{\text{atoms}} \|\langle \boldsymbol{R}_i \rangle \boldsymbol{P}_b + \langle \boldsymbol{t}_i \rangle - (\boldsymbol{R}_{gt} \boldsymbol{P}_b + \boldsymbol{t}_{gt})\|^2}$$

Where the sum is taken over the interface atoms, defined as the atoms that in the ground truth configuration are less than 8Å away from any of the target atoms.

The main loss term while training the network is then

$$l_{\text{align}} = \sum_i P(i) \cdot \log(\text{RMSD}_i^{\text{pred}})$$

When sampling decoys from the generator network π we sample the expected transformations for each binder point according to their expected binding energy, that is the sample space of π is $\{(\langle \boldsymbol{R}_i \rangle, \langle \boldsymbol{t}_i \rangle) : \forall i\}$, where

$$\text{P}(\langle \boldsymbol{R}_i \rangle, \langle \boldsymbol{t}_i \rangle) = \frac{\exp(-\langle \Delta E_i \rangle / \tau)}{\sum_i \exp(-\Delta E_i / \tau)}$$

Ranking decoys. Each decoy is given an energy score through a trained discriminator network. The total score of a transformation is decomposed into two terms. The first one is given by a sum over the binding energies of the interacting surface points after transformation

$$\text{Energy}(\boldsymbol{R})_1 = \sum_{i=1}^{\text{N}} \sum_{j=1}^{\text{M}} \frac{k(\boldsymbol{R}\boldsymbol{x}_i, \boldsymbol{y}_j)}{d_i d_j} \big[-\boldsymbol{a}_i^T \boldsymbol{a}_j + \boldsymbol{b}_i^T \boldsymbol{b}_j - s_i - s_j \big]$$

where $k(\boldsymbol{x}_i, \boldsymbol{y}_j)$ is a Gaussian Kernel with a learnable variance, and d_i and d_j are local densities used to avoid varying sampling rates:

$$d_i = \sum_{t=1}^{\text{N}} k(\boldsymbol{x}_i, \boldsymbol{x}_t) \quad \text{and} \quad d_j = \sum_{t=1}^{\text{M}} k(\boldsymbol{y}_j, \boldsymbol{y}_t) \, .$$

The second term takes into account the Van Der Waals energy in order to model short-range attractive and repulsive forces that vary with atom-pair distance. To do this, we used the Lennard-Jones 12-6 potential which calculates the interaction energy of atoms i and j as

$$\text{E}_{\text{vdw}}(i,j) = \epsilon_{i,j} \left[\left(\frac{\sigma_{i,j}}{d_{i,j}} \right)^{12} - 2 \left(\frac{\sigma_{i,j}}{d_{i,j}} \right)^{6} \right]$$

where $\sigma_{i,j}$ is the sum of the atomic radii, d_{ij} is the atom-pair distance and $\epsilon_{i,j}$ is the geometric mean of well depths. During training of the discriminator we allow both $\sigma_{i,j}$ and $\epsilon_{i,j}$ to be learnable. The total energy of a configuration is the sum of the Van Der Waals energy over all atoms pairs and the second energy term can be formulated as

$$\text{Energy}(R)_2 = \text{E}_{\text{vdw}}(\text{P1P2}) - \text{E}_{\text{vdw}}(\text{P1}) - \text{E}_{\text{vdw}}(\text{P2})$$

The total score of the transformation is then given by

$$\text{Score}(R) = X\text{Energy}(R)_1 + Y\text{Energy}(R)_2$$

where X and Y are learnable weights.

3.4 TRAINING PROCEDURE

Training the generator network. The generator network is pre-trained to generate plausible alignments. Our main loss is the alignment loss l_{align} described before. In addition we have two auxiliary losses.

First, there's an auxiliary loss that increases the unbound energy of points that are on the interface of the two binding proteins. Interface points are defined as points that are within 1Å distance from any surface point of the other protein. The unbound energy loss is then

$$l_{\text{iface}} = -\frac{1}{N_{\text{iface}}} \sum_{i \in \text{interface}} \log \left(\frac{e^{S_i}}{\frac{1}{N} \sum_j e^{S_j}} \right)$$

where N is the total number of points in the surface point cloud and N_{iface} is the number of interface points.

The other auxiliary loss makes interface points that come into contact with each other have a lower binding energy than points on the interface that don't come into contact:

$$l_{\text{contact}} = -\frac{1}{N_{\text{iface}_1}} \sum_{i \in \text{interface}_1} \sum_{j \in \text{interface}_2} \log \left(\frac{e^{-\Delta E_{ij}}}{\frac{1}{N_{\text{iface}_2}} \sum_k e^{-\Delta E_{ik}}} \right) \cdot \frac{\exp\left(-\|x_i - x_j\|^2/\sigma\right)}{\sum_k \exp\left(-\|x_i - x_k\|^2/\sigma\right)}$$

Note that we have a soft matching between points based on the Eucledian distance between them, during training we set $\sigma = 0.1$

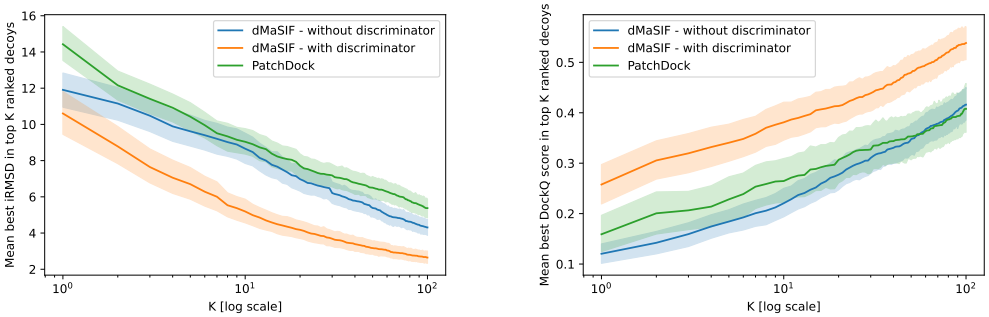
Training the ranking network (discriminator). The generator samples a proposal distribution on the group of 3D rigid transformations, $(\boldsymbol{R}_i, \boldsymbol{t}_i) \sim \pi(P_1, P_2)$. While training the discriminator network the weights of the generator network are frozen. The discriminator network is then trained using the same loss l_{align} as when training the generator. As such it is trained to give energy scores that minimize the expected log-RMSD of the proposed docoys.

4 EXPERIMENTAL EVALUATION

Datasets We train our networks using the Protein Interface Prediction (PIP) dataset from Townshend et al. (2020). The final evaluation is done on the bound forms of the structures in the Docking Benchmark 5 (DB5) (Vreven et al., 2015). The PIP dataset is split such that no structure in the training set has more than 30% sequence identity with any structure in the DB5.

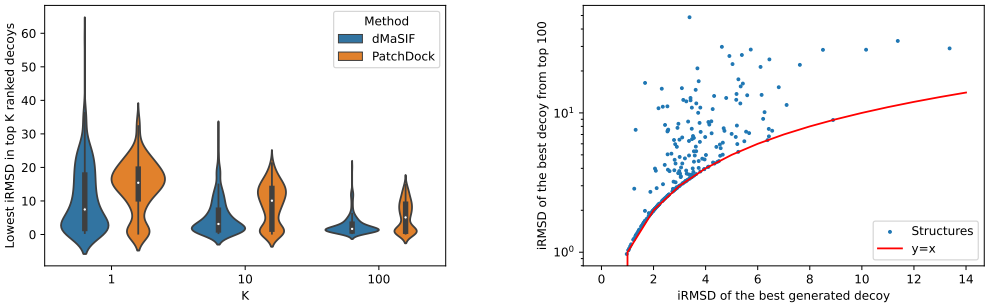
We protonate the protein structures using Reduce (Word et al., 1999) before passing them through our network. From the PIP dataset, we filter out complexes where either of the two binding partners has over 15000 atoms. We additionally filter out complexes where the minimum distance between any atom from the binder to any atom from the target is over 2.5Å. This leaves us with 70,730 pairs of complexes in the PIP training set, 24,466 complexes in the validation set and 14,392 complexes in the testing set. The average size of the proteins in our training set is 3872 ± 2444 atoms.

Baseline We benchmark our method against PatchDock (Duhovny et al., 2002; Schneidman-Duhovny et al., 2005). PatchDock is a good baseline for a few different reasons: 1) Like our method it does not allow for intra-molecular flexibility, and only does rigid-body docking. 2) It has fast running times relative to other classical docking tools. 3) It is an inherently surface-based method. PatchDock has three main steps: Surface generation and detection of concave, convex and flat surface patches; matching of surface patches and hotspot interface detection; Filtering decoys against unacceptable penetrations and scoring for shape complementarity. Each step in this pipeline has a clear correspondence to the steps in our method which as a result allows for an informative comparison.



(a) (b)

Figure 4: a) Mean minimum interface RMSD for the top K ranked decoys (lower is better). Shading represents 95% confidence interval. b) Mean maximum DockQ score for the top K ranked decoys (higher is better).



(a) (b)

Figure 5: a) Distribution of lowest iRMSDs in the top K ranked decoys for both dMaSIF and PatchDock. b) Shows for each structure in the DB5 on the x-axis the lowest iRMSD of the transformations generated by our generator network and the y-axis the lowest iRMSD picked in the top 100 by our discriminator network. With a perfect discriminator, each dot would end up on the red line.

Implementation We implement our architectures with PyTorch (Paszke et al., 2019) and use KeOps (Charlier et al., 2021) for fast geometric computations. For data processing and batching, we use PyTorch Geometric (Fey & Lenssen, 2019). The models are trained on 4 Tesla V100s. Both the DockQ score and interface RMSD were measured using an implementation from Basu & Wallner (2016).

Performance The comparison of the performance of our method to PatchDock is found in Fig. 4. Our generator network is already powerful enough to outperform PatchDock slightly on both the iRMSD and DockQ metrics. We emphasize that the generator network is *alignment-free*, in the sense that it does not actually perform the alignments it proposes in order to rank them but only gives each transformation an expected binding energy score. The discriminator further improves the performance. In our experiments, we use a very simple discriminator network with only 15 learnable parameters. By increasing the expressivity of the discriminator, we expect further improvement in performance, in particular, reducing the length of long tails seen in Fig 5a.

5 Conclusion

In this paper, we have presented an extension to our previous method dMaSIF (Sverrisson et al., 2021), which allowed us to perform rigid-body docking of protein-protein pairs. The proposed method has multiple attractive attributes: It is fully-differentiable back to the atomic structure, it

allows the generation of ensembles of docked poses, and it is structured to incorporate physical priors. The method shows strong performance in the benchmark we presented and the results indicate the performance could still be improved. We expect that this work could be a useful framework for numerous derived applications and further extensions. There are still challenges left when it comes to incorporating intra-molecular flexibility or designing new protein binders. We hope to be able to address of these challenges in future works.

6 ACKNOWLEDGEMENTS

The authors would like to thank Sergei Grudinin for useful discussions. FS is funded by a fellowship from the Swiss Data Science Center. MB is supported in part by the ERC Consolidator Grant No. 724228 (LEMAN). Most of the computation for this work was performed on the STI computing clusters at EPFL. JS is supported by the UKRI CDT in AI for Healthcare http://ai4health.io (Grant No. P/S023283/1).

REFERENCES

Sankar Basu and Björn Wallner. Dockq: a quality measure for protein-protein docking models. *PloS one*, 11(8):e0161879, 2016.

Benjamin Charlier, Jean Feydy, Joan Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021.

Rong Chen and Zhiping Weng. Docking unbound proteins using shape complementarity, desolvation, and electrostatics. *Proteins: Structure, Function, and Bioinformatics*, 47(3):281–294, 2002.

Stephen R Comeau, David W Gatchell, Sandor Vajda, and Carlos J Camacho. Cluspro: a fully automated algorithm for protein–protein docking. *Nucleic acids research*, 32(suppl_2):W96–W99, 2004.

Dina Duhovny, Ruth Nussinov, and Haim J Wolfson. Efficient unbound docking of rigid molecules. In *International workshop on algorithms in bioinformatics*, pp. 185–200. Springer, 2002.

Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew W Senior, Timothy Green, Augustin Žídek, Russell Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *BioRxiv*, 2021.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. *Advances in Neural Information Processing Systems*, 33, 2020.

Henry A Gabb, Richard M Jackson, and Michael JE Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *Journal of molecular biology*, 272(1):106–120, 1997.

Pablo Gainza, Freyr Sverrisson, Frederico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

Octavian-Eugen Ganea, Xinyuan Huang, Charlotte Bunne, Yatao Bian, Regina Barzilay, Tommi Jaakkola, and Andreas Krause. Independent se (3)-equivariant models for end-to-end rigid protein docking. *arXiv preprint arXiv:2111.07786*, 2021.

Susan Jones and Janet M Thornton. Principles of protein-protein interactions. *Proceedings of the National Academy of Sciences*, 93(1):13–20, 1996.

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.

Ephraim Katchalski-Katzir, Isaac Shariv, Miriam Eisenstein, Asher A Friesem, Claude Aflalo, and Ilya A Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences*, 89 (6):2195–2199, 1992.

Elodie Laine, Stephan Eismann, Arne Elofsson, and Sergei Grudinin. Protein sequence-to-structure learning: Is this the end (-to-end revolution)? *Proteins: Structure, Function, and Bioinformatics*, 89(12):1770–1786, 2021.

Michael C Lawrence and Peter M Colman. Shape complementarity at protein/protein interfaces, 1993.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. Patchdock and symmdock: servers for rigid and symmetric docking. *Nucleic acids research*, 33(suppl_2):W363–W367, 2005.

Freyr Sverrisson, Jean Feydy, Bruno E Correia, and Michael M Bronstein. Fast end-to-end learning on protein surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15272–15281, 2021.

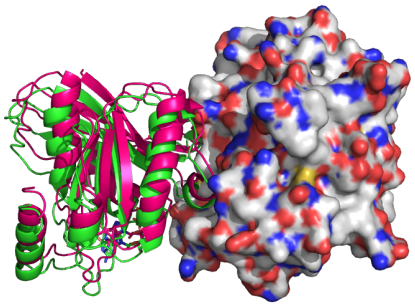
Raphael JL Townshend, Martin Vögele, Patricia Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Bowen Jing, Brandon Anderson, Stephan Eismann, et al. Atom3d: Tasks on molecules in three dimensions. *arXiv preprint arXiv:2012.04035*, 2020.

Thom Vreven, Iain H Moal, Anna Vangone, Brian G Pierce, Panagiotis L Kastritis, Mieczyslaw Torchala, Raphael Chaleil, Brian Jiménez-García, Paul A Bates, Juan Fernandez-Recio, et al. Updates to the integrated protein–protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2. *Journal of molecular biology*, 427(19):3031–3041, 2015.

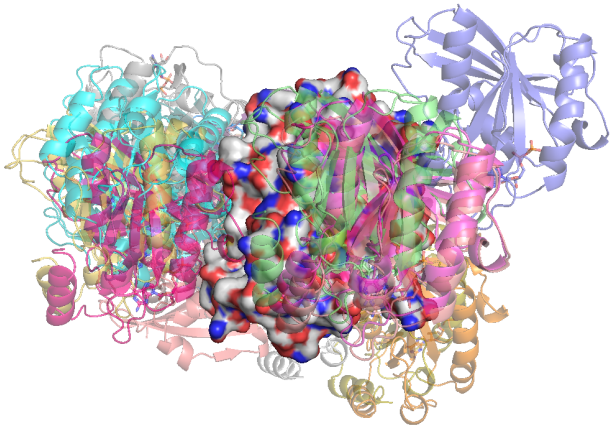
J Michael Word, Simon C Lovell, Jane S Richardson, and David C Richardson. Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation. *Journal of molecular biology*, 285(4):1735–1747, 1999.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

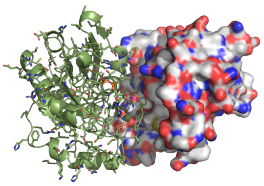
A APPENDIX



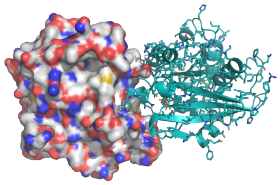
(a)



(b)



(c)



(d)

Figure 6: The figure shows a few examples of decoys for the complex 1A2K a) Shows the target structure in a surface representation, the native binder in green along with the best alignment in the top 20 in red. The decoy is ranked number 15 and has an RMSD of 0.88Å. b) shows with transparency the rest of the decoys in top 20 around the target. c) and d) show repectively the decoys ranked number 1 and 2. Although they are docked in the wrong configuration they show high interface complementarity and minimal clashes with the target.