Journal of Molecular Graphics and Modelling 105 (2021) 107865

Contents lists available at ScienceDirect

Journal of Molecular Graphics and Modelling

journal homepage: www.elsevier.com/locate/JMGM

OctSurf: Efficient hierarchical voxel-based molecular surface representation for protein-ligand affinity prediction

Qinqing Liu [a], Peng-Shuai Wang [b], Chunjiang Zhu [a], Blake Blumenfeld Gaines [a], Tan Zhu [a], Jinbo Bi [a, c], Minghu Song [c, *]

[a] Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06279, USA
[b] Microsoft Research Asia, Beijing, China
[c] Department of Biomedical Engineering, University of Connecticut, Storrs, CT 06279, USA

ARTICLE INFO

Article history:
Received 17 November 2020
Received in revised form
3 February 2021
Accepted 4 February 2021
Available online 9 February 2021

Keywords:
Protein-Ligand affinity prediction
Convolution neural networks
3D volumetric representation
Octree
Molecular surface

ABSTRACT

Voxel-based 3D convolutional neural networks (CNNs) have been applied to predict protein-ligand binding affinity. However, the memory usage and computation cost of these voxel-based approaches increase cubically with respect to spatial resolution and sometimes make volumetric CNNs intractable at higher resolutions. Therefore, it is necessary to develop memory-efficient alternatives that can accelerate the convolutional operation on 3D volumetric representations of the protein-ligand interaction. In this study, we implement a novel volumetric representation, OctSurf, to characterize the 3D molecular surface of protein binding pockets and bound ligands. The OctSurf surface representation is built based on the octree data structure, which has been widely used in computer graphics to efficiently represent and store 3D object data. Vanilla 3D-CNN approaches often divide the 3D space of objects into equal-sized voxels. In contrast, OctSurf recursively partitions the 3D space containing the protein-ligand pocket into eight subspaces called octants. Only those octants containing *van der Waals* surface points of protein or ligand atoms undergo the recursive subdivision process until they reach the predefined octree depth, whereas unoccupied octants are kept intact to reduce the memory cost. Resulting non-empty leaf octants approximate molecular surfaces of the protein pocket and bound ligands. These surface octants, along with their chemical and geometric features, are used as the input to 3D-CNNs. Two kinds of CNN architectures, VGG and ResNet, are applied to the OctSurf representation to predict binding affinity. The OctSurf representation consumes much less memory than the conventional voxel representation at the same resolution. By restricting the convolution operation to only octants of the smallest size, our method also alleviates the overall computational overhead of CNN. A series of experiments are performed to demonstrate the disk storage and computational efficiency of the proposed learning method. Our code is available at the following GitHub repository: https://github.uconn.edu/mldrugdiscovery/OctSurf.

1. Introduction

In computer graphics, the shape of three-dimensional (3D) objects can be represented with various formats, such as point clouds, 3D meshes, and voxel occupancy grids. This 3D geometric data can be used by modern deep learning algorithms for 3D object recognition. For example, ShapeNets [1] and VoxNet [2] are two pioneering works that integrate the volumetric occupancy grid representation and 3D convolutional neural networks (CNNs) to detect 3D objects. Recent years have seen growing interest in the computational chemistry community to apply machine learning-based scoring functions, including emerging voxel-based CNN techniques, for drug discovery problems [3]. AtomNet [4] is the first deep CNN model developed for structure-based drug design. In AtomNet, the protein-ligand complexes are first discretized into a 3D grid of voxels with various physicochemical atomic properties. Then a 3D-CNN model is built to learn spatial features of protein-ligand interaction near the drug binding site area and prospectively predict the protein-ligand binding affinity. Since the introduction of AtomNet, several other studies have applied volumetric neural networks for molecular representation [5], protein-ligand

* Corresponding author. A.B. Bronwell Building, Room 217, 260 Glenbrook Road, Unit 3247, University of Connecticut, Storrs, CT 06269-3247, USA.
E-mail address: minghu.song@uconn.edu (M. Song).

https://doi.org/10.1016/j.jmgm.2021.107865
1093-3263/

affinity prediction [6–11], amino acid environment similarity analysis [12], binding site comparison or classification [13,14], and functional site detection [15,16]. In the above studies, 3D representations of drug binding sites or protein-ligand interaction are derived based on voxelization of the entire protein binding pocket or bound ligand complex. In molecular modeling, the molecular surface representation is another alternative approach that has been widely used to encode chemical features and geometric morphometrics of molecules involving protein recognition. Several geometric descriptors and fingerprints [17–21] had been developed to describe protein surface features. For example, Gainza et al. [22] proposed a geometric deep learning framework named MaSIF, to embed precomputed geometric and chemical input features on surface patches of proteins into 2D interaction fingerprints for protein pocket-ligand prediction, protein-protein interaction site prediction, and ultrafast scanning of protein surfaces. Very recently, Stelios et al. [23] proposed DeepSurf, a method that generates a new surface representation by using a set of localized voxel grids around certain sample points on the protein solvent-accessible surface and utilizing the ResNet architecture to predict the ligandability score of these surface points. These previous works inspire us to develop a novel surface-based volumetric representation for the prediction of protein-ligand binding affinity.

In most current voxel-based CNNs, a 3D object is represented by the traditional dense voxel method. This method involves a cubic space that contains the 3D object, such as a protein pocket with a bound ligand, being divided into equal-sized grid cells. The memory usage of dense voxel-based approaches has cubic complexity with respect to spatial resolution, which sometimes renders them intractable at high resolutions. Thus, although voxel-based CNNs have quickly gained popularity in many 3D object recognition applications, the volumetric representation suffers from a memory bottleneck. Additionally, for many 3D objects, especially 3D surfaces, such as the molecular surface, there can be a lot of empty voxels within the grid enclosing the 3D object. Traditional CNN algorithms designed for the dense data type cannot handle such sparse 3D data efficiently; therefore, a significant amount of computation will be wasted on the convolution operation and zero multiplications of those non-occupied voxels.

Several variants of volumetric CNNs have been proposed to address the sparsity issue of 3D data representation. For instance, Li et al. propose a resolution-agnostic approach, Field Probing Neural Network (FPNN) [24], which replaces the convolutional layers in CNNs with field probing layers and employs a small set of field probing filters to extract relevant features of the 3D object efficiently. Kuzminykh et al. propose using the wave transform to modify the 3D representation of the molecular structure and allow atoms to be extended to fill nearby voxels [5]. In addition, a number of alternative 3D data structures have been utilized to accelerate the convolution on 3D sparse data. Sparse 3D CNNs [25] and Submanifold Sparse CNNs [26,27] encode sparse 3D data using a hash table so that convolution will be computed only on the non-empty voxels. Octree-based CNNs, such as OctNet [28] and O–CNN [29], employ the octree representation to efficiently generate the high-resolution volumetric input for 3D-CNN. Octree is a tree data structure that has been widely used in computer graphics that recursively partitions a 3D bounding box into eight sub-volumes or octants. At each recursive iteration, only non-empty octants will be partitioned, so groups of empty octants are stored with less data at a lower resolution. Eventually the octants at the finest level of the octree encode the shape information of the 3D object with high detail. OctNet adopts a combination of the grid and octree data structure by using a maximal tree depth of three to encode the 3D representation of objects with binary features, and limits its convolutional network operations within the interior volume of 3D objects. Because the sparsity issue is more obvious when representing only the surface of a 3D object, O–CNN uses the pure octree structure to encode the surface information of 3D objects and limits 3D CNN operation to octants at the finest level of the octree. This strategy leads to a significant reduction of computation cost to $O(n^2)$, from $O(n^3)$ of the full voxel-based network (with n representing the number of voxels in a dimension). It enables deeper networks and higher resolution at regions of interest in new 3D-CNN architectures.

A direct volumetric representation of protein-ligand complexes or pockets will face the same sparsity issue as in 3D computer graphics. For comparative studies, all protein-ligand binding pockets with various sizes are often orientated within an initial cube that is large enough to enclose all training pockets. Hence, there will be a large number of unoccupied voxels within the enclosing grid when adopting the traditional dense voxel representation. For example, in Fig. 1, a protein-ligand binding pocket example (PDB ID: 4II9) is orientated in the center of a 3D bounding box. When the grid is partitioned into voxels with variate resolutions such as 1, 0.5, and 0.25 Å, less than 7% of voxels overlap with protein and ligand surfaces. As the resolution increases, the occupancy rate of these surface voxels becomes much lower. This observation motivates us to apply a solution tailored for these 3D sparsity issues to improve efficiency.

Here we propose a computational framework that combines a novel surface-based volumetric representation and efficient 3D-CNN operations to predict protein-ligand binding affinity. Most current 3D-CNN methods employ the traditional voxel representation localized at protein or ligand atoms to capture the intermolecular atomic interaction between the protein and ligand. Instead, we adopt an alternative way to characterize 3D molecular recognition at the surface level. Protein and ligand surfaces are transformed into the octree-based volumetric representation, which we call OctSurf, to delineate their surface contact and shape complementary. Compared to conventional CNN models with dense voxel input, the OctSurf voxel representation consumes much less memory at the same resolution. It also enables us to generate the volumetric representation of molecular surfaces at much higher resolutions. Moreover, by restricting CNN operations over octants near the molecular surface region, our implementation also significantly reduces the computational overhead of 3D-CNN modeling while maintaining similar, or even better, prediction accuracy. More details about our implementation will be discussed in the Material and Method or the Supplementary Materials sections.

Finally, several recent studies [30–34] have pointed out the high performance of some machine learning-based scoring functions or deep learning models for the protein-ligand affinity prediction may be due to hidden biases within the training set. For example, Sieg et al. [30] showed that ligands with a molecular weight larger than 500 in the DUD-E [35] data are dominated by active compounds. Such non-causal bias can be easily fit by machine learning models to distinguish active and decoy compounds, and result in the unnoticed “Clever Hans” phenomenon [36]. Chen et al. [31], Su et al. [32], and Shen et al. [33] have exemplified that the potential analogue bias or the protein similarity between training and test data can make a significant impact on the performance of machine learning-based models. More recently, Yang et al. [34] demonstrated that models derived from protein or ligand structures alone will still yield unexpectedly good performance on test datasets. These works remind us that it is necessary to evaluate the potential issue of bias and the generalizability when developing new machine learning-based models. Therefore, we carry out several additional control experiments to inspect the sensitivity of our generated 3D-CNN models to the decoyed or biased training sets.

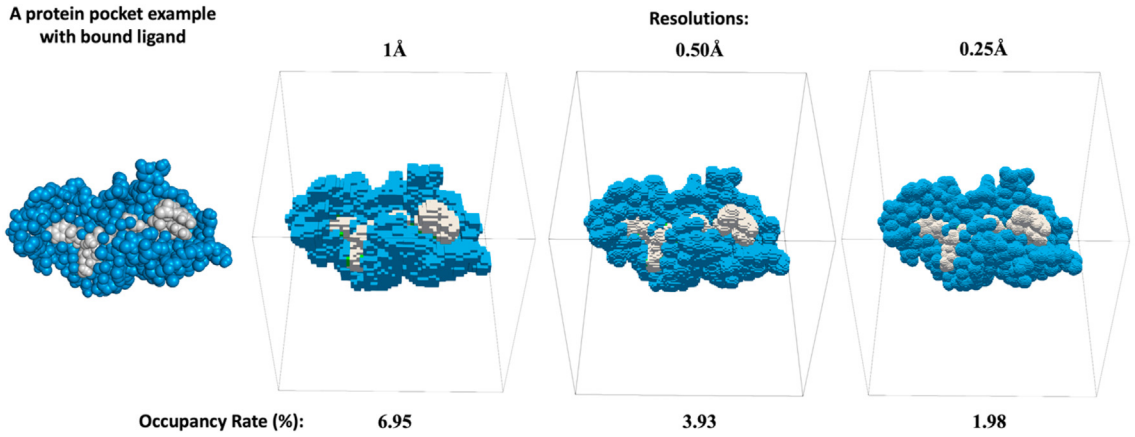

Fig. 1. The sparseness of protein-ligand surfaces in the volumetric representation. An example protein-ligand pocket is orientated in the center of a bounding box with a size of $64 \times 64 \times 64$ Å^3. As shown in the above figure, the occupancy rate decreases as the resolution increases. The occupancy rate here is defined as the ratio between the number of occupied voxels and the total number of voxels in the enclosing grid.

2. Materials and method

The overall strategy of our approach is illustrated in Fig. 2. First, 3D coordinates of the binding pocket and bound ligand pre-processed by the PDBbind database [37] are transformed into the 3D surface point cloud. Second, obtained 3D dot surfaces are originated at the center of a bounding box that is large enough to enclose all training protein-ligand pockets. Third, these surface point clouds are further rasterized into a digital volumetric representation, OctSurf, which is based on the Octree data structure. Finally, the OctSurf representations are fed into 3D-CNNs architectures for the protein affinity prediction, e.g., VGG and ResNet, by restricting the CNN operation on octants of the smallest size. We also implement the dense voxel version of molecular surface representation for the result comparison.

2.1. Datasets

Two major benchmark datasets of protein-ligand complexes, PDBbind [37] and Comparative Assessment of Scoring Functions (CASF-2016) [38], are employed here for model construction and assessment. The PDBbind set (v2018) includes two subsets, the general set with 11,663 complexes and the refined set with another 4463 complexes. The refined collection was selected based on a few quality filters regarding experimental binding data or potency, the crystallography resolution, the inherent nature of the protein-ligand interaction, etc. The CASF benchmark data, also known as the PDBbind-v2016 core set, consists of 285 high-quality protein-ligand complexes sampled from 57 protein targets within the PDBbind refined set. In this study, those 285 complexes in the core set are held out as the independent test dataset. We exclude protein complexes in the test set from the general and refine sets of PDBbind v2018 and randomly sampled 600 complexes out of the refined set as the validation set for fine-tuning hyperparameters. The remaining protein-ligand complexes in both the general and refined sets of PDBbind v2018 comprise the training set for 3D-CNN modeling. After we constructed our predictive models, we also tested them on newly deposited protein-ligand complexes in the latest version of PDBbind (v2019), which includes 1146 and 394 extra complexes in updated general and refined sets respectively. We used the high-quality refined set of PDBbind v2018 as the reference set to compare the storage space difference between our proposed OctSurf and the conventional dense voxel representations.

Similar to the published Pafnucy convolution model [39], we compute twenty-one atomic features to describe the properties of each surface point. These twenty-one features indicate the presence of protein or ligand atoms, their specific atom types (e.g., H, B, C, N, O, P, S, Se, halogen, metal), related physicochemical categories (hydrogen bond acceptor or donor, hydrophobic, aromatic, ring), specific atomic hybridization (e.g., sp1, sp2, and sp3), the connection valence with heavy atoms and hetero-atoms, partial atomic charge, and van der Waals atomic radius. In addition, to describe geometric information of a molecular surface, we calculate the normal vector of each surface point. Each normal vector of surface points includes three coordinate directions, which describe the surface curvatures and shape complementarity between protein and ligand surfaces. A total of twenty-four features are associated with each octant and are derived from the averaged feature values of surface points within this octant, as described in more detail in Section 2.2.2.

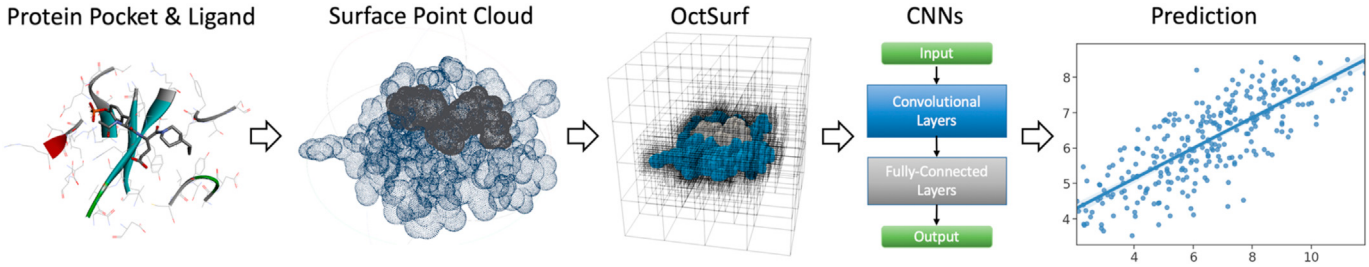

Fig. 2. The pipeline of our 3D-CNN implementation for the protein-ligand affinity prediction based on the OctSurf representation. Surface point clouds of binding pockets and bound ligands are rasterized into the octree-based volumetric representation, OctSurf, which are fed into the 3D-CNNs for binding affinity prediction.

During the 3D-CNN modeling process, we also implement on-the-fly data augmentation that rotates and translates original surface points along all axes to increase the number of training examples. To measure the performance of CNN models on the test set, we calculate the averaged prediction of 40 randomly augmented OctSurfs for each complex, then compare the average prediction with the true binding affinity label.

2.2. Molecular representation – OctSurf

2.2.1. The generation of van der Waals surface points

Preprocessed protein complex structures from PDBbind, including the pocket and ligand in pdb and mol2 file formats, are used to generate their the point cloud representations of their van der Waals surfaces. A point cloud is a collection of data points with geometric coordinates and other associated features that describe the shape of a 3D object. Each protein-ligand complex structure is placed within a 64 × 64 × 64 Å^3 bounding box centered on the geometric center of the ligand. In other 3D-CNN protein-ligand affinity prediction studies, the sizes of their bounding boxes are often in the range of 20 to 32 Å [7,39,40]. We conduct a statistical analysis of the pocket sizes over the entire PDBbind refined dataset, the size of 64 Å is chosen so that the bounding box is large enough to cover all protein-ligand pockets and we do not need to chop off any protein atoms. More details of the pocket size analysis are included in Supplementary Materials Figure S1. Point clouds for van der Waals surface of ligands and protein pockets are generated using an inhouse Java program based on the non-analytic Double Cubic Lattice Method (DCLM) [41] implemented in the CDK package [42]. Several structural and atomic property features are assigned to each surface point.

2.2.2. The octree data structure representing surface points

Octree is one of the most widely used space partitioning structures because of its flexible search hierarchy [28], where each node corresponds to a cubic search space called an octant. It has been used in other research areas, such as depth fusion [43], image rendering [44], and 3D reconstruction [45], to help improve memory efficiency. The key concept of the octree is to recursively partition non-empty cubic space into smaller cells by bisecting each cubic side. A significant amount of memory can be saved from the vacant cubic regions, which dominate the surface representation.

Fig. 3 is an illustration of the octree concept. Fig. 3(a) illustrates the recursive partition process of the octree algorithm at a different resolution or depth level. Only non-empty octants containing surface points from protein or ligand atoms are further partitioned at the next depth, while those empty octants remain intact. Fig. 3(b) and (c) exemplify the difference between the conventional dense voxel and our octree-based OctSurf representation. Compared to the conventional representation, the octree data structure uses many fewer voxels and thus reduces memory consumption. At the higher resolution, such a reduction will be substantial. The reconstructed molecular surface in around 0.1 Å resolution by the OctSurf is depicted in Fig. 3(f). It approximates the ground truth surface of the 1A1E protein-ligand complex, which is shown in Fig. 3(e). The Octsurf representation that shows the boundaries of all octants is shown in Fig. 3(g).

From an algorithm point of view, the two most popular strategies for constructing an octree are from the top-down [28] and from the bottom-up [29,30]. In the classical top-down way, as illustrated in Fig. 3(a), the octree starts from the bounding box that contains the 3D object as the root node (depth = 0), and then recursively subdivides all non-empty octants into child nodes in a breadth-first search process until reaching the desirable resolution at terminal leaf nodes. For example, to reach 1 Å resolution for a 64 × 64 × 64 Å^3 bounding box, we do this recursive step 6 times. Those octants that are not divided are named leaf octants whereas the octants that are further divided are referred to as non-leaf octants. On the other hand, the bottom-up algorithm starts from non-zero voxels, or non-empty octants in the finest level, and then finds their corresponding parent octants and sibling octants under the same parent node. This process repeats each time on the parent octants found in the previous step until it reaches the root node. We adopt the bottom-up approach in this work because it is more friendly to constructing octrees in parallel.

In this study, the hierarchical information of octrees and the features of consisting octants are stored using the following three series of vectors also used in O–CNN [29]: shuffle keys, labels, and input signals. Shuffle keys and labels are assigned for every octant, and input signals are only for the octants at the finest level. Those elements will be briefly discussed below while more details can be found in Section 2-5 of Supplementary Materials.

Shuffle key is used to indicate the location of an octant within the bounding box so that each octant can be conveniently identified. Because each non-leaf octant has eight children, each child octant can be encoded with a 3-bit code (i.e. a number from 0 to 7). Fig. 4 (a) shows the order of how the eight child octants are indexed, and (b) shows the shuffle keys stored in a vector called $\boldsymbol{S_d}$ at depth $d = 0$, 1, and 2 respectively. At depth $d \geq 1$, a shuffle key is represented by a $3d$-bit code where the first $3(d-1)$ bits represent the parent octant at a depth of $d-1$, and the remaining 3 bits indicate the relative position of the child octants at current depth. For instance, in Fig. 4(b), the shuffle key 5 for the octant at the depth of $d = 1$is equivalent to 3-bit binary key 101. The 3 bits are for x,y,z coordinates separately, and 0/1 indicates the left/right position along the corresponding direction. Hence, key 101 means the octant is right in the x and z coordinates but left in the y coordinate within its parent octant. Thus, its children have 6-bit keys, and the first 3 bits are all 101. For example, the first child has a binary key to 101,000, which equals a decimal number of 40. The 2nd child has key 101,001 = 41. The shuffle keys at every depth are sorted in ascending order. By sorting the "label" and "input" signals in the same order as the shuffle keys, the neighboring and parent-child relationships can be immediately calculated for a given octant.

Label vectors store the order of non-empty octants at each level in order to accelerate the process of tracing children octants. As illustrated in Fig. 4(b), the labels are put below the shuffle keys. The labels of empty octants are 0 shown in yellow color, and the non-empty nodes with non-zero label values are marked in orange. For example, the 7th element (or the 2nd non-empty node) of the Label at depth $d = 1$, $\boldsymbol{L}_1[6]$, is equal to 2. Given a non-empty octant with index j at the depth d, we can compute the index k of its 1st child octant at the depth $(d+1)$ by $k = 8*(\boldsymbol{L_d}[j] - 1)$. In Fig. 4(b), the octant with index 6 (highlighted with a red box in Fig. 4(b)) at the depth of $d = 1$ has its 1st child indexed by $8*(\boldsymbol{L}_1[6] - 1) = 8$, so $\boldsymbol{S}_2[8]$ at the depth of $d = 2$ stores the shuffle key 48 of the 1st child. Since the octants are ordered so that 8 child octants that share a parent octant are stored together, the rest of the child octants can also be quickly located in the vector.

Input signal. Each surface point in the point cloud is associated with relevant atomic and curvature features. The corresponding features of the finest leaf octants are computed by averaging features of all points inside the specific octant. For those empty leaf octants which do not include any point, we simply set the feature values to 0. We put the features of all the finest leaf octants in a vector, referred to as the input signal vector. The input signal vector will have $N \times M$ dimension, where N represents the number of the finest leaf octants in the octree, and M is the number of features calculated for an octant.

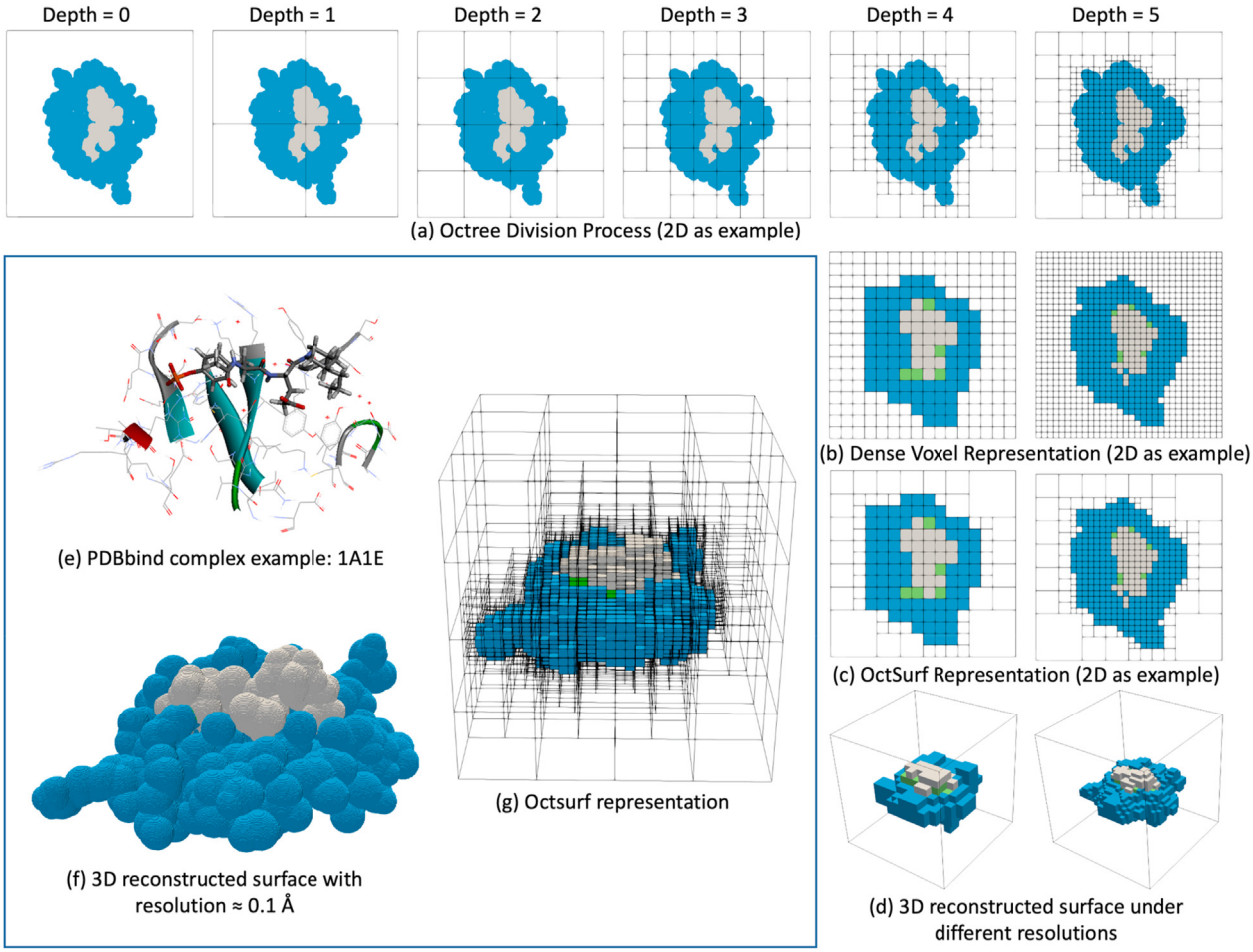

Fig. 3. The Illustration of the OctSurf representation for a protein-ligand pocket example (pdb id: 1A1E) in a 48*48*48 Å^3 bounding box. (a): the recursive division process at various resolutions or depths to generate an Octree-based volumetric representation of the molecular surface, an OctSurf. The surfaces of the protein and ligand are highlighted in blue and gray, respectively; (b): the conventional volumetric representation with equal-size voxels. Those voxels containing both protein and ligand surface points are colored in green; (c): the corresponding OctSurf representation at the same resolution. Compared with the conventional representation, the Octree data structure uses much fewer voxels; (e): the input binding pocket from PDBbind; (f): 3D OctSurf representation of the binding pocket with a resolution of around 0.1 Å; (g): example of the grid subdividing and resulting 3D OctSurf at the depth of 5. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

2.3. 3D-CNN operations on OctSurf

Although the proposed OctSurf data structure can be used in conjunction with any suitable 3D neural network architecture, we have focused on two types of classical CNN architecture in this work: the visual geometry group (VGG) network [46], and the residual network [47] (ResNet). These architectures have been widely adopted as the benchmark network structure for both 2D image recognition and 3D image classification [48]. The convolution and pooling operations used in these networks are modified to be adapted to the above OctSurf data structure. Our VGG-like and ResNet-like network structures for OctSurf are shown in Fig. 5. In both network structures, we use the stride of 1 and the same padding in convolution layers so that the output (OctSurf) of a convolution layer will have the same depth as that of the input OctSurf. The number of channels of the output may differ from the input OctSurf ($C = 24$), and depends on the number of convolution filters in the layer. For example, in Fig. 5(a), the first convolution layer takes an OctSurf of depth D with 24 channels as input and generates an OctSurf of depth D with $2^{max(11-D,\ 3)}$ channels. We use at least 8 channels in case $2^{11-D}$is too small. Every convolution layer is also followed by a batch normalization [49] layer and a Rectified Linear Unit (ReLU) activation layer. The max-pooling layers have a kernel size of 2 and stride of 2, which transforms an OctSurf of depth d into an OctSurf of depth $d-1$. After the pooling, convolutions with the OctSurf of depth $d-1$ can be performed in the next layer.

As shown in Fig. 5(a) below, in the VGG structure, we use $(D-2)$ pairs of convolution layers and max-pooling layers to process OctSurf data from the depth D down to 3, followed by two fully connected layers. In the ResNet structure, we apply 3 Residual Blocks (ResBlocks) at each depth from D down to 3 as shown in Fig. 7(b), except for the first layer where we have an additional pure convolution layer. A max-pooling layer is also applied after 3 ResBlocks at each depth, except in the layer next to the last we perform a global average pooling (with kernel size 8 on OctSurf of depth 3) to get a single octant. Then a fully connected layer is applied to this single octant to make the final prediction. In a ResBlock, there is a shortcut path and a residual path that handle the input independently. If the input channel N_a and the output channel N_b for the block are identical, the shortcut path is just an identity function. Otherwise, the shortcut path is a convolutional layer with the output channel number N_b. The residual path includes three convolutional layers. The outputs from the last convolution layers (when applicable) in both paths are not rectified until we sum them, and the rectified summation is the output of the ResBlock.

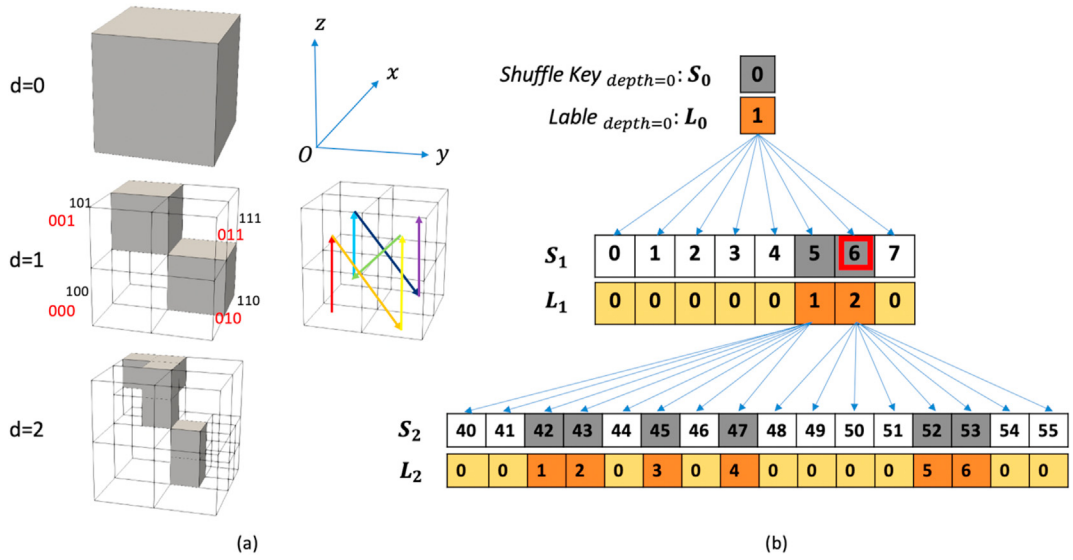

Fig. 4. *An illustration of Shuffle Key and Label in the OctSurf data structure. (a) How the eight child octants are indexed where red color numbers index the octants in the front and black numbers index the octants in the back. Indexing starts from the red arrow, and follows the rainbow color order to the final purple arrow, and (b) the shuffle key vectors* $\boldsymbol{S_d}$ *(white and gray), and the label vectors* $\boldsymbol{L_d}$ *(yellow and orange) at* $d = 0, 1, 2$*. White and yellow cells correspond to empty octants, which are not sub-divided; Gray and orange cells correspond to non-empty octants, sub-divided into 8 children (see arrows). For empty octants, their labels are zero. For non-empty octants, the label value p of an octant indicates that it is the p-th non-empty octant at this depth (starting from 1). The Label is used to find the children of parent octants.* (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

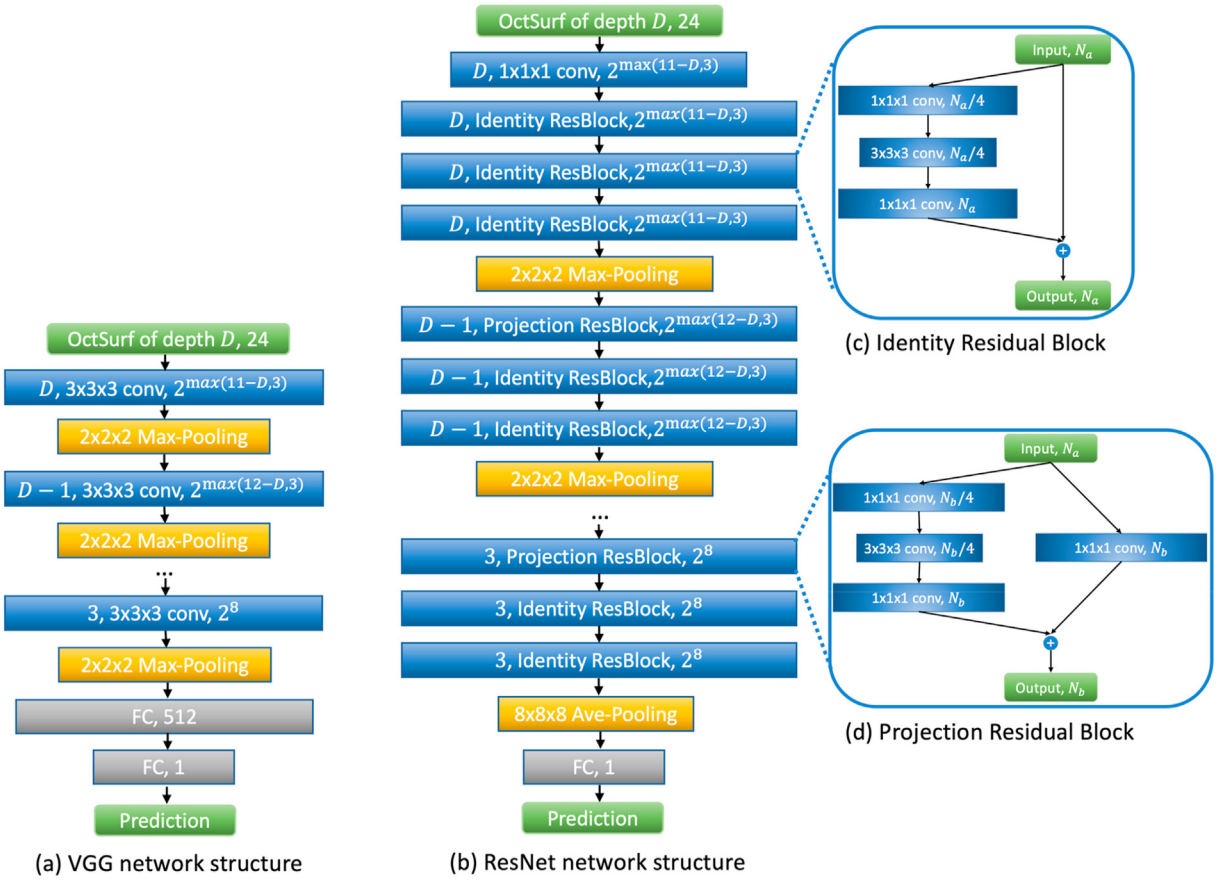

Fig. 5. *The configuration of VGG and ResNet network structure. (a) VGG configuration; (d,* $3 \times 3 \times 3$ *Conv., c) means that with an OctSurf at depth d as inputs, we perform convolution operation with the filter size of* $3 \times 3 \times 3$ *and the number of output channels c in this layer. FC: fully connected layers; (b) ResNet configuration; If the number of input channels and output channels are identical, an Identity Residual Block is applied. Otherwise, the Projection Residual Block will be utilized. (c) and (d): Configurations of Residual Block with Identity and Projection shortcuts.*

In the 3D convolution with OctSurf, each filter is a small cube of m-by-m-by-m (here m = 1 or 3), and the center of this cube moves along the order of sorted keys $S_D$for finest level octants. The convolution $\Phi(O)$ will be computed as a weighted sum of all of the feature values in the finest octants covered by the cube, and the weights are determined by the learned filter as follows:

$$\Phi(O) = \sum_n \sum_i \sum_j \sum_k \boldsymbol{W}_{\boldsymbol{ijk}}^{(\boldsymbol{n})} \boldsymbol{T}^{(\boldsymbol{n})}\left(O_{ijk}\right)$$

where O_{ijk} represents a neighboring octant of O, $\boldsymbol{T}^{(\boldsymbol{n})}(\cdot)$ represents the n-th feature associated with O_{ijk}, and $\boldsymbol{W}_{\boldsymbol{ijk}}^{(\boldsymbol{n})}$'s are the weights in a 3D convolution filter. A 2D illustration of the convolution and pooling operations are shown in Fig. 6. Note that the regular 3D convolution traverses through all voxels in all grids, while 3D convolution on OctSurf only focuses on those finest-level octants. Thus, a large amount of calculation on dominating empty regions can be omitted to achieve efficiency. Moreover, regular 3D convolution on sparse objects brings the dilation problem [27] that some empty grids will also be assigned non-zero features after the convolution if any of their neighbors are not empty. This problem could become more severe in the deeper ResNet structure with more convolution layers because non-zero feature values could propagate to empty voxels far from the surface of the object, which might add noise to the predictions. However, convolution on OctSurf only focuses on the finest octants, which are either the surface or direct neighbor of the surface, so the dilation issue is alleviated.

The CNNs perform max-pooling and average-pooling at appropriate layers. Similar to the convolution layer, a pooling layer also acts on the finest level octants of its input. For a cubic region containing p finest-level octants, the max (or average) pooling operation creates a single octant with the maximal (or average) feature value of all the p octants. In OctSurf computation, in max-pooling layers with a kernel size of 2, the cubic regions are redefined to be parents of the eight octants as indicated by the shuffle keys. When performing a global pooling operation with a kernel size of 8 on OctSurf of depth 3, the cubic region covers the whole OctSurf. After a pooling layer, the OctSurf moves up to a coarser layer in the hierarchy, e.g., depth d to $d-1$ after the max-pooling, or 3 to 0 after the global average pooling using our network configurations.

We adopt the 3D CNN convolution and pooling operations implemented in the recent TensorFlow version of O−CNN [29] where the octree data structure is implemented in C++. We revise the octree implementation to operate on molecular surface point clouds. Our experiments are executed on a Linux server that consists of an Intel(R) Xeon(R) Gold 6150 CPU (2.70 GHz) with 36 cores and 192 GB RAM, and one Nvidia V100 GPU card with 32 GB memory.

3. Experiment results and discussion

3.1. 3D OctSurf volumetric representation of protein and ligand surfaces

Four protein-ligand complexes, 1A1E, 2FXU, 3EL1, and 5ORV, are randomly selected from the PDBbind dataset as examples to evaluate the generated OctSurf representation. Their reconstructed molecular surfaces from the OctSurf volumetric representation at various resolutions are depicted in Fig. 7. Based on the figure, we observe that resolutions beyond 1 Å can reasonably capture the shape information of protein pockets and bound ligands. When the resolution is increased to be higher than 0.25 Å, more fine-grained contextual detail can be revealed.

3.2. Performance evaluation

We evaluate and compare the performance of OctSurf-based 3D-CNN models against the CNN models on dense voxel representations from the following three aspects at various resolutions or depth levels: (1) the data storage size of the generated OctSurf and dense voxel, along with the corresponding relationship between the file size and the occupancy rate; (2) the GPU memory consumption and computation time required for 3D-CNN training; and (3) the predictive performance of the 3D-CNN models based on both OctSurf and dense voxel representations. We also compile two decoy sets by removing either the ligand or protein from the original training co-crystal structures, and check whether our 3D-CNN models maintain the performance when switching to decoyed examples. In addition, since the protein similarity between the training and test sets has been found to have a significant effect on the performance of some machine learning-based scoring functions, we carry out a experiment to inspect how models behave when applying varied protein topological similarity thresholds to exclude training complexes that are structurally similar to any proteins in the test set.

3.2.1. Comparison of data storage for OctSurf vs. dense voxel representations

As discussed in Section 2.2, a series of vectors, including shuffle keys, labels, and input signals, comprise OctSurf representation. Both shuffle keys and octants' labels are recorded as integers while the input signals for octants at the finest level are real values, so the latter are saved as floating-point numbers. Some other information, such as the total depth and the number of octants at each depth, are also necessary to split the vector series and parse the shuffle key and label for each depth. In theory, we need $2n_{total} \times size_{int} + n_{finest} \times n_{feature} \times size_{float} + 784$ bytes of disk space

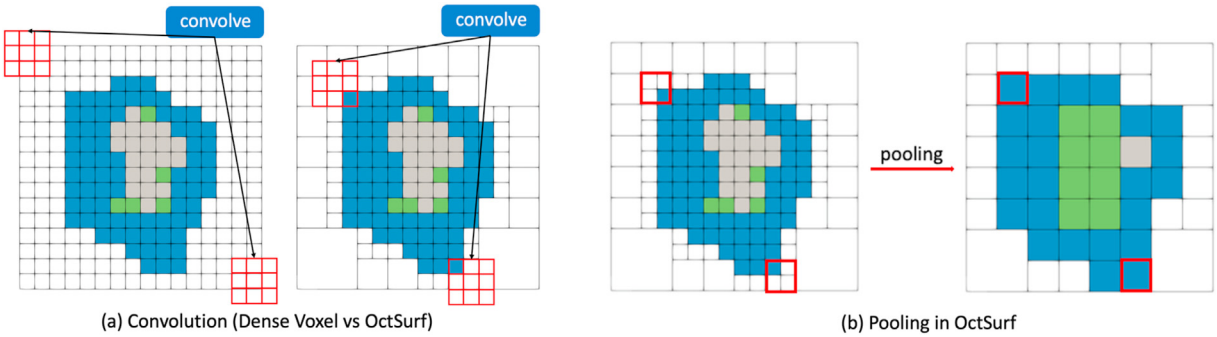

Fig. 6. Convolutional and pooling operations. (a) Conventional convolution in dense voxel starts from the left top corner, and ends at the right bottom corner while our networks with OctSurf input only convolve over the finest octants and thus reducing computation cost; (b) During the pooling computation, the features of all sibling octants are pooled to be the feature of their parent octant. Here the size of the kernel is set to 2. After pooling, the depth of the octree structure is reduced by one.

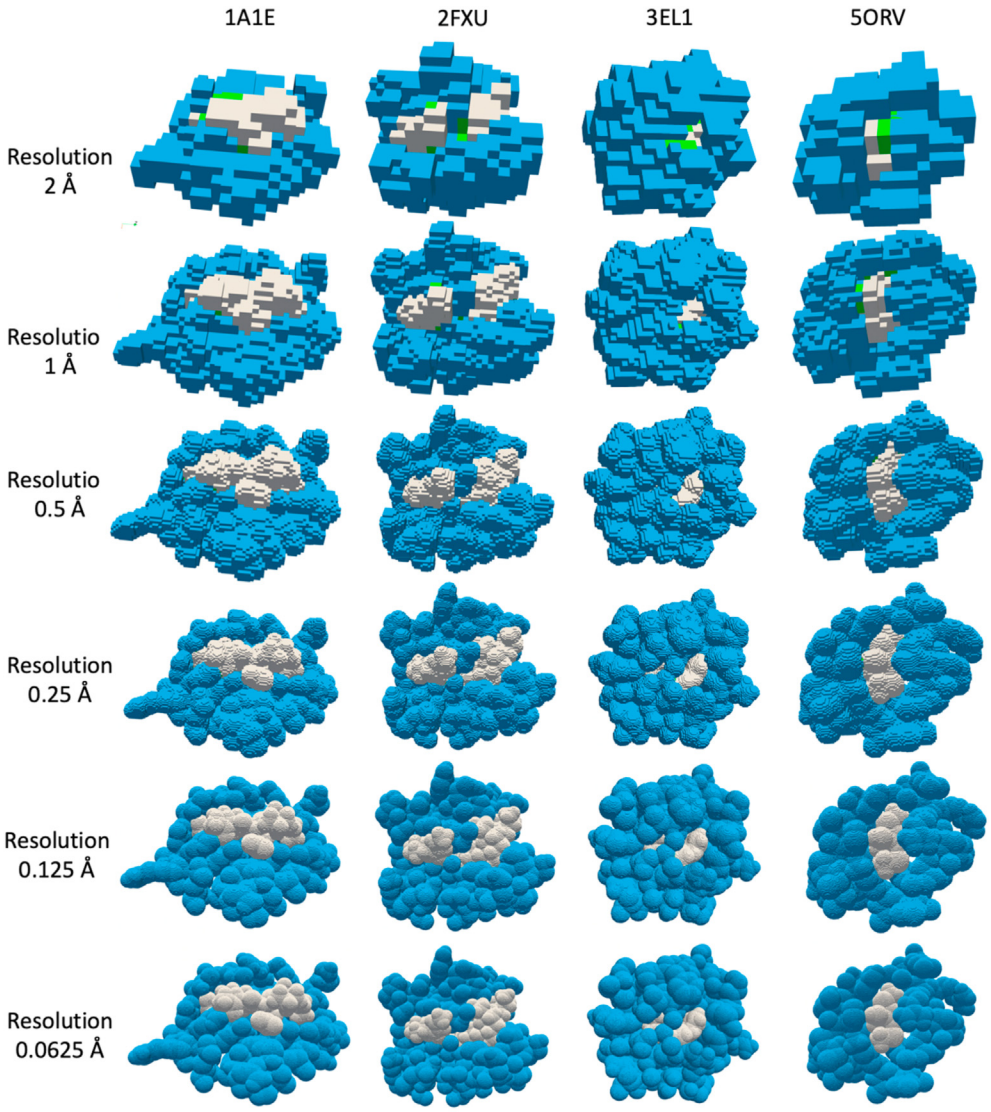

Fig. 7. Reconstruction of the molecular surface of the protein pocket (blue) and bound ligand (gray) under different resolutions. Boundary surfaces of protein pockets are represented by blue voxels, while gray voxels highlight ligand surfaces. At a lower resolution, e.g., 2 Å, some voxels may contain surface points from both protein and ligand atoms. These voxels are highlighted in green. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

or memory to store the OctSurf data representation, where n_{total} is the total number of octants at all depth levels, n_{finest} is the number of octants at the finest level, $n_{feature}$ is the number of features, $size_{int}$ and $size_{float}$ are both 4 bytes, and the remaining 784 bytes are used to store all other information.

We first analyze the storage usage of the OctSurf molecular representation of the example protein-ligand complexes (PDB IDs: 1A1E, 2FXU, 3EL1, and 5ORV) discussed in Section 3.1. The results are summarized in Table 1. The measured disk storage of OctSurf is consistent with the above theoretical expectation. The dense voxel representation consumes the same disk storage for different complexes at a specific resolution regardless of their occupancy rates. However, different complexes can have distinct file sizes for OctSurf representations. Compared with the dense voxel representation, an OctSurf representation uses a substantially smaller number of consisting elements. For instance, at the resolution of 0.0625 Å, dense voxel representations of these complexes require nearly 100 GB of disk storage, while the corresponding OctSurf only consumes less than 0.5 GB. Therefore, the OctSurf data structure enables us to generate volumetric molecular representation at much higher resolutions that are prohibitive with the dense voxel representation. The required disk storage ratio for OctSurf representations versus for dense voxel representations is roughly linearly correlated with the occupancy rate. The ratio is around twice as much as the occupancy rate in Table 1. The occupancy rate drastically decreases as the resolution gets higher, so the OctSurf representation saves a large amount of space compare to the dense voxel representation for the same protein-ligand complex. For instance, OctSurf takes around 5% of the storage used by the dense voxel representation at the resolution of 1 Å, but less than 0.5% of the dense voxel storage at the resolution of 0.0625 Å. OctSurf not only costs less disk space to store, but also requires less memory to perform CNN operations.

We further generate OctSurf representations of all 4463 complexes in the PDBbind refined set and analyze their data storage distribution under different resolution settings. Fig. 8(a) shows that the storage usage increases as the resolution increases while still keeps in a reasonably small size. The storage size of OctSurfs at the 1 Å resolution ranges from 0.47 MB to 2.71 MB. Under 0.25 Å resolution, it increases to between 10.06 MB and 62.54 MB. We also

Table 1
Comparisons of the storage needed for a pocket-ligand complex within a 64 × 64 × 64 Å^3 bounding box around the center of the ligand.

Resolution	PDB ID	1A1E	2FXU	3EL1	5ORV
2 Å	Dense Voxel (32^3)	3 MB			
	OctSurf (depth = 5)	0.18 MB	0.25 MB	0.26 MB	0.14 MB
	Occupancy Rate	3.69%	5.19%	5.43%	2.70%
	Ratio to Dense Voxel	6.14%	8.21%	8.55%	4.52%
1 Å	Dense Voxel (64^3)	24 MB			
	OctSurf (depth = 6)	0.98 MB	1.37 MB	1.44 MB	0.72 MB
	Occupancy Rate	2.35%	3.34%	3.55%	1.67%
	Ratio to Dense Voxel	4.08%	5.72%	5.99%	2.98%
0.5 Å	Dense Voxel (128^3)	192 MB			
	OctSurf (depth = 7)	4.98 MB	7.07 MB	7.51 MB	3.54 MB
	Occupancy Rate	1.29%	1.84%	1.94%	0.89%
	Ratio to Dense Voxel	2.59%	3.68%	3.91%	1.84%
0.25 Å	Dense Voxel (256^3)	1,536 MB			
	OctSurf (depth = 8)	21.88 MB	31.36 MB	33.03 MB	15.19 MB
	Occupancy Rate	0.64%	0.93%	0.98%	0.44%
	Ratio to Dense Voxel	1.42%	2.04%	2.15%	0.99%
0.125 Å	Dense Voxel (512^3)	12,288 MB			
	OctSurf (depth = 9)	87.54 MB	126.22 MB	133.09 MB	60.12 MB
	Occupancy Rate	0.30%	0.43%	0.45%	0.20%
	Ratio to Dense Voxel	0.71%	1.03%	1.08%	0.49%
0.0625 Å	Dense Voxel (1024^3)	98,304 MB			
	OctSurf (depth = 10)	325.66 MB	468.45 MB	493.64 MB	221.88 MB
	Occupancy Rate	0.12%	0.17%	0.18%	0.08%
	Ratio to Dense Voxel	0.33%	0.48%	0.50%	0.23%

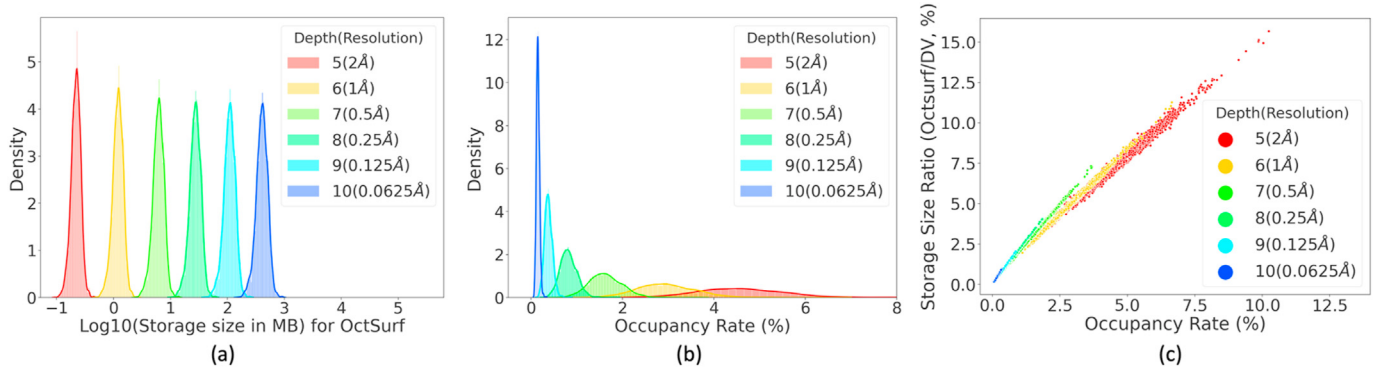

Fig. 8. The density plot of (a) storage usage of OctSurf over the entire PDBbind refined set after a log transformation with base 10 and (b) the occupancy rates of complexes at different resolutions; and (c) the linear relationship between the storage size ratio of OctSurf versus dense voxel (DV), and occupancy rate.

study the variation of occupancy rates of these complexes when increasing resolution. Fig. 8(b) shows the distribution of occupancy rates for the 4463 protein-ligand complexes. Consistent with the observation in Table 1, the overall occupancy rate is distributed over a narrower interval of smaller values at higher resolutions. Fig. 8(c) shows the rough linear relationship between the storage size ratio (with respect to dense voxel storage usage) and the occupancy rate. Table 2 shows the average storage usage and occupancy rates, and we see that the storage used by dense voxel representations generally grows cubically with respect to the resolution, e.g., from 24 MB at 1 Å to 8 times larger, 192 MB, at 0.5 Å. When approaching 0.0625 Å, the dense voxel representation of a single complex can reach nearly 100 GB. We observe that the OctSurf's storage intake only grows quadratically.

3.2.2. GPU memory consumption and computation time of 3D-CNN training

We also compare the GPU memory consumption and time needed to train CNN models on OctSurf and dense voxel representations under various resolutions. The GPU memory is measured in the unit of Mebibytes (MiB), which are each equivalent to 2^{20} bytes or 1.048 MB. The results of VGG and ResNet training are

Table 2
Average storage usage of the pocket-ligand complexes in the PDBbind refine set in the OctSurf and dense voxel (DV) representations.

Resolutions of Voxel Outputs	Size of Dense Voxels (MB)	Size of OctSurf (MB)	Ratio (%) of OctSurf/DV	Occupancy Rate (%)
2 Å	3	0.22	7.33	4.59
1 Å	24	1.22	5.08	2.96
0.5 Å	192	6.27	3.27	1.64
0.25 Å	1536	27.79	1.81	0.82
0.125 Å	12,288	111.55	0.91	0.38
0.0625 Å	98,304	413.64	0.42	0.15

listed in Table 3 and Table 4, respectively. The term OOM in Tables 3 and 4 indicates "out of memory" during computation. Except for cases at the resolution of 0.0625 Å, all models are trained in 10 epochs with a batch size of 8. At the highest resolution of 0.0625 Å, due to the large size of the representation data, the training batch size is reduced to 4.

As shown in Tables 3 and 4, the 3D-CNN modeling with OctSurf takes much less memory than modeling with dense voxels. Note that the GPU memory does hold not only 3D data representations but also hidden layer feature maps and neural network parameters, such as stochastic gradients and filters used in hidden layers. Therefore, GPU memory consumption is larger than just the file storage. As discussed earlier, the dense voxel representation often requires a larger storage size; thus it often causes the out of GPU memory issue at higher resolutions during the 3D-CNN modeling. In contrast, multiple lightweight OctSurfs even at 0.0625 Å can be fit into a single GPU's memory. In addition, because of the efficiency of CNN operations designed for the OctSurf representation, the training time is significantly reduced. For example, the VGG model drops by a factor of 4 at the 1 Å resolution. For ResNet with a more complex network structure, the reduction in the training time is more substantial.

3.2.3. Predictive performance of CNN

A series of VGG and ResNet models are trained using both OctSurf and dense voxel inputs at different resolutions and then tested on 285 protein-ligand complexes in the core test dataset to predict their binding affinities. We replicate the training and test process five times for each model and report their averaged predictive performance in Table 5. Five statistical metrics [11,40,50] are derived from the observed and predicted binding affinities. They include the square of the *Pearson* correlation coefficient (*Pearson* r^2), Spearman's rank correlation coefficient ρ, Kendall rank correlation coefficientτ, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Note that we use the mean squared error as the loss function together with an L2 regularizer that penalizes large values for network parameters during the training. We multiply the regularizer by a weight of 0.01 in the training objective function, which plays the tradeoff between the loss function and the regularizer. Both models are trained by the stochastic gradient descent with the momentum algorithm, with a decaying learning rate (from 0.01 to 1e-7). We run the training algorithm 100 epochs and monitor performance on the validation dataset to make sure that the loss value becomes stable before we report model prediction performance. At the highest resolution at 0.0625 Å, because of the changing of the batch size from 8 to 4, we run 50 epochs to maintain the same number of iterations.

A recent study [27] has pointed out the potential dilation problem when performing convolution operations on sparse objects. They find that convolution focusing only on those surface voxels often outperforms the full convolution over the entire 3D grid for a surface representation. Their finding is consistent with our observations in Table 5 that CNN models built on the OctSurf representation outperform the counterparts trained on the dense voxel data at all different resolutions. The t-tests are conducted to compare the RMSE metrics between OctSurf and dense voxel representations. The small p-values, as shown in Supplementary Materials Table S2 indicate that the null hypothesis that two representations achieving the same RMSE should be rejected, thus we conclude that the performance of two representations are different at the 0.01 significance level. Although ResNet has more convolutional layers, it is prone to the dilation issue for dense voxel inputs as discussed in Section 2.3. It may explain why ResNet models with dense voxel inputs produce worse performance than VGG models. On the other hand, we do not observe a similar trend with the OctSurf input.

When comparing among the different resolutions of OctSurf, the ResNet models perform the best at the 1 Å resolution. It does not meet our original expectation that higher resolution may boost the prediction performance. There are a few possible reasons for this. Octant-associated features in this study are derived from the atomic features, which may not accurately reflect the information in the granularity that a high resolution deserves (e.g., many neighbor octants have the same atomic features). One of the possible ways to improve the performance is to map electrostatic, hydrophobic, and hydrogen bonding potentials on the protein and ligand surfaces, and then derive the voxel features as new input for 3D-CNN modeling. Meanwhile, models developed at a higher resolution tend to include more parameters, which might cause overfitting considering that the size of the current protein-ligand training set is not quite large. With more protein-ligand complexes becoming available in public databases in the near future, the overfitting problem may be alleviated. Since proteins in the PDBbind core or CASF are grouped into 57 protein classes, each of which consists of five protein-ligand complexes. We also analyze how well models predict against each specific protein target in the core test data. We group the predicted binding affinity of the core set according to the protein target and calculate the *Pearson* correlation of predicted and observed binding constants for each target class. The histogram of *Pearson* Correlation for each of fifty-seven proteins in the core set is summarized in Fig. 9. More than 50% of target clusters have a Pearson correlation larger than 0.8. Predicted binding constants of each tested complex as well as the *Pearson* coefficient for each protein target are available in Table S3 of Supplementary Materials.

Besides the PDBbind-v2016 core set, later we extract new 1146 and 394 protein-ligand complexes from the general and refined sets of the latest version of PDBbind (v2019). These two protein-ligand complex datasets are used as additional test sets for model evaluation. As shown in Fig. 10, we obtain the same observation on these two new test sets that the OctSurf representation yields better performance than the Dense Voxel representation. For this particular experiment, we only build ResNet and VGG models at the resolution of 1 Å. More detailed metrics can be found in Supplementary Materials Table S4.

Table 3
The efficiency of the VGG network training with OctSurf.

Resolutions of Voxel Outputs	GPU Memory Usage of Dense Voxels (MiB)	GPU Memory Usage of OctSurf (MiB)	Training Time of Dense Voxels (hour:min:second)	Training Time of OctSurf (hour:min:second)
2 Å	3915	1099	0:16:04	0:12:08
1 Å	5067	1867	0:44:43	0:12:02
0.5 Å	OOM	2379	OOM	0:57:13
0.25 Å	OOM	3531	OOM	3:24:27
0.125 Å	OOM	5067	OOM	8:07:24
0.0625 Å	OOM	17,355	OOM	28:47:50

OOM: out of memory.

Table 4
The efficiency of the ResNet network training with OctSurf.

Resolutions of Voxel Outputs	GPU Memory Usage of Dense Voxels (MiB)	GPU Memory Usage of OctSurf (MiB)	Training Time of Dense Voxels (hour:min:sec)	Training Time of OctSurf (hour:min:sec)
2 Å	4985	1113	0:30:23	0:10:26
1 Å	17,241	1929	2:08:26	0:11:57
0.5 Å	OOM	3161	OOM	0:51:18
0.25 Å	OOM	5476	OOM	3:20:34
0.125 Å	OOM	17,241	OOM	14:05:23
0.0625 Å	OOM	31,397	OOM	56:41:58

OOM: out of memory.

Table 5
The mean and standard deviation of different model evaluation metrics for the five replicates of every VGG and ResNet model at the specific resolution.

		Dense Voxels					OctSurf				
Resolution	Metrics	r^2	ρ	τ	RMSE	MAE	r^2	ρ	τ	RMSE	MAE
2 Å	VGG	0.47 ± 0.006	0.68 ± 0.005	0.49 ± 0.003	1.67 ± 0.006	1.35 ± 0.005	0.58 ± 0.004	0.75 ± 0.003	0.56 ± 0.002	1.51 ± 0.008	1.22 ± 0.007
	ResNet	0.42 ± 0.002	0.65 ± 0.003	0.46 ± 0.004	1.74 ± 0.011	1.40 ± 0.006	0.60 ± 0.010	0.77 ± 0.006	0.57 ± 0.006	1.48 ± 0.008	1.17 ± 0.005
1 Å	VGG	0.49 ± 0.007	0.70 ± 0.005	0.50 ± 0.004	1.64 ± 0.018	1.32 ± 0.018	0.62 ± 0.010	0.78 ± 0.008	0.58 ± 0.006	1.45 ± 0.012	1.16 ± 0.010
	ResNet	0.46 ± 0.021	0.68 ± 0.015	0.49 ± 0.011	1.69 ± 0.019	1.36 ± 0.012	**0.63 ± 0.012**	**0.79 ± 0.008**	**0.59 ± 0.009**	1.45 ± 0.024	**1.16 ± 0.019**
0.5 Å	VGG	OOM	OOM	OOM	OOM	OOM	0.62 ± 0.005	0.79 ± 0.005	0.59 ± 0.004	**1.44 ± 0.017**	**1.16 ± 0.016**
	ResNet	OOM	OOM	OOM	OOM	OOM	0.61 ± 0.013	0.78 ± 0.006	0.58 ± 0.006	1.48 ± 0.027	1.19 ± 0.021
0.25 Å	VGG	OOM	OOM	OOM	OOM	OOM	0.60 ± 0.012	0.77 ± 0.007	0.58 ± 0.007	1.48 ± 0.026	1.18 ± 0.020
	ResNet	OOM	OOM	OOM	OOM	OOM	0.60 ± 0.016	0.77 ± 0.012	0.58 ± 0.011	1.48 ± 0.025	1.18 ± 0.018
0.125 Å	VGG	OOM	OOM	OOM	OOM	OOM	0.55 ± 0.017	0.75 ± 0.009	0.55 ± 0.010	1.57 ± 0.050	1.27 ± 0.037
	ResNet	OOM	OOM	OOM	OOM	OOM	0.49 ± 0.045	0.70 ± 0.030	0.51 ± 0.027	1.66 ± 0.103	1.35 ± 0.086
0.0625 Å	VGG	OOM	OOM	OOM	OOM	OOM	0.49 ± 0.007	0.71 ± 0.007	0.51 ± 0.005	1.66 ± 0.026	1.33 ± 0.024
	ResNet	OOM	OOM	OOM	OOM	OOM	0.42 ± 0.038	0.65 ± 0.029	0.46 ± 0.025	1.83 ± 0.107	1.49 ± 0.088

OOM: out of memory.

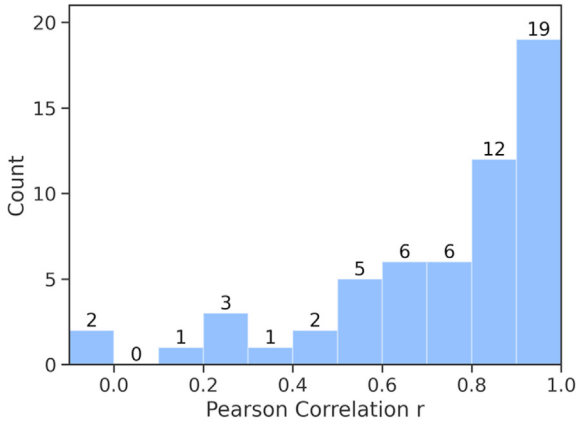

Fig. 9. Histogram of Pearson Correlation of predicted vs. observed binding constants for each of 57 protein members within the core set. Predictions are chosen from the ResNet model trained with OctSurf at 1 Å resolution.

3.2.4. The assessment of model sensitivity toward potential biases in the training data

To explore the effect of potential hidden data bias on our models, we implement a set of experiments and inspect constructed 3D-CNN models based on recompiled decoy training data. Yang et al. [34] showed that models derived from protein or ligand structures alone can still produce a comparatively good performance on a test dataset. So we are interested in checking whether this would happen to our models or not. We split the training protein-ligand complexes into two decoy sets that contain ligand or protein structures alone. Then we re-train both ResNet and VGG models using the new decoy data and test their performances on core set followed the same decoy treatment. For all the bias assessment experiments in this section, we always set the volumetric resolution at 1 Å because ResNet and VGG models often yield better performance on the core set under this resolution in our previous analyses. To ensure the model consistency, we repeat each model's training process five times and generate the box plot, as illustrated in Fig. 11, to summarize different models' predictive performance. Compared with the original models, the performance of new models built on the two decoy sets drops significantly. The mean Pearson correlation coefficient of observed and predicted binding constants by five ResNet models trained with the protein-ligand complex is about 0.63, while the same correlation coefficients of ResNet models built from pocket or ligand structures are only approximately 0.42 and 0.54, respectively. Although our observation contradicts with Yang's previous findings, it does not necessarily mean that our models capture real protein-ligand interaction patterns.

The second control experiment is designed to investigate the potential impact of protein similarity between training and test sets on our model performance. We first calculate the pairwise topological similarity between each pair of proteins in the general/refined set and core set. The topological similarity of protein structures is measured by TM-score [51] to avoid the power-law dependence with the length of the protein. If the similarity coefficient between a protein-ligand complex in the general/refined set and any complex in core set is higher than the specified cutoff, this complex will be removed from the training set. A series of TM-score similarity thresholds ranging from 1 to 0.5 are used to control the degree of similarity between two data sets. As shown in Fig. 12(a), the higher the cutoff, the more structurally similar proteins will remain in the training set. For each new training set, we follow a similar procedure as previously described to build the model. Finally, we check how the model's predictive performance varies with the topological similarity between the training and test protein complexes and illustrate the results in Fig. 12(b). In our experiment, both ResNet and VGG models are not very sensitive to the selection of similarity thresholds, although models derived from larger training sets seem to show slightly better performance.

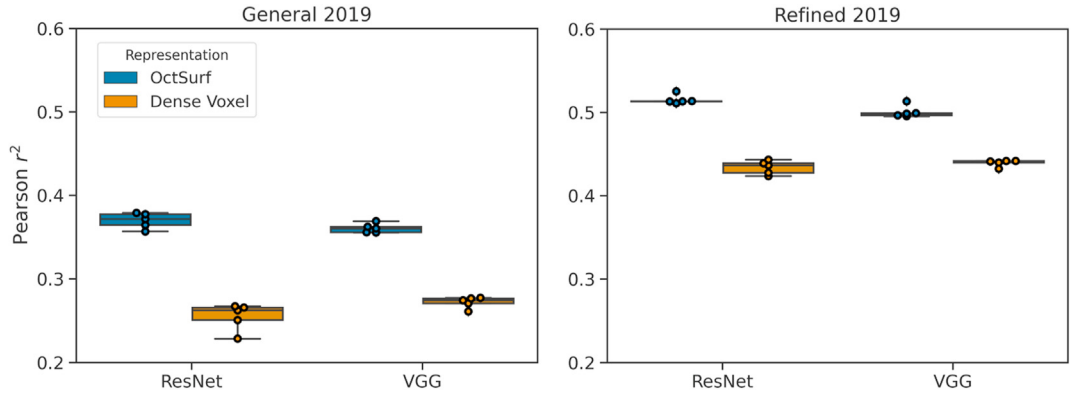

Fig. 10. *Prediction Pearson r^2 performance on General 2019 and Refined 2019, based on 1 Å resolution.*

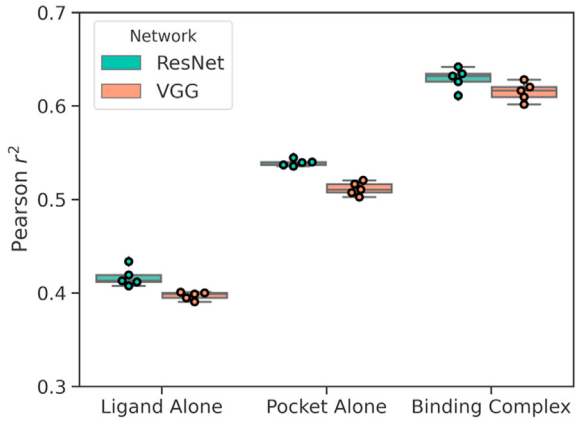

Fig. 11. The boxplot to demonstrate the performance change of OctSurf-based 3D-CNN models trained with decoy sets of ligand or protein pocket structures alone.

Interestingly, Su et al. [32] showed such bias sensitivity level is also dependent on selected machine learning approaches. For instance, it seems that tree-based models, including Random Forest and Decision Tree, are somehow more prone to be affected by the similarity cutoff. Several other nonlinear machine learning methods, such as linear support vector regression and Bayesian Ridge Regression, do not show significant performance variation in the same study.

Even though some remarkable progress has been achieved, AI application in drug discovery is still in its infancy. There are still a few major bottlenecks limiting the successful application of machine learning in protein-ligand affinity prediction and other scientific fields. Besides the lack of sufficiently reliable datasets to train robust models, current machine learning models excel in recognizing statistical patterns amongst vast reams of data, but are not capable of reasoning about the cause-effect relations, e.g., why ligands will have strong or weak binding affinity against relevant protein targets. From the algorithm development point of view, future intelligence systems for the protein-ligand affinity prediction require the appropriate incorporation of underlying prior domain knowledge, biophysics causality, and interpretable visualization techniques into data-driven machine learning frameworks.

4. Conclusion

In this paper, we describe a new 3D-CNN framework to predict the protein-ligand binding affinity. We implement an octree-based volumetric representation (OctSurf) to capture the contact and shape complementary between protein pocket and bound ligand surfaces. It provides an alternative volumetric representation to describe the interaction and recognition between the protein and bound ligand. The obtained volumetric surface representation can be further fed into two deep learning architectures, VGG and ResNet, to construct 3D-CNN models and predict protein-ligand binding affinity. Our OctSurf representation shows storage and

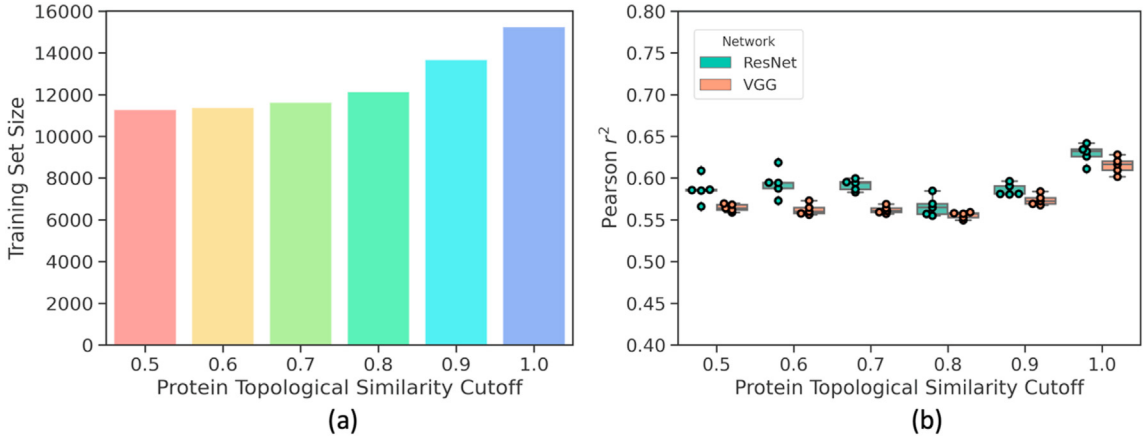

Fig. 12. *(a) The number of remaining protein-ligand complexes in the training set after applying different TM-score similarity cutoffs to remove the training cases that are structurally similar to any test proteins. (b) The box blot diagram of Pearson r^2 for five duplicate ResNet and VGG models built on each of the different training sets.*

memory efficiency over the traditional dense voxel representation of 3D molecular surfaces at the same resolution. It also reduces the computational overhead of 3D-CNN by restricting the convolution operation over octants near the molecular surface region. Moreover, it now allows us to reconstruct the 3D protein-ligand surface representation in much finer resolutions and build 3D-CNN predictive models, which are often computationally prohibited by conventional voxel-based methods. With much less GPU memory usage and computation cost, constructed CNN models on OctSurf demonstrate better prediction performance than the models trained using dense voxels of the same molecular surface dataset. We also run several control experiments to show that the predictive performance of our models deteriorates when using the protein or ligand structures alone as the training set, but they are not sensitive to protein similarity between training and test sets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported by research funds from the University of Connecticut and has not received any grant from federal funding agencies in the US, commercial, or not-for-profit sectors.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jmgm.2021.107865.

References

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: a deep representation for volumetric shapes, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1912–1920.
[2] D. Maturana, S. Scherer, Voxnet: a 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2015, pp. 922–928.
[3] C. Shen, J. Ding, Z. Wang, D. Cao, X. Ding, T. Hou, From machine learning to deep learning: advances in scoring functions for protein–ligand docking, Wiley Interdiscip. Rev. Comput. Mol. Sci. 10 (2020) e1429.
[4] I. Wallach, M. Dzamba, A. Heifets, AtomNet: a Deep Convolutional Neural Network for Bioactivity Prediction in Structure-Based Drug Discovery, ArXiv Prepr. ArXiv1510.02855, 2015.
[5] D. Kuzminykh, D. Polykovskiy, A. Kadurin, A. Zhebrak, I. Baskov, S. Nikolenko, R. Shayakhmetov, A. Zhavoronkov, 3D molecular representations based on the wave transform for convolutional neural networks, Mol. Pharm. 15 (2018) 4378–4385.
[6] J. Jimenez, S. Doerr, G. Mart\'inez-Rosell, A.S. Rose, G. De Fabritiis, DeepSite: protein-binding site predictor using 3D-convolutional neural networks, Bioinformatics 33 (2017) 3036–3042.
[7] M. Ragoza, J. Hochuli, E. Idrobo, J. Sunseri, D.R. Koes, Protein–ligand scoring with convolutional neural networks, J. Chem. Inf. Model. 57 (2017) 942–957.
[8] J. Hochuli, A. Helbling, T. Skaist, M. Ragoza, D.R. Koes, Visualizing convolutional neural network protein-ligand scoring, J. Mol. Graph. Model. 84 (2018) 96–108.
[9] A.H. Mahmoud, M.R. Masters, Y. Yang, M.A. Lill, Elucidating the multiple roles of hydration for accurate protein-ligand binding prediction via deep learning, Commun. Chem. 3 (2020) 1–13.
[10] H. Hassan-Harrirou, C. Zhang, T. Lemmin, RosENet: improving binding affinity prediction by leveraging molecular mechanics energies with an ensemble of 3D convolutional neural networks, J. Chem. Inf. Model. (2020).
[11] J. Jimenez, M. Skalic, G. Martinez-Rosell, G. De Fabritiis, K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks, J. Chem. Inf. Model. 58 (2018) 287–296.
[12] W. Torng, R.B. Altman, 3D deep convolutional neural networks for amino acid environment similarity analysis, BMC Bioinf. 18 (2017) 302.
[13] M. Simonovsky, J. Meyers, DeeplyTough: Learning Structural Comparison of Protein Binding Sites, BioRxiv., 2019, p. 600304.
[14] L. Pu, R.G. Govindaraj, J.M. Lemoine, H.-C. Wu, M. Brylinski, DeepDrug3D: classification of ligand-binding pockets in proteins with a convolutional neural network, PLoS Comput. Biol. 15 (2019), e1006718.
[15] W. Torng, R.B. Altman, High precision protein functional site detection using 3D convolutional neural networks, Bioinformatics 35 (2019) 1503–1512.
[16] M.M. Stepniewska-Dziubinska, P. Zielenkiewicz, P. Siedlecki, Improving detection of protein-ligand binding sites with 3D segmentation, Sci. Rep. 10 (2020) 1–9.
[17] S. Yin, E.A. Proctor, A.A. Lugovskoy, N. V Dokholyan, Fast screening of protein surfaces using geometric invariant fingerprints, Proc. Natl. Acad. Sci. Unit. States Am. 106 (2009) 16622–16626.
[18] V. Venkatraman, Y.D. Yang, L. Sael, D. Kihara, Protein-protein docking using region-based 3D Zernike descriptors, BMC Bioinf. 10 (2009) 407.
[19] D. Kihara, L. Sael, R. Chikhi, J. Esquivel-Rodriguez, Molecular surface representation using 3D Zernike descriptors for protein shape comparison and docking, Curr. Protein Pept. Sci. 12 (2011) 520–530.
[20] X. Zhu, Y. Xiong, D. Kihara, Large-scale binding ligand prediction by improved patch-based method Patch-Surfer2. 0, Bioinformatics 31 (2014) 707–713.
[21] S. Daberdaku, C. Ferrari, Antibody interface prediction with 3D Zernike descriptors and SVM, Bioinformatics 35 (2019) 1870–1876.
[22] P. Gainza, F. Sverrisson, F. Monti, E. Rodola, D. Boscaini, M.M. Bronstein, B.E. Correia, Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning, Nat. Methods 17 (2020) 184–192.
[23] S.K. Mylonas, A. Axenopoulos, P. Daras, DeepSurf: A Surface-Based Deep Learning Approach for the Prediction of Ligand Binding Sites on Proteins, ArXiv Prepr. ArXiv2002.05643, 2020.
[24] Y. Li, S. Pirk, H. Su, C.R. Qi, L.J. Guibas, Fpnn: field probing neural networks for 3d data, Adv. Neural Inf. Process. Syst. (2016) 307–315.
[25] B. Graham, Sparse 3D Convolutional Neural Networks, ArXiv Prepr. ArXiv1505.02890, 2015.
[26] B. Graham, L. van der Maaten, Submanifold Sparse Convolutional Networks, ArXiv Prepr. ArXiv1706.01307, 2017.
[27] B. Graham, M. Engelcke, L. Van Der Maaten, 3d semantic segmentation with submanifold sparse convolutional networks, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9224–9232.
[28] G. Riegler, A. Osman Ulusoy, A. Geiger, Octnet: learning deep 3d representations at high resolutions, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 3577–3586.
[29] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, X. Tong, O-cnn: octree-based convolutional neural networks for 3d shape analysis, ACM Trans. Graph. 36 (2017) 72.
[30] J. Sieg, F. Flachsenberg, M. Rarey, In need of bias control: evaluating chemical data for machine learning in structure-based virtual screening, J. Chem. Inf. Model. 59 (2019) 947–961.
[31] L. Chen, A. Cruz, S. Ramsey, C.J. Dickson, J.S. Duca, V. Hornak, D.R. Koes, T. Kurtzman, Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening, PLoS One 14 (2019), e0220113.
[32] M. Su, G. Feng, Z. Liu, Y. Li, R. Wang, Tapping on the black box: how is the scoring power of a machine-learning scoring function dependent on the training set? J. Chem. Inf. Model. 60 (2020) 1122–1136.
[33] C. Shen, Y. Hu, Z. Wang, X. Zhang, H. Zhong, G. Wang, X. Yao, L. Xu, D. Cao, T. Hou, Can machine learning consistently improve the scoring power of classical scoring functions? Insights into the role of machine learning in scoring functions, Briefings Bioinf. (2020).
[34] J. Yang, C. Shen, N. Huang, Predicting or pretending: artificial intelligence for protein-ligand interactions lack of sufficiently large and unbiased datasets, Front. Pharmacol. 11 (2020) 69.
[35] M.M. Mysinger, M. Carchia, J.J. Irwin, B.K. Shoichet, Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking, J. Med. Chem. 55 (2012) 6582–6594.
[36] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn, Nat. Commun. 10 (2019) 1–8.
[37] Z. Liu, M. Su, L. Han, J. Liu, Q. Yang, Y. Li, R. Wang, Forging the basis for developing protein–ligand interaction scoring functions, Acc. Chem. Res. 50 (2017) 302–309.
[38] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li, R. Wang, Comparative assessment of scoring functions: the CASF-2016 update, J. Chem. Inf. Model. 59 (2018) 895–913.
[39] M.M. Stepniewska-Dziubinska, P. Zielenkiewicz, P. Siedlecki, Development and evaluation of a deep learning model for protein–ligand binding affinity prediction, Bioinformatics 34 (2018) 3666–3674.
[40] Y. Li, M.A. Rezaei, C. Li, X. Li, DeepAtom: a framework for protein-ligand binding affinity prediction, in: 2019 IEEE Int. Conf. Bioinforma. Biomed., 2019, pp. 303–310.
[41] F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, M. Scharf, The Double cubic Lattice method: efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies, J. Comput. Chem. 16 (1995) 273–284, https://doi.org/10.1002/jcc.540160303.
[42] E.L. Willighagen, J.W. Mayfield, J. Alvarsson, A. Berg, L. Carlsson, N. Jeliazkova, S. Kuhn, T. Pluskal, M. Rojas-Chertó, O. Spjuth, others, the Chemistry Development Kit (CDK) v2. 0: atom typing, depiction, molecular formulas, and substructure searching, J. Cheminf. 9 (2017) 33.
[43] G. Riegler, A.O. Ulusoy, H. Bischof, A. Geiger, Octnetfusion: learning depth fusion from data, in: 2017 Int. Conf. 3D Vis., 2017, pp. 57–66.
[44] C. Crassin, F. Neyret, S. Lefebvre, E. Eisemann, Gigavoxels: ray-guided

streaming for efficient and detailed voxel rendering, in: Proc. 2009 Symp. Interact. 3D Graph. Games, 2009, pp. 15–22.
[45] G.K.M. Cheung, T. Kanade, J.-Y. Bouguet, M. Holler, A real time system for robust 3D voxel reconstruction of human motions, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2000 (Cat. No. PR00662), 2000, pp. 714–720.
[46] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, ArXiv Prepr. ArXiv1409. 1556 (2014).
[47] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[48] C. Yang, A. Rangarajan, S. Ranka, Visual explanations from deep 3D convolutional neural networks for Alzheimer's disease classification, AMIA Annu. Symp. Proc. (2018) 1571.
[49] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ArXiv Prepr. ArXiv1502.03167, 2015.
[50] M. Karimi, D. Wu, Z. Wang, Y. Shen, DeepAffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks, Bioinformatics 35 (2019) 3329–3338.
[51] Y. Zhang, J. Skolnick, Scoring function for automated assessment of protein structure template quality, Proteins Struct. Funct. Bioinforma. 57 (2004) 702–710.