Bioinformatics, 33(19), 2017, 3036–3042
doi: 10.1093/bioinformatics/btx350
Advance Access Publication Date: 31 May 2017
Original Paper

OXFORD

Structural bioinformatics

DeepSite: protein-binding site predictor using 3D-convolutional neural networks

J. Jiménez[1], S. Doerr[1], G. Martínez-Rosell[1], A. S. Rose[2] and G. De Fabritiis[1,3,*]

[1]Computational Biophysics Laboratory (GRIB-IMIM), Universitat Pompeu Fabra, Barcelona Biomedical Research Park (PRBB), 08003 Barcelona, Spain, [2]San Diego Supercomputer Center, UC San Diego, MC 0505, 9500 Gilman Drive, La Jolla, CA 92093-0505. USA and [3]ICREA, 08010 Barcelona, Spain

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

Received on January 10, 2017; revised on May 9, 2017; editorial decision on May 29, 2017; accepted on May 30, 2017

Abstract

Motivation: An important step in structure-based drug design consists in the prediction of druggable binding sites. Several algorithms for detecting binding cavities, those likely to bind to a small drug compound, have been developed over the years by clever exploitation of geometric, chemical and evolutionary features of the protein.

Results: Here we present a novel knowledge-based approach that uses state-of-the-art convolutional neural networks, where the algorithm is learned by examples. In total, 7622 proteins from the scPDB database of binding sites have been evaluated using both a distance and a volumetric overlap approach. Our machine-learning based method demonstrates superior performance to two other competitive algorithmic strategies.

Availability and implementation: DeepSite is freely available at www.playmolecule.org. Users can submit either a PDB ID or PDB file for pocket detection to our NVIDIA GPU-equipped servers through a WebGL graphical interface.

Contact: gianni.defabritiis@upf.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

One of the first steps in the structure-based drug design (SBDD) pipeline is identifying viable druggable binding sites on the target protein. This task is defined as identifying and delimiting protein cavities, potentially at the surface that are likely to bind to a small compound. Over the years, plenty of methods have been developed for this particular task, most of which rely on four different and complimentary approaches: geometric, evolutionary, energetic or statistical. These approaches typically exploit known binding site properties. fpocket (Le Guilloux *et al.*, 2009), for instance, uses the concept of alpha-spheres (spheres that contact four atoms but contain none) introduced by the MOE Site Finder (Edelsbrunner *et al.*, 1995) methodology and Voronoi tesellation to find cavities. POCKET (Levitt and Banaszak, 1992) and LigSite (Hendlich *et al.*, 1997) search for protein-solvent-protein events on a determined enclosing of the protein. Pocket-Picker (Weisel *et al.*, 2007) uses a uniform grid of points which get assigned a buriedness index, while SURFNET (Laskowski, 1995) works by placing gap spheres on a 3D grid between all pairs of atoms midway between their Van der Waals surfaces, reducing their raddii until they stop intersecting with any atom and finally keeping only those surpassing a given size threshold. PASS (Brady and Stouten, 2000) computes a coating of probe spheres with protein as substrate, with additional layers of probes accreted onto previously found probe spheres, keeping in the end low solvent exposure spheres at the point when an accretion layer no longer produces new probe spheres. Concavity (Capra *et al.*, 2009) additionally makes use of evolutionary sequence conservation information in combination with other structure-based methods.

Over the last few years, deep convolutional neural networks (DCNNs) have become the de-facto model for computer vision

applications. They have been proven to provide state-of-the-art results in many artificial intelligence problems, such as the ImageNet challenge classification task (Krizhevsky *et al.*, 2012). Reinforcement Learning is also a subfield where these models are gaining traction, especially by the use of the Deep Q-Network (DQN) architecture (Mnih *et al.*, 2015). Generative modeling using DCNN architectures are drawing increasing attention as well (Goodfellow *et al.*, 2014; Radford *et al.*, 2015). Furthermore, the application of these Deep Learning architectures is becoming increasingly common in computational biology (Angermueller *et al.*, 2016). In this work, we propose a machine learning algorithm based on DCNNs for predicting ligand-binding sites in proteins, and show that given enough training data, they are able to capture binding site characteristics and can outperform other two competitive algorithms by providing an extensive test set based on more than 7000 proteins of the scPDB database. DeepSite is freely available for online use at www.playmolecule.org.

2 Materials

For the work described here, we make use of the scPDB v.2013 database (Desaphy *et al.*, 2015). This database is an up to date selection of high-quality, non-redundant druggable binding sites extracted from the Protein Data Bank (PDB), focusing mostly on small synthetic or natural ligands. It contains information for the pocket, the corresponding ligand and its binding mode. The 2013 version of scPDB comprises 9283 different binding sites, most of them used in the present work. The amount of data provided in the database makes it ideal for a machine-learning approach, and especially a deep neural network one, which typically requires large amounts of examples for training.

2.1 Data selection and splitting

The scPDB database annotations, available as a CSV file, were used for selecting the subset of structures used in this experiment. In more detail, out of a total of 9190 structures found in the file, 12 were discarded as they contained multiple erroneous entries. Furthermore, the scPDB database also provides a clustering of binding sites for all PDB structures by Uniprot entry. To exclude identical binding sites in the training and test sets for an unbiased evaluation of the performance of our method, 1556 structures without clustering information were removed resulting in a final set of 7622 structures for analysis. These structures are listed in the Supplementary Material accompanying this article.

Using the structures selected after the initial filtering, we carry out a $k = 10$-fold cross-validation, splitting the structures in training sets of 6860 structures and test sets of 762 structures, approximately, based on the scPDB clusters i.e. we train 10 different models with the same architecture on each of the training/test-set splits. The separation method ensures that the same protein pocket (possibly existing in different PDB structures) does not occur on both the training and test set in a split, therefore limiting the possibility of over-fitting. In particular, we checked whether training and test sets were dissimilar enough in each of the splits by using directly the Shaper similarity metric matrix (Desaphy *et al.*, 2012) provided by the scPDB database which reports a structural distance between binding pockets. No identical pair of binding sites was found on both training and test set in any split. Detailed per split similarity distributions can be consulted in Supplementary Figure S1. To the best of our knowledge, this provides the most extensive comparison of binding site detector performance in terms of number of tested proteins to date.

3 Methods

In this section, we will first describe the 3D structural descriptors computed as input for the DCNN model. Then we provide an introduction to CNNs and the most common types of layers used during their architecture design. Training and optimization details of the final chosen architecture are discussed last.

3.1 Descriptor computation and labeling

We treat protein structures from a computer vision perspective, as if they were 3D images. Coordinates of this 3D image are defined to span the bounding box of the protein plus a buffer of 8 Å to account for pockets located close to its edges. The 3D image is then discretized into a grid of $1 \times 1 \times 1$ Å^3 sized voxels. For each of said voxels, a compendium of atomic-based pharmacophoric properties is defined. Voxel occupancies are defined with respect to the atoms in the protein depending on their excluded volume and other seven atom properties: hydrophobic, aromatic, hydrogen bond acceptor or donor, positive or negative ionizable and metallic. These are called *channels*, to draw a comparison to computer vision, where an image can be represented with three different color arrays: red, green and blue. The AutoDock 4 (Morris *et al.*, 2009) atom types found in Table 1 were used with the rules of Table 2 to assign each atom to a specific channel. Non-protein atoms are filtered out of the calculation. Atom occupancies were calculated by taking the simplest approximation for the pair correlation function defined by

$$g(r) = \exp\left(-\beta V(r)\right), \tag{1}$$

where $V(r) = \epsilon(r_{\text{vdw}}/r)^{12}$ is the repulsive component of a Lennard-Jones potential and r_{vdw} is the Van der Waals atom radius.

Table 1. AutoDock 4 atom types

Element	Description
C	Non H-bonding aliphatic carbon
A	Non H-bonding aromatic carbon
NA	Acceptor 1 H-bond nitrogen
NS	Acceptor S Spherical nitrogen
OA	Acceptor 2 H-bonds oxygen
OS	Acceptor S Spherical oxygen
SA	Acceptor 2 H-bonds sulfur
HD	Donor 1 H-bond hydrogen
HS	Donor S Spherical hydrogen
MG	Non H-bonding magnesium
ZN	Non H-bonding zinc
MN	Non H-bonding manganese
CA	Non H-bonding calcium
FE	Non H-bonding iron

Table 2. Property-atom type (AutoDock 4) correspondence used for Deepsite's 3D descriptor computation

Property	Rule
Hydrophobic	atom type C or A
Aromatic	atom type A
Hydrogen bond acceptor	atom type NA or NS or OA or OS or SA
Hydrogen bond donor	atom type HD or HS with O or N partner
Positive ionizable	atom with positive charge
Negative ionizable	atom with negative charge
Metal	atom type MG or ZN or MN or CA or FE
Excluded volume	all atom types

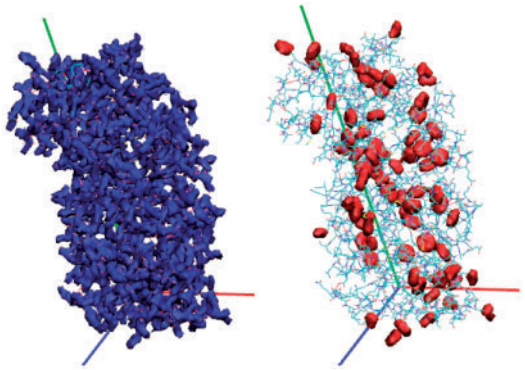

Fig. 1. Example of descriptor computation output for the hydrophobic and aromatic channels, respectively for PDB ID 4NIE

For simplicity we take the same ϵ for every atom type and such that $\beta\epsilon = 1$. The single atom occupancy estimate is therefore given by

$$n(r) = 1 - \exp\left(-(r_{\text{vdw}}/r)^{12}\right). \tag{2}$$

Finally, the occupancy for each property of each voxel is calculated as the maximum of the contribution of all atoms belonging to that channel at its center. We provide pseudo-code for the calculation of these descriptors in Algorithm 1. For user convenience, these descriptors are implemented and available in the high-throughput molecular dynamics (HTMDs) (Doerr *et al.*, 2016) Python package for molecular manipulation and computing. Examples of descriptor computation for the hydrophobic and aromatic channels can be seen in Figure 1. Similar plots for the rest of the channels can be found in Supplementary Figure S2.

Subgrids of $16 \times 16 \times 16$ voxels out of these arrays are then sampled, defining smaller protein areas with local properties. We use the fact that for all proteins in the database we know the location of its corresponding binding site to label each of the subgrids as positive, if its geometric center is closer than 4 Å to the pocket geometric center and negative otherwise. The threshold was chosen taking into account that it is common in binding site literature to determine a prediction successful if it is closer than our given threshold to the real binding site (Chen *et al.*, 2011). We then design a DCNN which uses as input the various features (channels) of the subgrids and outputs the binding site label probability, in the hope of capturing local patterns in the structure of the grid that may help characterize binding pockets.

Algorithm 1. Descriptor pseudo-code computation.

```
function Occupancy(atomCoords, centerCoords, radii, channels)
    for each atom A in protein do
        a ← atomCoords_A
        h ← channels_A
        r_vdw ← radii_A
        for each center c in centerCoords do
            r ← L2Dist(c, a)
            x ← r_vdw / r
            n ← 1 − exp(−x^12)
            for each channel p in h do
                O_c,p ← max{n, O_c,p}
            end for
        end for
    end for
end function
```

3.2 DCNN architecture and training

We provide a small introduction to CNNs here to motivate the design choices made in DeepSite's internal architecture. For a more in-depth introduction to the topic, the reader is referred to (Goodfellow *et al.*, 2016). In principle, CNNs are very similar to regular, fully connected networks: they are organized in hierarchical layers, each of them performing a dot product of some inputs $\boldsymbol{x}$ with some learnable weights of the layer, adding a bias and applying some non-linear function f. The output of a particular neuron ϕ can be written as

$$\phi = f\left(\sum_i w_i x_i + b\right), \tag{3}$$

where we sum over all the outputs x_i of the preceding layer. One of the known problems with fully connected networks is that they do not tend to scale well to full-sized images. CNNs take advantage of the fact that their input contains latent local structure to arrange its architecture in a more sensible way: neurons are arranged spatially and they are locally connected to areas of the preceding layer, while sharing some weights. Different types of layers are typically used in CNNs, the most common ones being:

- **Convolutional layers**. They compute the output of neurons locally connected to regions in the previous layer, by means of a dot product of their weights and the region they are connected to. In practice, one can control several layer hyper-parameters, such as the number of filters in the layer (the equivalent to neurons in fully connected layers), the spatial extent of each of these filters (also called filter size), and the stride and padding when computing the dot-product.
- **Activation layers**. They apply element-wise some non-linear function f to the output of a convolutional layer. Common choices are ReLU (rectifier unit) or ELU (exponential linear unit).
- **Pooling layers**. They perform downsampling according to some function (maximum, average…) over spatial dimensions. It is common for architectures to include pooling layers after successive convolutional layers to reduce the amount of parameters and therefore computation in the network.
- **Other layers**. The extent of types of layers and its hyperparameters that can be included in a modern DCNN architecture is wide. Special mention to the Dropout layer, which randomly sets neuron activations to 0 during the training stage, in practice behaving as a model regularizer.

Common choices and standard practices in DCNN literature (Iandola *et al.*, 2016; Szegedy *et al.*, 2016) motivate DeepSite's network design, as shown in Figure 2, where the full specification is provided. Our architecture comprises four convolutional layers with max pooling and dropout after every two convolutional layers, followed by one regular fully connected layer. All layers but the last use ELU activation function. The output of our network however uses a sigmoid activation function, with values ranging from 0 to 1, representing, once trained, the probability of a subgrid being close to a binding pocket. The architecture of our model resembles most VGG-like networks (Simonyan and Zisserman, 2015). The latter stacks several simple convolutional layers, then applying pooling operations after ReLU non-linearity, with increasing number of filters each time these operations are computed. Overall, our goal was to

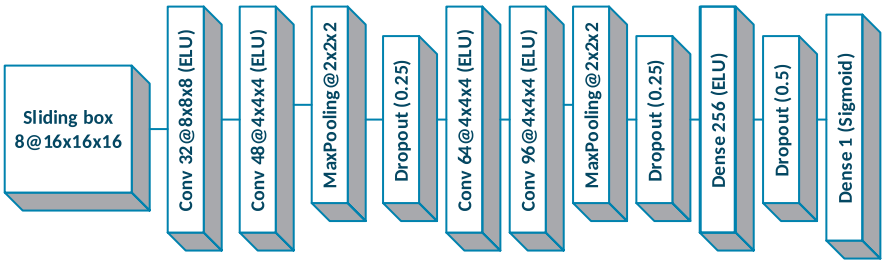

Fig. 2. DeepSite's internal DCNN architecture. The input is comprised of eight property channels of size 16 Å^3. Several convolutional layers with filter size of 8, 4 and 2 reduce the dimensionality of the subgrid before flattening in the 1D dense layers

design a simple enough DCNN that could still capture binding site characteristics, by imitating these architectures. Since training data in this work are significantly scarcer than the datasets typically used in computer vision, we simplified our architecture accordingly by reducing its depth and number of learnable parameters.

For the training of DeepSite's network, we use subgrids sampled with a sliding window step of four voxels from the computed descriptors. This procedure causes heavy unbalanced labeling of subgrids, since most of them are not located close to a binding pocket, therefore forcing a classifier towards marking most instances as the majority class. As such, the positive/negative class ratio in our dataset is below 0.1%. In the machine learning community, this is commonly referred to as the class imbalance problem (Longadge *et al.*, 2013). Many remedies can be proposed in these situations, the most simple being undersampling the majority class to obtain a balanced 50/50 class ratio dataset. We apply this strategy in each of the $k = 10$ splits for training instances.

The number of design choices in a DCNN model is wide: number of filters, filter size, strides, padding, dropout rate or choice of activation function are some examples. Other important parameters (Bengio, 2012) are the learning rate of the optimizer, the batch size, the number of training epochs and the momentum schedule. As for the choice of optimizer, we found the most stable results using Adam (Kingma and Ba, 2014), with the default learning rate of 0.001 and a batch size of 128 samples. We explored also changing batch sizes (32, 64, 256), with no apparent differences. As for the training epochs, we found that 25 were more than enough for the network to achieve reasonable convergence. The momentum schedule was set to the default in the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$). Training time for all 10 splits is ~8 h with tensor operations accelerated on GPU. Once the model is trained on a subset of data, it can be used to perform point predictions for every single voxel of a given protein. This yields an entire volumetric map of predicted-binding sites of a protein. In order to distinguish different binding sites in the same protein, we use the Mean-Shift (Comaniciu and Meer, 2002) clustering algorithm after filtering out voxels with low predicted probability P. Using this clustering procedure, allows us to provide both a point prediction of a single predicted pocket as the maximum probability voxel of the cluster and its volumetric map. From a practical point of view, we found this particular clustering algorithm well suited for the work described here, requiring little to none hyperparameter optimization, while being fast enough to be used in a production setting.

4 Experiments

As described in the Materials section, we used a $k = 10$-fold cross-validation scheme to validate our model. We have extensively tested our algorithm by defining two different and complimentary evaluation criteria: point prediction distance to the binding site and volumetric overlap. An evaluation from a subgrid classification point-of-view is provided in the Supplementary Material.

4.1 Evaluating pocket prediction performance

We use the following evaluation metrics in order to evaluate whether DeepSite is able to correctly locate the annotated-binding site per test protein:

- **Distance to the center of the binding site** (DCC). This metric considers a prediction successful if a point prediction of the pocket is closer than a given distance threshold to the geometric center of the real-binding site. Values ranging between 4 and 20 Å are typically used for success rate plots. This evaluation ignores altogether the shape of the predicted pocket.
- **Discretized volumetric overlap** (DVO). For this metric, we discretize protein space into $1 \times 1 \times 1$ Å^3 voxels, and consider the convex hulls determined by both the real and predicted-binding site volume. We then compute a Jaccard Index, defined by

$$J = \frac{\#|V_r \cap V_p|}{\#|V_r \cup V_p|}, \tag{4}$$

where V_r and V_p are the set of $1 \times 1 \times 1$ Å^3 voxels that fall inside the convex hull of the real and predicted binding pocket respectively. This measure takes into account both pocket shape and size of the pocket prediction. The average of this measured is considered across all different test splits.

We compare DeepSite against fPocket and Concavity as both methods provided state-of-the-art performance in previous studies (Chen *et al.*, 2011). For both metrics described above, only the top n pockets in each method is considered, where n is the number of annotated-binding sites per test protein. Averaged over folds results for the DCC metric can be seen in Figure 3. In particular, these results show that DeepSite is able to provide more accurate binding site point predictions than the other two methods. Individual split results for the DCC metric can be checked in Supplementary Figure S3.

For the volumetric overlap comparison, DeepSite also has to take into account one extra hyperparameter, the probability threshold P over which subgrids will be considered a binding pocket. This affects the way the posterior clustering algorithm behaves and the resulting volumetric shape. A plot with the average DVO measure using different probability thresholds for DeepSite is provided in Supplementary Figure S4. We find the optimum of this hyperparameter to be fairly similar across splits, suggesting DeepSite's predictions are fairly robust regardless of this choice. We finally chose $P = 0.4$ by averaging over splits, for which results are presented in Table 3. Choosing any other P threshold between 0.3 and 0.6 will

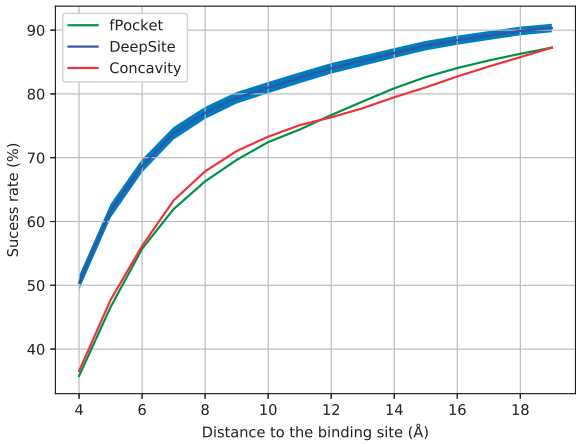

Fig. 3. DCC metric averaged over all 10 splits for DeepSite, fPocket and Concavity. Metric shown for distances ranging between 4 and 20 Å

Table 3. Evaluation of DeepSite volumetric performance using the DVO metric

	Average DVO	SD	*P*-value
DeepSite	**0.652**	0.129	—
fPocket	0.619	0.169	0
Concavity	0.489	0.172	0

Note: DVO is defined as a discretized Jaccard Index of the predicted and real pocket volumes. Mann–Whitney's U *P*-value shown for the unilateral test using DeepSite as baseline ($n = 7622$).

yield similar results, with reported DVO values of 0.648 and 0.646, respectively. A simple two sample Mann–Whitney's U unilateral test was performed to determine if the difference of location is significant. With a *P*-value of 0 for both comparisons, we conclude that DeepSite's median DVO is significantly larger than those of fPocket and Concavity.

Since at the core of DeepSite's model lies a DCNN architecture we additionally compute standard classification metrics of subgrids. We report accuracy, AUC (area under the ROC curve), sensitivity/recall, specificity, precision, F1 score, Matthew's correlation coefficient (MCC), log-loss and a scaled Cohen's Kappa coefficient (Brennan and Prediger, 1981) that takes into account class imbalance. For validation, both balanced and unbalanced class test sets are used. Many classification metrics such as precision, F1 or MCC are not informative on strongly imbalanced datasets, this being the reason why we provide both balanced and unbalanced evaluations as well as the scaled Cohen's Kappa coefficient. Individual split metrics and their average can be consulted in Supplementary Tables S1 and S2, respectively.

4.2 SCOPe fold performance

It is of interest to also evaluate how well our method performs on different groups of protein structures to see if there is a given bias to specific folds. The SCOPe (Fox *et al.*, 2014) database provides a curated classification of PDB structures, defining multiple hierarchical levels (class, fold, super-family, family, protein and species) moving from structure based classification (classes) to evolution based classification (species). Twelve different structural classes are defined (letters *a–l*) which are then subdivided into hundreds of different folds which are groupings of structural similar protein superfamilies (not to be confused with splits/folds of the *k*-fold cross-validation described earlier). In total 1431 different structural folds are defined in the SCOPe database, of which only 278 can be found in the scPDB dataset used in this study.

A plot with disaggregated performance using the DVO metric can be found in Figure 4. On average the performance of the method seems stable across different protein folds with some outliers which can be attributed to those proteins being strongly under-represented in the scPDB database (for instance fold e.37 comprising a single structure). We find no strong bias towards any individual fold, suggesting that the network is learning invariant properties and generalizing well, our original intention by learning atom-level properties. Another plot with the same disaggregation but using the DCC metric at distance $k = 4$ can be consulted in Supplementary Figure S5.

5 Implementation and usage

DeepSite was fully developed in Python using HTMD for all the molecular and descriptor related calculations. Implementation of 3D DCNNs was made using deep learning library Keras with the Theano (Bergstra *et al.*, 2010) tensor library which allows for GPU accelerated computing via CUDA/cuDNN. We made DeepSite freely available via a web server in the www.playmolecule.org domain. Users can submit jobs to our GPU-equipped servers and get a full volumetric map of the most probable-binding sites of the protein. The protein is interactively visualized and manipulated in three-dimensions via the NGL-Viewer (Rose and Hildebrand, 2015) interface with the user being able to modify a probability threshold for predictions. Point predictions of pocket sites as well as the volumetric map prediction can also be downloaded for use in other external applications.

All training and testing of the models described here were computed on a machine with an Intel(R) Core i7-3930K @ 3.20 GHz CPU, 64 GiB of RAM and a NVIDIA GeForce GTX 1080 graphics card. Depending on the size of the input protein, prediction time may vary, but most predictions ran within 2 min of wall clock time with the latest stable version of Theano (0.9). Regarding computational costs of our prediction algorithm, the internal DCNN model of DeepSite benefits greatly from parallel tensor computations performed on modern GPUs, since it is standard practice in the deep learning community to both train and test models using these. In practice, this means that DeepSite requires significantly more computational resources than other methods for the same task of predicting ligand-binding sites. Although it is technically possible to run CNN-related operations using CPU only, training and testing times increase substantially. Volumetric prediction example for PDB ID 3KS9 is provided in Figure 5. In particular, it can be appreciated that the main cavity predicted by DeepSite covers most of the annotated ligand (residue name Z99).

6 Discussion and future work

In this work, we have presented DeepSite, a pure machine-learning approach for predicting protein-ligand-binding sites. We have extensively tested our algorithm according to different criteria and conclude that it is able to outperform other existing state-of-the-art strategies while learning purely from examples and without encoding any problem specific knowledge. Furthermore, we believe the 3D descriptors described in this work are of very general nature, and have potential to be applied in other structural biology problems in conjunction with DCNN models. However, we also believe there is plenty of room for improvement, e.g. on encoded descriptor

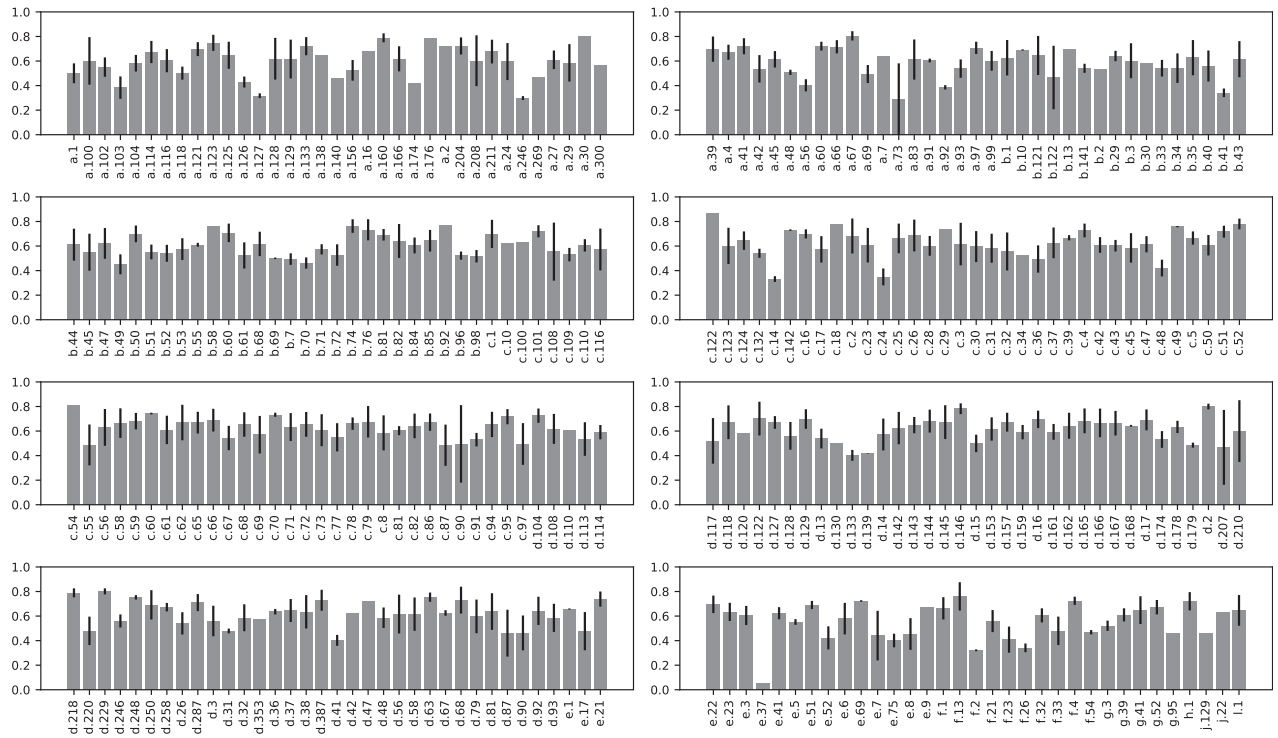

Fig. 4. Disaggregated average and SD of the DVO metric for SCOPe classification of proteins in the scPDB database

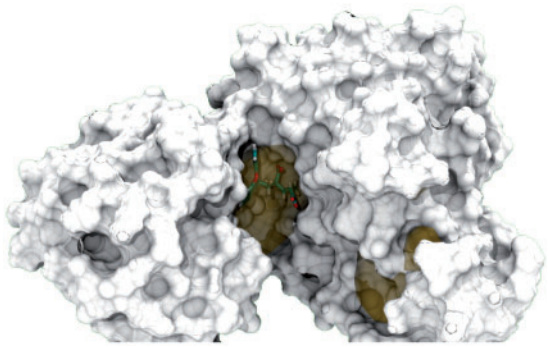

Fig. 5. Prediction example PDB ID 3KS9. Shaded volume represents DeepSite's cavity predictions, for which the largest covers most of the annotated ligand

properties. In particular it would be interesting to describe electronic density around the atoms to take into account the specific positions of atoms in the proteins and effects such as polarizability. In addition, water molecules could potentially also be important in binding site detection, since they play a critical role in protein-ligand recognition. A reasonable idea is to incorporate water molecules by having both hydrogen bond donor and acceptor channels on stable crystal waters.

Funding

G.D.F. acknowledges support from MINECO (BIO2014-53095-P) and FEDER.

Conflict of Interest: none declared.

References

Angermueller,C. *et al.* (2016) Deep learning for computational biology. *Mol. Syst. Biol.*, **12**, 878.

Bengio,Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU, pp. 437–478.

Bergstra,J. *et al.* (2010). Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, (Scipy), pp. 1–7.

Brady,G.P. and Stouten,P.F.W. (2000) Fast prediction and visualization of protein binding pockets with PASS. *J. Comput. Aid. Mol. Des.*, **14**, 383–401.

Brennan,R.L. and Prediger,D.J. (1981). Coefficient kappa: some uses, misuses, and alternatives. *Educational and Psychological Measurement*, **41**, 687–699.

Capra,J.A. *et al.* (2009) Predicting protein ligand binding sites by combining evolutionary sequence conservation and 3D structure. *PLoS Comput. Biol.*, **5**,

Chen,K. *et al.* (2011) A critical comparative assessment of predictions of protein-binding sites for biologically relevant organic compounds. *Structure*, **19**, 613–621.

Comaniciu,D. and Meer,P. (2002) Mean shift: a robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**, 603–619.

Desaphy,J. *et al.* (2012) Comparison and druggability prediction of protein-ligand binding sites from pharmacophore-annotated cavity shapes. *J. Chem. Inform. Model.*, **52**, 2287–2299.

Desaphy,J. *et al.* (2015) Sc-PDB: a 3D-database of ligandable binding sites-10 years on. *Nucleic Acids Res.*, **43**, D399–D404.

Doerr,S. *et al.* (2016) HTMD: high-throughput molecular dynamics for molecular discovery. *J. Chem. Theory Comput.*, **12**, 1845–1852.

Edelsbrunner,H. *et al.* (1995). Measuring proteins and voids in proteins. *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, **5**, 256–264.

Fox,N.K. *et al.* (2014) SCOPe: Structural Classification of Proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42(Database issue)**, D304–D309.

Goodfellow,I. *et al.* (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, **27**, 2672–2680.

Goodfellow,I. *et al.* (2016). *Deep Learning*. Cambridge MA: MIT Press.

Hendlich,M. *et al.* (1997) LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins. *J. Mol. Graph. Model.*, **15**, 359–363.

Iandola,F.N. *et al*. (2016). SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <1MB Model Size. *arXiv*, pp. 1–5.

Kingma,D. and Ba,J. (2014). Adam: a method for stochastic optimization. *International Conference on Learning Representations*, Banff, Canada, pp. 1–13.

Krizhevsky,A. *et al*. (2012). ImageNet classification with deep convolutional neural networks. *Advances In Neural Information Processing Systems*, Lake Tahoe, USA, pp. 1–9.

Laskowski,R.A. (1995) SURFNET: a program for visualizing molecular surfaces, cavities, and intermolecular interactions. *J. Mol. Graph*., **13**, 323–330.

Le Guilloux,V. *et al*. (2009) Fpocket: an open source platform for ligand pocket detection. *BMC Bioinformatics*, **10**, 168.

Levitt,D.G. and Banaszak,L.J. (1992) POCKET: a computer graphies method for identifying and displaying protein cavities and their surrounding amino acids. *J. Mol. Graph*., **10**, 229–234.

Longadge,R. *et al*. (2013) Class imbalance problem in data mining: review. *Int. J. Comput. Sci. Netw*., **2**, 83–87.

Mnih,V. *et al*. (2015) Human-level control through deep reinforcement learning. *Nature*, **518**, 529–533.

Morris,G.M. *et al*. (2009) Software news and updates AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J. Comput. Chem*, **30**, 2785–2791.

Radford,A. *et al*. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. 3rd International Conference on Learning Representations, San Diego, USA, abs/1511.06434, *arXiv*, pp. 1–15.

Rose,A.S. and Hildebrand,P.W. (2015) NGL viewer: a web application for molecular visualization. *Nucleic Acids Res*., **43**, W576–W579.

Simonyan,K. and Zisserman,A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICRL)*, San Diego, USA, pp. 1–14.

Szegedy,C. *et al*. (2016). Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, pp. 2818–2826.

Weisel,M. *et al*. (2007) PocketPicker: analysis of ligand binding-sites with shape descriptors. *Chem. Cent. J*., **1**, 7.