Bioinformatics, 37(16), 2021, 2332–2339
doi: 10.1093/bioinformatics/btab118
Advance Access Publication Date: 23 February 2021
Original Paper

Structural bioinformatics

VoroCNN: deep convolutional neural network built on 3D Voronoi tessellation of protein structures

Ilia Igashov[1,2], Kliment Olechnovič[3], Maria Kadukova[1,2], Česlovas Venclovas[3,*] and Sergei Grudinin [1,*]

[1]Moscow Institute of Physics and Technology, 141701 Dolgoprudniy, Russia, [2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France and [3]Institute of Biotechnology, Life Sciences Center, Vilnius University, Vilnius, LT 10257, Lithuania

*To whom correspondence should be addressed.

Associate Editor: Lenore Cowen

Received on July 4, 2020; revised on January 8, 2021; accepted on February 22, 2021; editorial decision on February 18, 2021

Abstract

Motivation: Effective use of evolutionary information has recently led to tremendous progress in computational prediction of three-dimensional (3D) structures of proteins and their complexes. Despite the progress, the accuracy of predicted structures tends to vary considerably from case to case. Since the utility of computational models depends on their accuracy, reliable estimates of deviation between predicted and native structures are of utmost importance.
Results: For the first time, we present a deep convolutional neural network (CNN) constructed on a Voronoi tessellation of 3D molecular structures. Despite the irregular data domain, our data representation allows us to efficiently introduce both convolution and pooling operations and train the network in an end-to-end fashion without precomputed descriptors. The resultant model, VoroCNN, predicts local qualities of 3D protein folds. The prediction results are competitive to state of the art and superior to the previous 3D CNN architectures built for the same task. We also discuss practical applications of VoroCNN, for example, in recognition of protein binding interfaces.
Availability and implementation: The model, data and evaluation tests are available at https://team.inria.fr/nano-d/software/vorocnn/.
Contact: ceslovas.venclovas@bti.vu.lt or sergei.grudinin@inria.fr
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Protein structure prediction and protein structure analysis are very important problems in structural biology and bioinformatics. They have recently been subject to revolution thanks to multiple developments in several fields, most notably deep learning (Greener et al., 2019; Senior et al., 2020; Xu, 2019). Indeed, as the recent Critical Assessment of protein Structure Prediction (CASP) community-wide challenge has demonstrated, nowadays we are able to accurately predict protein structures even if they possess novel folds (Abriata et al., 2019; Hou et al., 2019; Kryshtafovych et al., 2019; Senior et al., 2019; Zheng et al., 2019).

In the protein structure prediction field, so far deep-learning techniques have been routinely applied to regular two-dimensional data, represented with matrices of multiple sequence alignments (Adhikari et al., 2018; Jones and Kandathil, 2018; Xu, 2019), or regular three-dimensional (3D) data of voxelized electron density maps (Derevyanko et al., 2018; Pagès et al., 2019; Pagès and Grudinin, 2019). Given the unprecedented success of the former approaches in the general structure prediction task, it was a bit surprising to see that the latter could not achieve the same accuracy as more classical methods in the last CASP13 protein model quality assessment (MQA) exercise (Cheng et al., 2019; Won et al., 2019). We believe that the data representation used in MQA methods that are based on 3D convolutional neural networks (CNN) is too complex for the currently available amount of data and computational resources. This work proposes a novel approach, called VoroCNN, that combines the advantages of the versatility of 3D CNNs with a simpler data representation based on *Voronoi tessellation* of 3D space (Poupon, 2004) and geometric deep learning (Bronstein et al., 2017).

Proteins fold into specific three-dimensional (3D) structures as a result of interatomic interactions. Protein atoms interact among themselves and with the solvent, and these interactions rapidly decay with the distance. A rigorous way to define interatomic interactions is to construct a *Voronoi tessellation* of the protein atoms and relate every *Voronoi cell* to an atom and every *Voronoi cell face* to an interatomic contact (Cazals et al., 2006; Richards, 1974, 1977). However, if a pair of contacting atoms is located near the surface of a protein structure, the corresponding Voronoi face may extend far away from the atoms. This problem can be circumvented by constraining the Voronoi cells of the atoms inside the boundaries

defined by the solvent-accessible surface, enabling calculation of the areas for every atom-atom and atom-solvent contact. Such a solution has been implemented in Voronota (Olechnovič and Venclovas, 2014), a software package specifically optimized to construct rapid tessellations for molecular structures when the radii of balls (atoms) are not very different from each other. Each Voronoi tessellation-derived contact area describes the magnitude of the corresponding interaction. The relatively larger contact area indicates that the interaction is less overshadowed by adjacent interactions and vice versa. This trait, specific to the tessellation-based analysis of protein structures, naturally introduces non-pairwise-additive molecular interactions. Interatomic contact areas as proxies for multibody interactions proved to be effective in various tasks, such as measuring the deviation of models from the reference structure (Olechnovič et al., 2013; Olechnovič and Venclovas, 2020) or the estimation of model accuracy in the absence of native structure (Olechnovič and Venclovas, 2017, 2019; Pontius et al., 1996; Zimmer et al., 1998).

Generally, it is rather difficult to operate on non-regular data structures in 3D space (Bronstein *et al.*, 2017; Griffiths and Boehm, 2019). Therefore, to construct an efficiently *trainable* end-to-end neural network, we decided to convert the initial 3D tessellations into *protein interaction graphs*. This allowed us to reuse all the knowledge already available for graph convolutional neural networks (Gilmer et al., 2017; Kipf and Welling, 2017; Li et al., 2018; Scarselli et al., 2008; Wu et al., 2021). We believe that a 3D tessellation can be reduced to a graph without losing too much information. Indeed, relative coordinates in 3D space can be reconstructed from a set of pairwise distance observations if we have a sufficient number of these, which are encoded as graph edges. This has been routinely demonstrated by various NMR-based techniques (Wüthrich, 1990), and also recently by solving protein structures from residue contact maps (Senior *et al.*, 2020; Xu and Wang, 2019). Derivation of a protein interaction graph from Voronoi tessellation-based atomic contacts is then straightforward. Naturally, each protein atom corresponds to a *graph node*, each atom-atom contact—to a *graph edge*, and each contact area—to the weight of the edge. Every node can also contain additional tessellation-derived features. These can be the corresponding atom solvent-accessible surface area, the volume of the constrained Voronoi cell, etc. Also, such representation is intrinsically equivariant and inherently overcomes many orientation-dependence problems characteristic to some other methods.

2 Materials and methods

The workflow of the VoroCNN method consists of the following steps. Firstly, given an atomistic 3D-model of a protein, we create the corresponding graph using the Voronoi 3D-tessellation method Voronota (Olechnovič and Venclovas, 2014). Then, we convert the output of Voronota to a graph. Finally, we pass this graph as an input to a graph CNN that predicts local folding qualities of the input protein model. As we are building our graph using 3D geometric information about atom contacts, it was rather natural for us to choose the ground-truth local quality measure for the graph CNN that also assesses atomic contacts. Also, it has been recently shown that interaction-based, superposition-free measures, such as local Distance Difference Test (lDDT) (Mariani et al., 2013), and contact area difference (CAD)-score in particular (Olechnovič *et al.*, 2013), have advantages over superposition-based measures in multiple respects. They are more consistent in selecting good models, more robust in dealing with flexibility of multi-domain structures as well as movements of loops or local structural motifs (Olechnovič et al., 2019), and also more suitable for training protein structural model quality prediction methods (Uziela et al., 2018). Therefore, we have naturally chosen *local* CAD-scores of each residue in the protein as the ground truth for the graph CNN. We should specifically mention that we primarily train the network to predict node-based scores, which are called local quality measures in the protein structure prediction community.Figure 1 shows a schematic representation of our method.

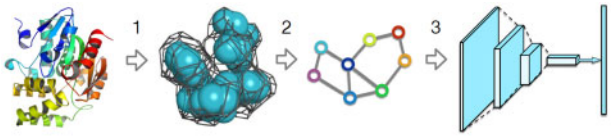


Fig. 1. Schematic representation of the VoroCNN quality assessment method. Firstly, a Voronoi tessellation of a 3D-model is computed with Voronota. Then, based on Voronoi 3D-tessellation, a graph is built. Finally, a graph neural network predicts local CAD-scores of all residues in the initial model

2.1 Graph representation

We represent the initial 3D-model of a protein as a *weighted unordered multigraph* with two types of edges, which are described in more detail below. The key property of the graph is that it implicitly keeps the information about the spatial relationship between atoms based on the Voronoi 3D-tessellation of the protein model.

Nodes in the graph correspond to atoms in the protein structure. Each node of the graph contains a vector of features that describe the corresponding atom. We associate each node with an atom type encoded with the one-hot representation—a binary vector with all zeros and a single '1' value at the position indicating the type of the atom. We use 167 types in total following Pagès *et al.* (2019). We represent the whole set of nodes as a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, where N is the number of atoms in the protein structure, and $d = 167$ is the dimension of the feature vector.

As mentioned above, our graph has two types of edges. Edges of the first type, the *contact edges*, correspond to the spatial relationship between atoms. To construct these edges, we built a Voronoi partitioning on a set of balls whose positions and sizes are defined by the locations of the protein atoms and their van der Waals radii, correspondingly. We say that two atoms are *in contact* if their Voronoi cells have a non-zero contact surface. We consider that two atoms have a contact edge if these atoms are in contact and there is no covalent bond between them. Edges of the second type, the *covalent edges*, correspond to the covalent bonds between the atoms that are in contact. We should note that in some bad-quality models that contain atomic clashes, two atoms with a covalent bond may not be in contact. In these cases, we do not assign any edge to these atoms. We represent the two sets of edges as two *adjacency matrices*. For the contact edges, we introduce a matrix $\mathbf{A}^{\mathrm{c}} \in \mathbb{R}^{N \times N}$, where *weights* a^{c}_{ij} are equal to the area of the contact surface between Voronoi cells of the ith and the jth atoms if there is a contact edge between them, and zero otherwise. For the covalent edges, we introduce a matrix $\mathbf{A}^{\mathrm{b}} \in \mathbb{R}^{N \times N}$, where *weights* a^{b}_{ij} are equal to the area of the contact surface between Voronoi cells of the ith and the jth atoms if there is a covalent edge between them, and zero otherwise.

In order to improve the numerical stability of the stochastic optimization and to add a certain level of regularization (Kipf and Welling, 2017), we normalize the adjacency matrices according to the following equation,

$$\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{1}$$

where $\mathbf{A}$ is an adjacency matrix and $\mathbf{D}$ is a diagonal matrix with nodes' degrees at the diagonal. We only consider edges of the same type when computing degrees of the nodes.

Finally, after the normalization, we split the covalent edges into three subtypes according to the types of covalent bonds, which can be single, double or aromatic. We also split the contact edges into five subtypes according to atoms' sequence-separation distances. For example, the first type of edges is composed of atoms belonging to the same residue, the second type of edges is composed of atoms that are in two consecutive residues, etc., the last type collects atoms with the sequence-separation ≥ 4. As a result, we obtain two adjacency tensors, $\hat{\mathbf{A}}^{\mathrm{b}} \in \mathbb{R}^{N \times N \times d_{\mathrm{b}}}$ for the covalent edges with $d_{\mathrm{b}} = 3$, and $\hat{\mathbf{A}}^{\mathrm{c}} \in \mathbb{R}^{N \times N \times d_{\mathrm{c}}}$ for the contact edges with $d_{\mathrm{c}} = 5$.

2.2 Graph convolutional neural network

Here we introduce a graph neural network that solves the following problem. Given a graph of a protein model in atom-level representation, the aim is to predict the local CAD-scores (Olechnovič *et al.*, 2013) of all the residues in the model. The key components of our network are the convolutional and the pooling layers, which are described below.

2.2.1 Convolutional layer

In the past years, convolutional neural networks became very popular as an efficient method for various image processing tasks (Alom et al., 2018; Fan et al., 2016; Wang et al., 2016; Zhang et al., 2017). The core idea of the convolutional layer is that it can learn local patterns that appear in different image segments. Very recently, similar approaches started to be adapted for graph structures (Bronstein *et al.*, 2017; Hamilton et al., 2017; Wu *et al.*, 2021). Contrary to images, graphs represent an irregular data domain, which makes the definition of convolution operation on graphs more complicated. One common way to define the convolution operation on a graph is based on the idea that one should combine information about a graph node with information about its neighbors. This basic observation has been further developed into various realizations of convolutional layers for graphs (Gilmer *et al.*, 2017; Kipf and Welling, 2017; Li *et al.*, 2018; Scarselli *et al.*, 2008). Today, graph convolutional networks are becoming a popular alternative to more classical approaches in structural bioinformatics too (Baldassarre et al., 2020; Cao and Shen, 2020; Fout et al., 2017; Sanyal et al., 2020; Zamora-Resendiz and Crivelli, 2019). This section introduces a graph convolutional layer that inherits from the same principles of sharing information between graph nodes and also takes into account the specificity of our data.

Our convolutional layer contains trainable tensors $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, $\mathbf{W}^{\text{b}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}} \times d_{\text{b}}}$ and $\mathbf{W}^{\text{c}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}} \times d_{\text{c}}}$, where d_{in} is the number of node features before passing them to the layer and d_{out} is the number of node features after the layer has been applied. Each layer transforms the feature matrix $\mathbf{Z} \in \mathbb{R}^{N \times d_{\text{in}}}$ into $\mathbf{Z}' \in \mathbb{R}^{N \times d_{\text{out}}}$ according to the following equation,

$$\mathbf{Z}' = \sigma[\mathbf{Z}\mathbf{W} + \sigma_\Sigma(\hat{\mathbf{A}}^{\text{b}} \diamond \mathbf{Z} \diamond \mathbf{W}^{\text{b}}) + \sigma_\Sigma(\hat{\mathbf{A}}^{\text{c}} \diamond \mathbf{Z} \diamond \mathbf{W}^{\text{c}}) + \mathbf{b}], \quad (2)$$

where $\mathbf{b} \in \mathbb{R}^{d_{\text{out}}}$ is a trainable bias vector, the result $\mathbf{X} \diamond \mathbf{Y}$ of the $\diamond$ operator is defined as

$$[\mathbf{X} \diamond \mathbf{Y}]_{ijk} = \begin{cases} \sum_l \mathbf{X}_{ilk} \mathbf{Y}_{lj}, & \text{if } \mathbf{Y} \text{ is an order-2 tensor (matrix)} \\ \sum_l \mathbf{X}_{ilk} \mathbf{Y}_{ljk}, & \text{if } \mathbf{Y} \text{ is an order-3 tensor,} \end{cases} \quad (3)$$

the function σ_Σ is defined as

$$[\sigma_\Sigma(\mathbf{X})]_{ij} = \sum_k \sigma(\mathbf{X}_{ijk}), \quad (4)$$

and σ is a non-linear activation function. In the present work we use exponential linear unit (ELU) as the activation function (Clevert et al., 2015).

2.2.2 Pooling layer

Downsampling operations are often used in classical CNN architectures to reduce the data representation, achieve a better translational invariance and extract hierarchical features. For images represented with pixels, downsampling can be implemented using convolutional filters with a stride that reduces the size of the output with respect to the input or also using additional pooling layers that return one pixel according to an operation applied to several input pixels. However, downsampling in a graph is an open research problem, because it is unclear how to define this operation on a non-regular domain in the general case. Nonetheless, there are several approaches based on clustering algorithms (Bach and Jordan, 2004; Dhillon et al., 2004, 2007). In this work, we introduce a pooling operation that uses prior information about the topology and the structure of the input graph.

Indeed, our graphs are very specific in the sense that we have a strict hierarchy of the representation. More precisely, atoms in the input data are grouped into residues. This allows us to introduce a pooling layer that downsamples the graph to the residue-level representation by averaging atoms' feature vectors within each residue. After applying this layer, covalent edges become primitive, i.e. they simply represent the peptide chain of the protein. Therefore, after the pooling layer, we keep working only with the contact edges. We should also specify that in this case the adjacency matrix is rewritten with the contact areas between the residues, which are computed as sums of the relevant inter-atom contact areas.

2.2.3 Network architecture

We have tested and assessed multiple graph network architectures that are described in more detail below. As is shown in Figure 2, the graph neural network is composed of three consecutive blocks: *Encoder*, *Body* and *Regressor*. *Encoder* consists of two linear layers with the activation function ELU. The purpose of *Encoder* is to reduce the size of feature vectors and embed them in a 10-dimensional vector space. *Body* contains convolutional layers described in Section 2.2.1 and the pooling layer described in Section 2.2.2. First of all, the sequence of four convolutional layers is applied to a graph in an atom-level representation. Then, the pooling layer downsamples the graph to the residue level. Finally, five consecutive convolutional layers process the coarsened graph. *Regressor* is intended for predicting local CAD-scores. It consists of a linear layer and a sigmoid function in the end. To train the network on local CAD-scores, we use Mean Squared Error (MSE) as a loss function,

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^{M} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2, \quad (5)$$

where $\mathbf{y}$ is a vector of ground-truth local CAD-scores, $\hat{\mathbf{y}}$ is a vector of VoroCNN predictions and M is the number of residues in a protein.

2.2.4 Training parameters and technical details

To train the model, we used Adam optimizer with a learning rate of 0.001 (Kingma and Ba, 2014). We stored and processed the adjacency matrices in sparse format, and the whole training process was conducted in seven parallel CPU threads. Thanks to the sparse representation of data, the efficiency of the CPU training turned out to be slightly higher compared to the GPU training. We shuffled the whole training dataset before training. Each thread in one iteration processed 2048 random models. To prevent overfitting, we trained the network with batches (64 models in a batch) and used L_2-regularization with a coefficient of 0.001. Adding dropout layers did not help. The final VoroCNN model contains 22 425 trainable parameters. One training iteration took on average 20 min on 15 Intel® Xeon® processors E5-2650 v2 @ 2.60 GHz, and we trained the network for 80 iterations. The code was written in Python using the PyTorch library (Paszke et al., 2019). All trained models, code and preprocessing binaries are available at https://team.inria.fr/nano-d/software/vorocnn/.

2.2.5 Datasets

To train and test our networks, we used submissions from the previous CASP challenges (Kryshtafovych *et al.*, 2019; Moult et al., 1995). For training and validation, we used models from CASP[8-11] server submissions (For CASP10 and CASP11, we took stage-2 server submissions.). We randomly split targets from CASP[8-11] into two parts: 80% for training and 20% for validation. We must mention that some of the CASP targets can form obligatory protein complexes, others belong to membrane proteins, and there are also targets with only low-quality models. To keep the physics of interactions, we tried to prune the training set as much as possible. Supplementary Information explains in detail the procedure of excluding suspicious models and targets. Overall, our *training dataset* consisted of 254 target structures and 44 202 models from

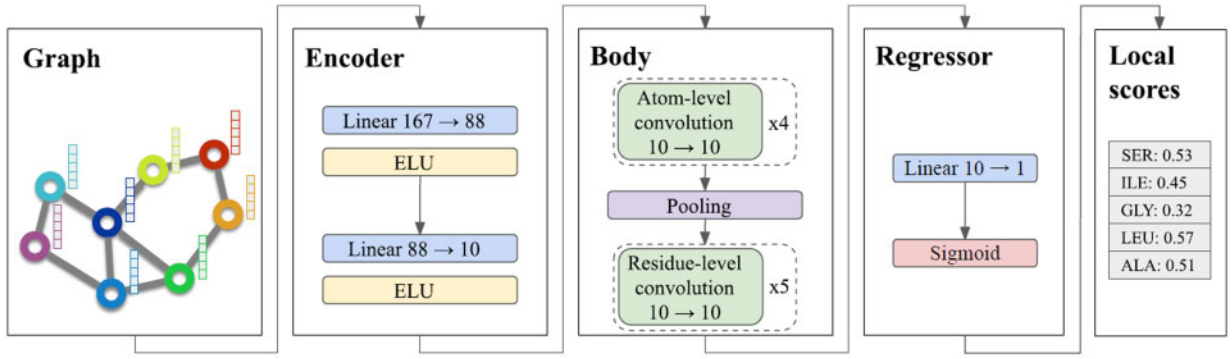


Fig. 2. Architecture of the VoroCNN network. The graph neural network consists of three blocks: Encoder, Body and Regressor. Encoder takes a graph as input and reduces node feature vectors with linear layers and ELUs. Body contains two sequences of convolutional layers and the pooling layer between them. Regressor consists of a linear layer followed by a sigmoid function and predicts local CAD-scores. A rounded colored rectangle represents a layer, dashed rounded rectangle means that it contains several identical layers, dimensions of input and output of each trainable layer are represented with the notation 'x → y'. The last column illustrates the output of the network—local CAD-score predictions

CASP[8-11]; the *validation dataset* consisted of 71 target structures and 12 666 models from CASP[8-11]. To evaluate the performance of our network, we have also constructed two *test datasets*. The first one included stage-2 server submissions of the CASP12 experiment (35 targets and 5 038 models), the second one included stage-2 server submissions from CASP13 (56 targets and 8 327 models). From all the sets, we excluded models with unrecognized atom names and heteroatoms. We included in test sets only those models and targets that were processed by each of the methods listed below in Section 3.1. Supplementary Tables S1–S3 from Supplementary Information list the targets included in the training, validation and test sets.

3 Results and discussion

3.1 Global scoring test sets and metrics

We evaluated the performance of VoroCNN on CASP12 and CASP13 stage-2 datasets. None of them were used for training or validation. Our main goal was to assess the ability of VoroCNN to select the best model from a pool submitted for a certain target structure. This can be fulfilled in several ways, and in this work, we report Pearson's and Spearman's correlations of model scores (Myers et al., 2010), global MSE and z-score. Per-target correlations were averaged over all targets using the Fisher transformation (Fisher, 1915). Here we provide results computed on global CAD-score (Olechnovič *et al.*, 2013) and global lDDT (Mariani *et al.*, 2013). Results computed on the global distance test (GDT) (Zemla et al., 2001, 1999) are reported in Supplementary Information.

For comparison with the state of the art, we have chosen several recent *single-model* methods. We want to specifically draw the reader's attention to the fact that we only compare our results with the best single-model methods. The second class of methods, the *consensus-based* approaches (Won *et al.*, 2019), selects the best models based on the analysis of the whole pool of structures. They often perform better than the single-model methods, but cannot assign a score to a single model without having access to the rest of the pool. Thus, we eliminated this class of methods from the comparison.

For comparison we have chosen the Voronoi diagram-based method VoroMQA (Olechnovič and Venclovas, 2017), the machine learning approach that uses orientational statistics of protein's backbones SBROD (Karasikov et al., 2019), descriptor-based methods ProQ3 (Uziela et al., 2016) and ProQ3D (Uziela et al., 2017), which also have access to sequence information, a 3D-CNN approach Ornate (Pagès *et al.*, 2019), a support-vector-machine-based SVMQA (Manavalan and Lee, 2017), an ensemble-learning MESHI-server (Elofsson et al., 2018) and a deep-learning-based MUfold2 (Wang et al., 2017). Since VoroCNN is trained to predict local per-residue scores, to obtain the global score of a model and compare it with the other methods, we averaged the local predictions. Supplementary Section S3 of Supplementary Information demonstrates that such averaging is valid, as the mean local CAD-scores correlate with the global CAD-scores with a coefficient of 0.995 – 0.999.

3.2 Global scoring results

Table 1 lists the results for CASP12. Table 2 compares the results for CASP13. For both benchmarks, VoroCNN outperforms or on par with other methods in CAD global and per-target correlations. Regarding CAD MSE, VoroCNN is in the top-2 on CASP12 and in the top-3 on CASP13. For the lDDT metrics, VoroCNN outperforms other methods in MSE, and is in top-2 for global correlations.

3.3 Global scores of predicted and native structures

An interesting question is how well can VoroCNN distinguish target (native) structures from the models. Let us consider VoroCNN as a binary classifier that predicts one of the two classes that a given model belongs to. In order to evaluate the quality of VoroCNN binary classification, we used global scores predicted by VoroCNN for all models and targets from our CASP12 and CASP13 test sets. Figure 3 (left) shows distributions of the global score predictions for CASP12. Figure 3 (right) shows the same distributions for CASP13. In both cases, we can see a clear separation between the two distributions.

3.4 Local scoring test set and metrics

For assessing predictions of per-residue quality scores, we downloaded the lDDT values from the CASP13 data archive. We computed several metrics: Pearson correlation coefficient; Spearman correlation coefficient; the best possible MCC (Matthews correlation coefficient) of binary classification, where the best value was found by varying the classification threshold. The ground truth for the binary classification was defined using the threshold of lDDT = 0.6, the arbitrary value used in CAMEO (Haas et al., 2018).

We compared our local quality prediction results with the results of the top four single-model methods that performed well in local scoring according to the official CASP13 assessment (Won *et al.*, 2019): ModFOLD7 (Maghrabi and McGuffin, 2020), ProQ3 (Uziela *et al.*, 2016), ProQ4 (Hurtado et al., 2018) and VoroMQA (Olechnovič and Venclovas, 2017). The local QA scores of these methods were downloaded from the CASP13 archive as distance deviations, therefore we could not consider Pearson correlation coefficients for them. Thus, we based our comparative analysis on the Spearman and Matthews correlation coefficients.

In addition, we performed a similar analysis using per-residue CAD-scores that we computed for the CASP13 server models with publicly available target structures. We chose an arbitrary CAD-score threshold for the binary classification to be 0.55 based on how

Table 1. CASP12 global CAD and lDDT predictions evaluated by z-score, MSE, Pearson's correlation r (global and per-target) and Spearman's correlation ρ (global and per-target)

	CAD						lDDT					
Method	$r_{per-target}$	r	$\rho_{per-target}$	ρ	MSE	z-score	$r_{per-target}$	r	$\rho_{per-target}$	ρ	MSE	z-score
VoroCNN	**0.828**	**0.871**	**0.784**	**0.866**	0.009	1.666	0.723	**0.838**	0.705	0.828	**0.011**	1.415
3DCNN (Ornate)	**0.828**	0.815	0.778	0.809	**0.007**	**1.745**	0.730	0.775	0.689	0.772	0.021	1.422
SVMQA	0.819	0.778	0.783	0.761	0.021	1.743	0.759	0.822	**0.737**	0.801	0.022	**1.480**
MUfold2	0.814	0.766	0.765	0.769	0.021	1.468	0.767	0.768	0.736	0.771	0.016	1.335
VoroMQA	0.794	0.668	0.759	0.691	0.053	1.359	0.754	0.692	0.725	0.711	0.031	1.148
ProQ3	0.792	0.796	0.744	0.806	0.036	1.625	**0.767**	0.837	0.732	**0.845**	0.014	1.424
SBROD	0.753	0.567	0.673	0.543	0.940	1.246	0.717	0.615	0.663	0.598	1.136	1.199
MESHI-server	0.732	0.777	0.681	0.783	0.029	1.519	0.717	0.818	0.690	0.828	0.026	1.282

Note: Results are sorted by per-target Pearson's r on CAD. The numbers for SBROD were computed by us for a previous publication (Karasikov *et al.*, 2019). We also locally calculated the corresponding numbers for Ornate. For all other methods, we used results from the server submissions archive downloaded from the official CASP website. The ground-truth CAD-scores and lDDT-scores were taken from the CASP website. Best column values are highlighhted in bold.

Table 2. CASP13 global CAD and lDDT predictions evaluated by z-score, MSE, Pearson's correlation r (global and per-target) and Spearman's correlation ρ (global and per-target)

	CAD						lDDT					
Method	$r_{per-target}$	r	$\rho_{per-target}$	ρ	MSE	z-score	$r_{per-target}$	r	$\rho_{per-target}$	ρ	MSE	z-score
VoroCNN	**0.827**	**0.803**	**0.797**	**0.806**	0.016	1.226	0.753	**0.802**	0.737	**0.801**	**0.013**	0.930
SBROD-server	0.804	0.386	0.763	0.401	0.054	**1.462**	0.771	0.460	**0.752**	0.470	0.041	**1.255**
VoroMQA-B	0.801	0.632	0.766	0.660	0.040	1.286	**0.775**	0.675	0.733	0.691	0.027	1.104
MUFold-server	0.792	0.718	0.752	0.718	0.015	1.350	0.759	0.747	0.739	0.752	0.015	1.169
VoroMQA-A	0.790	0.637	0.757	0.666	0.040	1.147	0.770	0.679	0.743	0.700	0.027	1.062
3DCNN (Ornate)	0.786	0.581	0.757	0.667	**0.010**	1.105	0.734	0.583	0.700	0.663	0.023	0.740
ProQ3D	0.785	0.733	0.740	0.731	0.032	1.396	0.747	0.768	0.734	0.772	0.017	1.222
ProQ3	0.775	0.704	0.741	0.705	0.036	1.411	0.733	0.738	0.723	0.743	0.021	1.168
MESHI-server	0.771	0.633	0.723	0.647	0.058	1.269	0.732	0.674	0.712	0.682	0.035	1.111

Note: Results are sorted by per-target Pearson's r on CAD. For all the methods, we used results from the server submissions archive downloaded from the official CASP website. The ground-truth CAD-scores and lDDT-scores were taken from the CASP website. Best column values are highlighhted in bold.

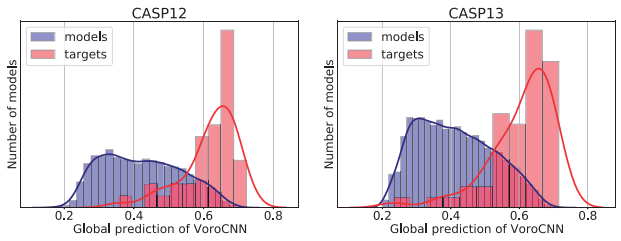


Fig. 3. Distribution of VoroCNN global scores on target structures and models from CASP12 (left) and CASP13 (right). Solid lines represent kernel density estimations of the corresponding distributions

CAD-score correlates with lDDT. Please see Supplementary Section S4 of Supplementary Information for more detail.

3.5 Local scoring results

Table 3 lists the quantitative results of the local scoring performance. In the case of lDDT, we use models of 73 CASP13 targets, with 2 604 778 residues in total. In the case of CAD-score, we were limited to the models of the 50 CASP13 targets with publicly available target structures. Thus, the considered models contain only 1 834 156 residues. The quantitative analysis results indicate that the local VoroCNN predictions are better than those of the state-of-the-art methods. We also computed Pearson correlation values for the VoroCNN per-residue scores: $r = 0.731$ when compared with lDDT, $r = 0.782$ when compared with CAD-score. VoroCNN achieves the best MCC scores using the thresholds 0.510 (for lDDT) and 0.507 (for CAD-score).

Table 3. Correlations of predicted per-residue QA scores with the reference lDDT and CAD-score values, calculated using the CASP13 data

	lDDT		CAD-score	
Method	$\lvert\rho\rvert$	MCC_{best}	$\lvert\rho\rvert$	MCC_{best}
VoroCNN	**0.724**	**0.577**	**0.778**	**0.616**
ModFOLD7	0.713	0.569	0.720	0.600
ProQ3	0.703	0.529	0.699	0.553
ProQ4	0.678	0.500	0.646	0.505
VoroMQA-A	0.614	0.434	0.563	0.411

Note: Reported values are the Spearman's ρ absolute values $\lvert\rho\rvert$ and the best Matthews correlation coefficients MCC_{best}. The table is sorted by the first numeric column. Best column values are highlighhted in bold.

Predicted per-residue folding qualities can be visually illustrated in all major molecular visualization systems. Figure 4A–B provides a visual comparison of VoroCNN predictions for the crystallographic structures, their CASP13 models, and also ground-truth local CAD-scores. Figure 4A shows a structure and models of a strigolactone receptor (pdb code 5CBK), which consists of multiple alpha-helices. Figure 4B demonstrates a beta-propeller structure and models of Wdr5 protein (pdb code 2O9K). One can see that the VoroCNN predictions for native structures and corresponding models are visually very similar to the ground truth.

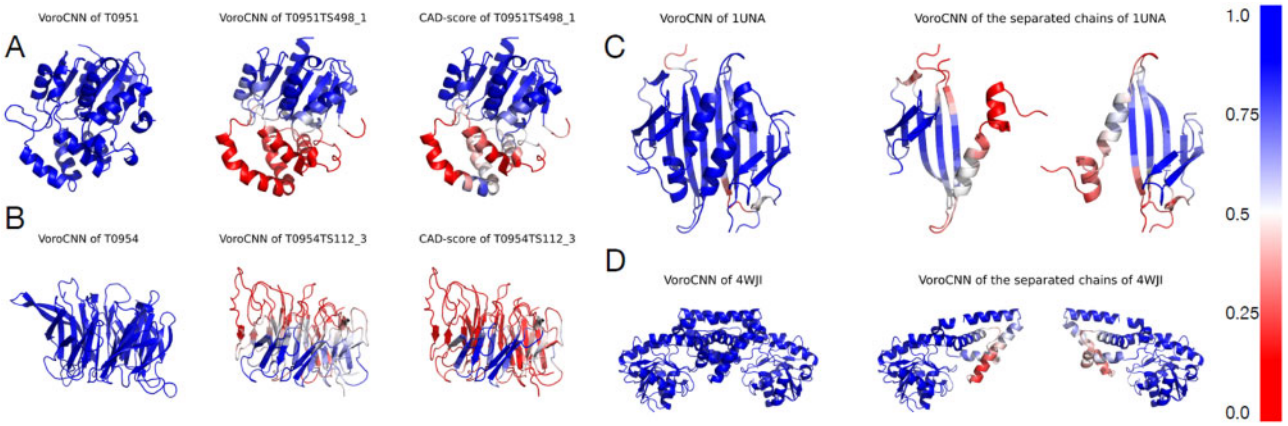


Fig. 4. Illustration of local scores predictions. The color-bar on the right corresponds to the values of the scores. (**A**) VoroCNN predictions for the T0951 CASP target (left), T0951TS498_1 model (center) and the ground-truth local CAD-scores (right). (**B**) VoroCNN predictions for the T0954 CASP target (left), T0954TS112_3 model (center) and the ground-truth local CAD-scores (right). (**C**) VoroCNN predictions for the obligatory complex of bacteriophage RNA-binding protein (pdb code 1UNA, left) and its individual subunits (right). (**D**) VoroCNN predictions for the obligatory complex of cyclohexadienyl dehydrogenase (pdb code 4WJI, left) and its individual subunits (right)

A generally interesting question is how much the predictions of local scores can be useful for the structural bioinformatics community. An obvious application of local scores is to highlight the local structural inaccuracies in protein models, as demonstrated in Figure 4A–B. Another practical example can be an analysis of protein binding interfaces. Figure 4C–D shows VoroCNN predictions for the two obligatory complexes and their individual subunits. We can clearly see that the binding interfaces have lower scores compared to the rest of the structure and are visually distinguishable. This can be explained by the specificity of these interfaces. Very often they are hydrophobic, and it is energetically unfavorable for them to be exposed to the solvent. That is why they can be detected by the local-scoring schemes, e.g. VoroCNN.

3.6 Tested architectures and ablation study

To design the final architecture, we studied a number of variations of the network. All the architectures were trained for 80 iterations, and their performance was measured on the validation dataset described in Section 2.3. All the networks followed the pattern '*Encoder-Body-Regressor*' described in Section 2.2.3 and represented in Figure 2. During these experiments, we assessed the performance of variations using per-target Pearson's *r*.

3.6.1 Body configuration

At the very beginning, we experimented with the size of *Body* and the position of the pooling layer. First of all, we put the pooling layer in the middle of *Body* and varied sizes of atom- and residue-level parts. Figure 5A shows these experiments. Each box corresponds to one architecture and is built based on the last 15 training iterations. Architectures presented in Figure 5A differ only by the configuration of *Body*—pairs of numbers on the x-axis relate to numbers of atom- and residue-level convolution layers correspondingly. Secondly, we varied the position of the pooling layer and the balance between the number of atom- and residue-level convolution layers. Figure 5B shows these experiments. For the sake of time and computational resources, we did not split graph edges into different types by the sequence separation factor or covalent types. Based on these experiments, we concluded that configurations with 4-5 atom- and residue-level layers and pooling in the middle achieve the best results.

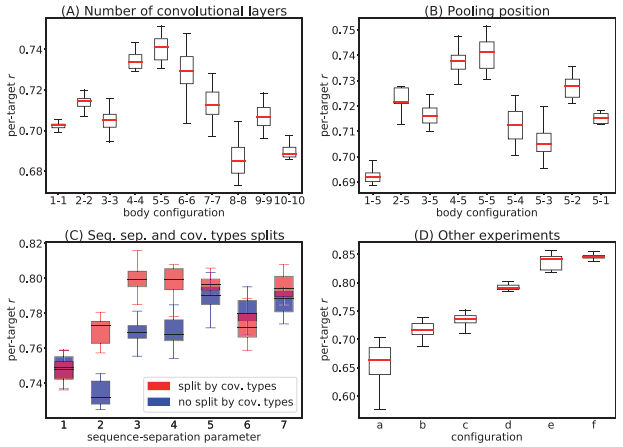
3.6.2 Sequence separation and covalent types

After that, we experimented with splitting the edges by the sequence separation factor and covalent types. Figure 5C represents these experiments. Here, each box corresponds to one architecture with the *Body* configuration '5-5' and a certain value of the sequence separation parameter (from 1 to 7). Red boxes represent networks with additional splitting by covalent types, blue boxes—without them. Based on these experiments, we concluded that additional splits do play a role. Namely, we decided to split edges by covalent types and set the sequence separation parameter to five.

3.6.3 Other experiments

We should note that we trained almost all the networks with two additional settings that we eventually omitted for the final version of VoroCNN. Our first assumption was that additional geometric descriptors could help to learn geometric information more efficiently. Hence, we conducted almost all our experiments with three additional descriptors that were assigned to nodes along with atom type one-hot vectors. These descriptors were: the solvent-accessible surface area computed with Voronota (Olechnovič and Venclovas, 2014), the volume of atom's Voronoi cell, also computed with Voronota, and the 'buriedness' of the atom, which is a topological distance in the graph to the nearest solvent-accessible atom. These features were normalized and concatenated to the nodes' embeddings, i.e. to the *Encoder* outputs. However, finally, we observed that additional descriptors were redundant and did not improve quality. Our second assumption was that data augmentation with models of high quality could diversify the training data and improve the generalization ability of networks. Hence, we generated random perturbations of target structures using the non-linear Normal Mode Analysis method NOLB (Hoffmann and Grudinin, 2017) and enriched the training data with these models (up to 50 synthetic models per each target). Surprisingly, we observed that such augmentation deteriorated the quality of predictions on the validation sets without near-native models.

In order to illustrate how geometric descriptors and data augmentation affect the performance, we provide Figure 5D. In this boxplot, we also put some other experiments that we conducted in order to test several additional hypotheses. This plot can also serve as an illustration of our overall progress in performance depending on the network configurations we tested. Here, one box includes several models with similar configurations that are thoroughly explained in the table in Figure 5. We include here (a) models with binary contact edges, (b) models with a smaller number of parameters, i.e. the dimensionality of *Encoder* output is reduced to four, (c) several models from Figure 5A and B, (d) the same models but trained without augmentation, (e) models with multiple edge types, (f) models without geometric descriptors.



Box	(a)	(b)	(c)	(d)	(e)	(f)
Number of networks included	4	4	5	5	5	5
Sequence separation	1	1	1	1	1-5	1-5
Split by covalent types	-	-	-	-	+	+
Encoder output size	10	4	10	10	10	10
Binary contact edges	+	-	-	-	-	-
Geometric descriptors	+	+	+	+	+	-
NOLB	+	+	+	-	-	-
Body configurations	.	.	2-5	2-5	.	.
	3-4	3-4	3-4	3-4	.	.
	4-4	4-4	4-4	4-4	.	.
	4-5	4-5	4-5	4-5	.	4-5
	5-5	5-5	5-5	5-5	5-5	.

Fig. 5. Ablation study. Boxplots show experiments on (**A**) the number of convolution layers in Body, (**B**) the position of the pooling layer, (**C**) splitting edges by sequence separation and covalent types, (**D**) binary edges, reduced Encoder output, data augmentation, and geometric descriptors. The table explains boxes from plot (D)

4 Conclusion

This work suggests a novel way to learn on 3D protein folds and 3D macromolecular data in general. For the first time, we demonstrate the applicability of learning on 3D Voronoi tessellations using graph convolutional networks. Our results confirm a high potential of using 3D tessellation and graph representation in general in various learning tasks in structural bioinformatics. Indeed, despite the complexity of the VoroCNN model and a rather big number of free parameters, our results are comparable to the state of the art and better than those of the very recent 3D CNN architectures trained on regular volumetric representations, e.g. Ornate. Thus, we believe that currently, given the available amounts of training data and computational resources, Voronoi tessellation is a better representation of 3D protein structure than raw volumetric data.

This work also illustrates the potential of methods that predict local folding accuracies for various structural bioinformatics applications. As we have demonstrated, VoroCNN can highlight structural inaccuracies in protein models, and can also distinguish protein binding interfaces.

Acknowledgements

The authors thank Elodie Laine from Sorbonne Université for the discussions during the study and proof-reading the manuscript.

Funding

This work was supported by the French-Lithuanian project PHC GILIBERT 2019 N° 42128UM/S-LZ-19-5, and by Inria International Partnership program BIOTOOLS.

Conflict of Interest

None declared.

References

Abriata,L.A. *et al.* (2019) A further leap of improvement in tertiary structure prediction in CASP13 prompts new routes for future assessments. *Proteins Struct. Funct. Bioinf.*, **87**, 1100–1112.

Adhikari,B. *et al.* (2018) DNCON2: improved protein contact prediction using two-level deep convolutional neural networks. *Bioinformatics*, **34**, 1466–1472.

Alom,M.Z. *et al.* (2018) The history began from AlexNet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.

Bach,F.R. and Jordan,M.I. (2004) Learning spectral clustering. In: *Advances in Neural Information Processing Systems 17, ed. Saul,L.K., Weiss,Y. and Bottou,L. Proceedings of the 2004 Conference, MIT Press, Cambridge, MA*, pp. 305–312.

Baldassarre,F. *et al.* (2020) GraphQA: protein model quality assessment using graph convolutional network. *Bioinformatics*. in press.

Bronstein,M.M. *et al.* (2017) Geometric deep learning: going beyond Euclidean data. *IEEE Signal Process. Mag.*, **34**, 18–42.

Cao,Y. and Shen,Y. (2020) Energy-based graph convolutional networks for scoring protein docking models. *Proteins Struct. Funct. Bioinf.*, **88**, 1091–1099.

Cazals,F. *et al.* (2006) Revisiting the Voronoi description of protein–protein interfaces. *Protein Sci.*, **15**, 2082–2092.

Cheng,J. *et al.* (2019) Estimation of model accuracy in CASP13. *Proteins Struct. Funct. Bioinf.*, **87**, 1361–1377.

Clevert,D.-A. *et al.* (2015) Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

Derevyanko,G. *et al.* (2018) Deep convolutional networks for quality assessment of protein folds. *Bioinformatics*, **34**, 4046–4053.

Dhillon,I.S. *et al.* (2004) Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, Association for Computing Machinery, New York, NY, USA, pp. 551–556.

Dhillon,I.S. *et al.* (2007) Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, **29**, 1944–1957.

Elofsson,A. *et al.* (2018) Methods for estimation of model accuracy in casp12. *Proteins Struct. Funct. Bioinf.*, **86**, 361–373.

Fan,Y. *et al.* (2016) Video-based emotion recognition using CNN-RNN and C3D hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, Association for Computing Machinery, New York, NY, USA, pp. 445–450.

Fisher,R.A. (1915) Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, **10**, 507–521.

Fout,A. *et al.* (2017) Protein interface prediction using graph convolutional networks. In: *Advances in Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, pp. 6530–6539.

Gilmer,J. *et al.* (2017) Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. JMLR.org, pp. 1263–1272.

Greener,J. *et al.* (2019) Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nat. Commun.*, **10**, 3977–3977.

Griffiths,D. and Boehm,J. (2019) A review on deep learning techniques for 3D sensed data classification. *Remote Sens.*, **11**, 1499.

Haas,J. *et al.* (2018) Continuous automated model evaluation (cameo) complementing the critical assessment of structure prediction in casp12. *Proteins Struct. Funct. Bioinf.*, **86**, 387–398.

Hamilton,W.L. *et al.* (2017) Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin, 40, 52–74*.

Hoffmann,A. and Grudinin,S. (2017) NOLB: nonlinear rigid block normal-mode analysis method. *J. Chem. Theory Comput.*, **13**, 2123–2134.

Hou,J. *et al.* (2019) Protein tertiary structure modeling driven by deep learning and contact distance prediction in CASP13. *Proteins Struct. Funct. Bioinf.*, **87**, 1165–1178.

Hurtado,D.M. *et al.* (2018) Deep transfer learning in the assessment of the quality of protein models. *arXiv preprint arXiv:1804.06281*.

Jones,D.T. and Kandathil,S.M. (2018) High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics*, **34**, 3308–3315.

Karasikov,M. *et al.* (2019) Smooth orientation-dependent scoring function for coarse-grained protein quality assessment. *Bioinformatics*, **35**, 2801–2808.

Kingma,D.P. and Ba,J. (2015) Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego, 2015*.

Kipf,T.N. and Welling,M. (2017) Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Kryshtafovych,A. *et al.* (2019) Critical assessment of methods of protein structure prediction (CASP)–Round XIII. *Proteins Struct. Funct. Bioinf.*, **87**, 1011–1020.

Li,R. *et al.* (2018) Adaptive graph convolutional neural networks. In: *Thirty-Second AAAI Conference on Artificial Intelligence. AAAI Press, Palo Alto, California, USA*.

Maghrabi,A.H. and McGuffin,L.J. (2020) Estimating the quality of 3D protein models using the ModFOLD7 server. In: Kihara,D. (ed) Protein Structure Prediction. Methods in Molecular Biology, vol 2165. Humana, New York, NY, pp. 69–81.

Manavalan,B. and Lee,J. (2017) Svmqa: support–vector-machine-based protein single-model quality assessment. *Bioinformatics*, **33**, 2496–2503.

Mariani,V. *et al.* (2013) lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, **29**, 2722–2728.

Moult,J. *et al.* (1995) A large-scale experiment to assess protein structure prediction methods. *Proteins Struct. Funct. Bioinf.*, **23**, ii–iv.

Myers,J.L. *et al.* (2010) *Research Design and Statistical Analysis*. Routledge, New York.

Olechnovič,K. and Venclovas,Č. (2014) Voronota: a fast and reliable tool for computing the vertices of the Voronoi diagram of atomic balls. *J. Comput. Chem.*, **35**, 672–681.

Olechnovič,K. and Venclovas,Č. (2017) VoroMQA: assessment of protein structure quality using interatomic contact areas. *Proteins Struct. Funct. Bioinf.*, **85**, 1131–1145.

Olechnovič,K. and Venclovas,Č. (2019) VoroMQA web server for assessing three-dimensional structures of proteins and protein complexes. *Nucleic Acids Res.*, **47**, W437–W442.

Olechnovič,K. and Venclovas,Č. (2020) Contact area-based structural analysis of proteins and their complexes using CAD-score. In: Gáspári,Z. (ed) Structural Bioinformatics. Methods in Molecular Biology, vol 2112. Humana, New York, NY, pp. 75–90.

Olechnovič,K. *et al.* (2013) CAD-score: a new contact area difference-based function for evaluation of protein structural models. *Proteins Struct. Funct. Bioinf.*, **81**, 149–162.

Olechnovič,K. *et al.* (2019) Comparative analysis of methods for evaluation of protein models against native structures. *Bioinformatics*, **35**, 937–944.

Pagès,G. and Grudinin,S. (2019) DeepSymmetry: using 3D convolutional networks for identification of tandem repeats and internal symmetries in protein structures. *Bioinformatics*, **35**, 5113–5120.

Pagès,G. *et al.* (2019) Protein model quality assessment using 3D oriented convolutional neural networks. *Bioinformatics*, **35**, 3313–3319.

Paszke,A. *et al.* (2019) PyTorch: an imperative style, high-performance deep learning library. In: Wallach,H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, **Vol. 32**. Curran Associates, Inc., pp. 8024–8035

Pontius,J. *et al.* (1996) Deviations from standard atomic volumes as a quality measure for protein crystal structures. *J. Mol. Biol.*, **264**, 121–136.

Poupon,A. (2004) Voronoi and Voronoi-related tessellations in studies of protein structure and interaction. *Curr. Opin. Struct. Biol.*, **14**, 233–241.

Richards,F.M. (1974) The interpretation of protein structures: total volume, group volume distributions and packing density. *J. Mol. Biol.*, **82**, 1–14.

Richards,F.M. (1977) Areas, volumes, packing, and protein structure. *Annu. Rev. Biophys. Bioeng.*, **6**, 151–176.

Sanyal,S. *et al.* (2020) ProteinGCN: Protein model quality assessment using graph convolutional networks. *BioRxiv 2020.04.06.028266*.

Scarselli,F. *et al.* (2008) The graph neural network model. *IEEE Trans. Neural Netw.*, **20**, 61–80.

Senior,A.W. *et al.* (2019) Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (CASP13). *Proteins Struct. Funct. Bioinf.*, **87**, 1141–1148.

Senior,A.W. *et al.* (2020) Improved protein structure prediction using potentials from deep learning. *Nature*, **577**, 706–705.

Uziela,K. *et al.* (2016) ProQ3: improved model quality assessments using rosetta energy terms. *Sci. Rep.*, **6**, 1–10.

Uziela,K. *et al.* (2017) Proq3d: improved model quality assessments using deep learning. *Bioinformatics*, **33**, 1578–1580.

Uziela,K. *et al.* (2018) Improved protein model quality assessments by changing the target function. *Proteins Struct. Funct. Bioinf.*, **86**, 654–663.

Wang,J. *et al.* (2016) CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 2285–2294.

Wang,J. *et al.* (2017) New deep neural networks for protein model evaluation. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE Computer Society, pp. 309–313.

Won,J. *et al.* (2019) Assessment of protein model structure accuracy estimation in CASP13: challenges in the era of deep learning. *Proteins Struct. Funct. Bioinf.*, **87**, 1351–1360.

Wu,Z. *et al.* (2021) A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, **32**, 4–24.

Wüthrich,K. (1990) Protein structure determination in solution by NMR spectroscopy. *J. Biol. Chem.*, **265**, 22059–22062.

Xu,J. (2019) Distance-based protein folding powered by deep learning. *Proc. Natl. Acad. Sci. USA*, **116**, 16856–16865.

Xu,J. and Wang,S. (2019) Analysis of distance-based protein structure prediction by deep learning in CASP13. *Proteins Struct. Funct. Bioinf.*, **87**, 1069–1081.

Zamora-Resendiz,R. and Crivelli,S. (2019) Structural learning of proteins using graph convolutional neural networks. *bioRxiv*, 610444.

Zemla,A. *et al.* (1999) Processing and analysis of CASP3 protein structure predictions. *Proteins Struct. Funct. Bioinf.*, **37**, 22–29.

Zemla,A. *et al.* (2001) Processing and evaluation of predictions in CASP4. *Proteins Struct. Funct. Bioinf.*, **45**, 13–21.

Zhang,K. *et al.* (2017) Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, **26**, 3142–3155.

Zheng,W. *et al.* (2019) Deep-learning contact-map guided protein structure prediction in CASP13. *Proteins Struct. Funct. Bioinf.*, **87**, 1149–1164.

Zimmer,R. *et al.* (1998) New scoring schemes for protein fold recognition based on Voronoi contacts. *Bioinformatics (Oxford, England)*, **14**, 295–308.