Bioinformatics, 34(23), 2018, 4046–4053
doi: 10.1093/bioinformatics/bty494
Advance Access Publication Date: 19 June 2018
Original Paper

Structural bioinformatics

Deep convolutional networks for quality assessment of protein folds

Georgy Derevyanko[1,*], Sergei Grudinin[2], Yoshua Bengio[3,†] and Guillaume Lamoureux[1,*]

[1]Department of Chemistry and Biochemistry and Centre for Research in Molecular Modeling (CERMM), Concordia University, Montréal, Québec H4B 1R6, Canada, [2]Inria, Université Grenoble Alpes, CNRS, Grenoble INP, LJK, Grenoble 38000, France and [3]Department 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' (EMA) (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 physicochemical 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) uses a sligthly more raw input, consisting of the eigenvectors of the Cα-to-Cα distance matrix. The model itself is an autoencoder (Hinton and Salakhutdinov, 2006) trained to classify the structures into either 'near native' or 'not near native'.

More in line with the 'end-to-end' spirit of deep learning, methods using as input a 3D representation of the structure have been developed to score protein-ligand poses (Ragoza *et al.*, 2017; Wallach *et al.*, 2015), to predict ligand-binding protein pockets (Jiménez *et al.*, 2017) and to predict the effect of a protein mutation (Torng and Altman, 2017). The molecules of interest are treated as 3D objects represented on a grid and the predictions are obtained from that information only. While a rigorous comparison of these methods is not always possible, they appear to improve on the state of the art: both AtomNet (Wallach *et al.*, 2015) and the 3D convolutional neural network by Ragoza *et al.* (2017) perform consistently better than either Smina (Koes *et al.*, 2013) or AutoDock Vina (Trott and Olson, 2010). For small molecules, Schütt *et al.* (2017a,b), and Smith *et al.* (2017) have recently developed deep neural networks to predict the molecular energy of a variety of chemical compounds in various conformations (or even various isomeric states). These models, intended to be used as universal force fields, are trained on ab initio quantum energies and forces, and use only the nuclear charges and the interatomic distance matrix as input.

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 (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

$$\rho(r) = \begin{cases} e^{-r^2/2} & \text{if } r \leq 2.0\,\text{Å} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The atomic density is projected to the grid corresponding to its atom type. Each grid has a resolution of 1 Å and has $120 \times 120 \times 120$ 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

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 number of processing steps and their complexity, while staying within hardware constraints. The number of 3D convolutional layers in each block increases approximately geometrically (for a total of eight convolutions).

Table 1. Atom types used in this work

Type	Description	Atoms
1	Sulfur/selenium	CYS:SG, MET:SD, MSE:SE
2	Nitrogen (amide)	ASN:ND2, GLN:NE2, backbone N (including N-terminal)
3	Nitrogen (aromatic)	HIS:ND1/NE1, TRP:NE1
4	Nitrogen (guanidinium)	ARG:NE/NH*
5	Nitrogen (ammonium)	LYS:NZ
6	Oxygen (carbonyl)	ASN:OD1, GLN:OE1, backbone O (except C-terminal)
7	Oxygen (hydroxyl)	SER:OG, THR:OG1, TYR:OH
8	Oxygen (carboxyl)	ASP:OD*, GLU:OE*, C-terminal O, C-terminal OXT
9	Carbon (sp2)	ARG:CZ, ASN:CG, ASP:CG, GLN:CD, GLU:CD, backbone C
10	Carbon (aromatic)	HIS:CG/CD2/CE1, PHE:CG/CD*/CE*/CZ, TRP:CG/CD*/CE*/CZ*/CH2, TYR:CG/CD*/CE*/CZ
11	Carbon (sp3)	ALA:CB, ARG:CB/CG/CD, ASN:CB, ASP:CB, CYS:CB, GLN:CB/CG, GLU:CB/CG, HIS:CB, ILE:CB/CG*/CD1, LEU:CB/CG/CD*, LYS:CB/CG/CD/CE, MET:CB/CG/CE, MSE:CB/CG/CE, PHE:CB, PRO:CB/CG/CD, SER:CB, THR:CB/CG2, TRP:CB, TYR:CB, VAL:CB/CG*, backbone CA

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

Each 3D convolutional layer takes N input density maps f^{in} and transforms them using M filters F according to the following formula:

$$f_i^{\text{out}}(\mathbf{r}) = \sum_{j=1}^{N} \int F_i(\mathbf{r}-\mathbf{r}') \cdot f_j^{\text{in}}(\mathbf{r}')\ d\mathbf{r}', \quad \forall i \in [1, M] \tag{2}$$

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

$$f_i^{\text{out}}(\mathbf{r}) = \begin{cases} f_i^{\text{in}}(\mathbf{r}) & \text{if } f_i^{\text{in}}(\mathbf{r}) \geq 0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in [1, M] \tag{3}$$

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'):

$$\hat{f}_k^{\text{in}}(\mathbf{r}) = \frac{f_k^{\text{in}}(\mathbf{r}) - \mu_{\text{B}}(\mathbf{r})}{\sqrt{\sigma_{\text{B}}^2(\mathbf{r}) + \epsilon}}, \quad \forall k \in [1, N_{\text{B}}] \tag{4}$$

where $\mu_{\text{B}}(\mathbf{r})$ is the mean of all $f^{\text{in}}(\mathbf{r})$ maps from the batch (calculated at each position $\mathbf{r}$) and $\sigma_{\text{B}}^2(\mathbf{r})$ 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:

$$f_k^{\text{out}}(\mathbf{r}) = \gamma \hat{f}_k^{\text{in}}(\mathbf{r}) + \beta, \quad \forall k \in [1, N_B] \tag{5}$$

Parameters γ and β 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 $\mathbf{x}_{\text{in}}$ as follows:

$$\mathbf{x}_{\text{out}} = W \cdot \mathbf{x}_{\text{in}} + \mathbf{b} \tag{6}$$

where W is a rectangular matrix and $\mathbf{b}$ is a vector, learned during the training. Each output vector is then transformed by a ReLU layer.

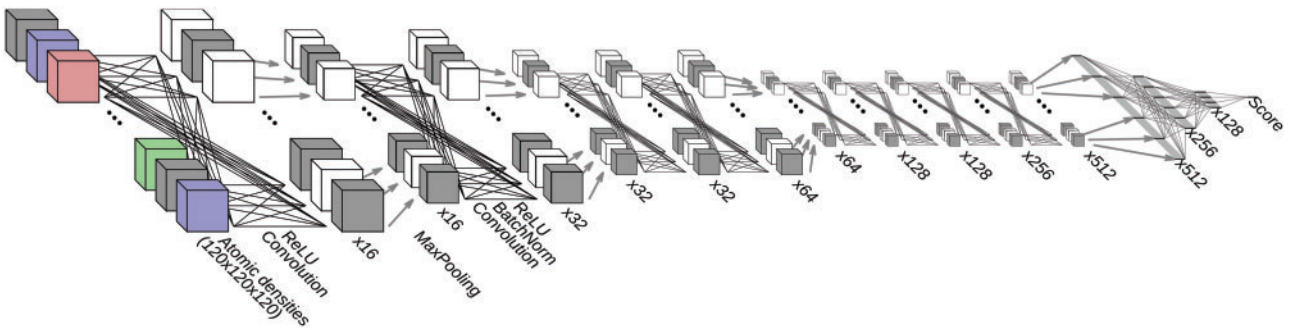

Fig. 1. Schematic representation of the convolutional neural network architecture used in this work. Unless otherwise specified, line connections across boxes denote the consecutive application of a 3D convolutional layer ('Convolution'), a batch normalization layer ('BatchNorm') and a ReLU layer. Grey arrows between boxes denote maximum pooling layers ('MaxPooling'). Labels '$\times M$' 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

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_{ij} be the ordering coefficient of decoys i and j, equal to $+1$ if $\text{GDT\_TS}_i \leq \text{GDT\_TS}_j$ 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:

$$L_{ij} = w_{ij} \max[0, 1 - y_{ij} \cdot (s_i - s_j)] \tag{7}$$

The coefficient w_{ij} is defined so that decoys with similar scores are excluded from the training: w_{ij} is one if $|\text{GDT\_TS}_i - \text{GDT\_TS}_j| > 0.1$ 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:

$$L = \frac{1}{N_B^2} \sum_{i=1}^{N_B} \sum_{j=1}^{N_B} L_{ij} \tag{8}$$

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:

$$\text{Loss} = \max_i(\text{GDT\_TS}_i) - \text{GDT\_TS}_{\text{argmin}_i(s_i)} \tag{9}$$

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.

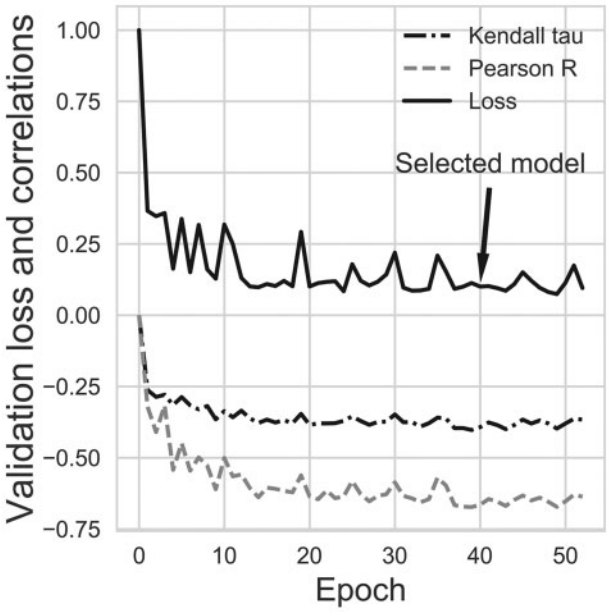

Fig. 2. Evaluation loss (Eq. 9), Kendall τ and Pearson R coefficients evaluated on the validation subset during the training procedure. One epoch corresponds to a cycle over all targets in the training subset. Models are saved every 10 epochs and the arrow shows the minimum validation loss for which 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 one pass through all targets in the dataset. The decoys are sampled in a homogeneous way, by dividing all decoys of a given target into N_B bins according to the value of their GDT_TS and by picking one decoy from each bin at random. Precisely, decoy i belongs to bin number

$$1 + \left\lfloor N_B \times \frac{\text{GDT\_TS}_i - \min(\text{GDT\_TS})}{\max(\text{GDT\_TS}) - \min(\text{GDT\_TS})} \right\rfloor \tag{10}$$

where max(GDT_TS) and min(GDT_TS) are computed on all decoys of the chosen target. If a bin is empty, the decoy is picked from another non-empty bin chosen at random. The order of targets and the order of decoys in the bins are shuffled at the end of each epoch.

Decoy structures are randomly rotated and translated each time they are used as input. The rotations are sampled uniformly (Shoemake, 1992) and the translation are chosen such that the translated protein fits inside the $120\,\text{Å} \times 120\,\text{Å} \times 120\,\text{Å}$ input grid (see text in Supplementary Material for details).

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 τ 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.)

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

Data	Loss (Eq. 9)	Pearson R	Spearman ρ	Kendall τ
Training subset	0.146	–0.71	–0.61	–0.45
Validation subset	0.135	–0.71	–0.59	–0.44

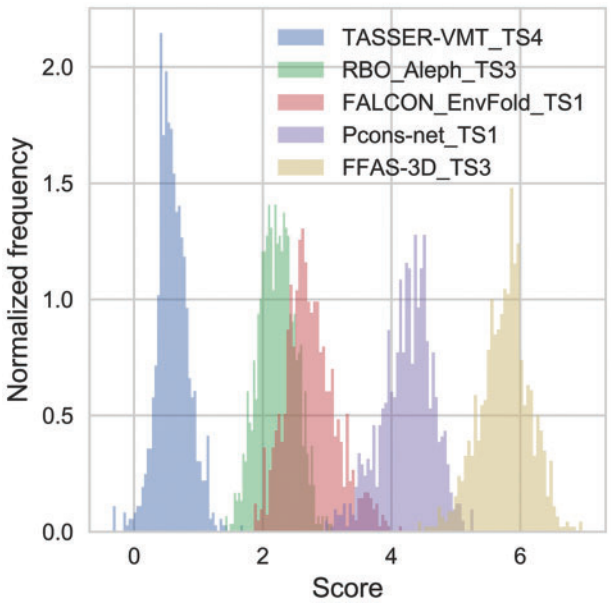

Fig. 3. Distributions of the *s* scores of five decoys for target T0832 under random translations and rotations. A lower score represents a higher quality

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 (Olechnovič 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)

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

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 multimeric 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 (Olechnovič 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 1383 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

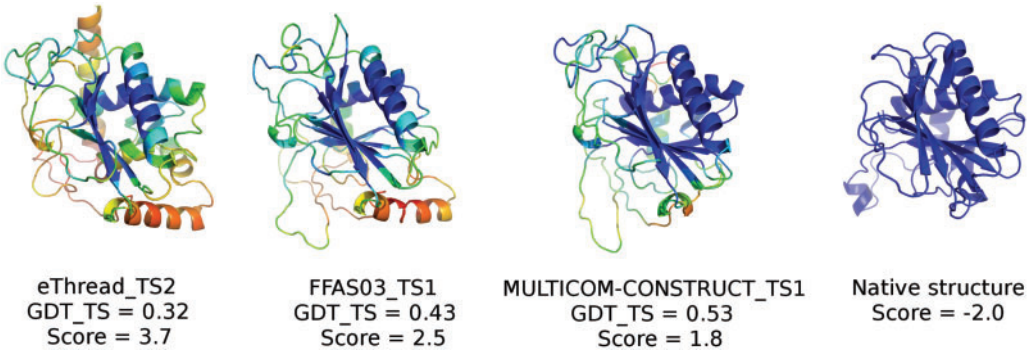

Fig. 4. Output of the Grad-CAM analysis for layer 10 of the network projected on four decoys of target T0786 (PDB code 4QVU). The values are represented using the 'rainbow' color scheme. The decoys are arranged in increasing quality from left to right, with the native structure on the right. Each decoy is aligned on the native structure and viewed in the same orientation

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 lDDT 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).

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)

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

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

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).

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 intepretability 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.L. 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

Alipanahi,B. *et al.* (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.*, **33**, 831–838.

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Bau,D. *et al.* (2017) Network Dissection: Quantifying Interpretability of Deep Visual Representations. *arXiv e-prints 1704.05796*.

Bengio,Y. *et al.* (1990) A neural network to detect homologies in proteins. In: *Advances in Neural Information Processing Systems*, pp. 423–430.

Cao,R. *et al.* (2016) DeepQA: improving the estimation of single protein model quality with deep belief networks. *BMC Bioinformatics*, **17**, 495.

Cheng,H. *et al.* (2014) ECOD: an evolutionary classification of protein domains. *PLoS Comput. Biol.*, **10**, e1003926.

Deng,H. *et al.* (2016) 3DRobot: automated generation of diverse and well-packed protein structure decoys. *Bioinformatics*, **32**, 378–387.

Dill,K.A. and MacCallum,J.L. (2012) The protein-folding problem, 50 years on. *Science*, **338**, 1042–1046.

Elofsson,A. *et al.* (2017) Methods For Estimation of Model Accuracy In CASP12. *arXiv e-prints 143925*.

Finn,R.D. *et al.* (2015) HMMER web server: 2015 update. *Nucleic Acids Res.*, **43**, W30–W38.

Finn,R.D. *et al.* (2016) The Pfam protein families database: towards a more sustainable future. *Nucleic Acids Res.*, **44**, D279–D285.

Gong,Y. *et al.* (2013) Deep convolutional ranking for multilabel image annotation. *arXiv e-prints 1312.4894*.

Haas,J. *et al.* (2013) The Protein Model Portal—a comprehensive resource for protein structure and model information. *Database*, **2013**, bat031.

Hinton,G.E. and Salakhutdinov,R.R. (2006) Reducing the dimensionality of data with neural networks. *Science*, **313**, 504–507.

Hinton,G.E. *et al.* (2006) A fast learning algorithm for deep belief nets. *Neural Comput.*, **18**, 1527–1554.

Honig,B. and Nicholls,A. (1995) Classical electrostatics in biology and chemistry. *Science*, **268**, 1144–1149.

Huang,S.-Y. and Zou,X. (2006) An iterative knowledge-based scoring function to predict protein–ligand interactions: I. Derivation of interaction potentials. *J. Comput. Chem.*, **27**, 1866–1875.

Huang,S.-Y. and Zou,X. (2008) An iterative knowledge-based scoring function for protein–protein recognition. *Proteins*, **72**, 557–579.

Ioffe,S. and Szegedy,C. (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv e-prints 1502.03167*.

Jiménez,J. *et al.* (2017) DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics*, **33**, 3036–3042.

Jing,X. *et al.* (2016) Sorting protein decoys by machine-learning-to-rank. *Sci. Rep.*, **6**, 31571.

Joachims,T. (2002) Optimizing search engines using clickthrough data. In: *Proceedings of the Eighth ACM SIGKDD International Conference on KNOWLEDGE DISCOVERY and Data Mining*. Association for Computing Machinery, pp. 133–142.

Kingma,D. and Ba,J. (2014) Adam: a method for stochastic optimization. *arXiv e-prints 1412.6980*.

Koes,D.R. *et al.* (2013) Lessons learned in empirical scoring with Smina from the CSAR 2011 benchmarking exercise. *J. Chem. Inf. Model.*, **53**, 1893–1904.

Krivov,G.G. *et al.* (2009) Improved prediction of protein side-chain conformations with SCWRL4. *Proteins*, **77**, 778–795.

Krizhevsky,A. *et al.* (2012) ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Kryshtafovych,A. *et al.* (2016) Methods of model accuracy estimation can help selecting the best models from decoy sets: assessment of model accuracy estimations in CASP11. *Proteins*, **84**, 349–369.

Leaver-Fay,A. et al. (2011) Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. *Methods Enzymol.*, **487**, 545–574.

LeCun,Y. *et al.* (1989) Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, **1**, 541–551.

LeCun,Y. *et al.* (2015) Deep learning. *Nature*, **521**, 436–444.

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*, **23**, ii–iv.

Moult,J. *et al.* (2014) Critical assessment of methods of protein structure prediction (CASP)—round x. *Proteins*, **82**, 1–6.

Nguyen,S.P. *et al.* (2014) DL-PRO: A Novel Deep Learning Method for Protein Model Quality Assessment. In: *2014 International Joint Conference on Neural Networks (IJCNN)*.

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

Ragoza,M. *et al.* (2017) Protein–ligand scoring with convolutional neural networks. *J. Chem. Inf. Model.*, **57**, 942–957.

Ray,A. *et al.* (2012) Improved model quality assessment using ProQ2. *BMC Bioinformatics*, **13**, 224.

Schütt,K.T. *et al.* (2017a) MolecuLeNet: A Continuous-Filter Convolutional Neural Network for Modeling Quantum Interactions. *arXiv e-prints 1706.08566*.

Schütt,K.T. *et al.* (2017b) Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.*, **8**, 13890.

Selvaraju,R.R. *et al.* (2016) Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *arXiv e-prints 1610.02391*.

Shen,M-Y. and Sali,A. (2006) Statistical potential for assessment and prediction of protein structures. *Protein Sci.*, **15**, 2507–2524.

Shoemake,K. (1992) Uniform random rotations. In: Kirk,D. (ed.) *Graphics Gems III*. Academic Press Professional, Inc., San Diego, CA, pp. 124–132.

Smith,J.S. *et al.* (2017) ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.*, **8**, 3192–3203.

Stumpe,M.C. *et al.* (2011) Calculation of local water densities in biological systems: a comparison of molecular dynamics simulations and the 3D-RISM-KH molecular theory of solvation. *J. Phys. Chem. B*, **115**, 319–328.

Torng,W. and Altman,R.B. (2017) 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC Bioinformatics*, **18**, 302.

Trott,O. and Olson,A.J. (2010) AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.*, **31**, 455–461.

Uziela,K. *et al.* (2016) ProQ3: improved model quality assessments using Rosetta energy terms. *Sci. Rep.*, **6**, 33509.

Uziela,K. *et al.* (2017) ProQ3D: improved model quality assessments using deep learning. *Bioinformatics*, **33**, 1578–1580.

Wallach,I. *et al.* (2015) AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery. *arXiv e-prints 1510.02855*.

Worrall,D.E. *et al.* (2016) Harmonic Networks: Deep Translation and Rotation Equivariance. *arXiv e-prints 1612.04642*.

Xiong,H.Y. *et al.* (2015) The human splicing code reveals new insights into the genetic determinants of disease. *Science*, **347**, 1254806.

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

Zhang,J. and Zhang,Y. (2010) A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction. *PloS One*, **5**, e15386.

Zhang,Y. and Skolnick,J. (2004) Scoring function for automated assessment of protein structure template quality. *Proteins Struct. Funct. Bioinf.*, **57**, 702–710.

Zhou,H. and Zhou,Y. (2009) Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Sci.*, **11**, 2714–2726.

Zhou,J. and Troyanskaya,O.G. (2015) Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods*, **12**, 931–934.