Structural bioinformatics

Deep convolutional networks for quality assessment of protein folds

Georgy Derevyanko1,*, Sergei Grudinin2, Yoshua Bengio3,† and Guillaume Lamoureux1,*

1Department of Chemistry and Biochemistry and Centre for Research in Molecular Modeling (CERMM), Concordia University, Montréal, Québec H4B 1R6, Canada, 2Inria, Université Grenoble Alpes, CNRS, Grenoble INP, LJJK, Grenoble 38000, France and 3Department of Computer Science and Operations Research, Université de Montréal, Montréal, Québec H3C 3J7, Canada

*To whom correspondence should be addressed.
†CIFAR Fellow.
Associate Editor: Alfonso Valencia

Received on October 5, 2017; revised on April 30, 2018; editorial decision on June 8, 2018; accepted on June 15, 2018

Abstract

Motivation: The computational prediction of a protein structure from its sequence generally relies on a method to assess the quality of protein models. Most assessment methods rank candidate models using heavily engineered structural features, defined as complex functions of the atomic coordinates. However, very few methods have attempted to learn these features directly from the data.

Results: We show that deep convolutional networks can be used to predict the ranking of model structures solely on the basis of their raw three-dimensional atomic densities, without any feature tuning. We develop a deep neural network that performs on par with state-of-the-art algorithms from the literature. The network is trained on decoys from the CASP7 to CASP10 datasets and its performance is tested on the CASP11 dataset. Additional testing on decoys from the CASP12, CAMEO and 3DRobot datasets confirms that the network performs consistently well across a variety of protein structures. While the network learns to assess structural decoys globally and does not rely on any predefined features, it can be analyzed to show that it implicitly identifies regions that deviate from the native structure.

Availability and implementation: The code and the datasets are available at https://github.com/lamoureux-lab/3DCNN_MQA.

Contact: georgy.derevyanko@gmail.com or guillaume.lamoureux@concordia.ca

Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction

The protein folding problem remains one of the outstanding challenges in structural biology (Dill and MacCallum, 2012). It is usually defined as the task of predicting the three-dimensional (3D) structure of a protein from its amino acid sequence. Progress in the field is monitored through the Critical Assessment of protein Structure Prediction (CASP) competition (Moult et al., 1995), in which protein folding methods are evaluated in terms of their accuracy at predicting structures ahead of their publication. Most methods participating in CASP include a conformational sampling step, which generates a number of plausible protein conformations, and a quality assessment step, which attempts to select the conformations closest to the unknown native structure.

In this work we explore the application of deep learning to the problem of ‘model quality assessment’ (MQA), also called ‘estimation of model accuracy’ (Kryshtafovych et al., 2016).
Deep learning has recently garnered considerable interest in the research community (LeCun et al., 2015), particularly in computer vision and natural language processing. Unlike more ‘shallow’ machine learning approaches, deep learning improves performance by learning a hierarchical representation of the raw data at hand. It alleviates the need for feature engineering, which has traditionally constituted the bulk of the work done by researchers.

Deep learning has been applied to biological data and has yielded remarkable results for predicting the effects of genetic variations on human RNA splicing (Xiong et al., 2015), for identifying DNA- and RNA-binding motifs (Alipanahi et al., 2015) and for predicting the effects of non-coding DNA variants with single nucleotide precision (Zhou and Troyanskaya, 2015). These successes have one thing in common: they use raw data directly as input and do not attempt to engineer features from them.

Deep-learning-inspired methods have been used for protein structure quality assessment as well. For instance, DeepQA (Cao et al., 2016) uses nine scores from other MQA methods and seven physicocal features extracted from the structure as input features to a deep restricted Boltzmann machine (Hinton et al., 2006). The method has been reported to outperform ProQ2 (Ray et al., 2012), which was the top-performing method in the CASP11 competition (Kryshtafovych et al., 2016). ProQ3D (Uziela et al., 2017) uses the same high-level input features as the earlier ProQ3 method (Uziela et al., 2016) but achieves better performance by replacing the support vector machine model by a deep neural network. Since the original ProQ3 method had one of the top performances in CASP12 (Elofsson et al., 2017), it can be expected that ProQ3D performs equally well.

Although both DeepQA and ProQ3D methods are based on deep neural networks, they use high-level features as input. In that sense, they use deep learning models more as traditional ‘shallow’ classifiers than as end-to-end learning models. It is likely that they do not benefit from all advantages offered by the deep learning approach. By comparison, the DL-Pro algorithm (Nguyen et al., 2014) provides a 5-tiered classification of all structures in the PDB using 3D convolutional neural networks (Moult et al., 2014). We use the CASP7 to CASP10 data as training set and the CASP11 data as test set, for a total of 564 target structures in the training set and 83 target structures in the test set. Additional testing is done on the CASP12 (Elofsson et al., 2017), CAMEO (Haas et al., 2013) and 3DRobot (Deng et al., 2016) datasets. Each target from the training set has 282 decoys on average. The test set (CASP11) is split into two subsets (Kryshtafovych et al., 2016): ‘stage 1’ with 20 decoys per target selected randomly from all server predictions and ‘stage 2’ with, for each target, the 150 decoys considered best by the Davis-QAconsensus evaluation method (Kryshtafovych et al., 2016). The native structures are excluded from both training and test datasets. To make the structural data more consistent we optimize the side chains of all decoys using SCWRL4 (Krivov et al., 2009).

The training and test sets cover a similar range in protein sequence length (see Supplementary Fig. S1). To get a sense of their degree of overlap, we have aligned all test sequences against all training sequences using blastp (Altschul et al., 1990). Less than 11% of the targets in the test set (9 out of 83) have sequence similarity with any target in the training set (see Supplementary Table S1). We have also computed the dataset overlap in terms of Pfam families (Finn et al., 2016). The families were found using HMMER (Finn et al., 2015) with an E-value cutoff of 1.0 (Finn et al., 2016). Ignoring targets for which no Pfam family could be determined, approximately 25% of the test set targets share a family with approximately 10% of the training set targets (see Supplementary Table S2).

Finally, we have compared the structures in the training and test sets using the ECOD database (Cheng et al., 2014). This database provides a 5-tiered classification of all structures in the PDB according to the following criteria: architecture (A-group), possible homology (X-group), homology (H-group), topology (T-group) and family (F-group). Among all test targets for which ECOD classes could be determined, approximately 16% belong to the same F-group of at least one training target, 61% belong to the same T-group (and same H-group), 72% belong to the same X-group and 96% belong to the same A-group (see Supplementary Figs S2 and S3).

2.2 Input
Each protein structure is represented by 11 density maps corresponding to the atom types defined in Table 1. These atom types are a simplification of the 20 types proposed by Huang and Zou (2006, 2008), to reduce the memory footprint of the model, yet to keep as rich an input as possible. Preliminary experiments using four atom types (corresponding to the chemical elements C, N, O and S) yielded worse performance on the validation set. The density of an atom is represented using the function

$$
\rho(r) = \begin{cases} 
  e^{-r^2/2} & \text{if } r \leq 2.0 \text{ Å} \\
  0 & \text{otherwise}
\end{cases}
$$

(1)

The atomic density is projected to the grid corresponding to its atom type. Each grid has a resolution of 1 Å and has 120 × 120 × 120 cells (see Supplementary Fig. S4 for an illustration).

2.3 Model
We score protein structures using 3D convolutional neural networks (CNNs). CNNs were first proposed for image recognition by LeCun et al. (1989) and first applied to biological data by Bengio et al. (1990). Convolutional neural networks have gained wider

2 Materials and methods

2.1 Datasets
We train and assess our method using the datasets of non-native protein conformations (‘decoys’) from the CASP competition
recognition after the ImageNet 2012 competition (Krizhevsky et al., 2012). The architecture of the model is shown in Figure 1. It is comprised of four blocks of alternating convolutional, batch normalization and ReLU layers (terminated by a maximum pooling layer), followed by three fully-connected layers with ReLU nonlinearities. The final output of the network is a single number, interpreted as the score of the input structure. (See Supplementary Table S3 for more details.) Details of the architecture were chosen to maximize the score of the input structure. (See Supplementary Table S3 for more details.)

The final output of the network is a single number, interpreted as the score of the input structure. (See Supplementary Table S3 for more details.) Details of the architecture were chosen to maximize the score of the input structure. (See Supplementary Table S3 for more details.)

The final output of the network is a single number, interpreted as the score of the input structure. (See Supplementary Table S3 for more details.) Details of the architecture were chosen to maximize the score of the input structure. (See Supplementary Table S3 for more details.)

Each 3D convolutional layer takes \( N \) input density maps \( f^m \) and transforms them using \( M \) filters \( F \) according to the following formula:

\[
f^m_{\text{out}}(r) = \sum_{i=1}^{N} \int f_i(r - r') \cdot f^m(r') \, dr', \quad \forall i \in [1,M]
\]  

(2)

In practice, these convolutions are approximated by sums on a 3D grid. The ReLU nonlinearity is computed as follows:

\[
f^m_{\text{out}}(r) = \begin{cases} 
    f^m(r) & \text{if } f^m(r) \geq 0 \\
    0 & \text{otherwise} 
\end{cases}, \quad \forall i \in [1,M]
\]  

(3)

The idea of batch normalization was introduced by Ioffe and Szegedy (2015) to reduce the shift in the distribution of subnetwork outputs during training. This layer normalizes each input value according to the mean and variance within the subset of examples used to estimate the gradient (the ‘batch’):

\[
f_k(r) = \frac{f_k(r) - \mu_k(r)}{\sqrt{\sigma_k^2(r) + \epsilon}}, \quad \forall k \in [1,N_b]
\]  

(4)

where \( \mu_k(r) \) is the mean of all \( f^m(r) \) maps from the batch (calculated at each position \( r \)) and \( \sigma_k^2(r) \) is the variance. \( N_b \) is the number of examples in the batch. The constant \( \epsilon = 10^{-5} \) is added to avoid division by zero. The output of the layer is computed by scaling the normalized inputs:

\[
f_k^{\text{out}}(r) = \gamma_k f_k(r) + \beta, \quad \forall k \in [1,N_b]
\]  

(5)

Parameters \( \gamma \) and \( \beta \) are learned along with other parameters of the network during the training.

The maximum pooling layer (‘MaxPool’) is used to build a coarse-grained representation of the input. The output of this layer is the maximum over the cubes of size \( d \times d \times d \) that cover the input domain with a stride \( l \) in each direction. This operation makes the output size approximately \( l \) times smaller than the input in each direction. All four ‘MaxPool’ layers of the model (Fig. 1) use \( d = 3 \) and \( l = 2 \).

During the coarse-graining procedure, the size of the individual data grids eventually shrinks to a single cell. The flattening layer reshapes the array of \( 1 \times 1 \times 1 \) density maps into a single vector. Afterwards, we compute several transformations using fully-connected layers. Each of these layers transforms a vector \( x_{\text{in}} \) as follows:

\[
x_{\text{out}} = W \cdot x_{\text{in}} + b
\]  

(6)

where \( W \) is a rectangular matrix and \( b \) is a vector, learned during the training. Each output vector is then transformed by a ReLU layer.

Table 1. Atom types used in this work

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Sulfur/selenium</td>
<td>CYS:SG, MET:SD, MSE:SE</td>
<td></td>
</tr>
<tr>
<td>2 Nitrogen (amide)</td>
<td>ASN:ND2, GLN:NE2, backbone N (including N-terminal)</td>
<td></td>
</tr>
<tr>
<td>3 Nitrogen (aromatic)</td>
<td>HIS:ND1/NE1, TRP:NE1</td>
<td></td>
</tr>
<tr>
<td>4 Nitrogen (guanidinium)</td>
<td>ARG:NE/NH*</td>
<td></td>
</tr>
<tr>
<td>5 Nitrogen (ammonium)</td>
<td>LYS:NZ</td>
<td></td>
</tr>
<tr>
<td>6 Oxygen (carbonyl)</td>
<td>ASN:OD1, GLN:OE1, backbone O (except C-terminal)</td>
<td></td>
</tr>
<tr>
<td>7 Oxygen (hydroxyl)</td>
<td>SER:OG, THR:OG1, TYR:OH</td>
<td></td>
</tr>
<tr>
<td>8 Oxygen (carboxyl)</td>
<td>ASP:OD*, GLU:OE*, C-terminal O, C-terminal OXT</td>
<td></td>
</tr>
<tr>
<td>9 Carbon (sp2)</td>
<td>ARG:CE, ASN:CG, ASP:CG, GLN:CD, GLU:CD, backbone C, H:CG(CD2)CE1</td>
<td></td>
</tr>
<tr>
<td>10 Carbon (aromatic)</td>
<td>PHE:CG/CD*/CE*/CZ, TRP:CG/CD*/CE*/CZ*/CH2, TYR:CG/CD*/CE*/CZ</td>
<td></td>
</tr>
</tbody>
</table>

Note: Atoms in each group are identified using their standard PDB residue names and atom names. Asterisks (*) correspond to either 1, 2 or 3.

![Fig. 1. Schematic representation of the convolutional neural network architecture used in this work. Unless otherwise specified, line connections across boxes denote the consecutive application of a 3D convolutional layer (‘Convolution’), a batch normalization layer (‘BatchNorm’) and a ReLU layer. Grey arrows between boxes denote maximum pooling layers (‘MaxPooling’). Labels ‘\*’ \( M \) denote the number of 3D grids and the number of filters used in the corresponding convolutional layer. The grey stripes denote one-dimensional vectors and crossed lines between them stand for fully-connected layers with ReLU nonlinearities. Details of the model can be found in Supplementary Table S3](https://academic.oup.com/bioinformatics/article/34/23/4046/5040325)
2.4 Training loss function

Decoy quality assessment is essentially a ranking problem: we have to arrange decoys according to their similarity to the native structure as quantified, for instance, by the global distance test total score (GDT_TS) (Zemla et al., 2001). Such a ranking approach has recently been used by the MQAPRank method (Jing et al., 2016), which, however, relies on a support vector machine model and uses high-level features as input.

We define the training loss function in terms of the margin ranking loss (Gong et al., 2013; Joachims, 2002) for each pair of decoys. Let GDT_TS_i denote the GDT_TS of decoy i and let y_i be the ordering coefficient of decoys i and j, equal to +1 if GDT_TS_i ≤ GDT_TS_j and to −1 otherwise. GDT_TS values are computed using the TM-score program (Zhang and Skolnick, 2004). The original GDT_TS covers the range [0, 100] but in this work we use a GDT_TS normalized to the range [0, 1]. Let s_i denote the output of the network for decoy i. We use the following expression for the pairwise ranking loss:

\[ L_{ij} = w_{ij} \max[0, 1 - y_i \cdot (s_i - s_j)] \]  

(7)

The coefficient \( w_{ij} \) is defined so that decoys with similar scores are excluded from the training: \( w_{ij} \) is one if |GDT_TS_i − GDT_TS_j| > 0.1 and is zero otherwise.

During the training procedure we load \( N_B \) decoy structures of a given target into memory (a ‘batch’) and compute the output of the network and the average ranking loss over all pairs:

\[ L = \frac{1}{N_B} \sum_{i=1}^{N_B} \sum_{j=1 \neq j}^{N_B} L_{ij} \]  

(8)

In principle, the ranking loss of Eq. 8 would be minimal for any output s decreasing monotonically with increasing GDT_TS. While an output strictly equal to the negative of the GDT_TS would produce a loss of zero, our preliminary experiments have shown better performance when the model is trained so that s orders like ‘–GDT_TS’ without necessarily being equal to it. This is consistent with the results by Jing et al. (2016), who show that, for the same input features, models based on GDT_TS ranking outperform models based on GDT_TS regression.

2.5 Evaluation criteria

We evaluate the model using various correlation coefficients of the scores and using an evaluation loss function distinct from the training loss function. The evaluation loss is defined, for any given protein, as the difference between the GDT_TS of the best decoy and the GDT_TS of the decoy with the lowest predicted score s:

\[ \text{Loss} = \max(GDT\_TS) - GDT\_TS_{\text{argmin}(s)} \]  

(9)

The correlation coefficients between the s value produced by the model and the GDT_TS are computed for all decoys of a given target in the test set and are then averaged over all targets. An ideal MQA algorithm would show a correlation coefficient of −1 and zero loss. These two criteria measure different qualities of the model. On the one hand, a correlation coefficient of −1 would be achieved if the algorithm ranks all decoys in the exact order of their GDT_TS (from best to worst). On the other hand, a zero loss would be achieved if the algorithm systematically assigns the lowest s value to the decoy with the highest GDT_TS, irrespective of the s value it assigns to the other decoys.

2.6 Optimization and dataset sampling

The model is optimized using the Adam algorithm (Kingma and Ba, 2014). The gradient of the average training loss function (Eq. 8) with respect to the model parameters is computed for the decoys in the batch. The batch size is set to \( N_B = 9 \) decoys. Preliminary experiments have shown that smaller batches tend to give too noisy gradients.

The training dataset is sampled by first choosing a random target from the dataset, then sampling decoys of this target. One epoch corresponds to a cycle over all targets in the training subset. Models are saved every 10 epochs and the arrow shows the minimum validation loss for which a model was saved (at epoch 40).

Fig. 2. Evaluation loss (Eq. 9), Kendall \( \tau \) and Pearson \( R \) coefficients evaluated on the validation subset during the training procedure. One epoch corresponds to a cycle over all targets in the training subset. Models are saved every 10 epochs and the arrow shows the minimum validation loss for which a model was saved (at epoch 40).

The training dataset is sampled by first choosing a random target from the dataset, then sampling decoys of this target. One epoch corresponds to a cycle over all targets in the training subset. Models are saved every 10 epochs and the arrow shows the minimum validation loss for which a model was saved (at epoch 40).

We select the final model based on its performance on a validation subset consisting of 35 targets (and their decoys) picked at random from the training set and excluded from the training procedure. The remaining 529 targets are called the training subset. Figure 2 shows the Kendall \( \tau \) and Pearson \( R \) coefficients and the evaluation loss on the validation subset over 52 epochs of training. Models are saved every 10 epochs and we pick the one that has the smallest evaluation loss (at epoch 40). Table 2 summarizes the performance metrics on the training and validation sets for the model at epoch 40. (See Supplementary Fig. S5 for results broken down by target.)

\[ 1 + \left[ N_B \times \frac{\max(GDT\_TS) - \min(GDT\_TS)}{\max(GDT\_TS) - \min(GDT\_TS)} \right] \]  

(10)
Table 2. Performance of the 3DCNN model from epoch 40 on the training and validation subsets

<table>
<thead>
<tr>
<th>Data</th>
<th>Loss (Eq. 9)</th>
<th>Pearson R</th>
<th>Spearman ρ</th>
<th>Kendall τ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training subset</td>
<td>0.146</td>
<td>-0.71</td>
<td>-0.61</td>
<td>-0.45</td>
</tr>
<tr>
<td>Validation subset</td>
<td>0.135</td>
<td>-0.71</td>
<td>-0.59</td>
<td>-0.44</td>
</tr>
</tbody>
</table>

Fig. 3. Distributions of the scores of five decoys for target T0832 under random translations and rotations. A lower score represents a higher quality.

3 Results

Ideally, the score assigned to a decoy should not depend on its position and orientation in space. To allow the 3DCNN model to learn this invariance, the rotational and translational degrees of freedom of all decoy structures are randomly sampled during the training. Figure 3 shows the distributions of scores for several decoys of the same target (T0832), calculated using the trained model for 900 rotations and translations sampled uniformly. While the score of a given structure is not strictly invariant under rotation and translation, it has a relatively narrow, unimodal distribution. (See Supplementary Fig. S6 for a distribution of scores under rotations and translations separately.) More importantly, the difference between the average scores of two decoys is usually larger than their standard deviations. To reduce the influence of position and orientation on the final ranking, we estimate the score of each decoy from the average of 90 scores calculated for random rotations and translations.

3.1 Performance on the CASP11 benchmark

Table 3 reports the performance of our model (3DCNN) compared to that of a number of state-of-the-art MQA methods: ProQ2D, ProQ3D (Uziela et al., 2017), VoroMQA (Olechnović and Venclovas, 2017) and RWplus (Zhang and Zhang, 2010). (See Supplementary Figs S7 and S8 for ranking results broken down by target.) ProQ2D uses a number of carefully crafted features such as atomic contacts, residue-residue contacts, surface accessibilities (as found in the structure and as predicted from the sequence) and secondary structure (observed and predicted). ProQ3D employs the same features as ProQ2D, as well as some Rosetta energy terms.

Table 3. Performance comparison of our method (3DCNN) with other state-of-the-art MQA methods on the CASP11 dataset stages 1 and 2 (see text)

<table>
<thead>
<tr>
<th>MQA method</th>
<th>Loss (Eq. 9)</th>
<th>Pearson R</th>
<th>Spearman ρ</th>
<th>Kendall τ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProQ3D</td>
<td>0.046</td>
<td>0.755</td>
<td>0.673</td>
<td>0.529</td>
</tr>
<tr>
<td>ProQ2D</td>
<td>0.064</td>
<td>0.729</td>
<td>0.604</td>
<td>0.468</td>
</tr>
<tr>
<td>3DCNN</td>
<td>0.064</td>
<td>0.535</td>
<td>0.425</td>
<td>0.325</td>
</tr>
<tr>
<td>VoroMQA</td>
<td>0.087</td>
<td>0.637</td>
<td>0.521</td>
<td>0.394</td>
</tr>
<tr>
<td>RWplus</td>
<td>0.122</td>
<td>0.512</td>
<td>0.402</td>
<td>0.303</td>
</tr>
<tr>
<td>Stage 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VoroMQA</td>
<td>0.063</td>
<td>0.457</td>
<td>0.449</td>
<td>0.321</td>
</tr>
<tr>
<td>3DCNN</td>
<td>0.064</td>
<td>0.421</td>
<td>0.409</td>
<td>0.288</td>
</tr>
<tr>
<td>ProQ3D</td>
<td>0.066</td>
<td>0.452</td>
<td>0.433</td>
<td>0.307</td>
</tr>
<tr>
<td>ProQ2D</td>
<td>0.072</td>
<td>0.437</td>
<td>0.422</td>
<td>0.299</td>
</tr>
<tr>
<td>RWplus</td>
<td>0.089</td>
<td>0.206</td>
<td>0.248</td>
<td>0.176</td>
</tr>
</tbody>
</table>

Note: The table reports the absolute, per-target average values of the correlation coefficients. Structures were optimized with SCWRL4 before scoring with each method.

(Leaver-Fay, 2011). RWplus, similar to DOPE (Shen and Sali, 2006) and DFIRE (Zhou and Zhou, 2009), uses a scoring approach based on statistical pairwise potentials. VoroMQA uses knowledge-based potentials that depend on the contact surface between pairs of heavy atoms in the protein (or the solvent). Its approach is distinct from both the machine-learning techniques exemplified by the ‘ProQ’ methods and the statistical potential techniques exemplified by the RWplus method. The methods chosen have available codes and could be re-evaluated on our CASP11 benchmark. Targets T0797, T0798, T0825 were removed from the benchmark because they were released for multimetric prediction. All methods were re-evaluated using the default settings proposed by their authors.

Methods ProQ2D and ProQ3D are trained on the CASP9 and CASP10 models (Uziela et al., 2017), using features pre-trained on a diverse set of protein structures (Ray et al., 2012; Uziela et al., 2016). The VoroMQA method is trained on high-resolution, non-redundant structures from the PDB (Olechnović and Venclovas, 2017) (2.5 Å resolution cutoff and 50% sequence identity cutoff, for a total of 12,825 PDB entries). The RWplus scoring function is trained on the CASP7 and CASP8 models (Zhang and Zhang, 2010), using a statistical potential trained on high-resolution structures from the PDB (1.6 Å resolution cutoff and 20% sequence identity cutoff, for a total of 1,383 PDB entries).

Despite relying solely on atomic coordinates, the 3DCNN model achieves a performance comparable to those of the heavily engineered ProQ2D and ProQ3D models, with evaluation losses either slightly above or slightly below, depending on the test set (see Table 3).

3.2 Analysis

To show that the 3DCNN network has learned a relevant description of protein structure and not merely artifacts of the dataset that correlate with the desired outcome, we first identify the regions of a decoy structure that are responsible for an increase of its score (a decrease in its quality). If the network has learned interpretable features of the input, we expect these parts of the decoy to deviate from the native structure. We use the Grad-CAM analysis technique proposed by Selvaraju et al. (2016). The key idea of this technique is to compute the gradient of the final score with respect to the output of a certain layer of the network, then compute the sum of this layer...
output weighted by the gradient. The weighted sum highlights the regions of the layer that are both strongly activated and highly influential on the final score. To generate an interpretable map, the weighted sum is then scaled up to the size of the input of the network, using tri-linear interpolation. This up-sampled map indicates which parts of the input contribute the most to the gradient of the score. In our case we choose to analyze layer 10, for which the output grid size is $25 \times 25 \times 25$. We tested the method on neighboring layers and layer 10 represents the best tradeoff between interpretability and coarseness. In line with our scoring procedure, we average the results from the Grad-CAM analysis over 90 rotations and translations of the decoy. We obtain the Grad-CAM output for each transformation and project it onto the atoms of the decoy.

Figure 4 shows a projection of the Grad-CAM results onto the atoms of four decoys of target T0786, represented as a color-coded value on the cartoon rendering of the structures. The orange/yellow regions are mainly found at the surface of the lower-quality decoys while the blue/green regions are found at the core. This indicates that the quality of the decoy would go up for any decrease in atomic density at the surface but would be unaffected by a change in density at the core (see Supplementary Fig. S9). It also suggests that the neural network recognizes and enforces packing. Interestingly, we find that the Grad-CAM outputs are mostly zero for decoys close to the native structure, despite the fact that no gradient information was included in the training procedure (see Supplementary Table S4). Moreover, low Grad-CAM outputs indicate high local model quality, as measured by the IDDT score (Mariani et al., 2013) (see Supplementary Figs S10 and S11).

### 3.3 Performance on the CASP12, CAMEO and 3DRobot datasets

To verify that the network does not rely on artifacts in the data to rank decoys, we have assessed its performance on three additional datasets.

The CASP12 dataset contains all server predictions from the CASP12 competition (Elofsson et al., 2017) available as of December 2, 2017. On this dataset, the pre-trained 3DCNN model displays an evaluation loss smaller than all other models tested (see Table 4).

The CAMEO dataset contains all structural models published on the CAMEO-QE webpage (Haas et al., 2013) in the 6-month period prior to December 10, 2017. On this dataset, the 3DCNN model performs significantly better than both VoroMQA and RWplus, the two other models tested (see Table 4).

The 3DRobot dataset, generated by the 3DRobot algorithm (Deng et al., 2016), consists of 300 decoys for each of 200 single-domain proteins selected from the PDB. The algorithm yields decoys that are uniformly distributed within an RMSD range of 0 to 12 Å away from the native structure. The evaluation loss for these 60 000 decoys is larger for the 3DCNN model than for VoroMQA and RWplus (see Table 4). However, all three models tested yield scores that are highly correlated with the GDT_TS (see Table 4), which suggests that the 3DRobot decoys are easier to rank and therefore are a less stringent benchmark (see Supplementary Fig. S12).

We have also examined how the performance on a given test target is affected by its level of structural similarity with the training set. While the average performance of the 3DCNN model is typically lower for test targets with no ECOD overlap with the training set than for test targets with ECOD overlap at any level, this trend is not robust enough to suggest that the model is overfitting the data (see Supplementary Figs S13 to S16). For instance, the performance of the model is not significantly lower for test targets matching the training set at the ‘architecture’ level only (A-group) than for test targets matching at the ‘family’ level (F-group).

---

**Table 4. Performance comparison of our method (3DCNN) with other state-of-the-art MQA methods on the CASP12, CAMEO and 3DRobot datasets (see text)**

<table>
<thead>
<tr>
<th>MQA method</th>
<th>Loss (Eq. 9)</th>
<th>Pearson R</th>
<th>Spearman ρ</th>
<th>Kendall τ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASP12</td>
<td>0.146</td>
<td>0.607</td>
<td>0.521</td>
<td>0.381</td>
</tr>
<tr>
<td>ProQ2D</td>
<td>0.151</td>
<td>0.619</td>
<td>0.607</td>
<td>0.453</td>
</tr>
<tr>
<td>VoroMQA</td>
<td>0.161</td>
<td>0.557</td>
<td>0.515</td>
<td>0.380</td>
</tr>
<tr>
<td>ProQ3D</td>
<td>0.164</td>
<td>0.609</td>
<td>0.602</td>
<td>0.451</td>
</tr>
<tr>
<td>RWplus</td>
<td>0.192</td>
<td>0.313</td>
<td>0.355</td>
<td>0.257</td>
</tr>
<tr>
<td>CAMEO</td>
<td>0.060</td>
<td>0.586</td>
<td>0.532</td>
<td>0.426</td>
</tr>
<tr>
<td>VoroMQA</td>
<td>0.099</td>
<td>0.456</td>
<td>0.427</td>
<td>0.346</td>
</tr>
<tr>
<td>RWplus</td>
<td>0.162</td>
<td>0.122</td>
<td>0.095</td>
<td>0.068</td>
</tr>
<tr>
<td>3DRobot</td>
<td>0.038</td>
<td>0.891</td>
<td>0.859</td>
<td>0.678</td>
</tr>
<tr>
<td>VoroMQA</td>
<td>0.076</td>
<td>0.844</td>
<td>0.829</td>
<td>0.651</td>
</tr>
<tr>
<td>RWplus</td>
<td>0.083</td>
<td>0.856</td>
<td>0.839</td>
<td>0.652</td>
</tr>
</tbody>
</table>

*Note: Structures from the CASP12 and CAMEO datasets were optimized with SCWRL4 before scoring with each method.*
4 Discussion

This work shows that it is possible to construct an algorithm that learns to assess the quality of protein models from a raw representation. Here, we have used 3D atomic densities broken down by atom types. However it is clear that any other physical quantity defined on a grid can be employed, such as the electrostatic potential calculated using the Poisson-Boltzmann equation (Honig and Nicholls, 1995) or the solvent density calculated using 3D-RISM (Stumpe et al., 2011). So far, no other MQA method has managed to include these crucial properties.

The loss function we used for training does not aim to predict the GDT_TS of a decoy but rather to sort decoys according to their GDT_TS. This ranking-based strategy allows the score to be interpreted as an energy function, which is not directly related to GDT_TS but which decreases when GDT_TS increases and has a local minimum for the native structure. In future work we plan to add terms to the loss function that penalize the first and second order derivatives of the loss at the native structure, to ensure that the score indeed reaches a local minimum there.

This work also identifies important avenues for improvement. First, the model captures the invariance of the score under translations and rotations only in an approximate way. This invariance problem can however be solved using the approach of Worrall et al. (2016), in which the coefficient space of the convolutional filters is restricted to circular harmonics, which encodes equivariance under rotations at each layer of the network and leads to invariance of the final output. Second, the output of the model remains difficult to interpret. While interpretation of deep neural networks remains an important research problem, the field is undergoing rapid progress. For instance, recently published work (Bau et al., 2017) has shown that interpretability can be quantified using extensively labeled image datasets that contain the bounding boxes and labels for fine-grained features such as body parts or car parts. In the case of protein models, many such labels (and bounding boxes) are readily available: amino acids, secondary structure elements, hydrogen bond networks, disulfide bonds, etc. Unlike in conventional machine learning models, these features would not be used for prediction but for interpretation of the prediction.

Funding

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN 355789 to G.D. and RGPIN 1016552 to Y.B.]; and the Canada Research Chair and Canadian Institute for Advanced Research (CIFAR) programs [to Y.B.]. Computational resources were provided by Calcul Québec and Compute Canada.

Conflict of Interest: none declared.

References

Mariani,V. et al. (2013) IDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. Bioinformatics, 29, 2722–2728.
Deep convolutional networks for quality assessment of protein folds


