Fast end-to-end learning on protein surfaces

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

Jean Feydy*
Imperial College London
jfeydy@ic.ac.uk
*equal contribution

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

Michael M. Bronstein
Imperial College London /
Twitter
m.bronstein@ic.ac.uk

Abstract

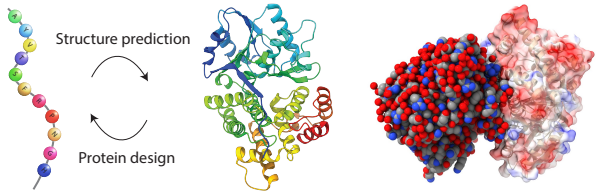
Proteins' biological functions are defined by the geometric and chemical structure of their 3D molecular surfaces. Recent works have shown that geometric deep learning can be used on mesh-based representations of proteins to identify potential functional sites, such as binding targets for potential drugs. Unfortunately though, the use of meshes as the underlying representation for protein structure has multiple drawbacks including the need to pre-compute the input features and mesh connectivities. This becomes a bottleneck for many important tasks in protein science.

In this paper, we present a new framework for deep learning on protein structures that addresses these limitations. Among the key advantages of our method are the computation and sampling of the molecular surface on-the-fly from the underlying atomic point cloud and a novel efficient geometric convolutional layer. As a result, we are able to process large collections of proteins in an end-to-end fashion, taking as the sole input the raw 3D coordinates and chemical types of their atoms, eliminating the need for any hand-crafted pre-computed features.

To showcase the performance of our approach, we test it on two tasks in the field of protein structural bioinformatics: the identification of interaction sites and the prediction of protein-protein interactions. On both tasks, we achieve state-of-the-art performance with much faster run times and fewer parameters than previous models. These results will considerably ease the deployment of deep learning methods in protein science and open the door for end-to-end differentiable approaches in protein modeling tasks such as function prediction and design.

1. Introduction

Proteins are biomacromolecules central to all living organisms. Their function is a determining factor in health and disease, and being able to predict functional properties of proteins is of the utmost importance to developing novel drug therapies. From a chemical perspective, proteins are polymers composed of a sequence of amino acids (Fig. 1.a). This sequence determines the structural conformation (fold) of the protein, and the structure in turn determines its function. In a folded protein, hydrophobic (water-repelling) residues typically cluster within the core of the protein, while hydrophilic (water-attracting) residues are exposed to water solvent on its surface. The properties of this surface dictate the type and the strength of the interactions that a protein can have with other molecules (Fig. 1.b). Analysing this complex 3D object is therefore a fundamental problem in biology: models for protein structures can be used to understand the possible interactions between a protein and its environment, and consequently predict the functions of these macromolecules in living organisms.



(a) Folding and design. (b) Interacting surfaces.

Figure 1: Three major problems in structural biology. (a) Protein design is the inverse problem of structure prediction. (b) Two interacting proteins represented as an atomic point cloud (left) and as a molecular surface (right) that abstracts out the internal fold (shown semi-transparently). Protein surfaces display a number of geometric (e.g. concave and convex regions) and chemical (e.g. charges) features. Identifying their binding is a complex problem that can be addressed with geometric deep learning.

Since proteins are predominant drug targets, the study of their interactions with other molecules is a key problem for fundamental biology and the pharmaceutical industry. Classical drugs are small molecules designed to bind to a protein of interest, with a binding site that usually has noticeable 'pocket-like' structure. Targets with flat surfaces that exhibit no pockets have long been a challenge for drug developers and are often deemed 'undruggable'. The possibility of addressing such targets with specifically designed

protein molecules (known as biological drugs or 'biologics') is a fast emerging field in drug-development holding the promise to provide novel therapeutic strategies for many important diseases (e.g. cancer, viral infections).

Deep learning methods have increasingly been applied to a broad range of problems in protein science [22], with the particularly notable success of DeepMind's AlphaFold to predict 3D protein structure from sequence [38]. Recently, Gainza et al. [21] introduced MaSIF, one of the first conceptual approaches for *geometric deep learning* on protein molecular surfaces allowing to predict their binding. The main limitations of MaSIF stem from its reliance on precomputed meshes and handcrafted features, as well as significant computational time and memory requirements.

Main contributions. In this paper, we present dMaSIF (differentiable molecular surface interaction fingerprinting), a deep learning approach to identify interaction patterns on protein surfaces that addresses the key drawbacks of MaSIF. Our architecture is free of any precomputed features. It operates directly on the large set of atoms that compose the protein, generates a point cloud representation for the protein surface, learns task-specific geometric and chemical features on the surface point cloud and finally applies a new convolutional operator that approximates geodesic coordinates in the tangent space. All these computations are performed on the fly, with a small memory footprint. Notably, we implement all core calculations as reductions of "symbolic matrices", supported by the recent KeOps library [19] for PyTorch [31]. These high performance routines let us design a method which is fully differentiable and an order of magnitude faster and more memory efficient than MaSIF. This in turn allows us to make predictions on larger collections of protein structures than was previously practical, and opens the door to end-to-end protein optimization and *de novo* protein design using geometric deep learning.

2. Related works

Deep learning in protein science. Proteins can be represented in different ways, the 1D amino acid sequence being the simplest and most abundant source of data. Recent methods have taken advantage of the wealth of protein sequences available in public databases and shown how unsupervised embeddings borrowed from the field of Natural Language Processing can improve function prediction [2, 8, 37]. Deep learning is also becoming a key component in many pipelines for protein folding (i.e. inferring the 3D structure from the amino acid sequence) [3, 48, 38, 49]. These methods often predict pair-wise distances and other geometric relations between different residues to use them as constraints in later structural refinements. Relations between amino acids of different proteins have also been predicted to handle protein-protein interactions [42, 20]. Protein design can be considered as 'inverse structure prediction' (i.e. predict a sequence that will fold into a particular structure) and has also benefited from deep learning methods [24]. We refer to [22] for a comprehensive overview.

Surface representations are relevant to the field: they abstract the internal parts of the protein fold which do not contribute to interactions. The Molecular Surface Interaction Fingerprinting (MaSIF) [21] method pioneered the use of mesh-based geometric deep learning to predict protein interactions. It was used to classify binding sites for small ligands, discriminate sites of protein-protein interaction in surfaces and predict protein-protein complexes.

Nevertheless, in spite of its conceptual importance and impressive performance, the MaSIF method has significant drawbacks that limit its practical applications for protein prediction and design. First, it takes as inputs mesh-based representations of a protein surface, that must be generated from the raw atomic point cloud as a preprocessing step. Second, it relies on hand-crafted chemical and geometric features that must also be pre-computed and stored on the hard drive. Third, it uses MoNet [30] mesh convolutions on precomputed geodesic patches, which becomes prohibitively expensive in terms of memory and run time when working with more than a few thousand proteins.

Deep learning on surfaces and point clouds. Deep learning on non-Euclidean structured data such as meshes, graphs and point clouds, known under the umbrella term *geometric deep learning* [11], has recently become an important tool in computer vision and graphics. Instead of considering geometric shapes as objects in a 3D Euclidean space and applying standard deep learning pipelines (e.g. based on 2D views [46], volumetric [39], space partitioning [36, 44, 40] and implicit representations [14]), geometric deep learning seeks to develop a non-Euclidean analogy of filtering and pooling operations. Boscaini et al. [27] proposed the first geometric CNN-like architecture (Geodesic CNN) based on intrinsic local charting on meshes. Follow-up works improved on these results using patch operators based on anisotropic diffusion (ACNN [10]), Gaussian mixtures (MoNet [30]), splines [17], graph message passing (FeastNet [43]), equivariant filters [32, 15], and primal-dual mesh operators [29]. We refer to [33] for a recent survey.

Point clouds are often used as a native representation of 3D data coming from range sensors, and have recently gained popularity in computer vision in lieu of surface-based representations. First works on deep learning on point clouds were based on deep learning on sets [50] (PointNet [34] and PointNet++ [35]). DGCNN [45] uses graph neural networks [6] on kNN graphs constructed on the fly to capture the local structure of the point cloud. Additional tangent space [40] and volumetric [4] convolution operators were also considered, see a recent survey paper [23].

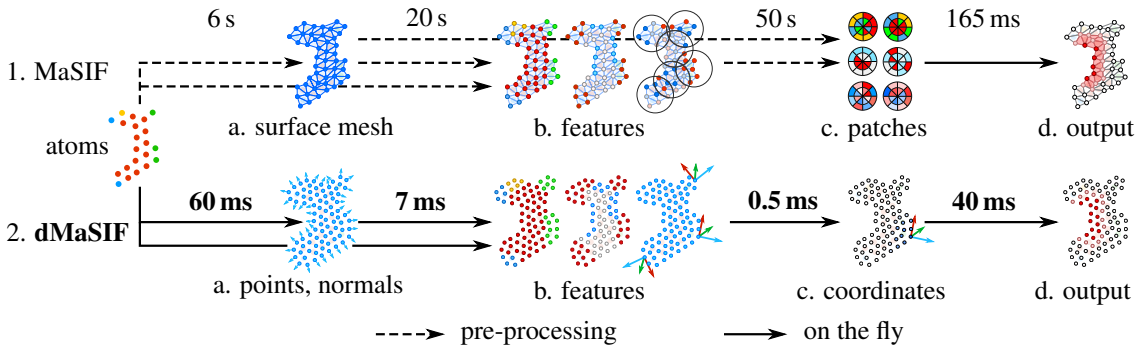
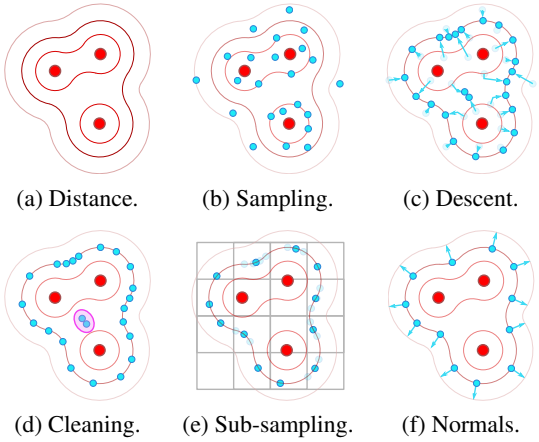


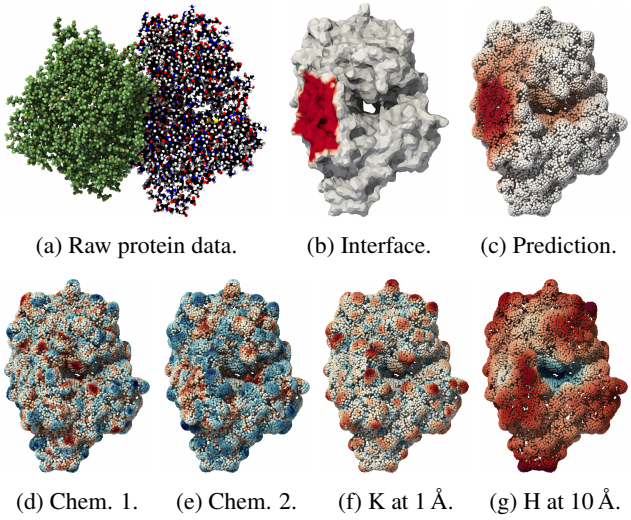
Figure 2: Both MaSIF and dMaSIF go through the same steps for interface prediction on protein surfaces. Starting from a raw atomic point cloud, we compute (a) a representation of the protein molecular surface, (b) geometric and chemical features, and (c) local coordinate systems; (d) a binding site is then predicted by a geometric convolutional neural network operating on (quasi-)geodesic patches on the protein surface. MaSIF precomputes steps (a)–(c), whereas we compute them on the fly 600 times faster. For every step, we display average run times per protein for inference on the site prediction task described in Section 4. Our method results in an accuracy level on par with MaSIF while alleviating the need for pre-calculations and providing significant speed-up for both inference and training.



(a) Distance. (b) Sampling. (c) Descent.

(d) Cleaning. (e) Sub-sampling. (f) Normals.

Figure 3: Sampling algorithm for protein surfaces. (a) Given the input protein (encoded as an atomic point cloud $\mathbf{a}_1, \dots, \mathbf{a}_A$, in red), its molecular surface is represented as a level set of the smooth distance function (1) to the atom centers. (b) To sample this surface, we first generate a point cloud $\mathbf{x}_1, \dots, \mathbf{x}_{N=AB}$ in the neighborhood of our protein (in blue): for every atom center, we draw $B = 20$ points from $\mathcal{N}(\mu = \mathbf{a}_k, \sigma = 10\text{Å})$ and (c) let this random sample converge towards the target level set by gradient descent on (2) – we use 4 gradient steps with a learning rate of 1. (d) We then remove points trapped inside the protein: we keep a sample if the distance function at this location is close to our target value of $r = 1.05$ Å within a margin of 0.10 Å, and if making four consecutive steps of size 1 Å in the direction of the gradient of the distance function increases it by more than 0.5 Å. (e) We then put all points in cubic bins of side length 1 Å and keep one average sample per cell; this ensures that our sampling has uniform density. (f) Finally, the gradient of the distance function at location $\mathbf{x}_i$ is normalized to be used as a normal $\hat{\mathbf{n}}_i$.



(a) Raw protein data. (b) Interface. (c) Prediction.

(d) Chem. 1. (e) Chem. 2. (f) K at 1 Å. (g) H at 10 Å.

Figure 4: Illustration on the binding of the 1OJ7 pair. (a) The Protein Data Bank documents interactions between proteins 1OJ7_D (right) and 1OJ7_A (left, green). Can we learn to predict this 3D binding configuration from the unregistered structures of both proteins? (b) MaSIF tackles this problem as a surface segmentation problem. The binding site (red) is the ground truth signal that MaSIF tries to predict from precomputed chemical and geometric features, such as the electrostatic potential. It relies on mesh convolutions on the preprocessed molecular surface of the protein. (c) Our method predicts the binding site without using any precomputed mesh structure or features. We perform all computations on an oriented point cloud, generated from the raw atom coordinates as in Figure 3. Data-driven chemical features (d-e) as well as Gaussian (f) and mean (g) curvatures at different scales are computed on the fly and given as inputs to a fast convolutional architecture that we describe in Figure 5. Rendering done with ParaView [5].

3. Our approach

Working with protein surfaces. In the following, we describe a new efficient end-to-end architecture for geometric deep learning on protein molecules. The premise of our work is that protein molecular surfaces carry important geometric and chemical information that is indicative of the way they interact with other molecules. Though we showcase our method on predicting binding properties (arguably, the most important task in structural biology and drug design), it is generic and can be trained on other problems – and in principle, be extended to other biomolecules.

Our method works on successive geometric representations of a protein, illustrated in Figure 2. The input is provided as a cloud of atoms $\{\mathbf{a}_1, \dots, \mathbf{a}_\mathrm{A}\} \subset \mathbb{R}^3$, with chemical types in the list [C, H, O, N, S, Se] encoded as one-hot vectors $\{\mathbf{t}_1, \dots, \mathbf{t}_\mathrm{A}\} \subset \mathbb{R}^6$. We then represent the surface of the protein as an oriented point cloud $\{\mathbf{x}_1, \dots, \mathbf{x}_\mathrm{N}\} \subset \mathbb{R}^3$ with unit normals $\hat{\mathbf{n}}_1, \dots, \hat{\mathbf{n}}_\mathrm{N}$ in $\mathbb{R}^3$. We associate feature vectors $\mathbf{f}_1, \dots, \mathbf{f}_\mathrm{N}$ to these points and progressively update them using convolution-like operations; the dimension of these features varies from 16 (10 geometric + 6 chemical features as input) to 1 (binding score as output) throughout our network. Our data comes from the Protein Data Bank [7], with protein structures that are typically made up of $\mathrm{A} = 3\text{K}–15\text{K}$ atoms and molecule sizes in the range 30 Å–300 Å (one ångström is equal to 10^{-10} m); we sample their surfaces at a resolution of 1 Å to work with $\mathrm{N} = 6\text{K}–15\text{K}$ points at a time.

We stress that unlike most other works for surface processing, our method *does not* rely on mesh structures, kNN graphs, or space partitioning of any kind. We compute exact interactions between all points of a protein surface efficiently using the recent KeOps library [13, 19] for PyTorch [31] that optimizes a wide range of computations on generalized distance matrices. [1]

3.1. Surface generation

Fast sampling. The surface of a protein can be described as the level set of a smooth distance function or *meta ball* [9] (Figure 3a). To represent the six different atom types accurately, we associate an atomic radius σ_k to each atom $\mathbf{a}_k$ and define the smooth distance function:

$$\mathrm{SDF}(\mathbf{x}) = -\sigma(\mathbf{x}) \cdot \log \textstyle\sum_{k=1}^{\mathrm{A}} \exp(-\|\mathbf{x} - \mathbf{a}_k\| \,/\, \sigma_k) \;, \quad (1)$$

for any $\mathbf{x} \in \mathbb{R}^3$, with a stable log-sum-exp reduction and with $\sigma(\mathbf{x}) = \sum_{k=1}^{\mathrm{A}} \exp(-\|\mathbf{x} - \mathbf{a}_k\|)\, \sigma_k \,/\, \sum_{k=1}^{\mathrm{A}} \exp(-\|\mathbf{x} - \mathbf{a}_k\|)$ the average atom radius in a neighborhood of point $\mathbf{x}$.

[1] The size 5K–20K and dimension 3 of our point clouds appear to be a sweetspot for KeOps in 'bruteforce mode', thanks to *contiguous* operations that stream much better on GPUs than the *scattered* memory accesses of graph-based and hierarchical methods.

As shown in Figure 3b, we sample the level set surface at radius $r = 1.05$ Å by minimizing the squared loss function:

$$E(\mathbf{x}_1, \dots, \mathbf{x}_\mathrm{N}) \;=\; \tfrac{1}{2} \textstyle\sum_{i=1}^{\mathrm{N}} (\,\mathrm{SDF}(\mathbf{x}_i) - r\,)^2 \quad (2)$$

on a random Gaussian sample. KeOps allows us to implement this sampling strategy efficiently on batches of more than 100 proteins at a time.

Descriptors. Point normals $\hat{\mathbf{n}}_i$ are computed using the gradient of the distance function (1). To estimate a local coordinate system $(\hat{\mathbf{n}}_i, \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$, we first smooth this vector field using a Gaussian kernel with $\sigma \in \{9, 12\}$ Å, i.e. use $\hat{\mathbf{n}}_i \leftarrow \mathrm{Normalize}(\sum_{j=1}^{\mathrm{N}} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2) \hat{\mathbf{n}}_j)$. We then compute tangent vectors $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{v}}_i$ using the efficient formulae of [16]. Let $\hat{\mathbf{n}}_i = [x, y, z]$ be a unit vector, $s = \mathrm{sign}(\mathrm{z})$, $a = -1/(s+z)$ and $b = a\,x\,y$, then

$$\hat{\mathbf{u}}_i = [\,1 + sax^2, sb, -sx\,], \quad \hat{\mathbf{v}}_i = [\,b, s + ay^2, -y\,]\,. \quad (3)$$

For each point $\mathbf{x}_i$, we then find the 16 nearest atom centers $\{\mathbf{a}_1^i, \dots, \mathbf{a}_{16}^i\}$ with types $\{\mathbf{t}_1^i, \dots, \mathbf{t}_{16}^i\}$ encoded as one-hot vectors in $\mathbb{R}^6$. We compute a vector of chemical features $\mathbf{f}_i$ in $\mathbb{R}^6$ by applying a Multi-Layer Perceptron (MLP) to the vectors $[\mathbf{t}_k^i, 1/\|\mathbf{x}_i - \mathbf{a}_k^i\|]$ in $\mathbb{R}^7$, performing a summation over the indices $k = 1, \dots, 16$ and applying a second MLP to the result. As illustrated in Figure 6, using simple MLPs with a single hidden layer of dimension 12 is enough to learn rich chemical features, such as the Poisson-Boltzmann electrostatic potential.

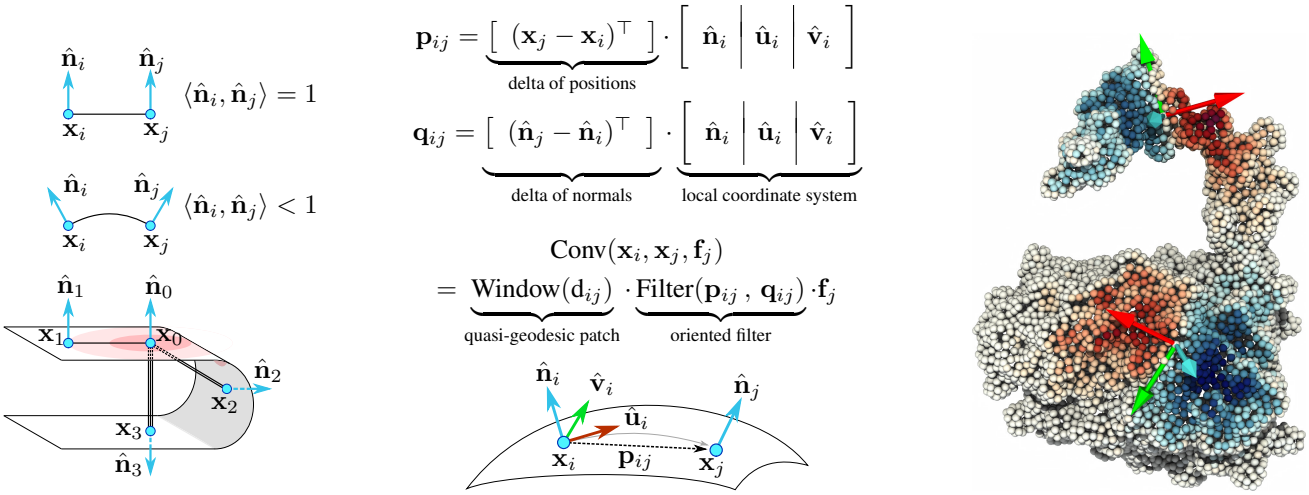
3.2. Quasi-geodesic convolutions on point clouds

Convolutions on 3D shapes. To update the feature vectors $\mathbf{f}_i$ and progressively learn to predict the binding site of a protein, we rely on (quasi-)geodesic convolutions on the molecular surface. This allows us to ensure that our model is fully invariant to 3D rotations and translations, takes decisions according to *local* chemical and geometric properties of the surface, and is not influenced by atoms located deep inside the volume of a protein. These modelling hypotheses hold for many protein interaction problems and prevent our network from overfitting on the few thousands of protein pairs that are present in our dataset.

In practice, geometric convolutional networks combine pointwise operations of the form $\mathbf{f}_i' \leftarrow \mathrm{MLP}(\mathbf{f}_i)$ with local inter-point interactions of the form:

$$\mathbf{f}_i' \;\leftarrow\; \sum_{j=1}^{\mathrm{N}} \mathrm{Conv}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{f}_j) \;, \quad (4)$$

where $\mathbf{f}_i$ and $\mathbf{f}_i'$ denote feature vectors associated to the point $\mathbf{x}_i$ and the $\mathrm{Conv}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{f}_j)$ operator puts a trainable weight on the relationship between the points $\mathbf{x}_i$ and $\mathbf{x}_j$. The sum can possibly be replaced by a maximum or any other reduction or *pooling* operation.



(a) Quasi-geodesic distance d_{ij}. (b) Quasi-geodesic convolution. (c) $\mathrm{Filter}(\mathbf{p}_{ij}) = \mathbf{p}_{ij}^{\hat{\mathbf{u}}}$.

Figure 5: We use an approximation of the geodesic distance (5) to implement fast quasi-geodesic convolutions on oriented point clouds. (a) The weighted distance d_{ij} between points $\mathbf{x}_i$ and $\mathbf{x}_j$ is equal to $\|\mathbf{x}_i - \mathbf{x}_j\|$ if the unit normal vectors $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_j$ point towards the same direction, but is larger otherwise. In this example, the points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ lay at equal distance of the reference point $\mathbf{x}_0$ in $\mathbb{R}^3$; but since the reference normal $\hat{\mathbf{n}}_0$ is aligned with $\hat{\mathbf{n}}_1$, orthogonal to $\hat{\mathbf{n}}_2$ and opposite to $\hat{\mathbf{n}}_3$, we have $\mathrm{d}_{0,1} = \|\mathbf{x}_0 - \mathbf{x}_1\| < 2 \cdot \mathrm{d}_{0,1} = \mathrm{d}_{0,2} < 3 \cdot \mathrm{d}_{0,1} = \mathrm{d}_{0,3}$. (b) We leverage this behaviour to prevent information leakage "across the *volume*" of a protein. We combine a Gaussian window on the weighted distance d_{ij} with a parametric "Filter" to aggregate features $\mathbf{f}_j$ between neighbors on a protein *surface*. (c) Our formulae induce local coordinate systems that closely mimic the structure of genuine geodesic patches – defined here by a Gaussian window of deviation $\sigma = 10$ Å. On smooth surfaces, they enable the computation of "quasi-geodesic" convolutions at a much lower cost than mesh-based methods.

Working with oriented point clouds. Numerous methods have been proposed to mimic surface operators with convolution operators on meshes or point clouds – see Section 2 and especially [40, 26, 47, 41]. In this work, we leverage the *normal vectors* that are produced by our sampling algorithm to define a fast quasi-geodesic convolutional layer that works directly on oriented point clouds. The KeOps library lets us implement this operation efficiently, *without any offline precomputation* on the surface geometry.

As illustrated in Figure 5, we approximate the geodesic distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ of a protein surface with unit normals $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_j$ as:

$$\mathrm{d}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \cdot (2 - \langle \hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j \rangle) \tag{5}$$

and localize our filters using a smooth Gaussian window of radius $\sigma \in \{9, 12\}$ Å, $w(\mathrm{d}_{ij}) = \exp(-\mathrm{d}_{ij}^2 \,/\, 2\sigma^2)$. In the neighborhood of any point $\mathbf{x}_i$ of the surface, two 3D vectors then encode the relative position and orientation of neighbor points $\mathbf{x}_j$ in the local coordinate system $(\hat{\mathbf{n}}_i, \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$:

$$\mathbf{p}_{ij} = [\, \mathbf{p}_{ij}^{\hat{\mathbf{n}}}, \mathbf{p}_{ij}^{\hat{\mathbf{u}}}, \mathbf{p}_{ij}^{\hat{\mathbf{v}}} \,], \quad \mathbf{q}_{ij} = [\, \mathbf{q}_{ij}^{\hat{\mathbf{n}}}, \mathbf{q}_{ij}^{\hat{\mathbf{u}}}, \mathbf{q}_{ij}^{\hat{\mathbf{v}}} \,].$$

Different choices for the trainable "Filter" on these 3D vectors let us encode a wide range of operations. We focus here on polynomial functions and MLPs instead of the popular Mixture-of-Gaussian filters [30], but note that this choice has little impact on the expressive power of our model.

Local orientation, curvatures. We must stress, however, that the pair of tangent vectors $(\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$ orthogonal to the normal $\hat{\mathbf{n}}_i$ is only defined up to a rotation in the tangent plane. To work around this problem at a low computational cost, we follow [28] and orient the first tangent vector $\hat{\mathbf{u}}_i = \hat{\mathbf{u}}(\mathbf{x}_i)$ along the geometric gradient $\nabla^{\hat{\mathbf{u}},\hat{\mathbf{v}}} P(\mathbf{x}_i)$ of a trainable potential $P(\mathbf{x}_i) = P_i = \mathrm{MLP}(\mathbf{f}_i)$, computed from the input features using a small MLP. We approximate its gradient using a derivative of Gaussian filter on the tangent plane, implemented as a quasi-geodesic convolution:

$$\nabla P(\mathbf{x}_i) \leftarrow \tfrac{1}{\mathrm{N}} \textstyle\sum_{j=1}^{\mathrm{N}} w(\mathrm{d}_{ij}) \, [\, \mathbf{p}_{ij}^{\hat{\mathbf{u}}}, \mathbf{p}_{ij}^{\hat{\mathbf{v}}} \,] \, P_j \in \mathbb{R}^2 \tag{6}$$

and then update the tangent basis $(\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$ using standard trigonometric formulae.

Local curvatures are computed in a similar fashion [12]. We use quasi-geodesic convolutions with Gaussian windows of radii σ that range from 1 Å to 10 Å and quadratic filter functions to estimate the local covariances $\mathrm{Cov}_{\sigma,i}^{\hat{\mathbf{u}},\hat{\mathbf{v}}}(\mathbf{p},\mathbf{p})$ and $\mathrm{Cov}_{\sigma,i}^{\hat{\mathbf{u}},\hat{\mathbf{v}}}(\mathbf{p},\mathbf{q})$ of the point positions and normals as 2×2 matrices in the tangent plane $(\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$. With $\lambda = 0.1$ Å a small regularization parameter, the 2×2 shape operator at point $\mathbf{x}_i$ and scale σ is then approximated as $\mathrm{S}_{\sigma,i} = (\lambda^2 \,\mathrm{Id}_{2\times 2} + \mathrm{Cov}_{\sigma,i}^{\hat{\mathbf{u}},\hat{\mathbf{v}}}(\mathbf{p},\mathbf{p}))^{-1} \mathrm{Cov}_{\sigma,i}^{\hat{\mathbf{u}},\hat{\mathbf{v}}}(\mathbf{p},\mathbf{q})$, which allows us to define the Gaussian $\mathrm{K}_{\sigma,i} = \det(\mathrm{S}_{\sigma,i})$ and mean $\mathrm{H}_{\sigma,i} = \mathrm{trace}(\mathrm{S}_{\sigma,i})$ curvatures at scale σ.

Trainable convolutions. Finally, the main building block of our architecture is a quasi-geodesic convolution that relies on a trainable MLP to weigh features in a geodesic neighborhood of the local reference point $\mathbf{x}_i$. We turn a vector signal $\mathbf{f}_i \in \mathbb{R}^{\mathrm{F}}$ into a vector signal $\mathbf{f}'_i \in \mathbb{R}^{\mathrm{F}}$ with:

$$\mathbf{f}'_i \leftarrow \sum_{j=1}^{\mathrm{N}} w(\mathrm{d}_{ij})\, \mathrm{MLP}(\mathbf{p}_{ij})\, \mathbf{f}_j \tag{7}$$

where MLP is a neural network with 3 input units, $\mathrm{H} = 8$ hidden units, ReLU non-linearity and $\mathrm{F} = 16$ outputs.

3.3. End-to-end convolutional architecture

Overview. We chain together the operations introduced in the previous sections to create a fully differentiable pipeline for deep learning on protein surfaces, illustrated in Figure 2. As a brief summary:

1. We sample surface points and normals as in Figure 3.
2. We use the normals $\hat{\mathbf{n}}_i$ to compute mean and Gaussian curvatures at 5 scales σ ranging from 1 Å to 10 Å.
3. We compute chemical features on the protein surface as described in Section 3.1. Atom types and inverse distances to surface points are passed through a small MLP with 6 hidden units, ReLU non-linearity and batch normalization [25]. Contributions from the 16 nearest atoms to a surface point $\mathbf{x}_i$ are summed together, followed by a linear transformation to create a vector of 6 scalar features.
4. We concatenate these chemical features to the 5 + 5 mean and Gaussian curvatures to create a full feature vector of size 16.
5. We apply a small MLP on this vector to predict orientation scores P_i for each surface point. We then orient the local coordinates $(\hat{\mathbf{n}}_i, \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)$ according to (6).
6. We apply successive trainable convolutions (7), MLPs and batch normalizations on the feature vectors $\mathbf{f}_i$. The numbers of layers, the radii of the Gaussian windows and the number of units for the MLPs are task-dependent and detailed in the Supplementary Material.
7. As a final step for site identification, we apply an MLP to the output of the convolutions to produce the final site/non-site binary output. For interaction prediction, we compute dot products between the feature vectors of both proteins to use them as interaction scores between pairs of points.

Asymmetry between binding partners. When trying to predict binding interactions for protein pairs, we process both interacting proteins identically up to the convolutional step. We then introduce some asymmetry by passing each one of the two binding partners through a separate convolutional network. This allows the network to find *complementary* (instead of similar) regions on both surfaces, such as convex bulges and concave pockets. We note that MaSIF encoded such an asymmetry by inverting the sign of the precomputed features on one of the two surfaces.

4. Experimental Evaluation

Benchmarks. We test our method on two tasks introduced in [21]. The tasks come from the field of structural bioinformatics and deal with predicting how proteins interact with each other.

Binding site identification: we try to classify the surface of a given protein into interaction sites and non-interaction sites. Interaction sites are surface patches that are more likely to mediate interactions with other proteins: understanding their properties is a key problem for drug design and the study of protein interaction networks. The identification of the interaction site is unaware of the binding partner.

Interaction prediction: we take as inputs two surface patches, one from each protein involved in a complex, and predict if these locations are likely to come into close contact in the protein complex. This task is key to prediction tasks like protein docking, i.e. predicting the orientation of two proteins in a complex.

Dataset. The dataset comprises protein complexes gathered from the Protein Data Bank (PDB) [7]. We use the training / testing split of [21], which is based on sequence and structural similarity and was assembled to minimize the similarity between structures of the interfaces in the training and testing set. For site identification, the training and test sets include 2958 and 356 proteins, respectively; 10% of the training set is reserved for validation. For interaction prediction, the training and test sets include 4614 and 912 protein complexes, respectively, with 10% of the training set used for validation.

The average number of points used to represent a protein surface is $\mathrm{N} = 11549 \pm 1853$ for our generated point clouds, compared to 6321 ± 1028 points for MaSIF.[2] Proteins are randomly rotated and centered to ensure that methods which rely on atomic point coordinates do not overfit on their spatial locations.

Baselines. Our main baselines are the MaSIF-site and MaSIF-search models [21]. For the MaSIF baselines, we use the pre-trained models and precomputed surface meshes and input features provided by the authors. Additionally, in order to show the benefits of our convolutional layer, we benchmark it against PointNet++ [35] and Dynamic Graph CNN (DGCNN) [45], two popular state-of-the-art convolutional layers for point clouds.

[2] This smaller sampling size of MaSIF stems from the large time and memory requirements of this method, which prohibits the use of finer meshes.

Implementation. We implement our architectures with PyTorch [31] and use KeOps [19] for fast geometric computations. For data processing and batching, we use PyTorch Geometric [18]. For the PointNet++ and DGCNN baselines, we use PyTorch Geometric implementations – but rely on KeOps symbolic matrices to accelerate the construction of kNN graphs and thus guarantee a fair comparison. For the MaSIF baselines, we use the reference implementation of [21].[3] All models are trained on either a single NVIDIA GeForce RTX 2080 Ti GPU or a single Tesla V100. Run times and memory consumption are measured on a single Tesla V100.

4.1. Surface and input feature generation

Precomputation. A key drawback of MaSIF is its reliance on the heavy precomputation of surface meshes and input features. These computations take a significant amount of time and generate large files that must be stored on disk. For reference, the pre-processed files used to train the MaSIF networks weigh more than 1TB. In sharp contrast, our method does not rely on any such precomputation. Table 1 compares corresponding run times for both pipelines: our method is three orders of magnitude faster than MaSIF for these geometric computations.

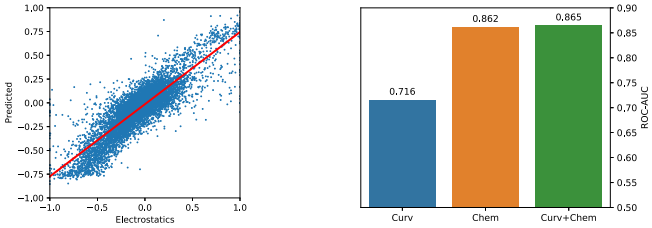
Scalability. Our surface generation algorithm scales beneficially with an increasing batch size. In SM we show that the running time and memory requirement per protein of our method both decrease significantly when processing dozens of proteins at time the batch size. This is a consequence of the increased usage of the GPU cores and the smaller influence of fixed PyTorch and KeOps overheads.

Moreover, our method of surface generation makes it easy to experiment with different point cloud resolutions. Different tasks could benefit from higher or lower resolution and tuning it as a hyperparameter could have significant effects on performance. We show the effects of resolution on time an memory requirements in SM.

Quality of learned chemical features. Another notable drawback of MaSIF is its reliance on 'handcrafted' geometric and chemical features (Poisson-Boltzmann electrostatic potential, hydrogen bond potential and hydropathy) that must be precomputed and provided as input to the neural network. In contrast, we do not use any handcrafted descriptors and learn problem-specific features directly from the underlying atomic point cloud, provided as the sole input of our method. We argue that this information alone is sufficient to compute an informative chemical and geometric description of the protein surface. To support this statement, we show in Figure 6 the results of an experiment where our chemical feature extractor is used to regress the Poisson-Boltzmann electrostatic potential on surface points. The quality of our predicition suggests that our data-driven chemical features are of similar quality to the descriptors used by MaSIF – or better.

We also note the results of an ablation study for chemical and geometric features, depicted in Figure 6. They suggest that the concatenation of geometric curvatures to the vector of learned chemical features does not significantly improve the performance of the network for the site prediction task: we will investigate this point in future works.

[3] Since MaSIF is implemented in TensorFlow [1], small discrepancies in measurements of memory consumption and running times are possible.



(a) Predicting electrostatics (b) Ablation study.

Figure 6: Our network can compute chemical properties of the protein surface from the underlying atomic point cloud. (a) Predicted Poisson-Boltzman electrostatic potential vs. the ground truth. Correlation cofactor r=0.83 and RMSE=0.16. (b) Ablation study showing how chemical and geometric features affect the performance in predicting interaction sites (ROC-AUC).

Computation	MaSIF	Ours
Surface generation	6.11±6.18 s	59.0±15.2 ms*
Input features	19.69±16.08 s	6.59±1.22 ms*
Local coordinates	50.65±45.15 s	0.46±0.09 ms*

Table 1: Average "pre-processing" time per protein. Our method is about 1000 times faster than MaSIF and allows these computations to be performed on the fly, as opposed to the offline precomputations of MaSIF. *With batches of 128 proteins at a time.

4.2. Performance

Binding site identification. Results for the identification of binding sites are summarized in Figures 7–9, which depict ROC curves and tradeoffs between accuracy, time and memory. We evaluate multiple versions of our architecture with varying numbers of convolution layers (1 vs 3) and patch sizes (5, 9, or 15Å). For comparison, we also show results when our convolutions are replaced by DGCNN and PointNet++ architectures, all other things being equal.

A first remark is that if we use a single convolution layer with a Gaussian window of deviation $\sigma = 15$ Å, our method matches the best accuracy of 0.85 ROC-AUC produced by

MaSIF – with 3 successive convolutional layers on patches of radius 9 Å. In this configuration, our network runs 10 times faster than MaSIF with an average time in the forward pass of 16 ms vs. 164 ms per protein. At the price of a modest increase of the model complexity (three convolution layers, and 36 ms on average per protein), we outperform MaSIF with a 0.87 ROC-AUC, detailed in Figure 7 (solid curves). Most remarkably, our models all have a small memory footprint (132 MB/protein), which is 11 times less than an equivalent MaSIF network (1492 MB/protein), 13 times less than DGCNN (1,681 MB/protein) and 30 times less than PointNet++ (3,995 MB/protein).

Interaction prediction. With a single convolutional layer architecture similar to that of MaSIF-search we reach a slightly higher performance of 0.82 vs. 0.81, as illustrated in Figure 7 (dashed). We remark that MaSIF-search reaches this level of accuracy using high dimensional feature vectors with 80 dimensions compared to our 16: understanding the influence of the number of convolutional "channels" on the performances of our network for different tasks will be an important direction for future works.

Note that MaSIF-search also relies on larger patches than MaSIF-site (12 Å vs. 9 Å), which causes a significant increase of run times to 727 ± 403 ms. On the other hand, our lightweight method runs in 17.5 ± 6.7 ms and is over 40 times faster at inference time.

5. Conclusion

We have introduced a new geometric architecture for deep learning on protein surfaces, enabling the prediction of their interaction properties. Our method is an order of magnitude faster and more memory efficient than previous approaches, making it suitable for the analysis of large-scale datasets of protein structures: this opens the door to the analysis of entire protein-protein interaction networks in living organisms, comprising over 10K proteins.

The fact that our pipeline works on raw atomic coordinates and is fully differentiable makes it amenable to *generative* tasks, with the possibility of performing a true end-to-end design of new proteins for diverse biological functions, namely in terms of the design of binders for specific targets. This opens fascinating perspectives in drug design, including biologics for targeting disease relevant targets (e.g. cancer therapy, antiviral) that display flat interaction surfaces and are impossible to target with small molecules.

More broadly, we believe that our new algorithmic and architectural ideas for deep learning on 3D shapes through fast on-the-fly computations on point clouds will be of general interest to computer vision and graphics experts. Conversely, we hope that our work will draw the attention of this community to some of the most important and promising problems in structural biology and protein science.

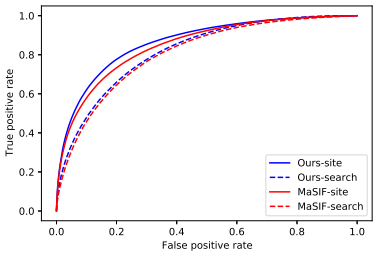


Figure 7: ROC curves comparing the performance of our method (blue) and MaSIF (red) on the task of binding site identification (solid curves) and search of binding partners (dashed). Our approach performs on par with MaSIF, achieving ROC-AUC of 0.87 (vs. 0.85) in site identification, and 0.82 (vs. 0.81) in identifying binding partners.

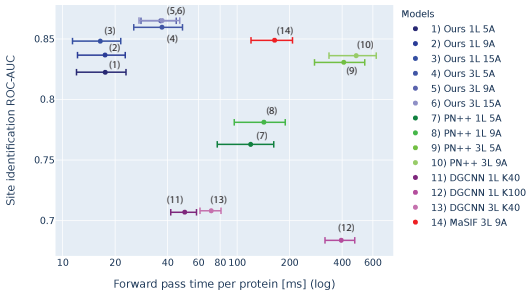


Figure 8: Accuracy (site identification ROC-AUC) vs. Run time (forward pass/protein in ms) of different architectures. Models are identified by the convolutional operator used, number of convolutional layers, and the value of σ used for the Gaussian window. PointNet++ models are identified by the radius of the neighborhood and DGCNN models by the number of nearest neighbours.

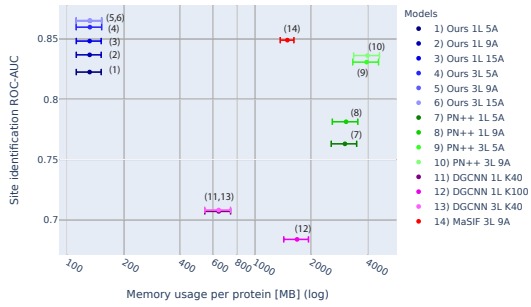


Figure 9: Accuracy (site identification ROC-AUC) vs. Memory footprint (MB/protein) of different architectures.

Acknowledgments. This work was supported in part by a Swiss Data Science Center fellowship, the Amazon Machine Learning Research Awards, ERC Consolidator grant No. 724228, ERC Starting Grant No. 716058, the Swiss National Science Foundation (310030_163139), and NCCR in Molecular Systems Engineering.

References

[1] Martín Abadi et al. Tensorflow: A system for large-scale machine learning. In *Proc. OSDI*, 2016.

[2] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, 16(12):1315–1322, 2019.

[3] Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell Systems*, 8(4):292–301, 2019.

[4] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv:1803.10091*, 2018.

[5] Utkarsh Ayachit. *The ParaView guide: a parallel visualization application*. Kitware, Inc., 2015.

[6] Peter W Battaglia et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

[7] Helen Berman, Kim Henrick, and Haruki Nakamura. Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*, 10(12):980–980, 2003.

[8] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-N protein engineering with data-efficient deep learning. *bioRxiv*, 2020.

[9] James F Blinn. A generalization of algebraic surface drawing. *ACM TOG*, 1(3):235–256, 1982.

[10] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Proc. NIPS*, 2016.

[11] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017.

[12] Yueqi Cao, Didong Li, Huafei Sun, Amir H Assadi, and Shiqiang Zhang. Efficient curvature estimation for oriented point clouds. *arXiv:1905.10725*, 2019.

[13] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *arXiv:2004.11127*, 2020.

[14] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. In *Proc. NeurIPS*, 2020.

[15] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh CNNs: Anisotropic convolutions on geometric graphs. *arXiv:2003.05425*, 2020.

[16] Tom Duff, James Burgess, Per Christensen, Christophe Hery, Andrew Kensler, Max Liani, and Ryusuke Villemin. Building an orthonormal basis, revisited. *JCGT*, 6(1), 2017.

[17] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proc. CVPR*, 2018.

[18] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *Proc. ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[19] Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. *Proc. NeurIPS*, 2020.

[20] Hiroyuki Fukuda and Kentaro Tomii. Deepeca: an end-to-end learning framework for protein contact prediction from a multiple sequence alignment. *BMC bioinformatics*, 21(1):1–15, 2020.

[21] 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.

[22] Wenhao Gao, Sai Pooja Mahajan, Jeremias Sulam, and Jeffrey J Gray. Deep learning in protein structural modeling and design. *arXiv:2007.08383*, 2020.

[23] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *Trans. PAMI*, 2020.

[24] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Proc. NeurIPS*, 2019.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[26] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *Proc. NeurIPS*, 2018.

[27] Jonathan Masci, Davide Boscaini, Michael M Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proc. ICCV Workshops*, 2015.

[28] Simone Melzi, Riccardo Spezialetti, Federico Tombari, Michael M Bronstein, Luigi Di Stefano, and Emanuele Rodolà. GFrames: Gradient-based local reference frame for 3D shape matching. In *Proc. CVPR*, 2019.

[29] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. In *Proc. NeurIPS*, 2020.

[30] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proc. CVPR*, 2017.

[31] 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. In *Proc. NeurIPS*, 2019.

[32] Adrien Poulenard and Maks Ovsjanikov. Multi-directional geodesic neural networks via equivariant convolution. *ACM TOG*, 37(6):1–14, 2018.

[33] Charles R Qi. *Deep learning on 3D data*. Springer, 2020.

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. CVPR*, 2017.

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. NIPS*, 2017.

[36] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3D representations at high resolutions. In *Proc. CVPR*, 2017.

[37] Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 2019.

[38] Andrew W Senior et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[39] Song, S., A Khosla, Xiao, and J. 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. CVPR*, 2015.

[40] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *Proc. CVPR*, 2018.

[41] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPconv: Flexible and deformable convolution for point clouds. In *Proc. CVPR*, 2019.

[42] Raphael JL Townshend, Rishi Bedi, Patricia A Suriana, and Ron O Dror. End-to-end learning on 3d protein structure for interface prediction. *arXiv preprint arXiv:1807.01297*, 2018.

[43] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3D shape analysis. In *Proc. CVPR*, 2018.

[44] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM TOG*, 36(4):1–11, 2017.

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019.

[46] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Proc. CVPR*, 2016.

[47] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *Proc. CVPR*, 2019.

[48] Jinbo Xu. Distance-based protein folding powered by deep learning. *PNAS*, 116(34):16856–16865, 2019.

[49] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *PNAS*, 117(3):1496–1503, 2020.

[50] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Proc. NIPS*, 2017.