PUCRS

FACULTY OF INFORMATICS
GRADUATE PROGRAM IN COMPUTER SCIENCE
MASTER IN COMPUTER SCIENCE

ALAN DIEGO DOS SANTOS

RANKING LIGANDS IN STRUCTURE-BASED VIRTUAL SCREENING USING SIAMESE NEURAL NETWORKS

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
GRADUATE PROGRAM IN COMPUTER SCIENCE

RANKING LIGANDS IN STRUCTURE-BASED VIRTUAL SCREENING USING SIAMESE NEURAL NETWORKS

ALAN DIEGO DOS SANTOS

Dissertation is submitted in partial fulfillment of requirements for the degree of Master in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Duncan Dubugras Ruiz

Porto Alegre
2017

Ficha Catalográfica

S237 Santos, Alan Diego dos

Ranking Ligands in Structure-Based Virtual Screening Using Siamese Neural Networks / Alan Diego dos Santos . – 2017.
30 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz.

1. Triagem Virtual. 2. Redes Neurais Siamesas. 3. Funções de Escore. 4. Docagem Molecular. I. Ruiz, Duncan Dubugras Alcoba. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Alan Diego dos Santos

Ranking Ligands in Structure-based Virtual Screening Using Siamese Neural Networks

This Dissertation/Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Computer Science of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 29th, 2017.

COMMITTEE MEMBERS:

Prof. Dr. Rafael Andrade Cáceres (UFCSPA)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS)

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PPGCC/PUCRS - Advisor)

I dedicate this work to my family. A special feeling of gratitude to my parents, José and Clair Santos, who supported me in achieving this goal with words of encouragements, and to my brother, who helped me with the text revision. I also dedicate this dissertation to my uncles and aunts, that supported me in during this program, and my friends, specially Rodrigo Silveira and Henrique Rodenbusch, that invited me to bars when I was locked in home studying. And, special feelings to my teachers, who have taught me almost everything I know and forced me into objectives that I thought were impossible to reach.

“Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid.”
(Unknown)

RANQUEANDO LIGANTES EM TRIAGEM VIRTUAL USANDO REDES NEURAIS SIAMESAS

RESUMO

Triagem virtual de bancos de dados de ligantes é amplamente utilizada nos estágios iniciais do processo de descoberta de fármacos. Abordagens computacionais 'docam' uma pequena molécula dentro do sítio ativo de um estrutura biológica alvo e avaliam a afinidade das interações entre a molécula e a estrutura. Todavia, os custos envolvidos ao aplicar algoritmos de docagem molecular em grandes bancos de ligantes são proibitivos, dado a quantidade de recursos computacionais necessários para essa execução. Nesse contexto, estratégias de aprendizagem de máquina podem ser aplicadas para ranquear ligantes baseadas na afinidade com determinada estrutura biológica e, dessa forma, reduzir o número de compostos químicos a serem testados. Nesse trabalho, propomos um modelo para ranquear ligantes baseados na arquitetura de redes neurais siamesas. Esse modelo calcula a compatibilidade entre receptor e ligante usando grades de propriedades bioquímicas. Nós também mostramos que esse modelo pode aprender a identificar interações moleculares importantes entre ligante e receptor. A compatibilidade é calculada baseada em relação à conformação do ligante, independente de sua posição e orientação em relação ao receptor. O modelo proposto foi treinado usando ligantes ativos previamente conhecidos e moléculas chamarizes (decoys) em um modelo de receptor totalmente flexível (Fully Flexible Receptor - FFR) do complexo InhA-NADH da *Mycobacterium tuberculosis*, encontrando ótimos resultados.

Palavras Chave: Triagem Virtual, Redes Neurais Siameses, Funções de Escore, Docagem Molecular.

RANKING LIGANDS IN STRUCTURE-BASED VIRTUAL SCREENING USING SIAMESE NEURAL NETWORKS

ABSTRACT

Structure-based virtual screening (SBVS) on compounds databases has been widely applied in early stage of the drug discovery on drug target with known 3D structure. In SBVS, computational approaches usually 'dock' small molecules into binding site of drug target and 'score' their binding affinity. However, the costs involved in applying docking algorithms into huge compounds databases are prohibitive, due to the computational resources required by this operation. In this context, different types of machine learning strategies can be applied to rank ligands, based on binding affinity, and to reduce the number of compounds to be tested. In this work, we propose a deep learning energy-based model using siamese neural networks to rank ligands. This model takes as inputs grids of biochemical properties of ligands and receptors and calculates their compatibility. We show that the model can learn to identify important biochemical interactions between ligands and receptors. Besides, we demonstrate that the compatibility score is computed based only on conformation of small molecule, independent of its position and orientation in relation to the receptor. The proposed model was trained using known ligands and decoys in a *Fully Flexible Receptor* model of InhA-NADH complex (PDB ID: 1ENY), having achieved outstanding results.

Keywords: Virtual Screening, Siamese Neural Network, Scoring Function, Molecular Docking.

LIST OF FIGURES

LIST OF ACRONYMS

ANN – Artificial Neural Network

AUC – Area Under the Curve

APBS – Automatic Poisson-Boltzmann Solver

CAD – Computer Aided Drug Design

CNN – Convolutional Neural Network

DCNN – Deep Convolutional Neural Network

DUD-E – Directory of Useful Decoys - Enhanced

FEB – Free Energy of Binding

FFR – Fully Flexible Receptor

HTS – High-Throughput Screening

MD – Molecular Dynamics

MTNN – Multi-Task Neural Network

PDB – Protein Data Bank

PINN – Pairwise Input Neural Network

ROC – Receiver Operating Characteristic

SNN – Siamese Neural Network

VS – Virtual Screening

CONTENTS

1. INTRODUCTION

The identification of compounds that show particular pharmacological activity and the optimization of the pharmacological properties of these compounds are the main focus of early-stage drug discovery. To this end, industry has been adopting high-throughput screening (HTS) approach, which consists of experimental screening of large libraries of compounds against a biological target. Despite the applicability of this method and potential improvements in drug research, its high cost continues to be a huge drawback [25].

This drawback of experimental HTS has led to the increasing employment of Computer Aided Drug Design (CADD) methods. Among many available computational methods, Structure-Based Drug Design (SBDD) is central to the efficient development of drug compounds and to the understanding of processes involved in molecular recognition [25]. SBDD focus on understanding the molecular basis behind the diseases and uses this knowledge in the process of drug identification. By using the 3D structural information about target, it is possible to computationally simulate the target's environment and investigate underlying interactions involved in molecular recognition.

One of most used techniques of SBDD is Structured-Based Virtual Screening (SBVS), which serves as an alternative approach to experimental screening. In SBVS, large libraries of compounds are computationally screened against drug targets. SBVS usually relies on molecular docking to screen chemical compounds [6]. Molecular docking predicts where and how a small molecule binds to target binding site [19]. In order to predict the best pose of molecule, docking techniques focus on generate different poses and evaluate each one to identify the best pose [1]. For each docked ligand, thousands of different poses are generated and evaluated during docking execution, which require too much computational resources. So, the application of this technique in huge compound libraries, like ZINC [16], which, nowadays, contains over 35 million of compounds, are prohibited without any approach to reduce the number of compounds.

In this context, machine learning methods could be used to reduce the total size of these libraries, selecting the most promising compounds based on molecular features. These models could analyze libraries of compounds, identify important molecular features and use this information to calculate a *score* based on binding affinity with a specific drug target. This score could be used to rank molecules in huge libraries, prioritizing compounds with greater chance of high-affinity binding. These scoring functions (SFs) can effectively exploit very large volumes of structural and interaction data and *learn*, based on these experiences, to calculate the score of each compound [1]. Similar to scoring functions used in molecular docking, this machine learning SFs evaluate the match between ligand and receptor. These ML scoring functions could also learn to identify important molecular features from raw-data [1]. The learning of new features from raw-data is the focus of a branch of machine learning algorithms known as *Deep Learning*. Deep Learning models uses a deep graph of multiple processing units to learn low-level features and combine them to create high-level representations based on training data. Indeed, the analysis of learned features could clarify some aspects involved in molecular recognition [1].

1.1 Problem Statement and Objectives

Structure-Based Virtual Screening relies on molecular docking to screen huge libraries of compounds against a molecular target. The significant cost of SBVS is computational time, even with advances in high-performance computing [48]. In order to reduce this cost, a machine learning model could rank the compounds based on their affinity with a drug target, making possible to reduce the size of library. This model should take as input a chemical compound and a biological structure and outputs a score based on the *match of chemical structures with a little or no human intervention*. In order to save computational resources used to predict the pose of ligand, this model should compute the score without the knowledge of position and orientation of molecule in target binding site.

So, to address these requirements, in this work, we propose the use of siamese neural networks in structure-based virtual screening. Siamese neural networks takes pairs of samples as input and outputs the compatibility between them. In addition, the proposed SNN can learn to identify important chemical features without human intervention, as shown in this work.

1.2 Organization

This work is organized into 6 chapters:

Chapter 2 - Chemoinformatics Detailed topics of chemoinformatics required to a better understanding of this work.

Chapter 3 - Machine Learning Presents machine learning models, how they are trained and evaluated. This chapter focuses on describing the architecture and training of artificial neural networks, focusing on siamese neural networks.

Chapter 4 - Virtual Screening using Siamese Neural Networks This chapter describes the proposed approach to rank ligands, the dataset used for training and testing, the results obtained and a comparison with other ML methods.

Chapter 5 - Related Work Enumerate and discuss related work on machine learning models applied in context of virtual screening. This chapter compares the results obtained by deep learning models and as well as the restrictions and applicability of each described approach.

Chapter 6 - Conclusions This chapter concludes the work and discuss some improvements to the proposed approach.

2. CHEMOINFORMATICS

This chapter describes some concepts of bioinformatics and drug development fundamental to a better understanding of this work. The first section describes topics of virtual screening, focused on structure-based virtual screening. The second section addresses about electrostatic potential grids. The third section details some aspects of the enzyme studied in this work, the InhA of *Mycobacterium tuberculosis*. And, the fourth and last section of this chapter deals about Fully Flexible Receptor (FFR) models and their use in molecular docking.

2.1 Virtual screening

Virtual screening (VS) is a computational technique to search for small molecules in huge libraries and select those structures which are most likely to bind to a drug target [37]. There are two main approaches of VS: ligand-based virtual screening and structure-based virtual screening.

In ligand-based VS approaches, candidates molecules are selected based on similarity with known active ligands on target. This approach is most employed when the 3D structure of the receptor is not available [27]. Structure-based VS techniques use information about target to search for candidate molecules that show potentially favorable interactions with the target [26]. In this work, we focus on structure-based virtual screening.

Structure-Based Virtual Screening relies on molecular docking to screen libraries of compounds against a drug target. Molecular docking is a computational approach that predicts the pose of small molecule (ligand) inside a drug target binding site (receptor) and evaluate interactions between them [46]. The docking process involves two building blocks: a search algorithm, which generate conformations of small molecule (also referred as *pose*), and a scoring function, that measures the binding affinity of each generated pose. The search algorithm generate poses by altering torsion angles, orientation and position of molecule in target's binding site. The scoring function estimates the binding affinity by observing the physical interactions between ligand and receptor. For the sake of efficiency, scoring functions do not fully account for certain physical processes involved in molecular recognition, which limit their ability to rank order and select small molecules. Nowadays, despite the existence of robust and accurate search algorithms for pose generation, the performance of scoring function is a major limiting factor for reliability of molecular docking [1]. Usually, docking approaches consider the structure of receptor as a rigid body, while the small molecule shows a partial flexibility [36]. During docking execution, the ligand assumes different conformations in different positions of the receptor binding site. Each one of these poses is evaluated by the scoring function. Due this iterative method of pose generation and pose evaluation, the computational time required by docking approaches is high [36] and its applicability in very large chemical libraries is prohibited.

Nowadays, there are more than 30 molecular docking programs available. In this work, we focus on Autodock 4.2.5 [30], due to its highly acceptance in scientific community and its free avail-

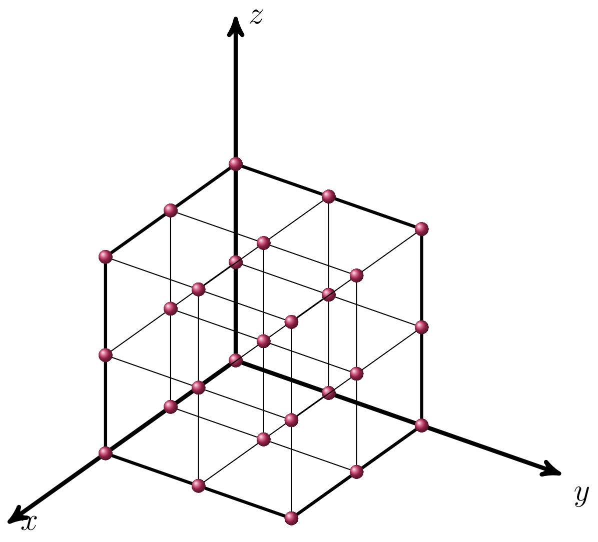


Figure 2.1: The structure of a 3D grid map using cartesian coordinates

ability to use. Autodock's results are sorted based on Estimated Free Energy of Binding (EFEB), calculated by an empirical scoring function [29]. The EFEB is calculated based on interactions between small molecule and receptor and smaller values means high affinity [24]. In case of existence of multiple poses with same energy, the results are sorted based on RMSD (Root Square Mean Deviation). The RMSD is calculated using mean of squared distance between atoms of pose generated by Autodock and a reference structure previously informed.

2.2 Electrostatic Properties Grids

Biological macromolecules, as proteins and enzymes, are composed of multiple atoms carrying a partial charge that interacts with each other. This interaction plays an important role in macromolecular function and molecular recognition. Due to the importance of these interactions in molecular systems, the study of electrostatic properties is well established [2]

In computational methods, electrostatic properties can be represented using grid maps. Using this approach, n points are equally distributed along each 3D coordinates, creating a grid. Figure 2.1 shows a square grid in Cartesian coordinates. The electrostatic potential is calculated on each purple point and is stored in a 3D matrix.

One of the most widespread models for the evaluation of electrostatic properties is the Poisson-Boltzmann equation (PBE) [9] [47], shown in equation 2.1, a second-order nonlinear elliptic partial differential equation that relates the electrostatic potential (ϕ) to dielectric properties of the solute and solvent (ε), the ionic strength of the solution and the accessibility of ions to the solute interior (κ^{-2}), and the distribution of solute atomic partial charges (f). The x refers to the coordinates of the grid point where electrostatic potential is calculated.

$$\nabla \cdot \varepsilon(x)\nabla\phi + \kappa^{-2} sinh\ \phi(x) = f(x) \tag{2.1}$$

2.3 InhA of *Mycobacterium Tuberculosis*

The tuberculosis, also known as TB, is an infectious disease caused by *Mycobacterium tuberculosis* (Mtb). Tuberculosis generally affects the lungs, but also affects other body parts. TB is considered a neglected disease due to the lack of development of new drugs. According to World Health Organization [33], there were 10.4 million new TB cases worldwide in 2015. In this same year, an estimated 1.8 million people died from TB, added to the 0.4 million deaths from TB disease among HIV-positive people. According to same report, 480 000 new cases of *multidrug-resistant* TB and 100 000 new cases of *rifampicin-resistance* TB were observed in 2015. From the combined total of 580 000 cases, India, China and Russia accounted for 45% of these new cases. Although the observed fell of 22% in the number of TB deaths between 2000 and 2015, TB remains as one of top 10 causes of death worldwide in 2015.

The observed new cases of drug-resistance TB evidence the necessity of the development of new drugs. Among Mtb internal structures, the InhA enzyme is a promising drug target, due to its importance in bacterial metabolism. This enzyme is involved in type II fatty acid synthesis (FAS-II) [41]. This fatty acid plays an important role as structural components of cell wall. So, the inhibition of this enzyme affects the cell wall formation, leading to a posterior death of the bacillus.

The InhA enzyme is composed by 268 aminoacids, which, when folded, create a structure with 7 β-sheet and 8 α-helices. The first 3D structure of InhA (PDB ID: 1ENY) was release in PDB site [4] by Dessen [10]. The binding site of InhA enzyme is localized between two helices that are sustained by protein loops. Inside the binding size, there is the co-enzyme NADH [10]. Figure 2.2 shows the 3D structure of InhA, represented as cartoon, and the co-enzyme NADH (PDB ID: 1BVR), highlighted in blue. Inside the binding site, along with NADH, is the ligand THT (PDB ID: 1BVR), which mimics the molecule involved in FAS-II.

2.4 Fully Flexible Receptor Model

During molecular docking, the receptor is treated as a rigid structure, due to the computational resources required to compute the flexibility of receptor using search algorithms. However, proteins and enzymes are inherently flexible in their environment and this flexibility is essential to perform their function [8]. In addition, macromolecules change their shape upon ligand binding, molding themselves to be complementary to ligand. This change in shape minimize the total Free Energy of Binding (FEB) of the complex, due to the increase of favorable contacts and to the reduce the adverse interactions [8].

There are a number of alternative ways to incorporate receptor flexibility in molecular docking experiments [42]. There are approaches that use a set of receptor conformations in docking simulations. These conformations can be determined experimentally either by X-ray diffraction or

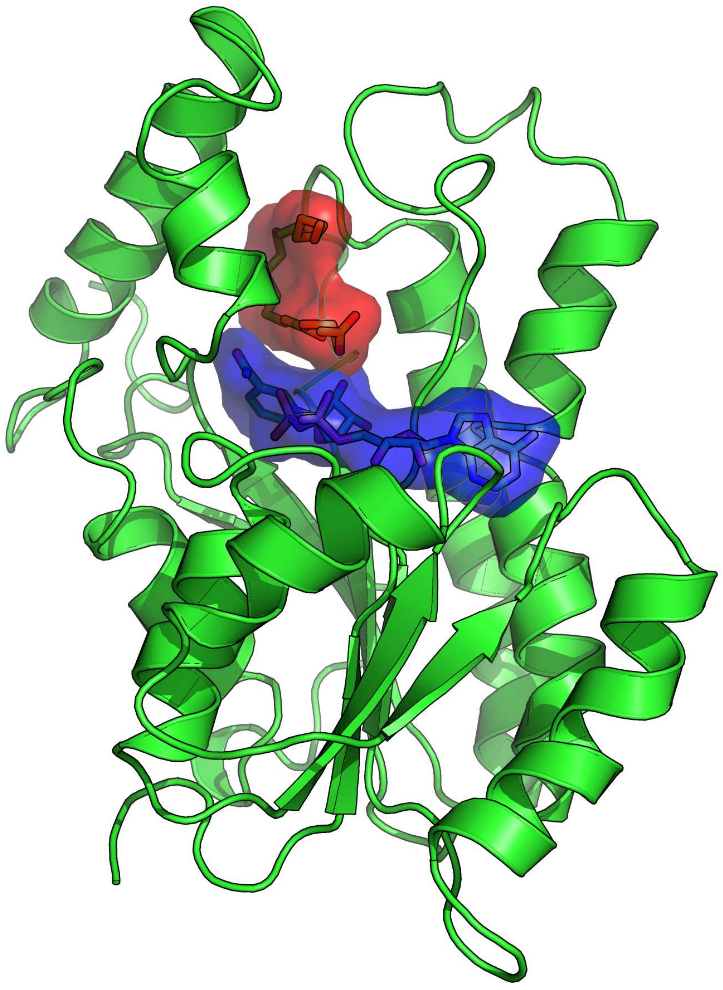


Figure 2.2: The InhA structure, represented as cartoon, with co-enzyme NADH (blue). In red, the ligand THT, which mimics the fatty acid molecule involved in function of InhA.

NMR experiments, or generated by computational methods, like molecular dynamic (MD) simulations [42].

Molecular dynamics (MD) is a computational method for studying the temporal evolution of coordinates and momentum of a macromolecular system [34]. A MD simulation can reproduce the flexibility of a macromolecular system in their environment, if the conditions are known. So, it is possible to obtain a set of observable conformations of the receptor, which shows different conformations of binding site. Each obtained state can be used in molecular docking. This set of states derived from an MD simulation is called Fully Flexible Receptor (FFR) model.

3. MACHINE LEARNING

According to Mitchell [28], "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E". Machine learning is the field of study of algorithms that can *learn* from and make predictions on data based on experience [20], instead of following strictly static program instructions. The main objective of a machine learning model is to generalize from experience, i.e., build a generalization model, using experience learned from training examples, that performs accurately on new and unseen data.

The performance of these models is highly dependent of data representation. Research in this area attempts to create better representations or create models that can learn representations from raw-data based on task. In this context, *deep learning models* can learn to identify low-level features in raw-data and combine these features to create high-level features [3], which are used in decision making.

These models are based on a deep graph of multiple processing units, similar to artificial neural networks (ANN). Due this similarity, algorithms used to train ANNs are used in training of deep learning models. During training, the weights of each processing unit is updated based on backwards propagated error of the model, leading to the reduction of *loss* of the model. This *loss* represents the performance P of the model in task T. In following sections, we detail the organization of ANNs and the algorithms used to train. In addition, we also detail some ANNs architectures, like convolutional neural networks and siamese neural networks.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational approaches of machine learning that are highly based on the human brain architecture. A general definition of artificial neural network is a computational graph whose nodes are processing units and directed edges describe the path of numerical information from node to node.

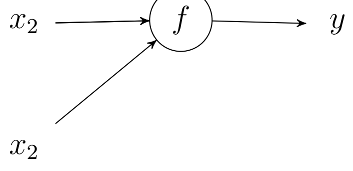


Figure 3.1: The basic architecture of a neuron

Consider the input X as a n-dimensional vector, i.e., $X \in \mathbb{R}^n$ and the output Y as a m-dimensional vector - $Y \in \mathbb{R}^m$. Neural networks consists of a chain of function compositions that transform the input X into Y. Figure 3.1 shows the basic architecture of a single neuron, which consists of input $X_i = [x_1, x_2, x_3]$, the activation function f and the output y. The activation function f is responsible for the non-linearity behaviour of ANNs. The choice of the activation function depends on the connection pattern between neurons and the performance on cross-validation data, between other variables.

A neural network consists of multiple neurons stacked both horizontally (next to each other - Figure 3.2a) and vertically (on top of each other - Figure 3.2b). The numerical path is then followed from input layer to output layer. For multiple neurons, the output activation a_i of neuron i is computed by the inner product between neuron's input X_i and weights W_i followed by an addition with its bias b_i and the application of an activation function f, as demonstrated below:

$$a_i = f(X_i^T \cdot W_i + b_i), \tag{3.1}$$

where $W_i \in \mathbb{R}^n$. Considering m neurons stacked horizontally, the multiplication and the non-linearity can be disentangled and written as a matrix multiplication, as shown below:

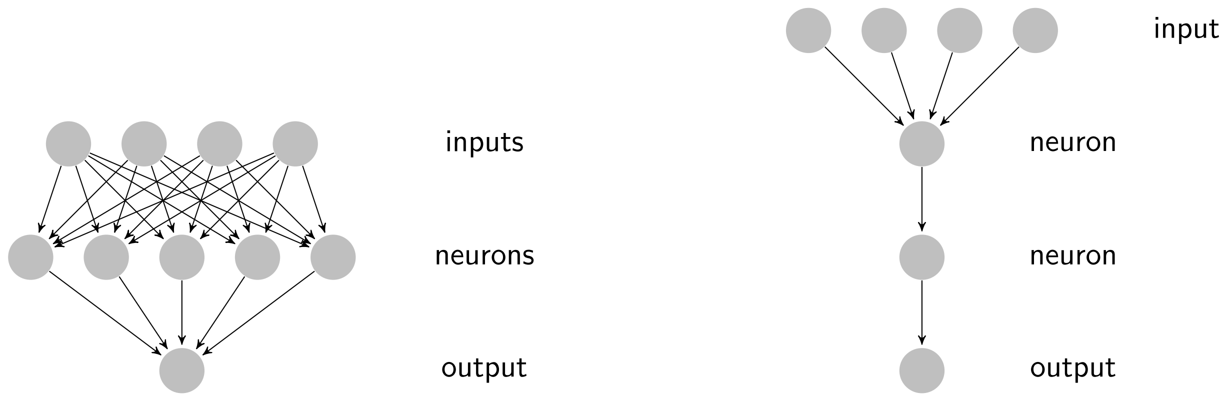
$$\begin{aligned} z &= W \cdot X + b && (3.2) \\ a &= f(z), && (3.3) \end{aligned}$$

where $W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $X \in \mathbb{R}^n$, $a \in \mathbb{R}^m$ and the activation function f is element-wise:

$$f(z) = f([z_1, z_2, ..., z_n]) = [f(z_1), f(z_2), ..., f(z_n)] \tag{3.4}$$

Many of these layers can be stacked on top of each other. In this case, the input of layer l is the output of layer $l-1$. So, we denote the multiple matrix operations to compute the predicted output Y', taking the input X, using n layers, where we denote $W^{(i)}$ and $b^{(i)}$ as the weight matrix and the bias matrix of layer i:

$$\begin{aligned} z^{(1)} &= W^{(1)} \cdot X + b^{(1)} \\ a^{(1)} &= f(z^{(1)}) \\ z^{(2)} &= W^{(2)} \cdot a^{(1)} + b^{(2)} \\ a^{(2)} &= f(z^{(2)}) \\ &\cdots \\ z^{(n)} &= W^{(n)} \cdot a^{(n-1)} + b^{(n)} \\ Y' &= f(z^{(n)}) \end{aligned}$$



(a) Horizontally stacked neurons

(b) Vertically stacked neurons

Figure 3.2: Different organizations of neurons showing the numerical path from the input nodes to the output node.

In this example, activation functions of all layers are the same. However, it is possible to use different activation functions in each layer. In classification tasks, the output layer usually uses the *softmax* function combined with *cross-entropy* loss function. This normalization of output shows good results in these tasks, due to the minimization of cross-entropy between the *true* distribution in data and estimated distribution of the trained model [17]. In following sections, we describe different architectures of ANNs and their applicability. For more information about neural networks, refer to [39].

3.2 Training an ANN

Once an artificial neural network has been structured, the training can be started. In the beginning of the training, all weights are randomly initialized, following different approaches, like [12] or [14]. So, an iterative process is started. At each step, a number of samples, which we called *batch*, is processed and the predicted output is compared against the desired output. The error is propagated backwards and is used to adjust all weights. This step is repeated over and over until a stop condition is reached. During training, the same set of data is processed many times as weights are continuously refined.

To train a multilayer neural network, the backpropagation algorithm [38] is used to compute Δw, that updates each weight w. This algorithm follows the inverse direction of numerical path in an ANN and computes the gradient of all network parameters. It is an efficient way to re-use some parts of gradients computating that are equal to different parameters, as shown later in this work.

To compute these gradients, first we change the target architecture of ANN, adding an extra layer at the end. This final layer is responsible for computing the *loss* of the model. So, instead of returning Y', the altered network outputs its own loss. If we follow backwards the numerical path, starting at this *extra layer* and ending at input, we can compute the partial gradient of loss with respect to any weight in ANN. So, in order to illustrate this process, we assume that each node is

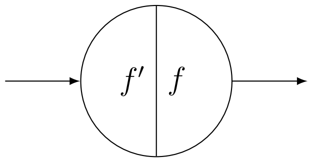


Figure 3.3: The right and left sides of processing unit

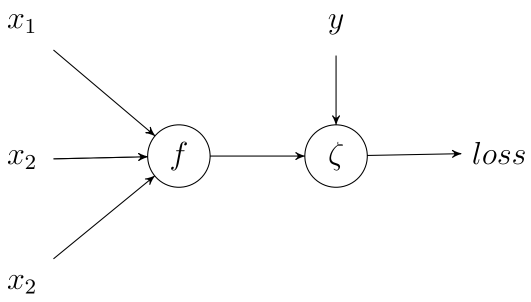


Figure 3.4: A single neuron network that computes its own loss

composed of two parts, as shown in Figure 3.3. The right side of this node computes the function associated with the node while the left side calculates the derivative of this function [38].

Now, considering a network composed of one single neuron, as shown in Figure 3.1, and as first step, the addition of the loss layer ζ, resulting in network shown in Figure 3.4. It is observed that the loss layer takes as input the output of network and desired output y and outputs a real value. The derivative of loss ζ with respect to weight w_i of input x_i is computed in two stages [38]:

Feed-forward the input X is fed into input neurons. The output of each node is evaluated and propagates through network. The derivative is stored in each node.

Backpropagation the constant 1 is fed into loss layer and the network is followed backwards. Incoming information is multiplied, due the chain rule of derivative, by the derivative stored in left part of the unit. The result in each node is the derivative of the loss function with respect to its weights and inputs.

Since we computed the gradient of loss function w.r.t. any parameter in ANN, we can update each parameter. For instance, we focus our attention on a weight w_{ij} that associates the node i with node j. To calculate Δw_{ij}, we can treat node i as an input node of sub-network made of all paths starting at w_{ij} and ending at the output of loss layer. The input of sub-network is o_i, which is the stored output of node i. The input of node j is $o_i w_{ij}$. The backpropagation computes the gradient of ζ with respect to this input, i.e, $\partial\zeta/\partial o_i w_{ij}$. Since, during backpropagation, o_i is treated as a constant, we have:

$$\frac{\partial \zeta}{\partial w_{ij}} = o_i \frac{\partial \zeta}{\partial o_i w_{ij}} \tag{3.5}$$

So, using the calculated gradient, it is possible to update w_{ij} using gradient descent by adding the complement

$$\Delta w_{ij} = -\gamma \frac{\partial \zeta}{\partial w_{ij}}, \tag{3.6}$$

where γ refers to learning rate, i.e, how much of gradient is used to update each weight. This parameter needs to be defined by the specialist. There are approaches to update the learning rate using different heuristics, like adadelta [49] and adam [18]. For more information about backpropagation and gradient descent, please refer to [38].

3.3 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a type of ANN in which the connectivity pattern between neurons favors the identification of local correlations on input [23]. CNNs take advantage on the fact that the input X is a multidimensional matrix, like images (2D matrices) or grid maps (3D matrices). These networks process these raw data matrices and use the extracted features to calculate the output y. A CNN consists of multiple layers of small neurons collections which process portions of input matrix. These processed portions are tiled so that their input regions overlap, in order to obtain better representations of data. Convolutional networks are usually composed of three main types of layers: convolutional layer, pooling layer and fully connected layer.

A convolutional neural network expects a specific organization of these multidimensional matrices in its input layer. To better explain this organization, let us consider a RGB image of width w and height h. This image actually consists of 3 different matrices $w \times h$, which one containing the pixel values for each color. In this case, this RGB image really is $3 \times w \times h$ matrix, once RGB images show 3 independent channels, one for each color. The same idea can be applied in biochemical grid maps. We can define each channel as a specific biochemical property. In this work, we also call this multidimensional matrix as input volume.

Convolutional Layer

The convolutional layer is the core building block of CNNs. Units in this layer are responsible for process the input matrix with a filter, which is spatially small but extends the full depth of channels. During the forward pass, each filter is slid across the spatial dimension of input volume and computes the dot product between filter and input at each position. The application of a filter in an input volume creates a *feature map*. Each convolution layer has an entire set of filter and each of them produces a separate and different feature map. These feature maps form a new volume, where each feature map corresponds to a single channel.

The size of output volume is controlled by three hyperparameters: *channels*, *stride* and *padding*. *Channels* refers to the number of filters in convolutional layers, which defines the number of feature maps. The size of each feature map depends on the *stride* with filter is slid. And sometimes, to enable the application of filter on border of feature map, the input volume can be padded with real values, controlled by hyperparameter *padding*.

These filters can deal with high-dimensional inputs without leading to an impractical numbers of free parameters. Besides, they can exploit local correlation by enforcing a local connectivity pattern between layers. It is important to emphasize that connections between layers are local in space, but always extend along all channels of the input volume. Such architecture ensures that filters produce a strong response to a spatial local pattern.

Pooling Layer

The objective of pooling layers is to progressively reduce the spatial size of input volume. Units in these layers operates independently in each channel performing a down sampling operation along spatial dimensional of input, using max or $average$ functions. This down sampling is done with filters of fixed size, which are applied in portions of input with no overlap.

Fully Connected Layer

The high level reasoning of convolutional networks is done using the fully connected layers, which combines the features of all feature maps to calculate the output y. Unlike convolution units, the input of each neuron in layer m consists of output of all neurons in layer $m-1$, similar to regular neural networks.

3.4 Siamese Neural Network

Traditional approaches to classification using neural networks generally require that all the categories be known in advance and that exist training examples for all classes. These approaches are intrinsically limited to a fairly number of categories (on the order of 1000). Furthermore, as these models learn parameters that best distinguish examples of different classes, they require a fair number of samples per category [13]. Those methods are unsuitable for applications where the number of categories is very large, the number of samples per category is small or only a subset of categories is known *a priori* [22].

Usually, approaches to this kind of problems, with huge number of categories or small samples per categories, are distance-based methods, which consist in creating a similarity metric that can be used to compare elements of data [13]. These methods can also be used to compare samples of *previously unseen-categories*. The idea behind a similarity measure is to map the input patterns into a target space where distance measures (such as Euclidean distance) can be applied.

Similarity metric can be seen as a function $E(X_1, X_2) = ||G(W, X_1) - G(W, X_2)||$ where $G(W, X)$ is the function parametrized by W, that maps the input X into the target space. In this target space, the distance between representations can be seen as similarity. So, the value of $E(X_1, X_2)$ must be small if X_1 and X_2 belong to same category, and large, otherwise.

The architecture of a siamese neural network is shown in Figure 3.5, where X_1 and X_2 is a pair of inputs, $G(W, X)$ is the map function and $E(X_1, X_2)$ is the calculated similarity between X_1 and X_2. In this architecture, the map function $G(W, X)$ is a model that can learn representations of data, like deep learning models. This kind of network was used to compute similarity between human faces [7], using a convolutional neural network as $G(W, X)$.

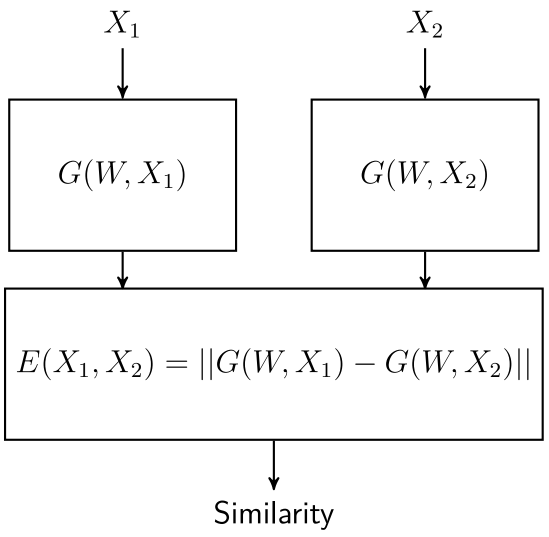


Figure 3.5: Diagram of a Siamese Neural Network to calculate $E(X_1, X_2)$

3.4.1 Training data

A siamese neural network computes the energy between two samples by calculating the distance between low-dimensional representations of these samples. To learn the best parametrization W of map function $G(W, x)$, a label to each pair is required. Due to this requirement, SNN is considered a supervised approach, i.e., approaches of machine learning that require a label or classification of samples in learning phase.

To train a SNN, we are given a set of training samples $S = \{(X_i, X_j), c : i = 1..N, j = 1..N\}$, where X_i and X_j refer to i^{th} and j^{th} samples of training data, respectively, N refers to total size of training data and c is the classification of each pair ($c = 1$ for compatible pair, $c = 0$, for non-compatible pair). The approach used to generate these pairs is highly dependent of nature, origin and organization training data.

3.4.2 Defining a loss function to train a SNN

In order to learn the parametrization W, an objective function must be defined. In this work, we use the contrastive loss function described by Hadsell *et al* in [13], shown in equation 3.7, where c refers to label assignment of pair ($c = 1$ if pair is compatible, $c = 0$, otherwise) and the constant m is the margin, which define an radius around $G(W, x)$ where dissimilar pairs contribute to loss functions. This loss function is composed by a sum of two terms: the first term is the partial loss for compatible pairs and the second, for non-compatible pairs. The minimization of this function decreases the energy of compatible pairs and increases the energy of non-compatible ones.

$$L(W, c, X_i, X_j) = c\frac{1}{2}(E(W, X_i, X_j))^2 + (1 - c)\{max(0, m - E(W, X_i, X_j))\} \tag{3.7}$$

An efficient way of performing the minimization of contractive loss consists in using gradient descent combined with backpropagation [22], that calculates the gradient of $\zeta(W, c, X_i, X_j)$ with respect to an arbitrary weight w_{ij}, corresponding to weight of input i on neuron j. This approach uses the computational graph to compute relations between weights, for the correct calculation of gradient. To evaluate the applicability of contrastive loss and verify that propagated gradient leads to a improvement in SNN performance, we applied the backpropagation algorithm to this function.

To calculate the gradient, let us consider an arbitrary siamese neural network. In this case, the merge layer computes the euclidean distance between low-dimensional vectors. At the end of network, a cost layer, that computes $\zeta(W, c, X_i, X_j)$, is inserted. So, the neural network is now computing the loss of input pair (X_i, X_j) with label c. This architecture is shown in Figure 3.6.

Firstly, the computational graph of contrastive loss function (equation 3.7) is shown in Figure 3.7. The inputs c and $E(X_i, X_j)$, which, for simplicity, we call E, are, respectively, the classification and the output energy of an arbitrary pair of training data. Using the graph, the gradient of ζ with respect to an arbitrary weight $w_{i,j}$ is

$$\frac{\partial \zeta}{\partial w_{ij}} = \frac{1}{2}\frac{\partial cE^2}{\partial w_{ij}} + \frac{\partial (1 - c)max(0, m - E)}{\partial w_{ij}} \tag{3.8}$$

Splitting both terms and applying the multiplication rule, we have

$$\frac{\partial cE^2}{\partial w_{ij}} = 2c\frac{\partial E}{\partial w_{ij}} + E^2\frac{\partial c}{\partial w_{ij}} \tag{3.9}$$

$$\frac{\partial (1 - c)max(0, m - E)}{\partial w_{ij}} = (1 - c)\frac{\partial E^2}{\partial w_{ij}} + max(0, m - E)\frac{\partial (1 - c)}{\partial w_{ij}} \tag{3.10}$$

Since the input c is not dependent of w_{ij}, the gradients $\partial c/\partial w_{ij}$ and $\partial(1 - c)/\partial w_{ij}$ are both equal to 0. The output of max function is defined by the sign of $m - E$: if $m - E > 0$, the gradient $-\partial E/\partial w_{ij}$ is propagated, and, otherwise, the gradient $\partial 0/\partial w_{ij} = 0$ is propagated. In

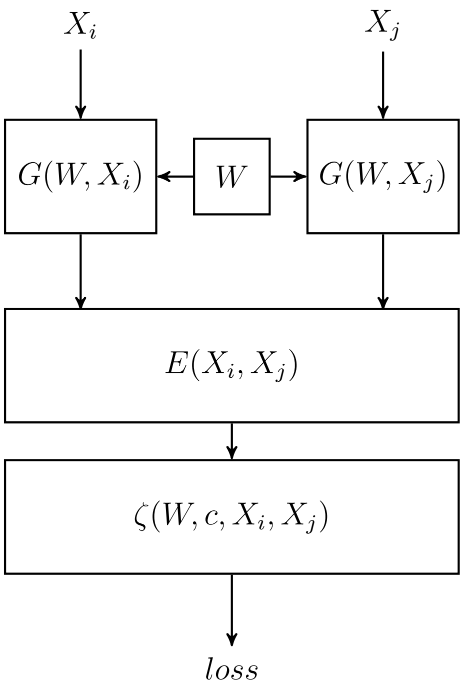


Figure 3.6: Diagram of Siamese Neural Network that computes the loss

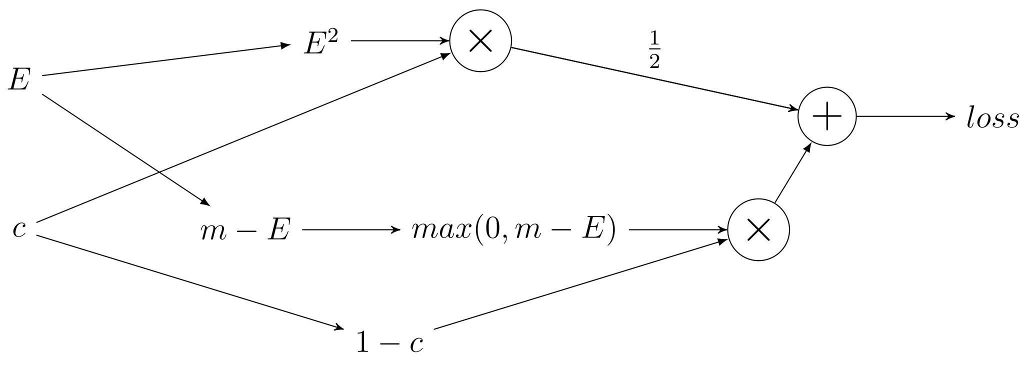


Figure 3.7: Computational graph of loss function ζ

this example, we assume that m is sufficiently large to guarantee that $m - E > 0$. So, we can demonstrate that

$$\frac{\partial \zeta}{\partial w_{ij}} = c\frac{\partial E}{\partial w_{ij}} - \frac{1}{2}(1-c)\frac{\partial E}{\partial w_{ij}} = \left[c - (1-c)\right]\frac{\partial E}{\partial w_{ij}}, \tag{3.11}$$

This derivative shows the desirable behavior of our loss function: if $c = 1$, i.e, a compatible pair, the gradient $\partial\zeta/\partial w_{ij}$ is the same sign of $\partial E/\partial w_{ij}$, so, the minimization of E leads to a minimization of ζ; if $c = 0$, the maximization of E leads to a minimization of ζ.

3.5 Evaluating a Machine Learning Model

One of the most important tasks in building and training a machine learning model is the evaluation of its performance. The performance metrics used in this evaluation are highly dependent of machine learning task. In this work, we focus on classification metrics. First of all, it is important to emphasize that the core objective of the learning is to generalize from the experience, i.e., perform accurately on new and unseen tasks after having experienced a data set. Due to this reason, the available data is divided in training set and testing set. During training, the testing set must remain totally hidden from the model, so, the evaluation metrics, applied in testing set, measure its generalization power.

As described before, the objective function, minimized during training, measures the performance. This function is used to guide the search of the best parametrization, even though, different machine learning models use different objective functions in their training. So, it is not desirable to compare losses calculated by different functions, once their value for two models with similar performance using other metrics can vary in order of magnitude.

Due to these differences in objectives functions, other performance metrics can be used to compare different models. There are specific metrics for tasks of classification, regression, clustering, ranking, among others. In this work, we focus on metrics for classification and ranking. To measure the performance of classification tasks, accuracy and AUC are usually applied. In ranking tasks, precision-recall is widely used.

Accuracy is the measure of how often the classifier correctly labels a sample. It is the ratio between the number of correct predictions and the size of data set. Accuracy can be computed using equation 3.12, where y_i and y'_i refers, respectively, the correct label and the predicted label of sample i, N is the total size of data set and $[y'_i == y_i]$ returns 1, if the logical proposition $y'_i == y_i$ is true, and 0, if otherwise.

$$accuracy = \frac{1}{N}\sum_{i=0}^{N}[y'_i == y_i] \tag{3.12}$$

The AUC metric stands from Area Under Curve, usually calculated based on the Receiver Operating Characteristic (ROC) curve. The ROC curve plots the rate of true positives to the rate of false positives, showing the sensitivity of the classifier. This metric shows how many correct positive labels are returned by the model as more and more false positives are allowed. A perfect classifier should maintain the true positive rate at 100% immediately, without incurring in any false positive. Good classifiers show a bigger AUC, since they show a high true positive rate. The range of calculated AUC varies in $[0, 1]$, although a random binary classifier achieves 0.5, since the rate of false positive and true positive increase equally as decision boundary is altered. The Figure 3.8 shows a ROC curve of an arbitrary classifier C1 and a random binary classifier, identified as C2.

Precision-recall actually refers to two metrics: precision and recall. Precision metric focus on number of true positives among the set of samples predicted as positive. Recall metric describes about the number of true positive samples among all available positive samples. Precision and recall are computed using equations 3.13 and 3.14 respectively, where y_i and y'_i refer the correct label and

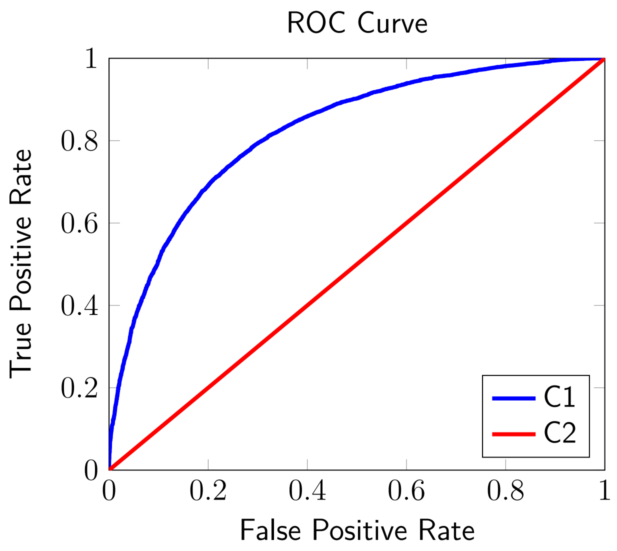


Figure 3.8: Receiver operating characteristic (ROC) curve of an arbitrary classifier C1 and a random binary classifier C2.

the predicted label of sample i, N is the total size of data set and $[(y'_i == 1) \wedge (y_i == 1)]$ returns 1, if the logical proposition $(y'_i == 1)$ and $(y_i == 1)$ are true, and 0, if otherwise.

$$precision = \frac{\sum_{i=0}^{N}[(y'_i == 1) \wedge (y_i == 1)]}{\sum_{i=0}^{N}[y'_i == 1]} \tag{3.13}$$

$$recall = \frac{\sum_{i=0}^{N}[(y'_i == 1) \wedge (y_i == 1)]}{\sum_{i=0}^{N}[y_i == 1]} \tag{3.14}$$

4. VIRTUAL SCREENING USING SIAMESE NEURAL NETWORKS

Structure based virtual screening (SBVS) is a computational approach to identify novel hit compounds when 3D structure of drug target is known. Molecular docking is the most used approach of SBVS applied to filter small molecules in compound libraries [25]. Despite the accuracy of this strategy in drug development, its implementation costs are prohibited. Molecular docking requires a lot of computational resources to dock a huge library of compounds, due to the successive application of the search algorithm and the scoring function.

The main objective of this work is to propose a cheaper alternative method to rank ligands based on affinity using siamese neural networks, reducing the size of compounds to dock. In the following sections, we describe the architecture of proposed approach.

4.1 Proposed architecture

The objective of proposed model is to rank small molecules based on binding affinity with a drug target. This ranking is based on a *distance-based metric* calculated between the target and the candidate molecule. We intend that our approach could be used along molecular docking, increasing the performance and accuracy, and reducing the computational time and human intervention in drug development. It is important to emphasize that the proposed model, unlike other approaches to VS [30] [36] [35] [44], *does not require* the location of each ligand atom in the binding site of the target.

The proposed model is composed of a siamese neural network that computes the *compatibility* between target binding site and a candidate molecule. As described in chapter 3, siamese neural networks compute the similarity between samples using a *distance-based metric*. In that case, both samples can be directly compared, due to the fact that similar samples show similar attributes. However, in case of ranking ligands, high affinity binding requires the match of complementary features between ligand and receptor [32]. So, due to this complementarity between ligand and receptor, the representations generated by map function must be complementary.

4.1.1 Distance Metric

In ligand classification task using siamese neural networks, the distance metric must account the required complementarity between vectors. Besides, as described in chapter 3, siamese neural network outputs lower energy to compatible (similar) pairs. And the minimization of contrastive loss function enforces this comportment. So, we trained different architectures of map function with each distance metric proposed. The training data and validation data of these SNN are both composed by a small portion of training data, described in details in section 4.2.1. Ev-

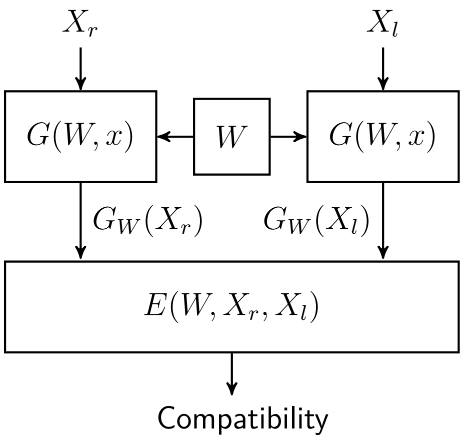


Figure 4.1: Diagram of the SNN to calculate $E(X_r, X_l)$

ery model was training using Stochastic Gradient Descent (SGD) algorithm and the gradients were computed by backpropagation. The learning rates were update using Adam method [18].

We evaluated two different distance metrics: cosine distance (equation 4.1), which is based in cosine similarity, and euclidean distance (equation 4.2). These metrics were altered to output a small value distances to complementary vectors and large value distances, otherwise. The cosine distance varies between $[0, 2]$. In training, we multiply this distance by a factor m and define the threshold of classification at $m/2$. The threshold of the model using euclidean distance, which varies in range $[0, +\infty)$, was also set to $m/2$.

We trained two architectures of SNN, each one with a different distance metric. The map function of both architectures were composed 4 fully connected layers with 512, 256, 64, 8 neurons, respectively. We trained both architectures during 2000 optimization steps. Each optimization step is composed of a feed-forward step and a backpropagation step. The loss reduces quickly using the distance metric shown in Equation 4.2. So, due to this behaviour, we chose this metric to compute the similarity calculated by SNN.

$$E(X_l, X_r) = 1 + \frac{V_r \cdot V_l}{||V_r||\,||V_l||} \tag{4.1}$$

$$E(X_l, X_r) = ||V_r + V_l|| \tag{4.2}$$

4.1.2 Map function

The map function $G(W, x)$ of SNN consists of a convolutional neural network. CNNs expect, as input, a matrix of real valued features. Due this requirement, we use grids of biochemical properties as input. In addition, once chemical interactions are predominantly local [5], convolutional layers of CNNs could capture spatially local correlations in these grids and use them to generate

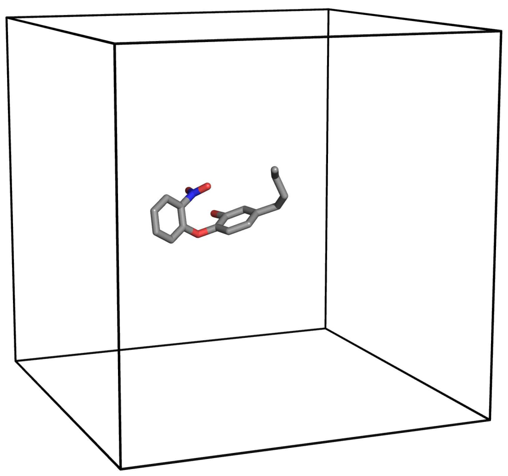


Figure 4.2: Bounding box of 3D grid is centered at center of mass of small molecule 1TN (PDB ID: 4OXY)

more improved representations. So, the use of CNN as $G(W, x)$ is appropriate [44]. In the following sections, we describe the experiments executed with this architecture.

4.1.3 Data encoding

The detection of important chemical interactions is a central key in development of an accurate drug identification model. These chemical interactions are predominantly local [5]. So, the convolutional layers of an SNN can be used to detect these features and use them in decision making. However, convolutional layer takes as input a matrix of number. So, due to this behavior, the involved molecular structures must be encoded as n-dimensional matrices in order to feed a neural network.

In this work, we chose to encode these structures as a 3D grid of electrostatic potential. Electrostatic interactions play an important role in molecular recognition and constitute a driving force underlying protein-ligand and protein-protein interactions [15]. For each point in the grid, the electrostatic potential is calculated and a real value is returned, creating a 3D matrix. These grids were generated by APBS (Automatic Poisson Boltzmann Solver) [2], using Poisson-Boltzmann Equation, described in more details in section 2.2. The generated grids were adjusted to fit a $25\ \text{Å} \times 25\ \text{Å} \times 25\ \text{Å}$ cube, using grid spacing of $1\ \text{Å}$, centered at the center of mass, in case of ligand (Figure 4.2), and at binding site, in case of protein grids (see Figure 4.3).

4.2 Experiments

This section describes the experiments performed using the proposed model. We compare the results obtained with our model to those obtained with a DCNN, similar to the approach described in [44].

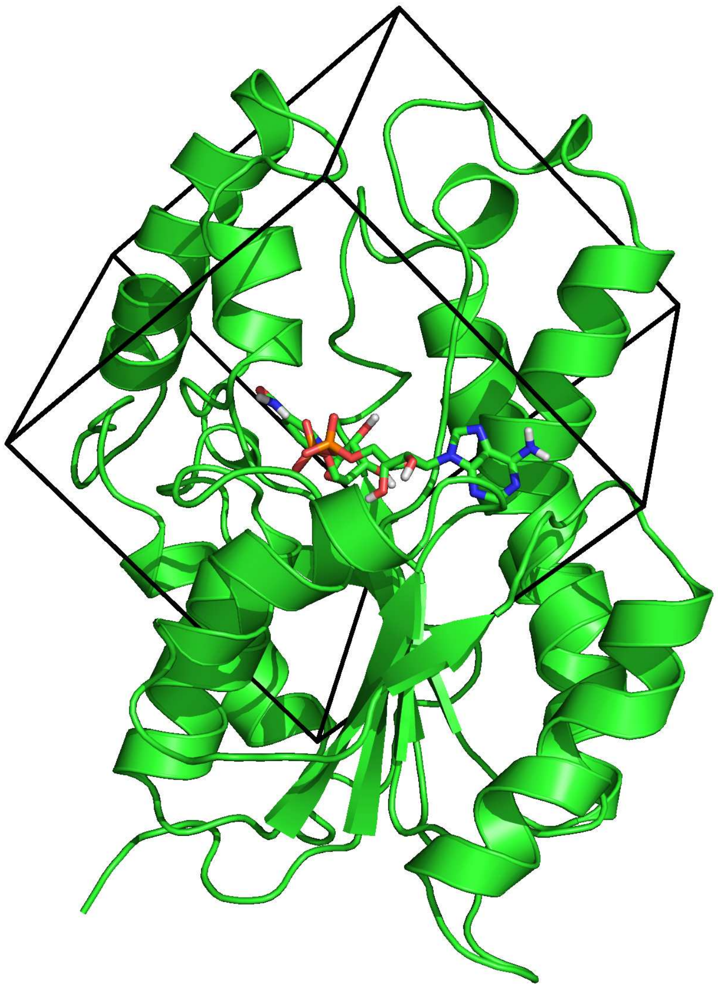


Figure 4.3: Bounding box of 3D grid is centered at binding site of InhA-NADH complex (PDB ID: 1ENY). InhA is represented as cartoon and NADH, as sticks.

4.2.1 Dataset

We demonstrate the application of the proposed model on a Fully Flexible Receptor (FFR) Model containing 19.5 nanoseconds molecular dynamics trajectories simulation of InhA-NADH enzyme complex of Mycobacterium Tuberculosis (PDB ID:1ENY) [11]. In this FFR model, there are 19 500 snapshots, one for each 1 picoseconds.

Once CNNs are used as map function, the model can capture local correlations and use it in decision making. Besides, differences in torsion angles of small molecules influence the position of chemical features and this change can be the difference between high-affinity binding and low-affinity binding. So, to train our model, we used conformations of ligands that could match with the target binding site.

To create the dataset used in the experiments, we selected 38 small molecules, equally distributed between actives and decoys. The 19 actives ligands were selected from structures of InhA available in RCSB PDB [4] and were studied in [36]. The 19 decoys were selected from decoy subset of InhA from DUD-E [31]. In DUD-E, decoys were selected based on similar physical properties but

Table 4.1: Small molecules in training set

ZINC ID	Classification	Source
ZINC21289745	active	PDB ID: 1P44
ZINC14961108	active	PDB ID: 2B36
ZINC02943677	active	PDB ID: 4U0J
ZINC00851723	active	PDB ID: 4TRJ
ZINC01103943	active	PDB ID: 4TZK
ZINC51492080	active	PDB ID: 4U0K
ZINC06700734	active	PDB ID: 4TZT
ZINC01295794	active	PDB ID: 2NSD
ZINC58632782	active	PDB ID: 2X22
ZINC29061009	active	PDB ID: 3FNF
ZINC39232487	active	PDB ID: 3FNG
ZINC16052312	active	PDB ID: 3FNH
ZINC95921165	active	PDB ID: 4BQP
ZINC95921221	active	PDB ID: 4OIM
ZINC98208018	active	PDB ID: 4OXN
ZINC04008680	decoy	DUD-E
ZINC08861490	decoy	DUD-E
ZINC12599848	decoy	DUD-E
ZINC15232810	decoy	DUD-E
ZINC19902859	decoy	DUD-E
ZINC20614024	decoy	DUD-E
ZINC32129225	decoy	DUD-E
ZINC37151917	decoy	DUD-E
ZINC37503812	decoy	DUD-E
ZINC43123516	decoy	DUD-E
ZINC50005899	decoy	DUD-E
ZINC53766965	decoy	DUD-E
ZINC59011110	decoy	DUD-E
ZINC59250224	decoy	DUD-E
ZINC62766324	decoy	DUD-E

different chemical structures from known active ligands. For our dataset, we selected 19 decoys based on pharmacophore fingerprints similarity of each selected active ligand. The training and testing set were split based on molecule, once, in virtual screening, new and unseen molecules must be compared to the target binding site. The molecules in training set and testing set are shown in table 4.1 and 4.2 respectively. These tables show the selected molecules identified by its ZINC ID, along with classification and source of each one.

The docking of 38 molecules were done using Autodock 4.2.5 [30], using 3 000 000 energy evaluations or 27 000 generations as stop criteria for genetic algorithm with population size of 150 individuals. The full list of parameters used is shown in appendix APPENDIX A. As receptors, we selected 976 snapshots of InhA-NADH complex from $19.5ns$ MD FFR model, one conformation for each $20ps$ of simulation.

Table 4.2: Small molecules in testing set

ZINC ID	Classification	source
ZINC14961116	active	PDB ID: 2B37
ZINC29060814	active	PDB ID: 3FNE
ZINC98209089	active	PDB ID: 4COD
ZINC29050100	active	PDB ID: 4OXY
ZINC03314334	decoy	DUD-E
ZINC21950587	decoy	DUD-E
ZINC21987975	decoy	DUD-E
ZINC64521482	decoy	DUD-E

For each molecular docking execution, we selected the pose with *best FEB*. In order to enable the model to tolerate small changes in conformation of ligands, we aligned and clustered all selected poses with average linkage algorithm using the RMSD as distance metric. We configure the average linkage algorithm to stop joining clusters if the distance between clusters was greater than $1\ \text{Å}$. The most representative conformation of each cluster, i.e, the most similar conformation with all other in cluster, was selected and randomly rotated around the center of mass. This rotation ensures that SNN can identify chemical features despite ligand position and orientation. Each rotated pose was paired with the FFR snapshot used in docking.

4.2.2 Training and Results

In this section, we discuss the results obtained using DCNN, considering just the grid of ligand as input, and the proposed SNN approach to classify and rank ligands. In case of probabilistic models, DCNN, the rank are based on calculated probability, i.e, the greater the probability of high-affinity binding, better the position achieved by a compound.

In this work, the accuracy and the AUC (Area Under Curve) are used to compare the models. In addition, we employed precision-recall metrics to evaluate the bias of each tested model.

Training Method

In order to define the best architecture and hyperparameters of optimization process of both tested networks, we tested different architectures and evaluate the performance of each model against a small subset of training data. Each ANN architecture was trained during 5 epochs with different hyperparameters. The best performing architecture was selected to another optimization process.

To define the best approach to update learning rate of gradient descent algorithm, we tested two different approaches, Adam [18] and Adadelta [49]. We use the grid search to define the best parameters of each approach and compared the results obtained after 2 epochs for each combination of parameters. The configuration with best performance was used in the training of network.

Siamese Neural Network

Using the dataset described above, we trained a siamese neural network as described above. In order to train this model, we used the contrastive loss, shown in equation 3.7, minimized using stochastic gradient descent (SGD) and backpropagation. The learning rate of SGD was updated using Adam method [18]. The parameters of Adam were optimized using the grid search. In this search, random values were generated for each parameter (initial lr α, β_1,β_2,ϵ) and all combinations of these values were tested using a small dataset. To achieve the best architecture of SNN, different architectures were trained using a subset of training data. The performance of each architecture in another subset of training data was evaluated and compared. In this work, we only describe the best performing architecture.

In our experiments, the map function that shows best performance consists of n convolutional layers followed by 3 fully connected layers. ReLU was the activation function used in all neurons, except the outputs, which uses the linear function. C_x denotes convolutional layers and F_x refers to fully connected layer, where x is the layer index.

C_1 Filters: 4; Kernel Size: $5 \times 5 \times 5$; Stride: $2 \times 2 \times 2$; Parameters: 504;

C_2 Filters: 8; Kernel Size: $5 \times 5 \times 5$; Stride: $2 \times 2 \times 2$; Parameters: 4008;

C_3 Filters: 8; Kernel Size: $5 \times 5 \times 5$; Stride: $1 \times 1 \times 1$; Parameters: 8008;

C_4 Filters: 16; Kernel Size: $3 \times 3 \times 3$; Stride: $1 \times 1 \times 1$; Parameters: 3472;

F_5 Number of units: 32; Parameters: 64032;

F_6 Number of units: 16; Parameters: 528;

F_7 Number of units: 8; Parameters: 136;

Siamese neural networks are considered an energy-based model, which lacks normalization in output. To enable direct comparison with other machine learning approaches, we normalized the output of SNN using the logistic function. The default decision boundary of logistic function is at 0 and the not normalized output of SNN varies in range $[0, +\infty)$. In this case, we set a threshold of energy, which defines the maximum energy of a pair (X_l, X_r) to be labelled as compatible or X_l be classified as active. During our tests, we set the threshold at $m/2$, where m is the margin in contrastive loss function. The normalization function used is shown in equation 4.3, where $E(X_l, X_r)$ refers to output of model feed with pair (X_l, X_r).

$$p_{active} = \frac{1}{1 + e^{(E(X_l,X_r) - \frac{m}{2})}} \tag{4.3}$$

Using the normalized output, it is possible to compute the AUC and accuracy of this model, which achieved 0.92 AUC and 85.81% of accuracy. The precision and recall are 0.85 and 0.81, respectively, which suggests that this trained model is more restrictive in classifying molecules as active, even if some of discarded molecules are, in fact, active.

DCNN

As described in chapter 3, deep convolutional neural networks are subclass of neural networks that can correlate spatial information on multi-dimensional matrix using convolutional layers. DCNN takes as input a multi-dimensional matrix and outputs a normalized probability, due to the use of softmax function in output neurons.

In our experiments, to avoid the requirement of knowing the location of each atom inside target binding site, the input of trained DCNN is only the 3D grid of ligand. Using this architecture, the trained DCNN learns to identify candidate molecules for a single target. However, this limitation do not affects our results, once dataset used in training and tests contains one single FFR model of a single receptor. As the used dataset contains a pair of grids (X_l, X_r) and a label c, we discard the grid X_r and train the model using just X_l and c.

Different network architecture was trained using the dataset. In this work, we describe the architecture with best performance. C_x denotes convolutional layers with ReLU activation, P_x denotes a max pooling layer where x is the layer index, O refers to 2-way softmax layer.

C_1 Filters: 4; Kernel Size: $3 \times 3 \times 3$; Padding: $1 \times 1 \times 1$; Parameters: 112;

P_2 Pool Filter: $2 \times 2 \times 2$

C_3 Filters: 8; Kernel Size: $3 \times 3 \times 3$; Padding: $1 \times 1 \times 1$; Parameters: 876;

P_4 Pool Filter: $2 \times 2 \times 2$

C_5 Filters: 16; Kernel Size: $1 \times 1 \times 1$; No Padding ; Parameters: 1142;

C_6 Filters: 16; Kernel Size: $3 \times 3 \times 3$; Padding: $1 \times 1 \times 1$; Parameters: 69282;

P_7 Pool Filter: $2 \times 2 \times 2$

C_8 Filters: 32; Kernel Size: $1 \times 1 \times 1$; No Padding ; Parameters: 554;

C_6 Filters: 16; Kernel Size: $3 \times 3 \times 3$; No Padding; Parameters: 55360;

O Number of units: 2; Parameters: 130;

The loss function used to train this model was *cross-entropy*, shown in equation 4.4, where y' is the output of neural network feed by X. The stochastic gradient-descent algorithm

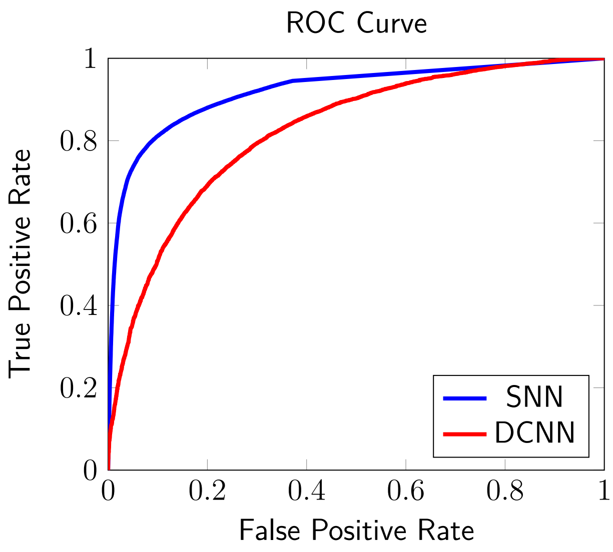


Figure 4.4: Receiver operating characteristic curve of both tested models.

and backpropagation were applied to minimize the loss. The learning rate in gradient-descent was updated using Adadelta method.

$$\zeta(W, X, c) = -c\log(y') - (1-c)\log(1-y') \tag{4.4}$$

This architecture of DCNN achieved 74.03% of accuracy and 0.82 of AUC. The precision and recall are both 0.74, what implies that this model mislabel the same number of actives and decoys molecules, showing no bias in classification.

4.2.3 Comparing the results

After training both models, we can measure and compare their performance. We summarize the obtained results in table 4.3. The proposed model achieved 0.92 AUC, outstanding the DCNN approach. Figure 4.4 shows the ROC curve of both approaches. Besides, the accuracy of the proposed model was greater too. The Precision-recall shows that the proposed model is more restrictive in considering a molecule as active, even if this restriction force the misclassification of true active molecules.

Table 4.3: Statistics of performance

	Accuracy	Precision	Recall	AUC
SNN	85.81%	0.85	0.81	0.92
DCNN	74.03%	0.74	0.74	0.82

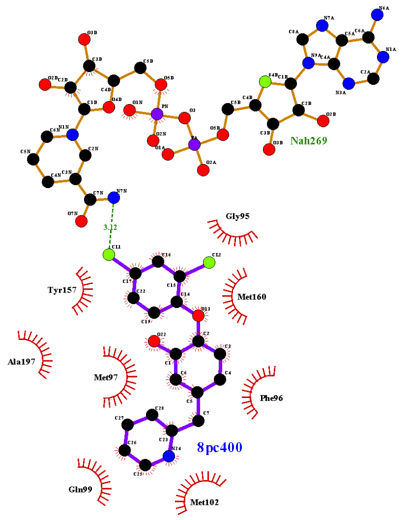


Figure 4.5: Representation of interactions between small molecule 8PC (PDB ID: 3FNE) and binding site of InhA-NADH complex (PDB ID: 1ENY).

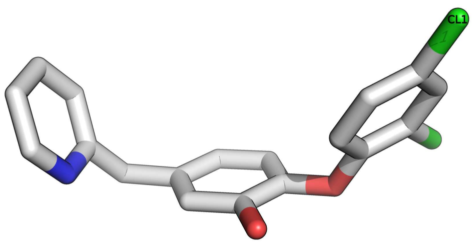


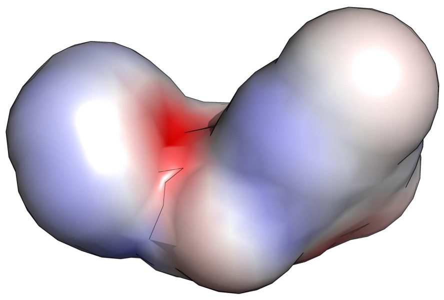
Figure 4.6: Conformation of small molecule 8PC (PDB ID: 3FNE). The atom CL1 is labelled

4.3 Understanding the model

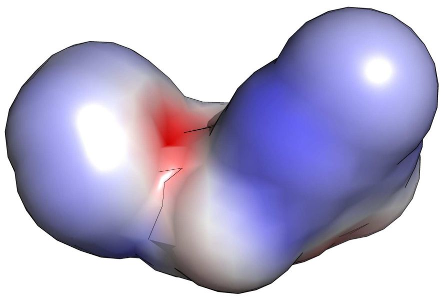
To achieve better understanding about map function $G(W, X)$ and how chemical features are combined to calculate compatibility, we artificially altered changes of a docked ligand and compared the calculated compatibility between an original grid and a grid with altered charges as input.

Firstly, we identified an important interaction involved in molecular recognition between a docked conformation of inhibitor 8PC (8PC400 from PDB ID: 3FNE) and a snapshot of FFR model. Using Ligplot+ [21], we observed a hydrogen bond between chlorine atom $CL1$ of 8PC, shown in Figure 4.6, and nitrogen atom $N7N$ of NADH complex, as shown in Figure 4.5.

In order to measure the influence of this interaction in compatibility calculated by SNN, we artificially altered the calculated charge of atom $CL1$ to avoid the existence of the hydrogen bond. To better visualize the difference between original and altered grid, we plotted the electrostatic surface in Figure 4.7 , visualized in PyMOL [40]. Both surfaces are aligned with Figure 4.6, so, it is possible to identify the region of atom $CL1$ on electrostatic surface. In this Figure, blue color represents positive electrostatic potential, and red color, negative electrostatic potential. It is possible to observe a slight difference between the original surface (Figure 4.7a) and the surface



(a) Original electrostatic surface of molecule 8PC



(b) Altered electrostatic surface of molecule 8PC

Figure 4.7: Electrostatic surfaces used to verify the influence of a hydrogen bond in compatibility calculated by the proposed model. Blue areas represent positive electrostatic potential and red areas, negative electrostatic potential. White areas show potential close to $0\ volts$.

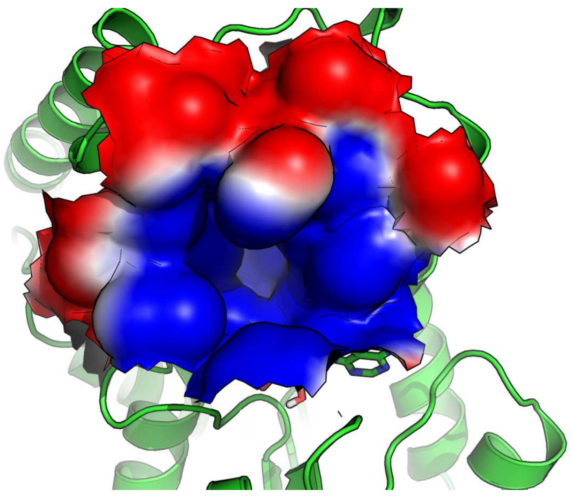


Figure 4.8: Electrostatic surface of binding site of InhA-NADH complex. InhA is represented as cartoon and NADH is represented as stick, using CPK color scheme. Blue areas represent positive electrostatic potential and red areas, negative electrostatic potential. White areas show potential close to $0\ volts$.

with altered charges (Figure 4.7b); the potential at region of atom $CL1$ is slightly higher. Besides, as shown in Figure 4.8, most of binding site of InhA shows positive electrostatic potential. So, this increase in potential of grid shown in Figure 4.7b should reduce the calculated compatibility (increase the output of SNN). As expected, the compatibility between the original pair is 0.71, while the compatibility of altered grid is 32.32. This difference shows that the model is identifying important features and matching complementary features to calculate the compatibility.

5. RELATED WORK

Virtual screening techniques focus on ranking a set of previously untested molecules according to the probability of successfully binding to a drug target [43]. In this context, there are two main approaches: *structure-based virtual screening*, which requires the knowledge of 3D structure of target; and *ligand-based virtual screening*, which is based on search of compounds that are similar to known ligands that exhibit the desired biological activity.

Nowadays, the most used computational method in SBVS is molecular docking, which uses scoring functions to evaluate each generated pose. For the sake of efficiency, these functions are based on theory-inspired mathematical functions that approximate physical forces underlying molecular interactions. To overcome these limitations, machine learning approaches to scoring functions were proposed. Different ML approaches were used to mimic scoring functions, including random forest, SVM and neural networks [1]. Nowadays, deep learning approaches are being used in this context. Wallach *et al* [44] and Pereira *et al* [35] proposed the use of deep convolutional neural networks to distinguish between active and decoys docked inside target binding site. These DCNNs were feed with 3D grids of biochemical descriptors. Both works used DUD-E [31] as training and testing data. The main difference between these works is the data encoding. AtomNet, proposed by Wallach, uses as input 3D grids protein-ligands descriptors, while DeepVS, proposed by Pereira, uses atom context representations, which includes features like atom types, atomic partial charges, amino acid types and distances from neighbors to the reference atom. Both works achieved more than 0.9 AUC on specific targets of DUD-E.

Despite the good results obtained using deep learning models to mimic scoring functions, these approaches require the knowledge of the positions of all atoms of ligands inside target binding site. To overcome this limitation, Wang *et al* [45] proposed an architecture of neural network that combine information about ligand and receptor. Wang's approach creates binary representations of receptor and ligand based on chemical fingerprints and combines these representations in a Pairwise Input Neural Network (PINN). Using a different approach, Untherthiner and Mayr *et al* [43] proposed a multi-task neural network for target prediction. This MTNN comprises one or multiple layers of ReLU hidden layers followed by one layer of 1 230 sigmoid output units, where each output unit refers to a specific drug target. The input of this network is a binary sparse vector of ligand fingerprints. According to this work, this approach achieves a mean AUC of 0.83 across all studied targets.

The approach proposed in this work is different from the described methods. The proposed model learns a function that maps 3D grids of biochemical properties into low-dimensional representations and computes the *compatibility* between these grids using these vectors. Due to the use of a convolutional neural network in map function, our approach can use spatial information available in these grids in decision making. Unlike other ML approaches to SBVS that use CNNs, our method do not require the knowledge of the location of every small molecule atom inside binding site. In addition, this approach can learn to identify implicitly molecular interactions involved in molecular recognition that are hard to model explicitly.

6. CONCLUSION

Structure-based virtual screening usually relies on molecular docking to screen libraries of compounds against a drug target. However, the computational docking requires a lot of computational resources [48], due to generation and evaluation of many poses of ligand. In this context, machine learning models could exploit the available biological data and learns, through experience, to identify implicit molecular features and use them to rank small molecules based on matching important chemical properties with drug target.

In this work, we propose a distance-metric used in siamese neural networks that can compute compatibility of complementary structures. This altered SNN can be used in virtual screening to rank ligands based on matches of complementary chemical features. The proposed model takes as input two 3D grids of biochemical properties and computes the compatibility between these grids. This compatibility is calculated using the euclidean distance between representations generated by a convolutional neural network. The convolutional layers of CNN capture local correlation in these grids and use this information to calculate the compatibility.

We also show that the proposed model can be used to rank and classify molecules based on information of target binding site. The training data consisted of a 30 small molecules docked on a FFR model of InhA-NADH complex of *Mycobacterium tuberculosis*. This model point out another deep learning approach based on convolutional neural network [44] [35], achieving 0.92 AUC. Besides, we demonstrate that the trained model identifies important chemical features, like hydrogen bonds.

This model showed promising results, although, further tests are required to identify its potential. These further tests may include the use of multiple drug targets and changes of biochemical properties used to generate the grids, however, the computational resources to generate new biological datasets that include docked conformations of active ligands and decoys are prohibited, due to the time spent in molecular docking process. In addition, changes in loss function to consider the FEB or other characteristics of complexes ligand-receptor should improve the performance and prioritize compounds that experimentally showed better results.

BIBLIOGRAPHY

[1] Ain, Q. U.; Aleksandrova, A.; Roessler, F. D.; Ballester, P. J. "Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening", *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 5–6, 2015, pp. 405–424.

[2] Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. "Electrostatics of nanosystems: application to microtubules and the ribosome", *Proceedings of the National Academy of Sciences*, vol. 98–18, 2001, pp. 10037–10041.

[3] Bengio, Y.; Courville, A.; Vincent, P. "Representation learning: A review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35–8, 2013, pp. 1798–1828.

[4] Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. "The protein data bank", *Nucleic acids research*, vol. 28–1, 2000, pp. 235–242.

[5] Bissantz, C.; Kuhn, B.; Stahl, M. "A medicinal chemist's guide to molecular interactions", *Journal of medicinal chemistry*, vol. 53–14, 2010, pp. 5061–5084.

[6] Cheng, T.; Li, Q.; Zhou, Z.; Wang, Y.; Bryant, S. H. "Structure-based virtual screening for drug discovery: a problem-centric review", *The AAPS journal*, vol. 14–1, 2012, pp. 133–141.

[7] Chopra, S.; Hadsell, R.; LeCun, Y. "Learning a similarity metric discriminatively, with application to face verification". In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 539–546.

[8] Cozzini, P.; Kellogg, G. E.; Spyrakis, F.; Abraham, D. J.; Costantino, G.; Emerson, A.; Fanelli, F.; Gohlke, H.; Kuhn, L. A.; Morris, G. M.; et al.. "Target flexibility: an emerging consideration in drug discovery and design", *Journal of medicinal chemistry*, vol. 51–20, 2008, pp. 6237–6255.

[9] Davis, M. E.; McCammon, J. A. "Electrostatics in biomolecular structure and dynamics", *Chemical Reviews*, vol. 90–3, 1990, pp. 509–521.

[10] Dessen, A.; Quemard, A.; Blanchard, J. S.; Jacobs Jr, W. R.; Sacchettin, J. C. "Crystal structure and function of the isoniazid target of mycobacterium tuberculosis", *Science*, vol. 267–5204, 1995, pp. 1638.

[11] Gargano, F.; Costa, A.; De Souza, O. N. "Effect of temperature on enzyme structure and function: a molecular dynamics simulation study". In: Annals of the 3rd International Conference of the Brazilian Association for Bioinformatics and Computational Biology, 2007.

[12] Glorot, X.; Bengio, Y. "Understanding the difficulty of training deep feedforward neural networks." In: Aistats, 2010, pp. 249–256.

[13] Hadsell, R.; Chopra, S.; LeCun, Y. “Dimensionality reduction by learning an invariant mapping”. In: IEEE Computer Society Conference on Computer vision and pattern recognition, 2006, pp. 1735–1742.

[14] He, K.; Zhang, X.; Ren, S.; Sun, J. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[15] Hildebrandt, A.; Blossey, R.; Rjasanow, S.; Kohlbacher, O.; Lenhof, H.-P. “Electrostatic potentials of proteins in water: a structured continuum approach”, *Bioinformatics*, vol. 23–2, 2007, pp. e99–e103.

[16] Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. “Zinc: A free tool to discover chemistry for biology”, *Journal of Chemical Information and Modeling*, vol. 52–7, 2012, pp. 1757–1768, pMID: 22587354, http://dx.doi.org/10.1021/ci3001277.

[17] Karpathy, A. “Stanford University CS231n: Convolutional Neural Networks for Visual Recognition”. Available in: http://cs231n.stanford.edu/syllabus.html, Sep 2015.

[18] Kingma, D. P.; Ba, J. “Adam: A method for stochastic optimization”. Available in: https://arxiv.org/abs/1412.6980, Feb 2017.

[19] Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. “Docking and scoring in virtual screening for drug discovery: methods and applications.”, *Nature reviews. Drug discovery*, vol. 3–11, 2004, pp. 935–949.

[20] Kohavi, R.; Provost, F. “Glossary of terms”, *Machine Learning*, vol. 30–2-3, 1998, pp. 271–274.

[21] Laskowski, R. A.; Swindells, M. B. “Ligplot+: multiple ligand–protein interaction diagrams for drug discovery”, *Journal of chemical information and modeling*, vol. 51–10, 2011, pp. 2778–2786.

[22] LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; Huang, F. “A tutorial on energy-based learning”, *Predicting structured data*, vol. 1, 2006, pp. 0.

[23] LeCun, Y.; et al.. “LeNet-5, convolutional neural networks”. Available in: http://yann.lecun.com/exdb/lenet, Feb 2017.

[24] Lengauer, T.; Rarey, M. “Computational methods for biomolecular docking”, *Current opinion in structural biology*, vol. 6–3, 1996, pp. 402–406.

[25] Lionta, E.; Spyrou, G.; K Vassilatis, D.; Cournia, Z. “Structure-based virtual screening for drug discovery: principles, applications and recent advances”, *Current topics in medicinal chemistry*, vol. 14–16, 2014, pp. 1923–1938.

[26] Lyne, P. D. “Structure-based virtual screening: an overview”, *Drug discovery today*, vol. 7–20, 2002, pp. 1047–1055.

[27] Martin, Y. C.; Kofron, J. L.; Traphagen, L. M. “Do structurally similar molecules have similar biological activity?”, *Journal of medicinal chemistry*, vol. 45–19, 2002, pp. 4350–4358.

[28] Mitchell, T. M. “Machine learning”. McGraw-Hill Boston, MA:, 1997.

[29] Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J.; et al.. “Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function”, *Journal of computational chemistry*, vol. 19–14, 1998, pp. 1639–1662.

[30] 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”, *Journal of computational chemistry*, vol. 30–16, 2009, pp. 2785–2791.

[31] 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”, *Journal of medicinal chemistry*, vol. 55–14, 2012, pp. 6582–6594.

[32] Oloff, S.; Zhang, S.; Sukumar, N.; Breneman, C.; Tropsha, A. “Chemometric analysis of ligand receptor complementarity: identifying complementary ligands based on receptor information (colibri)”, *Journal of chemical information and modeling*, vol. 46–2, 2006, pp. 844–851.

[33] Organization, W. H.; et al.. “Global tuberculosis report 2016”. Available in: http://www.who.int/tb/publications/global_report/en/, Feb 2016.

[34] Paquet, E.; Viktor, H. L. “Molecular dynamics, monte carlo simulations, and langevin dynamics: a computational review”, *BioMed research international*, vol. 2015, 2015.

[35] Pereira, J. C.; Caffarena, E. R.; dos Santos, C. N. “Boosting docking-based virtual screening with deep learning”, *Journal of chemical information and modeling*, vol. 56–12, 2016, pp. 2495–2506.

[36] Quevedo, C. V. “Triagem virtual em banco de dados de ligantes considerando propriedades físico-químicas de um modelo de receptor totalmente flexível (Virtual screening in ligand databases considering phisical-chemical properties of a fully-flexible receptor model)”, Ph.D. Thesis, Programa de Pós-Graduação em Ciências da Computação, PUCRS, 2016, 157p.

[37] Rester, U. “From virtuality to reality-virtual screening in lead discovery and lead optimization: a medicinal chemistry perspective.”, *Current opinion in drug discovery & development*, vol. 11–4, 2008, pp. 559–568.

[38] Rojas, R. “The backpropagation algorithm”. In: *Neural networks*, Springer, 1996, pp. 149–182.

[39] Samarasinghe, S. “Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition”. CRC Press, 2016.

[40] Schrödinger, LLC. “The PyMOL molecular graphics system, version 1.8”. Available in: http://www.pymol.org, November 2015.

[41] Takayama, K.; Wang, C.; Besra, G. S. “Pathway to synthesis and processing of mycolic acids in mycobacterium tuberculosis”, *Clinical microbiology reviews*, vol. 18–1, 2005, pp. 81–101.

[42] Teodoro, M. L.; Kavraki, L. E. “Conformational flexibility models for the receptor in structure based drug design”, *Current pharmaceutical design*, vol. 9–20, 2003, pp. 1635–1648.

[43] 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.

[44] Wallach, I.; Dzamba, M.; Heifets, A. “Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery”. Available in: https://arxiv.org/abs/1510.02855, Feb 2017.

[45] Wang, C.; Liu, J.; Luo, F.; Tan, Y.; Deng, Z.; Hu, Q.-N. “Pairwise input neural network for target-ligand interaction prediction”. In: IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2014, pp. 67–70.

[46] Waszkowycz, B.; Clark, D. E.; Gancia, E. “Outstanding challenges in protein–ligand docking and structure-based virtual screening”, *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1–2, 2011, pp. 229–259.

[47] Xiao, L.; Wang, C.; Luo, R. “Recent progress in adapting poisson–boltzmann methods to molecular simulations”, *Journal of Theoretical and Computational Chemistry*, vol. 13–03, 2014, pp. 1430001.

[48] Yuriev, E. “Challenges and advances in structure-based virtual screening”, *Future medicinal chemistry*, vol. 6–1, 2014, pp. 5–7.

[49] Zeiler, M. D. “Adadelta: An adaptive learning rate method”. Available in: https://arxiv.org/abs/1212.5701, Feb 2017.

APPENDIX A – AUTODOCK PARAMETERS USED IN DOCKING

```
outlev 1                              # diagnostic output level
intelec                               # calculate internal electrostatics
seed 71277 142557                     # seeds for random generator
ligand_types A C HD N NA OA SA        # atoms types in ligand
tran0 random                          # initial coordinates/A or random
quaternion0 random                    # initial orientation
dihe0 random                          # initial dihedrals or random
torsdof 3                             # torsional degrees of freedom
rmstol 2.0                            # cluster_tolerance/A
extnrg 1000.0                         # external grid energy
e0max 0.0 10000                       # max initial: energy; retries
ga_pop_size 150                       # number of individuals in pop
ga_num_evals 3000000                  # maximum number of energy evaluations
ga_num_generations 27000              # maximum number of generations
ga_elitism 1                          # number of top elitism
ga_mutation_rate 0.02                 # rate of gene mutation
ga_crossover_rate 0.8                 # rate of crossover
ga_window_size 10                     #
set_ga                                # set the above parameters for GA
sw_max_its 300                        # iterations of local search
sw_max_succ 4                         # successes before changing rho
sw_max_fail 4                         # failures before changing rho
sw_rho 1.0                            # size of local search space
sw_lb_rho 0.01                        # lower bound on rho
ls_search_freq 0.06                   # probability of local search
unbound_model extended                # state of unbound ligand
ga_run 25                             # do this many hybrid GA–LS runs
analysis                              # perform a ranked cluster analysis
tstep 0.2                             # translation step/A
qstep 1.0                             # quaternion step/deg
dstep 1.0                             # torsion step/deg
```

MARISTA

PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria Acadêmica
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: proacad@pucrs.br
Site: www.pucrs.br/proacad