JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

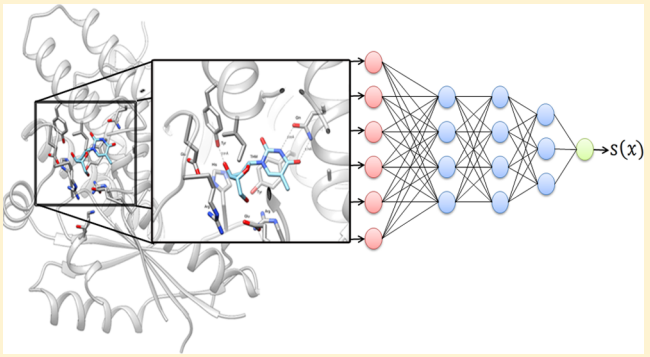
Boosting Docking-Based Virtual Screening with Deep Learning

Janaina Cruz Pereira,*[,†] Ernesto Raúl Caffarena,*[,†] and Cicero Nogueira dos Santos*[,‡]

[†]Fiocruz, 4365 Avenida Brasil, Rio de Janeiro, RJ 21040 900, Brazil

[‡]IBM Watson, 1101 Kitchawan Rd., Yorktown Heights, New York 10598, United States

ABSTRACT: In this work, we propose a deep learning approach to improve docking-based virtual screening. The deep neural network that is introduced, DeepVS, uses the output of a docking program and learns how to extract relevant features from basic data such as atom and residues types obtained from protein−ligand complexes. Our approach introduces the use of atom and amino acid embeddings and implements an effective way of creating distributed vector representations of protein−ligand complexes by modeling the compound as a set of atom contexts that is further processed by a convolutional layer. One of the main advantages of the proposed method is that it does not require feature engineering. We evaluate DeepVS on the Directory of Useful Decoys (DUD), using the output of two docking programs: Autodock Vina1.1.2 and Dock 6.6. Using a strict evaluation with leave-one-out cross-validation, DeepVS outperforms the docking programs, with regard to both AUC ROC and enrichment factor. Moreover, using the output of Autodock Vina1.1.2, DeepVS achieves an AUC ROC of 0.81, which, to the best of our knowledge, is the best AUC reported so far for virtual screening using the 40 receptors from the DUD.



INTRODUCTION

Drug discovery process is a time-consuming and expensive task. The development and even the repositioning of already-known compounds is a difficult chore.[1] The scenario gets worse if we take into account the thousands or millions of molecules capable of being synthesized in each development stage.[2,3]

In the past, experimental methods such as high-throughput screening (HTS) could help making this decision through the screening of large chemical libraries against a biological target. However, the high cost of the entire process associated with a low success rate makes this method inaccessible to academia.[2−4]

In order to overcome these difficulties, the use of low-cost computational alternatives is extensively encouraged, and it was adopted routinely as a way to aid in the development of new drugs.[3−5]

Computational virtual screening works basically as a filter (or a prefilter) consisting of the virtual selection of molecules, based on a particular predefined criterion of potentially active compounds against a determined pharmacological target.[2,4,6]

Two variants of this method can be adopted: ligand-based virtual screening and structure-based virtual screening. The first one involves the similarity and the physicochemical analysis of active ligands to predict the activity of other compounds with similar characteristics. The second is utilized when the three-dimensional (3D) structure of the target receptor was already elucidated somehow (experimentally or computationally modeled). This approach is used to explore molecular interactions between possible active ligands and residues of the binding site. Structure-based methods present a better performance, when compared to methods based solely on the structure of the ligand focusing on the identification of new compounds with therapeutic potential.[1,7−9]

One of the computational methodologies extensively used to investigate these interactions is molecular docking.[5,8,10] The selection of more-potent ligands using Docking-based Virtual Screening (DBVS) is made by performing the insertion of each compound from a compound library into a particular region of a target receptor with an elucidated 3D structure. In the first stage of this process, a heuristic search is carried out in which thousands of possible insertions are regarded. In the second, the quality of the insertion is described via some mathematical functions (scoring functions) that give a clue about the energy complementarity between the compound and the target.[10,11] The last phase became a challenge to the computational scientists, considering that it is easier to recover the proper binding mode of a compound within an active site than to assess a low energy score to a determined pose. This hurdle constitutes a central problem to the docking methodology.[12]

Systems based on machine learning (ML) have been successfully used to improve the outcome of DBVS for both, increasing the performance of score functions and constructing binding affinity classifiers.[1,3] The main strategies used in virtual screening are neural networks (NN),[13] support vector machines (SVM),[12] and random forest (RF).[14] One of the main advantages of employing ML is the capacity to explain the nonlinear dependence of the molecular interactions between the ligand and the receptor.[3]

Received: June 15, 2016
Published: November 4, 2016

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

Traditional ML strategies are dependent on how the data are presented. For example, in the virtual screening approaches, scientists normally analyze the docking output to generate or extract human engineered features. Although this process can be effective to some degree, the manual identification of characteristics is a laborious and complex process and cannot be applied on a large scale, resulting in the loss of relevant information, consequently leading to a set of features incapable of explaining the actual complexity of the problem.[1,3,15,16]

On the other hand, recent work on deep learning (DL), which is a family of ML approaches that minimizes feature engineering, has demonstrated enormous success in different tasks from multiple fields.[15−17] DL approaches normally learn features (representations) directly from the raw data with minimal or no human intervention, which makes the resulting system easier to adapt to new datasets.

In the past few years, DL has been attracting the attention of the academic community and large pharmaceutical industries, as a viable alternative to aid in the discovery of new drugs. One of the first works in which DL was successfully applied solved problems related to quantitative structure−activity relationships (QSAR) by Merck in 2012. Some years later, Dahl et al.[18] developed a multitask deep neural network to predict biological and chemical properties of a compound directly from its molecular structure. More recently, Multitask deep neural networks were employed to foresee the active-site directed pharmacophore and toxicity.[19,20] Also in 2015, Ramsundar et al.[21] predicted drug activity using Massively Multitask Neural Networks associated with fingerprints. The relevance of DL is also highlighted by recent applications to the discovery of new drugs and the determination of their characteristics such as Aqueous Solubility prediction[22] and Fingerprints.[23]

In this work, we propose an approach based on Deep Convolutional Neural Networks to improve DBVS. The method uses docking simulation results as input to a Deep Neural Network, referenced hereafter as DeepVS, which automatically learns to extract relevant features from basic data such as compound atom types, atomic partial charges, and the distance between atoms. DeepVS learns abstract features that are suitable to discriminate between active ligands and decoys in a protein-compound complex. To the best of our knowledge, this work is the first on using deep learning to improve docking-based virtual screening. Recent works on improving docking-based virtual screening have only used traditional shallow neural networks with human-defined features.[7,13,24]

We evaluated DeepVS on the Directory of Useful Decoys, which contains 40 different receptors. In our experiments, we used the output of two docking programs: Autodock Vina1.1.2 and Dock 6.6. DeepVS outperformed the docking programs in both AUC ROC and enrichment factor. Moreover, when compared to the results of other systems previously reported in the literature, DeepVS achieved the state-of-the-art area under the curve (AUC) of 0.81.

The main contributions of this work are (1) proposition of a new deep learning-based approach that achieves state-of-the-art performance on DBVS; (2) introduction of the concept of atom and amino acid embeddings, which can also be used in other deep learning applications for computational biology; and (3) proposition of an effective method to create distributed vector representations of protein-compound complexes that models the compound as a set of atom contexts that is further processed by a convolutional layer.

MATERIALS AND METHODS

DeepVS. DeepVS takes the data describing the structure of a protein-compound complex (input layer) as input and produces a score capable of differentiating ligands from decoys. Figure 1 details the DeepVS architecture. First, given an input

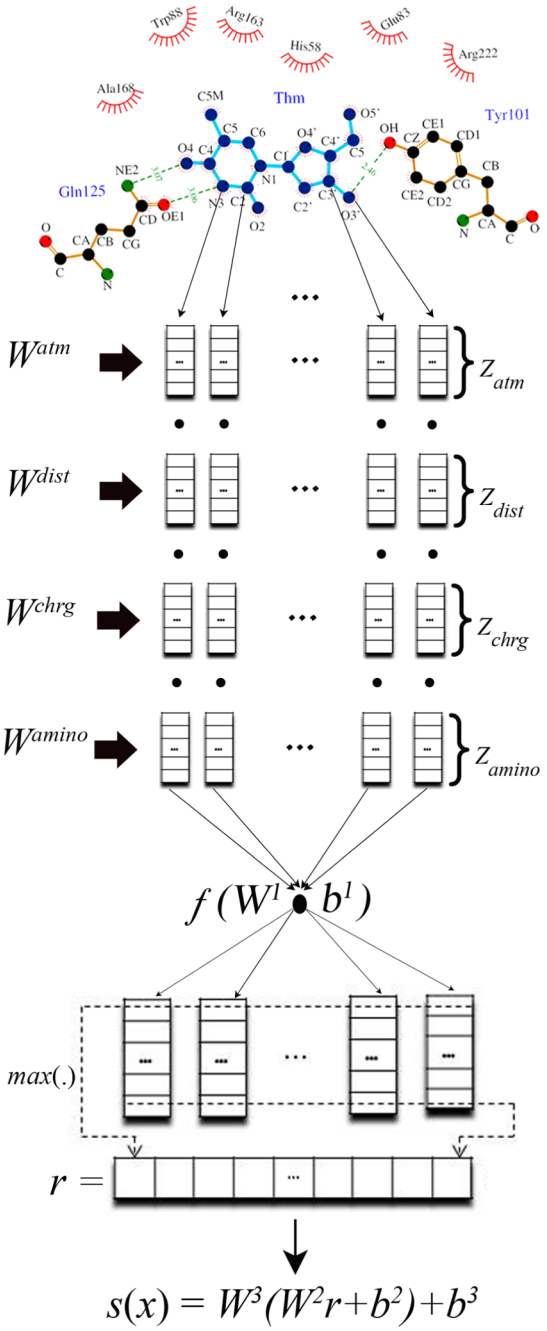


Figure 1. DeepVS architecture scheme. In this figure, we use, as an example, the compound thymidine (THM) in complex with thymidine kinase (TK) protein (PDB No. 1kim) compound atoms are marked in dark blue and their interactions in light blue.

protein-compound complex x, information is extracted from the local context of each compound atom (first hidden layer). The *context* of an atom comprises basic structural data (*basic features*) involving distances, neighbor atom types, atomic partial charges, and associated residues. Next, each basic feature from each atom context is converted to feature vectors that are

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

learned by the network. Then, a convolutional layer is employed to summarize information from all contexts from all atoms and generate a distributed vector representation of the protein-compound complex r (second hidden layer). Finally, in the last layer (output layer), the representation of the complex is given as input to softmax classifier, which is responsible for producing the score. In Algorithm 1, we present a high-level pseudocode with the steps of the feedforward process executed by DeepVS.

Algorithm 1 DeepVS feedforward process

Input: protein-compound complex x, where the compound contains m atoms
Given: trained network parameters $W^{atm} \in \mathbb{R}^{d^{atm} \times |A|}$, $W^{dist} \in \mathbb{R}^{d^{dist} \times |D|}$, $W^{chrg} \in \mathbb{R}^{d^{chrg} \times |C|}$, $W^{amino} \in \mathbb{R}^{d^{amino} \times |R|}$, $W^1 \in \mathbb{R}^{cf \times |z_i|}$, $W^2 \in \mathbb{R}^{h \times |cf|}$, $W^3 \in \mathbb{R}^{2 \times |h|}$, $b^1 \in \mathbb{R}^{cf}$, $b^2 \in \mathbb{R}^h$, $b^3 \in \mathbb{R}^2$
$Z = [\,]$
// generates the representations of atom contexts
for i=1 to m **do**
  z_{atm} = columns from W_{atm} corresponding to atom types of atom$_i$'s neighbors
  z_{dist} = columns from W_{dist} corresponding to distances of atom$_i$'s neighbors
  z_{chrg} = columns from W_{chrg} corresponding to charges of atom$_i$'s neighbors
  z_{amino} = columns from W_{amino} corresponding to amino acids of atom$_i$'s neighbors
  // representation of the atom$_i$ context
  $z_i = \{z_{atm}; z_{dist}; z_{chrg}; z_{amino}\}$
  $Z.add(z_i)$
end for
// U is initialized with zeros
$U = [..] \in \mathbb{R}^{cf \times m}$
// convolutional layer
for i=1 to m **do**
  $U[:, i] = f(W^1 Z[i] + b^1)$
end for
// column-wise max pooling
$r = max(U, axis = 1)$
// hidden and output layers
score $= W^3(W^2 r + b^2) + b^3$
// returns normalized score
return $\frac{e^{score[1]}}{e^{score[0]} + e^{score[1]}}$

Atom Context. First, it is necessary to perform some basic processing on the protein-compound complex to extract the input data for DeepVS. The input layer uses information from the context of each atom in the compound. The context of an atom a is defined by a set of basic features extracted from its neighborhood. This neighborhood consists of the k_c atoms in the compound closest to a (including itself) and the k_p atoms in the protein that are closest to a, where k_c and k_p are hyperparameters that must be defined by the user. The idea of using information from closest neighbor atoms of both compound and protein have been successfully explored in previous work on structure-based drug design.[25]

The basic features extracted from the context of an atom include the atom types, atomic partial charges, amino acid types, and the distances from neighbors to the reference atom. For instance (Figure 2), for the nitrogen atom (N3) from the THM compound, the vicinity with k_c = 3 and k_p = 2 is formed

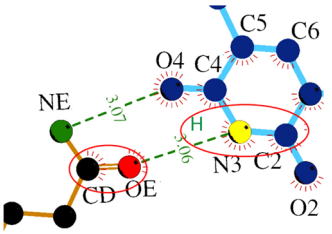


Figure 2. Context from the N3 atom (yellow) from the thymidine (THM) compound. Red circles represent the two closest neighbors from the THM compound and the two closest neighbors from the protein closest to the N3 atom.

by N3, H and C from the compound, and the two atoms OE and CD from the residue Gln_{125} in the TK protein (PDB No. 1kim). In this particular case, the context of atom N3 contains the following values for each basic feature:

- Atom type = [N; H; C; OE; CD]
- Charge = [-0.24; 0.16; 0.31; −0.61; 0.69]
- Distance = [0.00; 1.00; 1.34; 3.06; 3.90]
- Amino acid type = [Gln; Gln]

Representation of the Atom Context. The first hidden layer in DeepVS transforms each basic feature value of atom contexts into real-valued vectors (also known as *embeddings*) by a lookup table operation. These embeddings contain features that are automatically learned by the network. For each type of basic feature, there is a corresponding embedding matrix W that stores a column vector for each possible value for that basic feature. The matrices W^{atm}, W^{dist}, W^{chrg}, and W^{amino} contain the embeddings of the basic features atom type, distance, atomic partial charge, and amino acid type, respectively. These matrices constitute the weight matrices of the first hidden layer and are initialized with random numbers before training the network.

Each column in $W^{atm} \in \mathbb{R}^{d^{atm} \times |A|}$ corresponds to a feature vector of a particular type of atom, where A is the set of atom types and d^{atm} is the dimensionality of the embedding and constitutes a hyperparameter defined by the user. Given the context of an atom a, the network transforms each value of the basic feature atom type in its respective feature vector and then concatenates these vectors to generate the vector atom-type representation z_{atm}. As illustrated in Figure 3, retrieving the

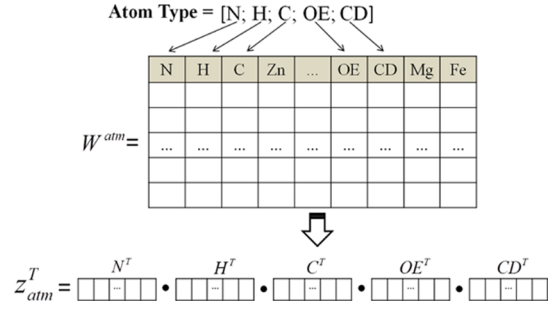


Figure 3. Illustration of the construction of the atom type representation (z_{atm}) for the context of the atom N3 from the compound THM. The solid black circle symbol (•) indicates the concatenation operation.

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

embedding of an atom type from W^{atm} consists of a simple lookup table operation. Therefore, the order of the atom-type embeddings (columns) in W^{atm} is arbitrary and has no influence on the result. However, the order in which the embeddings of the atom types are concatenated to form z_{atm} does matter. We always concatenate first the embeddings from atom types of the ligand, from the closest to the farthest, and then we concatenate the embeddings from atom types of the protein, from the closest to the farthest.

Similarly, the z_{dist}, z_{chrg}, and z_{amino} vectors are created from values of distance, charge, and amino acid types in the context of the target atom. Values of the basic features charge and distance must be discretized before being used as input for the network. We define minimum and maximum values, c_{min} and c_{max}, respectively, to perform the discretization of charge values. Bins equally distanced by 0.05 between minimum and maximum are built. For instance, with $c_{min} = -1$ and $c_{max} = 1$, there will be 40 bins. Similarly, to discretize distance values, bins equally distributed by 0.3 Å will be defined in the range between d_{tmin} and d_{tmax}. For example, with $d_{tmin} = 0$ and $d_{tmax} = 5.1$ Å, there will be 18 bins.

Finally, the representation of the context of the atom a is defined as $z_a = \{z_{atm}; z_{dist}; z_{chrg}; z_{amino}\}$, comprising the concatenation of the vectors previously described. Our hypothesis is that from the basic contextual features, the network can learn more abstract features (the *embeddings*) that are informative about the discrimination between compound and decoys. This type of strategy, where basic features (words) are transformed to more abstract features (word embeddings), has obtained enormous success in the field of Natural Language Processing (LNP).[26−30]

Representation of the Protein-Compound Complex. The second hidden layer in DeepVS is a convolutional layer, responsible for (1) extracting more abstract features from the representations of all atom contexts in the compound, and (2) summarizing this information in a fixed-length vector r. We call the vector r the *representation of the compound-protein complex*, and it is the output of the convolutional layer.

The subjacent goal in using a convolutional layer is its ability to address inputs of variable sizes.[31] In the case of virtual screening, different compounds can have a different number of atoms. Therefore, the number of representations of atom contexts can differ for different complexes. In DeepVS, the convolutional layer allows the processing of complexes of different sizes.

Given a complex x, whose compound is composed of m atoms, the input to the convolutional layer is a list of vectors $\{z_1, z_2, ..., z_m\}$, where z_i is the representation of the context of the ith atom in the compound. In the first stage of the convolutional layer, the generation of more abstract features from each vector z_i is carried out, according to

$$u_i = f(W^1 z_i + b^1) \tag{1}$$

where $W^1 \in \mathbb{R}^{cf \times |z_i|}$ is the weight matrix corresponding to the convolutional layer, b^1 is a bias term, f is the hyperbolic tangent function and $u_i \in \mathbb{R}^{cf}$ corresponds to the resulting feature vector. The number of units (also called filters) in the convolutional layer (cf) is a hyperparameter defined by the user.

The second stage in the convolutional layer, also known as pooling layer, summarizes the features from the various atom contexts. The input consists of a set of vectors $\{u_1, u_2, ..., u_m\}$. For the DeepVS, we use a max-pooling layer, which produces a vector $r \in \mathbb{R}^{cf}$, where the value of the jth element is defined as the maximum of the jth elements of the set of input vectors, i.e.,

$$[r]_j = \max_{1 \le i \le m} [u_i]_j \tag{2}$$

The resulting vector r from this stage is the representation of the compound-protein complex (eq 2). In this way, the network can learn to generate a vector representation that summarizes the information from the complex that is relevant to discriminate between ligands and decoys.

Scoring of Compound-Protein Complex. The vector r is processed by two usual neural network layers: a third hidden layer that extracts one more level of representation, and an output layer, which computes a score for each one of the two possible classifications of the complex: (0) inactive compound and (1) active compound. Formally, given the representation r generated for the complex x, the third hidden layer and the output layer execute the following operation:

$$s(x) = W^3(W^2 r + b^2) + b^3 \tag{3}$$

where W^2 is the weight matrix of the third hidden layer ($W^2 \in \mathbb{R}^{h \times |cf|}$), $W^2 \in \mathbb{R}^{2 \times |h|}$ is the weight matrix of the output layer, and $b^2 \in \mathbb{R}^h$ and $b^3 \in \mathbb{R}^2$ are bias terms. The number of units in the hidden layer, h, is a hyperparameter that is defined by the user. $s(x) \in \mathbb{R}^2$ is a vector containing the score for each of the two classes.

Let $s(x)_0$ and $s(x)_1$ be the scores for classes 0 and 1, respectively. We transform these scores in a probability distribution using the *softmax* function, as follows:

$$p(0|x) = \frac{e^{s(x)_0}}{e^{s(x)_0} + e^{s(x)_1}} \tag{4}$$

$$p(1|x) = \frac{e^{s(x)_1}}{e^{s(x)_0} + e^{s(x)_1}} \tag{5}$$

where we interpret $p(0|x)$ and $p(1|x)$ as the conditional probabilities of the compound be a decoy or a ligand, respectively, given the compound-protein complex data acquired from docking.

The likelihood of class 1 (active ligand) is the scoring used to rank ligands during Virtual Screening essays. The larger the scoring, the greater the chance the compound is an active ligand.

Training DeepVS. The common approach for training neural networks is the stochastic gradient descent (SGD) algorithm.[32] In our case, SGD is used to minimize a loss function over a training set D that contains both complexes of ligands and decoys. At each iteration, a new complex $(x, y) \in D$ is randomly chosen, where $y = 1$ if the complex contains an active ligand and $y = 0$ otherwise. Next, the DeepVS network with parameter set $\theta = \{W^{atm}, W^{chrg}, W^{dist}, W^{amino}, W^1, b^1, W^2, b^2, W^3, b^3\}$ is used to estimate the probability $p(y|x, \theta)$. Finally, the prediction error is computed as the negative log-likelihood, $-\log(p(y|x, \theta))$, and the network parameters are updated applying the backpropagation algorithm.[32] In other words, the set of parameters θ of the network is learned by using SGD to select a set of values that minimize the loss function with respect to θ:

$$\theta \longmapsto \sum_{(x,y) \in D} -\log p(y|x, \theta) \tag{6}$$

In our experiments, we applied SGD with minibatches, which means that, instead of considering only one compound-protein

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

complex at each iteration, we consider a small set of m_s randomly selected complexes and used the average prediction loss to perform backpropagation. We set $m_s = 20$ in our cases. Also, we used Theano[33] to implement DeepVS and perform all the experiments reported in this work.

Experimental Setup. *Dataset.* We used the Directory of Useful Decoys (DUD)[34] as a benchmark to evaluate our deep-learning-based virtual screening approach. One of the main reasons for using this dataset was the possibility of comparing our results with those from systems previously proposed for revalidation of scoring functions and reclassification in virtual screening.[7,11,24] There is a problem in the partial charges of the original version of DUD that makes it trivial to discriminate between ligands and decoys. Therefore, we use the version of DUD produced by Armstrong et al.,[35] which contains corrected atomic partial charges.

The DUD dataset has been developed specifically to validate VS methods in a rigorous way. The dataset is composed of 40 receptors distributed in six different biological groups: hormone nuclear receptors, kinases, serine proteases, metalloenzymes, flavoenzymes, and other classes of enzymes.[34] Also, it possesses 2950 annotated ligands and 95 316 decoys, which translates to a ratio of 36 decoys for each annotated ligand. Each of the 36 decoys was retrieved from the ZINC databank, to mimic some physical property of the associated ligand, such as molecular weight, cLogP, and the number of hydrogen-bond groups, although differing in its topology.[7,34]

Docking Programs. In this work, we used two different computer programs to perform molecular docking: Dock 6.6[36] and Autodock Vina1.1.2.[37] Both are open access and widely used in academia to perform VS. Dock 6.6 offers physics-based energy score functions based on force fields and scoring functions (GRID score and AMBER score).[36] Autodock Vina1.1.2 applies a hybrid scoring function that combines characteristics of knowledge-based and empiric scoring functions (Vina score).[37]

Dock 6.6 Setup. Protein and compound structures were prepared using computational tools from Chimera.[38] Receptors were prepared using the DockPrep module from Dock 6.6. Ligands, nonstructural ions, solvent molecules, and cofactors were removed. Absent atoms were added, and Gasteiger atomic partial charges were applied. Input files were mol2 formatted, except those receptors that were used to calculate molecular surface, in which H atoms were removed, and were finally saved in PDB format.[38] Spheres were created with radius in the range 13.5−15.5 Å, varying according to the size of the active site of the receptor. Box and grid parameters were taken directly from the DUD. The docking procedure was performed according to an available script provided by the Chimera program.[36]

Autodock Vina1.1.2 Setup. Receptors and compound were prepared following default protocols and Gasteiger atomic partial charges were applied.[37,39] A cubic grid with an edge dimension of 27 Å was defined. The center of the grid box coincided with the center of mass of the ligand. Docking runs were performed following the default settings defined in AutoDockTools.[39] The only hyperparameter that we changed was the global search exhaustiveness, which we set to a value of 16, as described in the work of Arciniega and Lange.[39] Note that, although Autodock Vina1.1.2 can output more than one pose, in our experiments, we only consider just one, which corresponded to the pose that Autodock Vina1.1.2 outputs as the best one.

Evaluation Approach. The performance of the proposed method is assessed using leave-one-out cross-validation with the 40 proteins from the DUD dataset. Figure 4 illustrates the process we follow to perform our leave-one-out cross-validation experiments. First, we applied either DOCK6.6 or Autodock Vina1.1.2 for each protein and its respective set of ligands and decoys to generate the docked protein-compound complexes. Next, in each run, one receptor was left out of the test set while the others were employed as the training set.

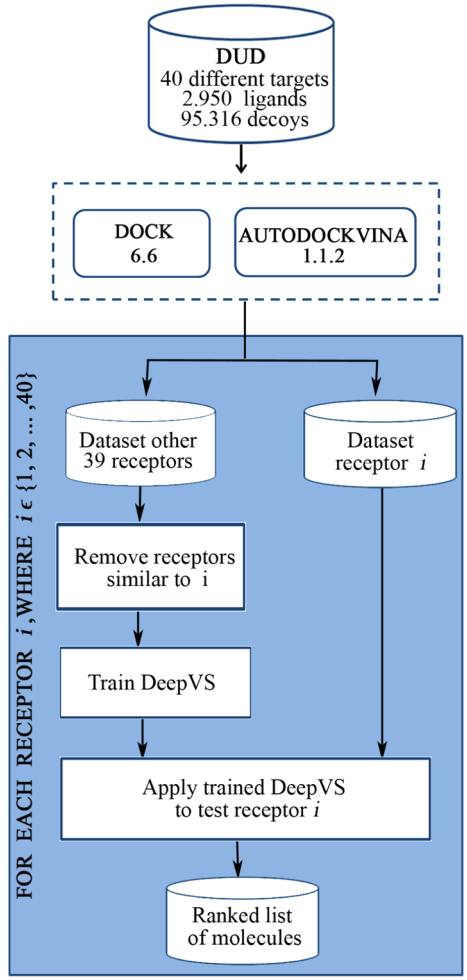


Figure 4. Leave-one-out cross-validation.

To avoid distortions in the performance results of DeepVS, it was essential to remove all receptors similar to that used as a test in a specific cross-validation iteration from the training set. Following Arciniega and Lange,[7] we regarded those sharing the same biological class or those with reported positive cross enrichment as similar receptors.[34] Once the network was trained, it was applied to the test receptor, producing a scoring for each of the potential ligands. Such scoring was used to rank the set compounds. The ranking was validated using metrics that indicate the algorithm's performance.

DeepVS Hyperparameters. The main advantage in using leave-one-out cross-validation is the possibility of tuning the neural network hyperparameters without being very concerned with overfitting. In fact, leave-one-out cross-validation is a suitable method for tuning hyperparameters of ML algorithms when the dataset is small.[40] In our experiments, we used the same set of hyperparameters for the 40 leave-one-out

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

cross-validation iterations. This is equivalent to performing 40 different experiments with different training/test sets using the same configuration for DeepVS. The hyperparameter values that provided our best results and were used in our experiments with both Autodock Vina1.1.2 and Dock 6.6 are specified in Table 1. Note that our evaluation approach was stricter than

Table 1. Hyperparameter Values for DeepVS Used To Train the Network

hyperparameter	description	value
d^{atm}	atom type embedding size	200
d^{amino}	amino acid embedding size	200
d^{chrg}	charge embedding size	200
d^{dist}	distance embedding size	200
cf	number of convolutional filters	400
h	number of hidden units	50
λ	learning rate	0.075
k_c	number of neighboring atoms from compound	6
k_p	number of neighboring atoms from protein	2

that used by Arciniega and Lange,[7] because they tuned the hyper-parameters using a hold-out set at each leave-one-out iteration.

In the next section, we present some experimental results that detail the difference in performance when we vary some of the main hyperparameters of DeepVS.

Evaluation Metrics. To validate DeepVS performance and compare it with other methods previously published in the literature, we used two well-established VS performance metrics: the enrichment factor (EF) and the area under the Receiver Operating Characteristic (ROC) curve.[41,42]

ROC curves are a way to represent the relationship between the selectivity (Se) and specificity (Sp) along a range of continuous values (eqs 7 and 8). It represents the ratio of true positives in function of the false positives.

$$\mathrm{Se} = \frac{\text{true positives}}{\text{total actives}} \tag{7}$$

$$\mathrm{Sp} = \frac{\text{true negatives}}{\text{total decoys}} \tag{8}$$

The area under the ROC curve (AUC) represents a quantification of the curve and facilitates the comparison of results. The AUC is calculated as given in eq 9, where N_{actives} depicts the number of actives, N_{decoys} represents the number of decoys, and $N^i_{\text{decoys\_seen}}$ describes the number of decoys that are higher ranked than the ith active structure.[41] An AUC value of $\leq$0.50 indicates a random selection, whereas an AUC value of 1.0 indicates the perfect identification of active compounds.

$$\mathrm{AUC} = 1 - \frac{1}{N_{\text{actives}}} \sum_{i}^{N_{\text{actives}}} \frac{N^i_{\text{decoys\_seen}}}{N_{\text{decoys}}} \tag{9}$$

Given that the set of compounds are ranked by score, the enrichment factor (EF) at x% (eq 10) is an indicator of how good the set formed by the top x% ranked compounds is, compared to a set of an equal size selected at random from the entire set of compounds.[41,42] The EF value is computed as

$$\mathrm{EF}_{x\%} = \frac{\text{actives at } x\%}{\text{compounds at } x\%} \times \frac{\text{total compounds}}{\text{total actives}} \tag{10}$$

RESULTS AND DISCUSSION

DeepVS vs Docking Programs. In Table 2 we report, for each of the 40 receptors in the DUD, the virtual screening performance for Dock 6.6, Autodock Vina1.1.2 (henceforth referenced as ADV), DeepVS using Dock 6.6 output (DeepVS-Dock) and DeepVS using AVD output (DeepVS-ADV). For each system, we report the AUC ROC and the enrichment factor (EF) at 2% and 20%. We also report the maximum enrichment factor (EF_{max}), which is the maximum EF value that can be achieved for a given ranked list of compounds. Among the four systems, DeepVS-ADV achieved the best average result for AUC, $\mathrm{EF}_{2\%}$, and $\mathrm{EF}_{20\%}$. DeepVS-ADV had the best AUC for 20 of the 40 DUD receptors.

Overall, the quality of the docking output impacts the performance of DeepVS, which is expected. The average performance of DeepVS-ADV, which had a better docking input from ADV (average AUC of 0.62) produced better AUC, $\mathrm{EF}_{2\%}$, $\mathrm{EF}_{20\%}$, and EF_{max} values than DeepVS-Dock, whose input is based on DOCK6.6 (average AUC of 0.48). On the other hand, there were some cases where the AUC of docking program was very poor, but DeepVS was able to boost the AUC result significantly. For instance, although DOCK6.6 produced an AUC of <0.40 for the receptors AR, COX1, HSP90, InhA, PDE5, PDGFrb, and PR, DeepVS-Dock resulted in an AUC of >0.70 for these receptors.

In Figure 5, we compare the AUC of DeepVS-ADV and ADV for the 40 receptors. DeepVS-ADV achieved an AUC of >0.70 for 33 receptors, while this number was only 13 for ADV. The number of receptors with AUC < 0.50 was 2 for DeepVS-ADV and 9 for ADV. The AUC of DeepVS-ADV was higher than that of ADV for 31 receptors. On average, the AUC of DeepVS-ADV (0.81) was 31% better than that for ADV (0.62). In addition, when we selected 20% of the data according to the ranked compounds, on average, the EF value of the DeepVS-ADV (3.1) was 55% larger than the EF value of the ADV (2.0).

A comparison of the AUC from DeepVS-Dock and Dock 6.6 for the 40 receptors is presented in Figure 6. On average, the AUC of DeepVS-Dock (0.74) was 54% better than that of Dock 6.6 (0.48). While Dock 6.6 achieved AUC > 0.70 for 10% (4) of the receptors only, DeepVS-Dock reached AUC > 0.70 for 68% (27) of the receptors. The number of receptors with AUC < 0.50 was 5 for DeepVS-Dock and 23 for Dock 6.6. The AUC of DeepVS-Dock was higher than that for Dock 6.6 for 36 receptors. Finally, when we select 20% of the data according to the ranked compounds, on average, the EF value of DeepVS-Dock (3.0) was more than two times larger than the Ef value from Dock 6.6 (1.3).

The experimental results presented in this section, which include outputs from two different docking programs, are strong evidence that DeepVS can be an effective approach for improving docking-based virtual screening.

DeepVS Sensitivity to Hyperparameters. In this section, we present experimental results regarding an investigation on the sensitivity of DeepVS concerning the main hyperparameters. In our experiments, we used the ADV output as input to DeepVS. Therefore, all results reported in this section were generated using DeepVS-ADV. However, we noticed that DeepVS behaved in a similar manner when the input from Dock 6.6 was used. In the experiments, we varied one of the hyperparameters and fixed all the others to the following values: $d^{atm/amino/chrg/dist}$ = 200, cf = 400, λ = 0.075, k_c = 6, and k_p = 2.

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

Table 2. Virtual Screening AUC ROC and Enrichment Factor (EF) Results for Dock 6.6, Autodock Vina1.1.2, and DeepVS

	Dock				DeepVS-Dock				ADV				DeepVS-ADV			
	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
average	20.1	5.3	1.3	0.48	16.9	5.9	3.0	0.74	16.4	6.0	2.0	0.62	16.0	6.6	3.1	**0.81**
ACE	36.2	3.0	0.6	0.41	3.6	2.0	2.7	**0.75**	9.1	3.0	1.4	0.38	2.2	1.0	1.8	0.72
AChE	12.1	0.5	0.8	0.50	1.6	0.5	0.8	0.48	5.4	4.8	3.0	**0.68**	4.0	1.9	1.1	0.51
ADA	24.5	13.0	3.3	0.86	8.5	6.5	4.1	**0.87**	1.1	0.0	1.1	0.47	9.2	2.2	3.3	0.83
ALR2	1.0	0.0	0.2	0.38	1.7	0.0	1.3	0.56	8.2	3.8	2.5	**0.70**	5.7	3.8	1.3	0.66
AmpC	5.1	4.8	1.4	**0.57**	1.3	0.0	0.0	0.44	1.0	0.0	0.2	0.23	1.1	0.0	0.0	0.42
AR	7.3	0.7	0.1	0.25	36.5	18.3	4.1	0.80	36.5	14.2	3.5	0.74	21.9	11.5	4.2	**0.88**
CDK2	36.6	11.9	2.1	0.56	11.4	8.9	3.7	**0.82**	18.3	8.9	2.1	0.66	6.1	4.0	2.9	0.79
COMT	20.0	17.8	2.7	0.64	40.1	13.4	3.2	0.88	40.1	8.9	1.4	0.41	20.0	8.9	4.6	**0.92**
COX1	1.1	0.0	0.2	0.35	35.0	8.2	3.0	0.75	20.0	10.3	3.6	**0.79**	5.0	2.1	2.8	0.77
COX2	1.3	0.7	1.2	0.58	18.4	12.4	3.6	0.78	36.8	21.4	3.8	0.84	36.8	12.7	4.3	**0.91**
DHFR	36.5	3.7	1.3	0.48	18.3	10.4	4.5	0.88	36.5	8.2	3.5	0.86	9.1	6.7	4.8	**0.94**
EGFr	34.5	12.0	1.9	0.49	23.0	8.4	4.6	**0.93**	4.9	2.5	1.5	0.58	8.6	5.5	3.6	0.86
$ER_{agonist}$	18.1	2.3	1.0	0.43	6.5	0.8	3.5	0.75	17.7	16.6	3.3	0.79	8.1	6.0	3.9	**0.88**
$ER_{antagonist}$	7.1	5.9	2.2	0.68	7.1	5.9	4.0	**0.90**	13.8	8.9	2.3	0.66	7.4	3.8	3.8	0.88
FGFrl	36.6	8.9	2.1	0.49	15.7	8.1	4.6	**0.91**	1.1	0.0	1.0	0.48	36.6	7.7	3.3	0.85
FXa	36.9	3.2	1.1	0.49	2.0	0.4	1.7	0.71	3.2	2.1	1.5	0.66	4.3	1.4	3.9	**0.86**
GART	10.5	7.4	4.5	0.90	12.3	4.9	4.8	**0.92**	2.9	0.0	2.6	0.77	2.6	0.0	2.4	0.77
GPB	1.0	0.0	0.1	0.23	2.7	1.9	1.2	0.51	3.1	2.9	1.2	**0.52**	1.0	0.0	0.9	0.42
GR	18.4	1.9	0.3	0.22	11.1	7.6	2.2	0.49	18.4	4.4	1.2	0.57	20.3	10.8	4.4	**0.91**
HIVPR	1.0	0.0	0.4	0.16	4.1	0.9	2.1	0.51	36.6	6.6	2.6	0.72	6.2	5.6	4.1	**0.88**
HIVRT	36.9	4.9	1.0	0.40	36.9	6.2	2.8	0.69	24.6	6.2	1.8	0.64	7.4	4.9	2.3	**0.73**
HMGR	18.2	4.2	0.4	0.16	36.5	2.8	1.0	0.24	1.0	0.0	0.6	0.42	36.5	19.6	4.9	**0.96**
HSP90	2.3	0.0	1.0	0.39	5.8	2.0	3.3	0.74	1.3	0.0	0.8	0.54	10.0	4.1	5.0	**0.94**
InhA	36.7	5.3	1.2	0.38	36.7	13.0	4.3	0.90	36.7	11.2	1.8	0.54	6.7	16.6	4.5	**0.94**
MR	18.3	3.3	1.0	0.36	14.7	6.7	2.3	0.55	36.7	20.0	4.0	**0.82**	24.4	16.7	3.3	**0.82**
NA	36.6	13.2	3.3	0.73	3.5	0.0	2.8	**0.78**	1.1	0.0	0.5	0.40	2.1	0.0	1.0	0.68
P38MAP	33.8	11.1	2.2	0.62	33.8	16.0	4.2	**0.91**	2.8	1.4	2.2	0.62	21.6	16.4	3.9	0.87
PARP	36.6	10.7	1.5	0.51	2.9	0.0	0.8	0.63	36.6	4.6	2.9	**0.74**	1.7	0.0	0.8	0.65
PDE5	36.5	3.0	0.9	0.39	12.2	3.9	2.9	0.75	36.5	6.9	1.7	0.57	36.5	8.9	3.1	**0.86**
PDGFrb	36.8	5.4	1.2	0.38	17.4	14.4	4.5	**0.92**	36.8	5.4	1.1	0.48	36.8	19.2	3.7	0.91
PNP	4.8	4.0	0.6	0.44	10.3	2.0	4.8	**0.94**	3.0	2.0	1.8	0.66	4.2	0.0	4.0	0.86
PPARg	2.6	1.8	0.4	0.49	36.9	1.2	2.0	0.79	4.1	3.1	1.7	0.64	36.9	2.5	4.4	**0.87**
PR	3.1	1.8	0.6	0.33	4.8	1.8	3.3	0.70	1.5	0.0	1.1	0.43	8.5	5.5	2.4	**0.77**
RXRa	36.4	12.1	2.0	0.73	6.4	4.9	4.2	0.91	36.4	24.3	4.2	**0.93**	12.1	9.7	3.0	0.85
SAHH	1.7	1.5	1.1	0.50	19.3	15.1	4.7	0.94	13.5	10.5	3.2	0.80	19.9	18.1	4.7	**0.95**
SRC	38.4	12.3	1.8	0.44	25.6	12.9	3.9	**0.88**	4.0	2.9	2.2	0.69	19.2	9.7	3.4	0.85
thrombin	3.6	2.3	1.4	0.60	36.3	5.4	1.1	0.59	18.1	6.2	2.8	0.73	36.3	6.2	3.2	**0.83**
TK	2.0	0.0	1.4	**0.63**	1.2	0.0	0.5	0.44	1.5	0.0	0.2	0.55	1.4	0.0	0.2	0.54
trypsin	36.1	4.5	1.6	0.51	18.0	5.6	1.9	0.65	9.8	3.4	1.6	0.63	36.1	6.8	2.4	**0.80**
VEGFr2	36.7	10.9	1.4	0.43	9.7	6.1	3.8	0.88	36.7	5.4	1.6	0.56	36.7	4.8	4.1	**0.90**

In Table 3, we present the experimental results of varying the basic feature embedding sizes. We can see that embedding sizes of >50 mainly improved the EF. Embedding sizes of >200 did not improve the results.

The experimental results of varying the number of filters in the convolutional layer (cf) are presented in Table 4. Notice that the AUC improves by increasing the number of convolutional filters up to 400. On the other hand, using cf = 200 results in the best $EF_{2\%}$ value.

In Table 5, we present the experimental results of DeepVS trained with different learning rates. We can see that larger learning rates work better for the DUD dataset. A learning rate of 0.1 resulted in the best outcomes, in terms of AUC and EF.

We also investigated the impact of using a different number of neighboring atoms from the compound (k_c) and the receptor (k_p). In Table 6, we present some experimental results where we vary both k_c and k_p. For instance, with $k_c = 0$ and $k_p = 5$, it means that no information from the compound is used, while information from the five closest atoms from the receptor is used. In the first half of the table, we keep k_p fixed at a value of 5 and vary k_c. In the second half of the table, we keep k_c fixed at a value of 6 and vary k_p. In the first half of Table 6, we note that, by increasing the number of neighboring atoms that we use from the compound (k_c), we significantly increase both EF and AUC. In the second half of the table, we note that using k_p values of >2 degrades DeepVS-ADV performance. As we conjecture in the next section, this behavior seems to be related to the quality of the docking program output.

In order to assess the robustness of DeepVS, with regard to the initialization of the weight matrices (network parameters), we performed 10 different runs of DeepVS-ADV using a different random seed in each run. In Table 7, we present the

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

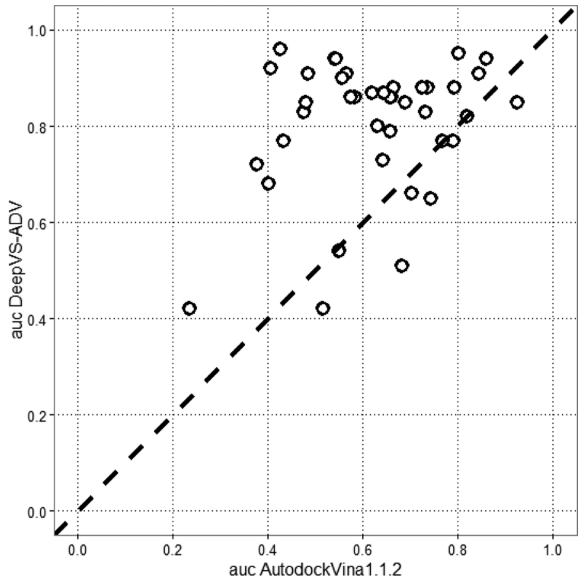


Figure 5. DeepVS-ADV vs Autodock Vina1.1.2 AUC. The circles represent each of the 40 DUD receptors.

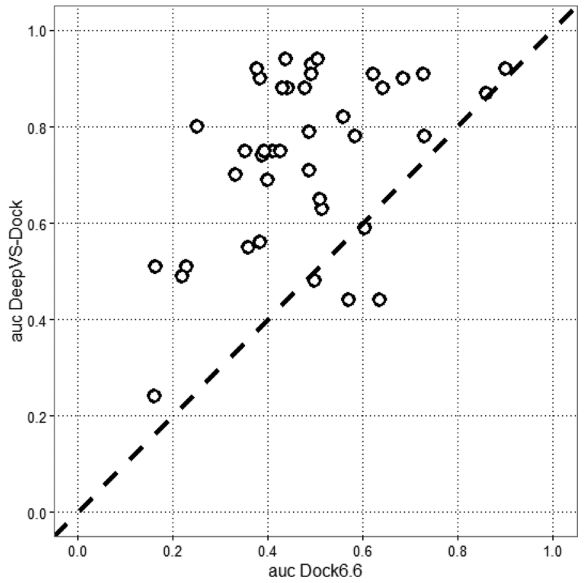


Figure 6. DeepVS-Dock vs DOCK6.6 AUC. The circles represent each of the 40 DUD receptors.

Table 3. DeepVS Sensitivity to Basic Feature Embedding Sizes

$d^{atm/amino/chrg/dist}$	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
50	15.4	6.5	3.0	0.803
100	15.6	6.8	3.1	0.799
200	16.0	6.6	3.1	0.807

Table 4. DeepVS Sensitivity to the Number of Convolutional Layer Filters (cf) in the Convolutional Layer

cf	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
100	14.7	6.4	3.0	0.797
200	16.3	7.3	3.0	0.799
400	16.0	6.6	3.1	0.807
800	16.8	7.0	3.1	0.804

Table 5. DeepVS Sensitivity to the Learning Rate

λ	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
0.1	17.3	6.9	3.2	0.809
0.075	16.0	6.6	3.1	0.807
0.05	15.5	6.6	3.0	0.801
0.025	14.7	6.4	3.0	0.795
0.01	15.7	6.2	3.1	0.800

Table 6. DeepVS Sensitivity to the Number of Neighboring Atoms Selected from the Protein-Compound

k_c	k_p	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
0	5	6.99	2.18	1.54	0.574
1	5	15.55	4.22	2.27	0.697
2	5	15.44	5.62	2.59	0.743
3	5	16.74	6.38	2.76	0.752
4	5	17.38	6.25	2.91	0.782
5	5	19.07	6.47	3.18	0.799
6	5	17.89	6.79	3.04	0.799
6	4	17.02	6.38	3.17	0.801
6	3	16.44	6.82	2.99	0.793
6	2	16.03	6.62	3.14	0.807
6	1	16.03	6.99	3.13	0.806
6	0	16.92	6.95	3.06	0.803

Table 7. DeepVS Sensitivity to Different Random Seeds

run	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
average	15.97	6.82	3.10	0.805
stadard deviation	0.73	0.16	0.04	0.003
1	16.53	6.88	3.19	0.807
2	16.32	6.63	3.06	0.799
3	15.30	6.76	3.10	0.805
4	14.92	6.89	3.12	0.807
5	17.08	6.74	3.10	0.807
6	16.82	6.84	3.05	0.804
7	15.90	6.52	3.07	0.803
8	15.59	7.00	3.06	0.805
9	16.08	6.86	3.12	0.806
10	15.18	7.04	3.11	0.803

Table 8. DeepVS-ADV Results for Proteins with Good Docking Quality

	AUC > 0.75		
	$k_p = 0$	$k_p = 2$	$k_p = 5$
average	0.83	0.86	0.87
COX1	0.78	0.77	0.80
COX2	0.89	0.91	0.91
DHFR	0.96	0.94	0.96
$ER_{agonist}$	0.89	0.88	0.89
GART	0.78	0.77	0.77
MR	0.80	0.82	0.80
RXRa	0.64	0.85	0.90
SAHH	0.93	0.95	0.95

experimental results of these 10 runs. This table shows that the standard deviation is very small for both EF and AUC, which demonstrates the robustness of DeepVS to different random seeds.

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

Table 9. DeepVS-ADV Results for Proteins with Poor Docking Quality

	AUC < 0.50		
	$k_p = 0$	$k_p = 2$	$k_p = 5$
average	0.80	0.78	0.77
ACE	0.71	0.72	0.66
ADA	0.80	0.83	0.83
AmpC	0.59	0.42	0.46
COMT	0.92	0.92	0.89
FGFrl	0.83	0.85	0.79
HMGR	0.97	0.96	0.96
NA	0.67	0.68	0.66
PDGFrb	0.91	0.91	0.91
PR	0.81	0.77	0.79

Table 10. Target Exchange Experiment

	HIVPR		HSP90	
	ADV	DeepVS	ADV	DeepVS
HIVPR	0.72	0.88	0.52	0.75
HSP90	0.31	0.82	0.54	0.94
	EGFr		$ER_{agonist}$	
	ADV	DeepVS	ADV	DeepVS
EGFr	0.58	0.86	0.33	0.80
$ER_{agonist}$	0.71	0.86	0.79	0.88

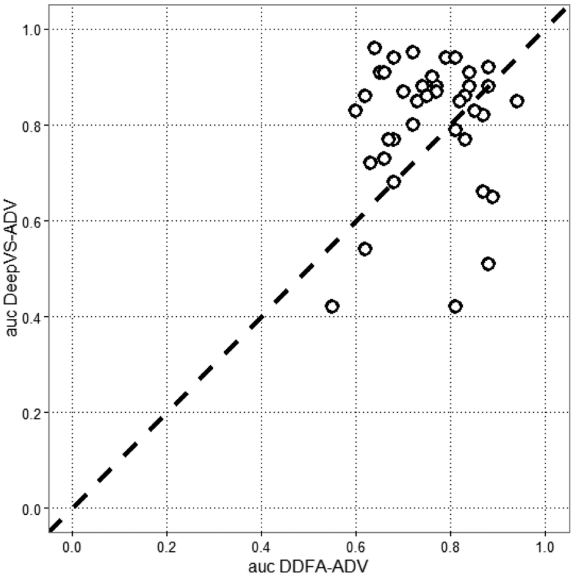


Figure 7. DeepVS-ADV vs DDFA-ADV.

Docking Quality Versus DeepVS Performance. In the experimental results reported in Table 6, we note that, when creating the atom contexts, using k_p values of >2 (the number of neighboring atoms coming from the protein) does not lead to improved AUC or EF values. In fact, if we use only information from the compound ($k_p = 0$), which is equivalent to perform ligand-based virtual screening, the AUC is already very good (0.803). We hypothesize that this behavior is related to the quality of the docking output, which varies greatly across the 40 DUD proteins. When attempting to test this hypothesis, we separately analyze the AUC of DeepVS for the DUD proteins for which the AUC of ADV is good (Table 8) or poor (Table 9).

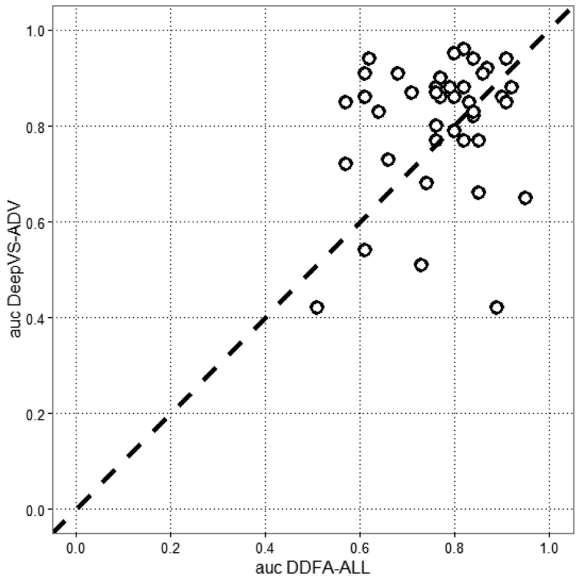


Figure 8. DeepVS-ADV vs DDFA-ALL.

Table 11. Reported Performance of Different Systems on the DUD

system	AUC	ref
DeepVS-ADV	**0.81**	
ICM[a]	0.79	43
NNScore1-ADV[b]	0.78	11
Glide SP[a]	0.77	44
DDFA-ALL	0.77	7
DDFA-RL	0.76	7
NNScore2-ADV[b]	0.76	11
DDFA-ADV	0.75	7
DeepVS-Dock	**0.74**	
DDFA-AD4	0.74	7
Glide HTVS[b]	0.73	11
Surflex[a]	0.72	44
Glide HTVS	0.72	44
ICM	0.71	43
RAW-ALL	0.70	7
Autodock Vina[b]	0.70	11
Surflex	0.66	44
Rosetta Ligand	0.65	7
Autodock Vina	0.64	7
ICM	0.63	44
Autodock Vina	**0.62**	
FlexX	0.61	44
Autodock4.2	0.60	7
PhDOCK	0.59	44
Dock4.0	0.55	44
Dock 6.6	**0.48**	

[a]Tuned by expert knowledge. [b]Determined using a different dataset of decoys.

In Table 8, we present the AUC of DeepVS-ADV for proteins whose AUC of ADV is larger than 0.75. We present results for three different values of k_p, namely, $k_p = 0$, 2, and 5; in the three experiments, we use $k_c = 6$. We can notice that for proteins that have a good docking quality (and likely the protein-compound complexes have good structural information), the average AUC of DeepVS-ADV increases as we

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

Table 12. Virtual Screening AUC ROC and Enrichment Factor (EF) Results for Autodock Vina1.1.2 and DeepVS Using the DUD-E Dataset

	ADV				DeepVS-ADV			
	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC	EF_{max}	$EF_{2\%}$	$EF_{20\%}$	AUC
average	21.0	6.0	2.4	0.68	17.9	6.2	3.0	0.77
AA2AR	1.4	0.6	1.1	0.56	5.3	4.7	3.2	0.81
CP3A4	23.5	2.7	1.6	0.61	23.5	4.1	3.0	0.81
FABP4	44.6	15.9	3.4	0.78	44.6	3.2	3.3	0.82
GRIK1	11.0	4.0	2.2	0.60	32.9	13.4	2.6	0.72
JAK2	61.7	9.8	3.1	0.78	18.5	8.4	4.1	0.90
MMP13	11.0	4.5	2.3	0.65	2.6	2.4	1.1	0.52
NOS1	4.1	2.5	1.6	0.59	5.5	4.5	2.4	0.68
PPARA	10.4	8.2	4.0	0.86	10.3	9.1	4.4	0.90

increase k_p. This result suggests that if the structural information is good, the neural network can benefit from it.

In Table 9, we present the AUC of DeepVS-ADV for proteins whose AUC value of ADV is <0.5. For these proteins, which have a poor docking quality (and likely the protein-compound complexes have poor structural information), the average AUC of DeepVS-ADV decreases as we increase k_p. This result suggests that, if the structural information is poor, the neural network works better without using it.

In Table 10, we present the results of an experiment where we exchanged the target protein for a given set of ligands and decoys. We perform this experiment with two pairs of proteins, where each protein in a pair is from a different class. The main purpose of this experiment is also to show that DeepVS leverages the protein-compound structural information. As we can see in Table 10, the AUC values of both Autodock Vina (ADV) and DeepVS decrease significantly when the ligands and decoys of a different target are used.

Comparison with State-of-the-Art Systems. In this section, we compare DeepVS results with those reported in previous work that also has employed DUD. First, we perform a detailed comparison between DeepVS and the Docking Data Feature Analysis (DDFA) system,[7] which also applied neural networks on the output of docking programs to perform virtual screening. Next, we compare the average AUC of DeepVS with one of other systems that also report results for the 40 DUD receptors.

DDFA uses a set of human-defined features that are derived from docking output data. Examples of the features employed in DDFA are compound efficiency, the best docking score of the compound poses and the weighted average of the best docking scores of the five most similar compounds in the docked library. The features are given as input to a shallow neural network that classifies the input protein-compound complex as an active or inactive ligand. DDFA uses data from the six best poses output by the docking program, whereas in DeepVS, we use data from the best pose only. In Figure 7, we compare the AUC of DeepVS-ADV vs DDFA-ADV, which is the version of DDFA that uses the output of Autodock Vina. In this figure, each circle represents one of the 40 DUD receptors. DeepVS-ADV produces higher AUC than DDFA-ADV for 27 receptors, which represents 67.5% of the dataset.

DDFA-ALL is a more robust version of DDFA[7] that simultaneously uses the output of three different docking programs: Autodock4.2 (AD4), Autodock Vina1.1.2 (ADV), and Rosetta Ligand 3.4 (RL). Therefore, DDFA-ALL uses three times more input features than DDFA-ADV. In Figure 8, we compare the AUC of DeepVS-ADV vs DDFA-ALL. Despite using data from one docking program only, DeepVS-ADV produces higher AUC than DDFA-ALL for 25 receptors, which represents 62.5% of the dataset. This is a strong indication that the DeepVS-ADV results for the DUD dataset is very robust.

In Table 11, we compare the average AUC of DeepVS and the docking programs with those from other systems reported in the literature. DeepVS-ADV produced the best AUC among all systems, outperforming commercial docking softwares ICM and Glide SP. NNScore1-ADV and NNScore2-ADV are also based on shallow neural networks that use human-defined features and the output of Autodock Vina. It is worth to note that NNScore1-ADV and NNScore2-ADV[11] results are based in a different set of decoys that are simpler than those available in the DUD. Therefore, these results are not 100% comparable with other results in the table. To the best of our knowledge, the AUC of DeepVS-ADV is the best value reported so far for VS using the 40 receptors from the DUD.

DeepVS for DUD-E Dataset. We further assessed the robustness of DeepVS by conducting an experiment where we trained DeepVS-ADV using the 40 proteins from the DUD, and applied it to perform virtual screening for 8 proteins randomly selected from the DUD-E dataset.[45] DUD-E is an enhanced version of DUD that contains 102 targets. In DUD-E, each target contains an average number of 224 ligands and 50 decoys per ligand. Each one of the 8 proteins selected for our experiment belong to a different protein class. Moreover, none of the 8 are contained in the set of 40 proteins from the DUD.

In Table 12, we summarize the results of DeepVS-ADV for the DUD-E proteins. DeepVS-ADV improved the AUC of ADV for 7 of the 8 proteins and produced an improvement of 0.09 in the average AUC. On average, DeepVS-ADV also produces better $EF_{2\%}$ and $EF_{20\%}$ values than Autodock Vina1.1.2.

CONCLUSIONS

In this work, we introduce DeepVS, which is a deep learning approach to improve the performance of docking-based virtual screening. Using DeepVS on top of the docking output of Autodock Vina, we were capable of producing the best area under the curve (AUC) reported so far for virtual screening on the DUD dataset. This result, together with the fact that (1) DeepVS does not require human-defined features, and (2) it achieves good results using the output of a single docking program, makes DeepVS an attractive approach for virtual screening. Moreover, different from other methods that use shallow neural networks with few parameters, DeepVS has a larger potential for performance improvement if more data are added to the training set. Deep learning systems are usually trained with large amounts of data. Although the number of

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

protein-compound complexes in the DUD is relatively large (more than 100 000), the number of different proteins is still very small (only 40).

In addition, this work also brings some very innovative ideas on how to model the protein-compound complex raw data to be used in a deep neural network. We introduce the idea of atom and amino acid embeddings, which can also be used in other deep learning applications for bioinformatics. Moreover, our idea of modeling the compound as a set of atom contexts that is further processed by a convolution layer proved to be an effective approach to learning representations of protein-compound complexes.

AUTHOR INFORMATION

Corresponding Authors

*E-mail: janaina.pereira@ioc.fiocruz.br (J. C. Pereira).
*E-mail: ernesto@fiocruz.br (E. R. Caffarena).
*E-mail: cicerons@us.ibm.com (C. N. dos Santos).

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

J.C.P. is funded through a Ph.D. scholarship from the Oswaldo Cruz Foundation. The research of E.R.C research is supported by the following grants: Faperj/E-26/111.401/2013 and CNPq.Papes VI./407741/2012-7.

REFERENCES

(1) Hecht, D.; Fogel, G. B. Computational Intelligence Methods for Docking Scores. *Curr. Comput.-Aided Drug Des.* **2009**, *5*, 56−68.

(2) Walters, W.; Stahl, M. T.; Murcko, M. A. Virtual Screening—An Overview. *Drug Discovery Today* **1998**, *3*, 160−178.

(3) Cheng, T.; Li, Q.; Zhou, Z.; Wang, Y.; Bryant, S. Structure-Based Virtual Screening for Drug Discovery: A Problem-Centric Review. *AAPS J.* **2012**, *14*, 133−141.

(4) Shoichet, B. K. Virtual Screening of Chemical Libraries. *Nature* **2004**, *432*, 862−865.

(5) Ghosh, S.; Nie, A.; An, J.; Huang, Z. Structure-Based Virtual Screening of Chemical Libraries for Drug Discovery. *Curr. Opin. Chem. Biol.* **2006**, *10*, 194−202.

(6) Bissantz, C.; Folkers, G.; Rognan, D. Protein-Based Virtual Screening of Chemical Databases. 1. Evaluation of Different Docking/Scoring Combinations. *J. Med. Chem.* **2000**, *43*, 4759−4767. (PMID: 11123984)

(7) Arciniega, M.; Lange, O. F. Improvement of Virtual Screening Results by Docking Data Feature Analysis. *J. Chem. Inf. Model.* **2014**, *54*, 1401−1411.

(8) Drwal, M. N.; Griffith, R. Combination of Ligand- and Structure-Based Methods in Virtual Screening. *Drug Discovery Today: Technol.* **2013**, *10*, e395−e401.

(9) Schneider, G. Virtual Screening: An Endless Staircase? *Nat. Rev. Drug Discovery* **2010**, *9*, 273−276.

(10) Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. Docking and Scoring in Virtual Screening for Drug Discovery: Methods and Applications. *Nat. Rev. Drug Discovery* **2004**, *3*, 935−949.

(11) Durrant, J. D.; Friedman, A. J.; Rogers, K. E.; McCammon, J. A. Comparing Neural-Network Scoring Functions and the State of the Art: Applications to Common Library Screening. *J. Chem. Inf. Model.* **2013**, *53*, 1726−1735.

(12) Kinnings, S. L.; Liu, N.; Tonge, P. J.; Jackson, R. M.; Xie, L.; Bourne, P. E. A Machine Learning-Based Method to Improve Docking Scoring Functions and Its Application to Drug Repurposing. *J. Chem. Inf. Model.* **2011**, *51*, 408−419.

(13) Durrant, J. D.; McCammon, J. A. NNScore: ANeural-Network-Based Scoring Function for the Characterization of Protein-Ligand Complexes. *J. Chem. Inf. Model.* **2010**, *50*, 1865−1871. (PMID: 20845954)

(14) Ballester, P. J.; Mitchell, J. B. A Machine Learning Approach to Predicting Protein-Ligand Binding Affinity with Applications to Molecular Docking. *Bioinformatics* **2010**, *26*, 1169−1175.

(15) Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *Pattern Anal. Mach. Intell., IEEE Trans.* **2013**, *35*, 1798−1828.

(16) LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436−444.

(17) Bengio, Y. Learning Deep Architectures for AI. *FNT Mach. Learn.* **2009**, *2*, 1−127.

(18) Dahl, G. E.; Jaitly, N.; Salakhutdinov, R. Multi-Task Neural Networks for QSAR Predictions. *arXiv Preprint*, 2014, arXiv:1406.1231.

(19) Unterthiner, T.; Mayr, A.; Klambauer, G.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Hochreiter, S. Deep Learning as an Opportunity in Virtual Screening. In *Proceedings of the Deep Learning Workshop at NIPS*, 2014.

(20) Unterthiner, T.; Mayr, A.; Klambauer, G.; Hochreiter, S. Toxicity Prediction Using Deep Learning. *arXiv Preprint* 2015, arXiv:1503.01445.

(21) Ramsundar, B.; Kearnes, S.; Riley, P.; Webster, D.; Konerding, D.; Pande, V. Massively Multitask Networks for Drug Discovery. *arXiv Preprint*, 2015, arXiv:1502.02072.

(22) Lusci, A.; Pollastri, G.; Baldi, P. Deep Architectures and Deep Learning in Chemoinformatics: the Prediction of Aqueous Solubility for Drug-like Molecules. *J. Chem. Inf. Model.* **2013**, *53*, 1563−1575.

(23) Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gomez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, Montreal, Canada, Dec. 7−12, 2015; pp 2215−2223.

(24) Durrant, J. D.; McCammon, J. A. NNScore 2.0: A Neural-Network Receptor-Ligand Scoring Function. *J. Chem. Inf. Model.* **2011**, *51*, 2897−2903.

(25) Weber, J.; Achenbach, J.; Moser, D.; Proschak, E. VAMMPIRE: a Matched Molecular Pairs Database for Structure-Based Drug Design and Optimization. *J. Med. Chem.* **2013**, *56*, 5203−5207.

(26) Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493−2537.

(27) Socher, R.; Huval, B.; Manning, C. D.; Ng, A. Y. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, July 12-14, 2012 Association for Computational Linguistics Stroudsburg, PA, 2012; pp 1201−1211.

(28) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, Lake Tahoe, NV, Dec. 5−10, 2013; 9 pp.

(29) dos Santos, C. N.; Gatti, M. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland, Aug. 23−29, 2014; pp 69−78.

(30) dos Santos, C. N.; Zadrozny, B. Learning Character-Level Representations for Part-of-Speech Tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014; pp 1818−1826.

(31) Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K. J. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Trans. Acoust., Speech, Signal Process.* **1989**, *37*, 328−339.

(32) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning Representations by Back-Propagating Errors. *Nature* **1986**, *323* (6088), 533−536.

(33) Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; Bengio, Y. Theano: A

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506

CPU and GPU Math Expression Compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010; p 3.
(34) Huang, N.; Shoichet, B. K.; Irwin, J. J. Benchmarking Sets for Molecular Docking. *J. Med. Chem.* **2006**, *49*, 6789−6801.
(35) Armstrong, M. S.; Morris, G. M.; Finn, P. W.; Sharma, R.; Moretti, L.; Cooper, R. I.; Richards, W. G. ElectroShape: Fast Molecular Similarity Calculations Incorporating Shape, Chirality and Electrostatics. *J. Comput.-Aided Mol. Des.* **2010**, *24*, 789−801.
(36) Lang, P. T.; Brozell, S. R.; Mukherjee, S.; Pettersen, E. F.; Meng, E. C.; Thomas, V.; Rizzo, R. C.; Case, D. A.; James, T. L.; Kuntz, I. D. DOCK 6: Combining Techniques to Model RNA-Small Molecule Complexes. *RNA* **2009**, *15*, 1219−1230.
(37) Trott, O.; Olson, A. J. AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading. *J. Comput. Chem.* **2010**, *31*, 455−461.
(38) Pettersen, E. F.; Goddard, T. D.; Huang, C. C.; Couch, G. S.; Greenblatt, D. M.; Meng, E. C.; Ferrin, T. E. UCSF Chimera—A Visualization System for Exploratory Research and Analysis. *J. Comput. Chem.* **2004**, *25*, 1605−1612.
(39) Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *J. Comput. Chem.* **2009**, *30*, 2785−2791.
(40) Arlot, S.; Celisse, A. A Survey of Cross-Validation Procedures for Model Selection. *Stat. Surv.* **2010**, *4*, 40−79.
(41) Jahn, A.; Rosenbaum, L.; Hinselmann, G.; Zell, A. 4D Flexible Atom-Pairs: An Efficient Probabilistic Conformational Space Comparison for Ligand-Based Virtual Screening. *J. Cheminf.* **2011**, *3*, 23.
(42) Nicholls, A. What do We Know and when do We Know It? *J. Comput.-Aided Mol. Des.* **2008**, *22*, 239−255.
(43) Neves, M. A.; Totrov, M.; Abagyan, R. Docking and Scoring with ICM: The Benchmarking Results and Strategies for Improvement. *J. Comput.-Aided Mol. Des.* **2012**, *26*, 675−686.
(44) Cross, J. B.; Thompson, D. C.; Rai, B. K.; Baber, J. C.; Fan, K. Y.; Hu, Y.; Humblet, C. Comparison of Several Molecular Docking Programs: Pose Prediction and Virtual Screening Accuracy. *J. Chem. Inf. Model.* **2009**, *49*, 1455−1474.
(45) Mysinger, M. M.; Carchia, M.; Irwin, J. J.; Shoichet, B. K. Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *J. Med. Chem.* **2012**, *55*, 6582−6594.

DOI: 10.1021/acs.jcim.6b00355
J. Chem. Inf. Model. 2016, 56, 2495−2506