CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Learning the shapes of protein pockets

Master's thesis in Computer science and engineering

Alejandro Corrochano
Yossra Gharbi

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Learning the shapes of protein pockets

Alejandro Corrochano
Yossra Gharbi

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Learning the shapes of protein pockets
Alejandro Corrochano
Yossra Gharbi

Supervisor: Peter Damaschke, Department of Computer Science and Engineering
Advisor: Jon Paul Janet, MolecularAI, AstraZeneca
Examiner: Alexander Schliep, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2022

Learning the shapes of protein pockets
Alejandro Corrochano
Yossra Gharbi
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The comparison of protein pockets plays an important role in drug discovery. Through the identification of binding sites with similar structures, we can assist in finding hits and characterizing the function of proteins. Traditionally, the geometry of cavities has been described with scalar features, which are not fully representative of the shape. In this work, we propose a method that creates geometrical descriptors of the pocket shape based on Euclidean neural networks, allowing us to encode their physical features. As a result, we can compare the cavities by computing the Euclidean distance between their respective embeddings. As a way of ensuring that the generated embeddings contain relevant geometrical information, our model was trained on a supervised classification task to predict whether given pockets are druggable. To do this, a new dataset was built from the existing sc-PDB database that served as a reference to set the druggable cavities. Then, the protein cavity detection algorithm Fpocket was applied to generate decoys. The supervised model is evaluated by predicting druggability on held-out data, while the utility of the learned embeddings is assessed by comparing how a pocket changes during a dynamic simulation. The findings obtained are encouraging and point to a possible paradigm shift in the way pocket shape can be learned. All code is available at `https://github.com/acorrochanon/Pocket-shapes`.

Keywords: Protein, cavity, ligand-binding, 3D-equivariance, shape, latent space, e3nn, Fpocket, sc-PDB.

Acknowledgements

This work would not have been possible without the guidance, constant support, and valuable insights from our supervisor Jon Paul Janet. We would like to express our gratitude to him for allowing us to be part of such an exciting and rewarding project. We were driven by his passion for drug discovery to give our most dedicated effort along the thesis journey.

We would also like to thank Peter Damaschke for taking the time to provide helpful feedback that improved the quality of our work.

Finally, we would like to thank our families for supporting us throughout this project.

Alejandro Corrochano and Yossra Gharbi, Gothenburg, August 2022

Contents

List of Figures

List of Tables

1
Introduction

1.1 Introduction

Drug discovery is driven by the desire to cure, prevent or manage diseases by developing new treatments or improving medicines that are already in use. It seeks to create drugs that inhibit or boost the activity of disease-related biological units, thereby reducing their impact and their associated symptoms. Such a process requires a comprehensive, multi-step analysis to characterize these units in order to understand their involvement in illnesses. These biological entities are called targets and are most often genes or proteins.

Specifically, proteins are macromolecules made up of chains of amino acids, connected together by peptide bonds. They fold to form different 3-dimensional (3D) structures, called conformations, which determine the overall shape of the molecule. Since proteins act as potential targets in the treatment of diseases, pharmaceutical companies are continuously designing and developing small molecules, termed ligands, that target these units by binding to particular areas in their structure, allowing the modulation of their biological function. These anchor points, where ligands can bind, are physical voids on the surface or in the interior of a protein that possess suitable properties for favorable ligand-target interactions. They are typically referred to as binding sites, cavities or pockets.

The identification and characterization of pockets are fundamental aspects in determining how to target the protein. In general, ligands must be designed to fit into pockets with suitable specificity and affinity. However, the distinction between cavities that allow ligand binding and those which do not is not fully understood. Therefore, the correct characterization is still an unsolved task. Through a better understanding of pocket shape, ligands and targets can be matched more precisely and rapidly, accelerating the drug discovery process.

1.2 Problem

Pocket shape is one of the most fundamentally essential characteristics of proteins for describing how they interact with preferred and non-preferred chemical compounds [1]. Specifically, it is vital to determine how protein pockets are shaped in order to fully understand the ligand-target binding process [2]. In fact, according to a study of drug-binding pockets, most key features to predict druggability (i.e.,

a protein's ability to bind small, drug-like molecules with a high affinity [3]) were related to the size and shape of the pocket [1]. Traditionally, standard methods of describing the pocket shape have relied on scalar features such as pocket volume or surface area [4]. However, these methods do not capture the geometrical features of the shape in full details.

For a protein with no known ligands, all potential pockets can be determined using computational methods that inspect their conformations (e.g., Fpocket [5]). These methods compute point clouds that represent each potentially druggable pocket, but this is not sufficient to unambiguously determine if they are able to accommodate small drug-like molecules with suitable specificity and affinity. Data-driven methods are constantly being employed to help infer this distinction by comparing pockets of interest to known cavities [4].

A further complication is that proteins in solution manifest a dynamic behavior that plays a vital role in their function. A range from fast vibrational motions to slow conformational changes over time are known to alter the ligand-target binding interaction [6]. It follows that pocket druggability prediction based on a static snapshot of protein structures may miss indicative features that occur from their motion. In practice, molecular dynamic simulations can be used to model the physical movements of atoms and molecules, resulting in multiple conformations of each pocket at different time frames being acquired (e.g., by repeated application of Fpocket, providing dynamic pocket information).

In that context, challenges arise in fully characterizing the shape of pockets using point clouds. A tool capable of establishing this type of shape encoding for examining the trajectories of druggable pockets can have a significant impact on determining which types of ligands could bind to new targets, thereby speeding up the drug discovery process.

1.3 Context

It is not uncommon for researchers to apply machine learning techniques in this area. Schmidtke and Xavier [4] worked on predicting the druggability of pockets in order to exclude binding sites that were unsuitable and focus on those that were most likely to yield positive results. They were able to develop a logistic regression model based on a dataset that included several parameters, such as hydrophobicity and polarity scores. According to their findings, the authors deemed their work to be state-of-the-art in terms of druggability prediction performance, achieving an Area Under the Curve (AUC) value of 0.91. However, there were only few scalar variables, such as volume and surface area, that were used to represent the geometry of a given pocket. This may be limiting because it doesn't represent shape details to full extent. This is due to the fact that while comparing pockets, some of them could share similar values despite being geometrically distinct from one another. For instance, two cavities could have totally different shapes but have nearly identical volumes. Besides, these high-level geometric summary properties are also likely to miss subtle variations (i.e., different structural conformations) that proteins might

exhibit as part of their living, dynamic system.

DeepSite [7] is one of the recent deep models used to detect binding sites. It classifies each point in 3D space based on its local environment to determine whether it belongs to a binding pocket. The probabilities of all points create a 3D cloud, which can be used to obtain the most probable locations of binding sites within the protein structure. DeepSite uses a deep 3D Convolutional Neural Network (CNN) architecture, which is typically used in image classification problems, i.e., convolutional and max-pooling layers fully connected, followed by a final neuron with the predicted class. Despite CNNs being transitionally invariant [8], data augmentation is needed to achieve rotation and inversion equivariance. Consequently, when treating pockets using CNNs, a bigger dataset is required to handle all possible transformations they may exhibit.

Tensor Field Networks [9] are an example of a breakthrough in machine learning architectures that could provide a starting point for abolishing data augmentation. They are built to capture all of the key aspects of any 3D input data by using just one instance of it. This eliminates the need to translate, rotate and invert every sample to get an accurate output, resulting in a faster solution. In one of their demonstrations, they proved the potential of the framework for geometry-related problems by classifying 3D Tetris shapes based on a unique orientation for each training sample. There were no difference in the results when the same samples were evaluated in various random orientations, yielding 100% accuracy.

1.3.1 Thesis contribution: Why is it relevant?

Drug discovery and development relies heavily on geometry for many of its processes. When it comes to designing and selecting compounds that act on certain therapeutic targets, the structure of biological entities often dictates how they interact with one another, making it a crucial tool in drug discovery. For this reason, geometric information is necessary not just to search for small compounds most likely to bind a target (i.e., virtual screening), but also to better understand molecular properties such as toxicity. Furthermore, a thorough knowledge of geometry permits clustering of molecules by groups with similar cavity shapes, allowing faster identification and comparison of analogues. These approaches, along with others that similarly focus on molecular geometry, aim to reduce development costs, therefore speeding up the time it takes to get new medicines to patients.

This research managed to go one step further in the use of shape as a factor in the discovery and development of new drugs. Using geometry-based descriptors of protein binding sites (e.g., cavity volume) is not novel, but the explicit encoding of shape information has not yet been addressed despite its importance in characterizing the geometric aspects of pockets. Given the potential of Euclidean neural networks (e.g., e3nn [10]), which allow precise classification of a 3D object regardless of the angle of rotation and orientation, a new geometry encoding and manipulation paradigm arises. Given the results we achieved, the model can encode the shape of pockets precisely. Sharing these insights with other researchers in the field would

help them address similar difficulties.

1.4 Goals and Challenges

This project was able to decipher the shape of protein binding pockets through the use of deep learning techniques that are able to encode meaningful information of the geometry, using point clouds of different sizes as input. To create a model that generates the latent space (i.e., an abstract multi-dimensional space that encodes an internal representation of what has been observed externally [11]) of pockets' geometrical characteristics, we trained it to discriminate between druggable and undruggable cavities. Indeed, the classification task encourages the relevant production of the embeddings of pocket shape. Since data is based on 3D point coordinates, the model was designed to support equivariance to 3D translations, rotations and inversion, avoiding the use of data augmentation to handle all possible instances of the object in the 3D space of Euclidean geometry.

In practice, pocket detection tools generate data for each cavity in the form of 3D point clouds, which are then used to build computational models (e.g., Concavity [12]). The challenge was to create low-dimensional equivariant embeddings (i.e., latent space) of point clouds. To accomplish this requirement, e3nn was used to retain the full geometric information of pockets.

Once the model had been trained to generate accurate embeddings of the shape, they were used to compare binding sites as well as investigate how some pockets from well-known protein structures change over time. This was achieved by analyzing different molecular dynamic trajectories at different time frames.

2

Theory

In the following sections, we examine e3nn's mathematical details in greater depth. This chapter depends on some fundamental aspects of group theory, which are covered, for example, in the book "Group Theory in a Nutshell for Physicists" [13]. However, subsequent chapters do not require familiarity with this material.

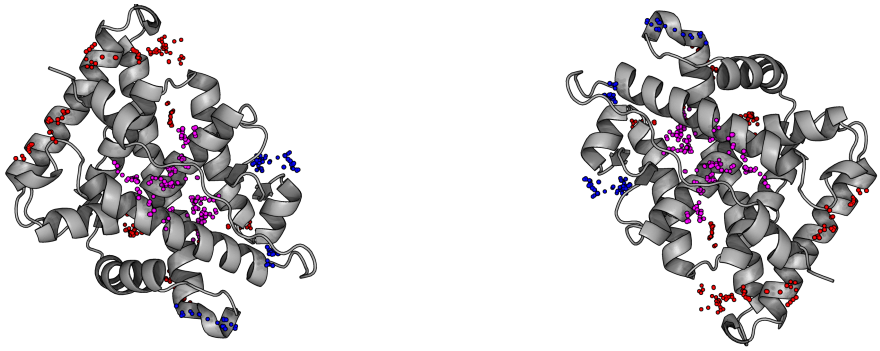
2.1 Overview

Machine learning is increasingly being applied not only in engineering applications such as image recognition and natural language processing, but also in the disciplines of natural sciences, including biology, chemistry, earth sciences, geology, and physics. Many problems in this domain use mathematical models to define natural phenomena. Since symmetry runs rampant in nature [14], these models should respect the symmetries of the system under study in order to faithfully describe the observable events. Similarly, machine learning, which involves inducing predictive models, must recognize and respect symmetry constraints in order to produce physically meaningful and valid results [15]. Consequently, a lot of recent work has focused on developing neural networks that are equivariant to certain groups of symmetries (e.g., equivariant to rotation means that a rotation of input features results in an equivalent rotation of outputs). E3nn is among them [10].

E3nn is able to consistently recognize two representations when they exhibit the same entity. For instance, given a protein and a rotated copy, the network learns that both representations are truly describing the same entity (Figure 2.1).

E3nn accomplishes this milestone by building a family of networks that can handle the symmetries of 3D Euclidean space efficiently. This includes equivariance to all translations, rotations and reflections, along with arbitrary finite combinations of them. This is far from trivial because rotations in 3D do not commute; that is the composition rule is not commutative. Written another way, for two 3D rotation matrices M and N, $MN \neq NM$ [13].

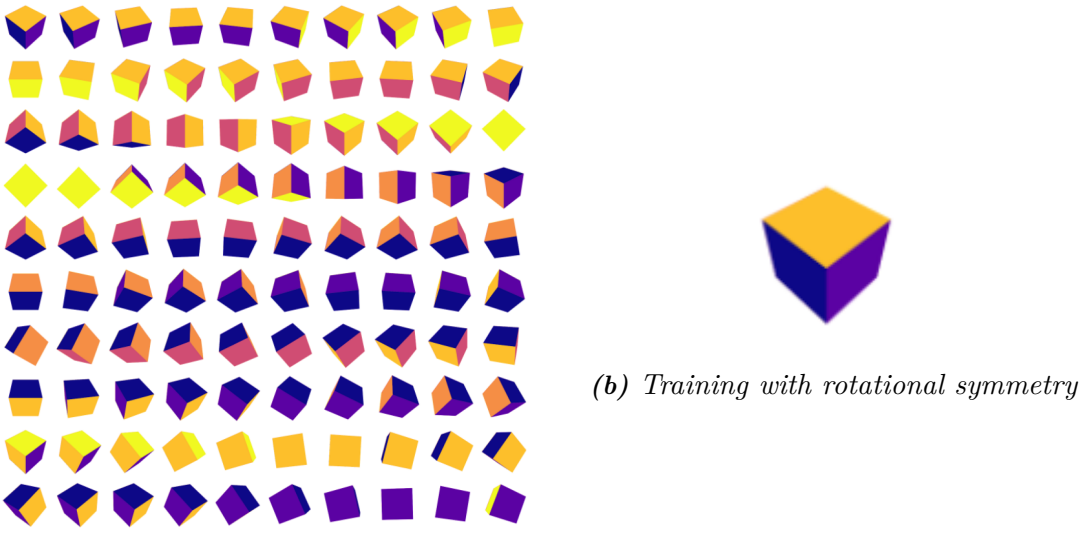
3D rotation equivariance removes the need of using data augmentation to identify features in any 3D rotation, guaranteeing a smooth transformation under rotation and reducing dramatically the computational and training effort [9]. Hence, the performance of the model is significantly improved. In contrast, machine learning



(a) Original *(b) Rotated copy*

Figure 2.1: *Images of a protein and a rotated copy. E3nn understands that the two representations describe the same entity. The points correspond to the pockets of the protein.*

models that were not built to control symmetry require data augmentation (Figure 2.2). For 3D data, it needs approximately 500 fold augmentation [16].



(a) Training without rotational symmetry *(b) Training with rotational symmetry*

Figure 2.2: *Example of data augmentation needed to train a neural network to recognize rotation equivariance for a cube without (a) and with (b) inbuilt rotational symmetry [16].*

In e3nn, scalars, vectors and geometric tensors (i.e., N-dimensional arrays) are used to represent data, and continuous convolutions are utilized to map between these representations [15]. The network consists of layers built on equivariant filters, which are the product of a learnable radial function and a spherical harmonic. Spherical harmonics are particular functions defined on the surface of a sphere to the real or complex numbers, whose Laplacian is zero [10]. They are specifically useful for solving rotationally equivariant problems.

The layers of the network act on 3D coordinates of points and features at those points. In other words, the input and output of each layer are a set of point coordi-

nates in $\mathbb{R}^3$ and a vector in a representation of $O(3)$ (i.e., the group of 3D rotation and inversion) associated with each point coordinates [9].

E3nn demonstrates that composing equivariant layers yields an equivariant network. Thus, proving each layer of a network is equivariant suffices to prove that the network as a whole is equivariant. Furthermore, if a layer is equivariant to translation, rotation, and inversion independently, then it is equivariant to all possible arbitrary combinations of these three transformations as well. This can be expressed mathematically as [9]

$$f(D(g)x, w) = D(g)f(x, w), \tag{2.1}$$

where

- f is the continuous convolution layer of e3nn,
- x is a point coordinates,
- g is an Euclidean group (i.e., group of all rotations, translations, inversion, and their combinations),
- $D(g)$ is the representation of g acting on x. For instance, if $x \in \mathbb{R}^3$ and g is a rotation, then $D(g)$ is a 3×3 rotation matrix. In general, the dimension of $D(g)$ depends on what it is acting on,
- w are the model weights.

In other words, computing the function first, and then applying a symmetry transformation has the same result as applying a symmetry transformation first, and computing the function afterwards.

To get a better understanding of what e3nn is, following are its key features.

2.2 Feature space

E3nn acts on point clouds and features at those points. In practice, it produces a stack of K features by assigning to each position $x \in \mathbb{R}^3$ a feature vector $f(x)$ that consists of a number of geometrical quantities, such as scalars, 3D vectors, and tensors, stacked into a single K-dimensional vector.

The question is how the network is able to precisely distinguish the behavior of distinct geometrical quantities of the feature space under actions of the group $O(3)$. The answer is to categorize the features by how they transform with respect to the group $O(3)$. The categorization aligns with the notion of irreducible representations.

2.3 Irreducible representations

There is a one-to-one correspondence between the type of a geometric quantity and the type of inducing representation D, which describes how data would behave under

rotational symmetry actions [15]. For instance, when rotated and inverted, a scalar remains invariant, whereas a 3D vector changes at the same rate with rotation and flips when inverted.
To acquire a better understanding of what it means to be a representation of the group $O(3)$, group representation theory is first introduced. Second, we explain how any such representation can be decomposed via a change of basis into elementary building blocks termed irreducible representations.

Given a group G, a group representation D associates each element g with an invertible $n \times n$ matrix, such that [13]

$$D(gf) = D(g)D(f), \tag{2.2}$$

where gf indicates the composition of two transformations $g, f \in G$, and $D(g)D(f)$ denotes matrix multiplication.

Furthermore, a geometrical quantity, under the action of $O(3)$, transforms with a representation of $O(3)$ [9]. In concrete, a transformation $g \in O(3)$ can always be decomposed into a rotation $r \in SO(3)$ (i.e., group of 3D rotation) and inversion $i \in \{e, I\}$ (i.e., group of inversion) [10], written as $g = ri$. Therefore, the representation of $O(3)$ is the product of the representations of $SO(3)$ and inversion; that is $D(g) = D(r)D(i)$, for all $g \in O(3)$.

In general, any representation of $SO(3)$ can be decomposed into elementary building blocks called irreducible representations, indexed by l, and of dimension $2l + 1$, for $l = 0, 1, 2, \cdots$. The irreducible representations of $SO(3)$ are known as the Wigner-D matrices of order l, denoted D^l [15]. Including parity inversion, we obtain for each l an even (i.e., do not change sign under a parity inversion) and odd (i.e., change sign under a parity inversion) irreducible representations [10]. For example, there are even and odd irreducible representations of $l = 0$, defined as a scalar and pseudoscalar, respectively.

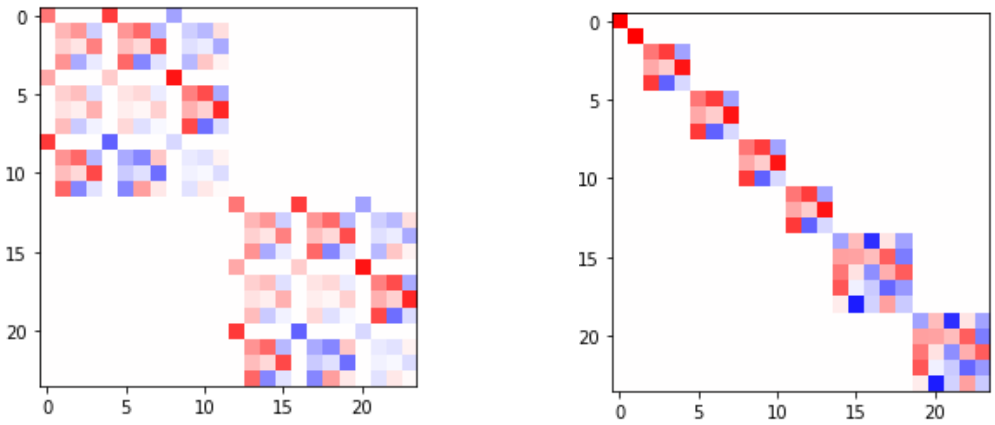
In e3nn, all operations are performed using irreducible representations; for instance, the tensor product of two irreducible representations of indices l_1 and l_2 is decomposed into direct sum of irreducible representations. Expressed mathematically [15],

$$l_1 \otimes l_2 = |l_1 - l_2| \oplus \cdots \oplus (l_1 + l_2)\,. \tag{2.3}$$

An illustration of this decomposition in the form of a concrete example is shown below.

Let x and y be two feature vectors, where x consists of two vectors (i.e., two odd $l = 1$) and y consists of one scalar and one pseudovector (i.e., an even $l = 0$, and an even $l = 1$, respectively). Let R be a random rotation matrix, and D_x and D_y are the representations of the rotation R appropriate to x and y, respectively. Let A be the tensor product of the two feature vectors x and y. The representation of R for A is $D_x \otimes D_y$, which is plotted as an image for the purpose of visualization. The image has one square for each element of the array. It is shown that the representation of R is not an irreducible representation (Figure 2.3 (a)).

However, through the use of a change-of-basis matrix, it can be transformed into a direct sum of irreducible representations. Rather than have to do this manually every time a tensor product needs to be computed, e3nn automates this process and reproduces the representation of R into a block diagonal structure of irreducible representations (Figure 2.3 (b)).



(a) Reducible representation *(b) Irreducible representation*

Figure 2.3: *Displaying the decomposition of the reducible representation into irreducible representation as images. Every square has a different color based on the corresponding array element value and the color map used. The image is stretched individually along axis to fill the bounding box in data coordinates.*

In practice, the Irrep class (singular) in e3nn represents the irreducible representations of $O(3)$, describing how features would behave under rotations and inversion. The Irreps class (plural) consists of a direct sum of irreducible representations; it combines different Irrep objects. This is the constitutional class in e3nn to be able to build a working network. A typical set of irreps has the following structure: $m \times lp$, where:

- m is the number of features,
- l is the rotation order (describing whether the features are scalars, vectors, etc),
- p describes the parity. Two choices are available: e or o referring to even and odd, respectively.

2.4 Continuous-filter convolutions

The use of continuous filters is the core of e3nn, as they are able to handle unevenly distributed data at random positions. Continuous convolutions are generally defined as [17]

$$(f * g)(x_i) = \int_{-\infty}^{+\infty} f(x_j) \odot g(x_i - x_j)\, dx_j. \tag{2.4}$$

They (i.e., continuous filters) assign to each position $x_j \in \mathbb{R}^D$ a feature vector $f(x_j) \in \mathbb{R}^F$, and the continuous kernel function g that maps a relative position in $\mathbb{R}^D$ to a kernel weight in $\mathbb{R}^F$. The Hadamard-product is denoted as $\odot$.
In the case of 3D point clouds, we have D = 3.

Monte-Carlo integration allows continuous convolution to be approximated as [17]

$$(f * g)(x_i) \approx \frac{1}{N} \sum_{n=1}^{N} f(x_n) \odot g(x_i - x_n), \tag{2.5}$$

where N is the number of point positions.

In the case of e3nn, the kernel function g is a learned parametric function composed of a multi-layer perceptron and a spherical harmonic. Spherical harmonics are implemented in order to design rotation-equivariant filters. Furthermore, since the filters and layer input inhabit representations of the group $O(3)$, the tensor product of representations are utilized as an alternative to the Hadamard product in order to reproduce outputs that can be fed into downstream layers and also transform fittingly [10].

3

Methods

3.1 Overview

In this work, we present a different deep learning-based approach for learning protein pockets shapes, inspired by the idea of equivariance to 3D translations, rotations and inversion instead of data augmentation. To fulfill these requirements, we built a model based on e3nn architecture. We adapted it to the problem of binding sites, and added functionalities that allow to easily generate a low-dimensional representation of pockets that encodes thoroughly their shapes and produce predictions for binding sites, being druggable or not. The input in our case is a 3D structure of a pocket represented with point clouds that the model can use to decipher the shape.

In order to learn meaningful embeddings, we ran a classification task, allowing us to encode properties about known pockets into the feature space. For this purpose, data from real protein structures is required.

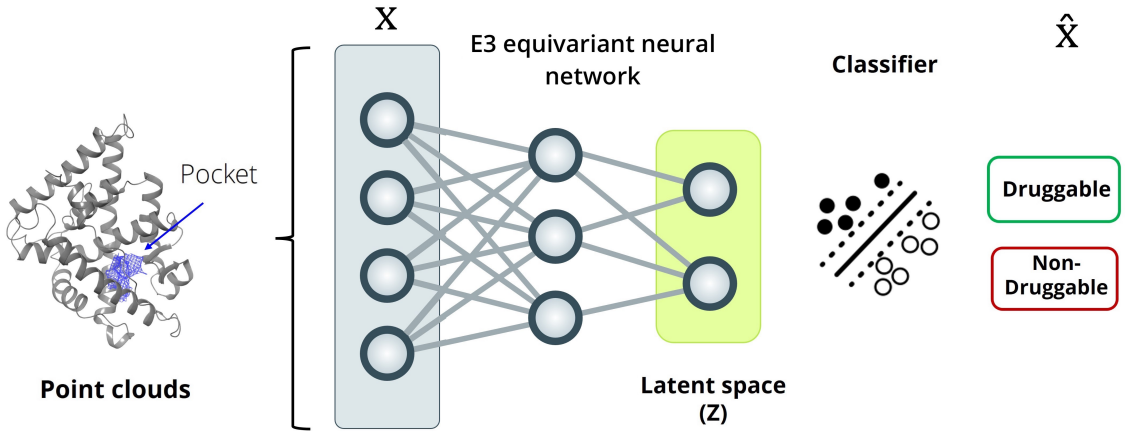


Figure 3.1: *Overview of the designed model's mechanism to learn the shapes. Features from latent space are used to classify pockets.*

3.2 Dataset and preprocessing

3.2.1 Sc-PDB dataset

The sc-PDB [18] is a specialized database that focuses on ligand-binding sites in proteins derived from X-ray crystallography [19]. The database incorporates known binding sites, accompanied with their protein structures and ligands, in the form of point clouds. The 3D shapes of binding site structures were generated by VolSite [20]. After projecting the protein on a 3D grid lattice of resolution 2 Å, VolSite characterizes each voxel by a pharmacophoric property, based on the properties of the neighboring protein atoms. Data is displayed in mol2 files, with atoms encoding each property.

The data set is composed of 17,594 binding sites, conforming to 16,612 protein structures. There are 942 duplicate protein structures in the dataset because some proteins have more than one ligand-binding site complex (Figure 3.2), but each sc-PDB protein/pocket pair is unique.

Datasets such as sc-PDB are suitable for machine-learning approaches, particularly those that rely on deep neural networks, which typically require a large number of examples for training. Sc-PDB's wealth of data makes it an excellent starting point and is available to download from `http://bioinfo-pharma.u-strasbg.fr/scPDB/`.

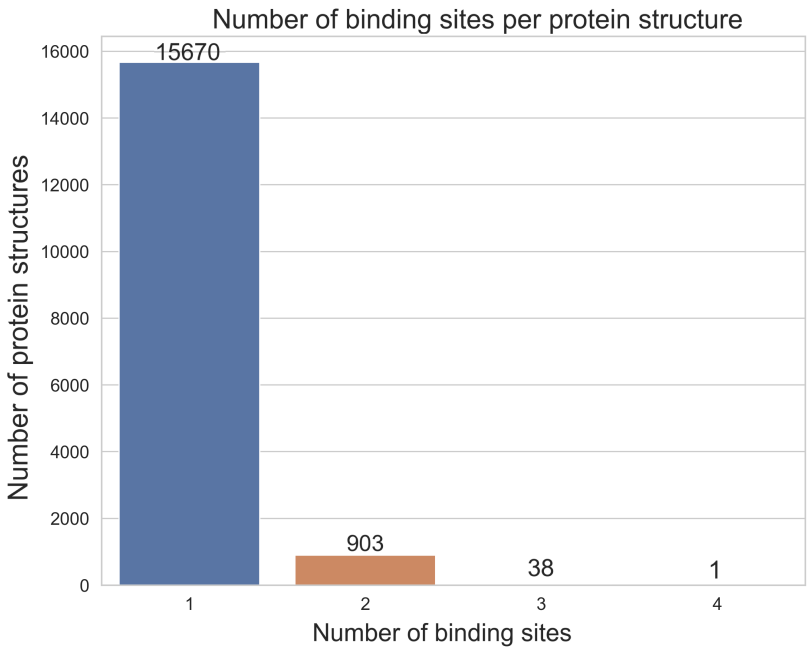


Figure 3.2: *Number of binding sites present in each protein in the sc-PDB database.*

3.2.2 UniProt IDs clustering

The sc-PDB database contains instances of the same protein, or closely related proteins, being crystallized multiple times. In order to create a fair train/test split and ensure any excessively similar proteins are not in the same split, we grouped proteins with similar structures by UniProt IDs.

UniProt ID is the unique identifier assigned to the set of proteins derived from the same gene [21]. The mapping of protein structures to their respective UniProt IDs was conducted by the Retrieve/ID mapping service (`https://www.uniprot.org/uploadlists/`), but was performed using the UniProt Python Application Programming Interface (API) to access and query data programmatically. Protein structures were then clustered according to their UniProt IDs. This means that all proteins that share a common uniprot ID will therefore belong to the same partition. The cases where protein structures belong to multiple UniProt IDs were removed. Finally, a total of 5650 clusters were obtained, of which 3213 contained a single protein structure and 2437 contained multiple protein structures. The number of structures per UniProt ID ranges from 1 to 280, with a median of 1 and a mean of 3.27. The cluster of UniProt ID (P24941) is the largest of all, containing 280 protein structures.
The code of the UniProd IDs clustering can be found in the file unisplit.py (Appendix B).

3.2.3 Fpocket

Sc-PDB only provides known binding sites (i.e., druggable pockets). Undruggable pockets are required in order to ran a classification task such that the neural network is forced to encode meaningfully the physical information of the pocket shape. Based on the learned geometrical features, the model is able to discriminate whether cavities are druggable or not.

To handle this issue, Fpocket was exploited to find pockets of both types. Fpocket is a geometry-based software that uses the concept of alpha spheres and Voronoi tessellation to detect cavities. It principally relies on a two-step algorithm: it first finds cavities in a protein structure and then scores them from the most probable druggable pockets to the least probable druggable pockets. For instance, a structure with 20 cavities has pocket with 0 ID as the most druggable and 19 as the least. Aside from producing point clouds for each detected pocket, Fpocket also provides chemical and geometrical summary features for each cavity. Specifically, for the classification task, we used the number of Voronoi vertices, mean alpha-sphere radius, mean alpha-sphere apogee area, and convex hull-computed volume.

To rigorously determine which of the Fpocket cavities were druggable and undruggable, these were compared to pockets in sc-PDB to find the closest match. Any pocket that was similar to the sc-PDB pockets was marked as druggable, whereas the rest were marked as undruggable. Considering the Fpocket ranking system previously described, it is reasonable to presume that the matches between Fpocket and sc-PDB pockets will occur with those that hold the best druggability score.

For this comparison to happen, a workflow carefully designed and developed to carry out this task is detailed below.

3.2.4 Assigning labels to pockets

The method labels new pockets generated by Fpocket using sc-PDB cavities as a reference. Following the steps outlined below, we were able to generate undruggable cavities:

1. Pockets detection using Fpocket for each protein contained in sc-PDB.
2. Calculation of the centroids of each cavity detected by Fpocket and the known druggable pockets in sc-PDB.
3. Measurement of distances between the centroid of each pocket and the known binding sites provided by sc-PDB.
4. Given a protein structure, labeling as druggable the pocket generated by Fpocket whose centroid has the smallest distance to its corresponding sc-PDB centroid.
5. Labeling as undruggable all the remaining pockets of the same protein structure.

3.2.4.1 Format conversion

Fpocket requires protein structure data to be stored in a pdb format. However, data was supplied as mol2 files by sc-PDB. There is a significant difference between the two formats, as they encode data differently. As a result, the Openbabel [22] library was used to perform the conversion. Initially, the files were assumed to be properly converted, but after studying certain strange results, it was found that multi-chain structures were being incorrectly converted to single-chains. In the case of protein structures with nearly-identical chains, this may lead to an error, as there is a risk of creating several copies of the same pocket with different labels when applying Fpocket to it. For instance, if a protein has two identical chains, A and B, and the cavity is located at A, Fpocket may find the mirror image of the cavity at B. As a result of this, two different labels are assigned to the same shape. Hence, since sc-PDB is only provided for one chain, working with the chain at which the cavity is situated is indispensable.

To solve this issue, we first stored all the residues corresponding to the chain where the ligand is located. Next, we removed all the unwanted chains from the pdb file using BioPandas [23].

Figure 3.3 illustrates the result of treating a multichain protein with the described approach.

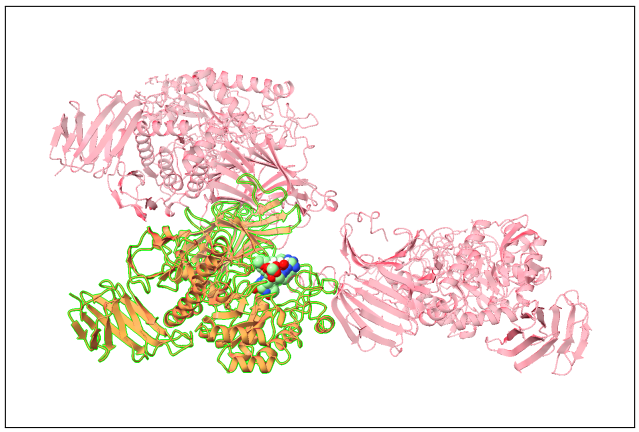


Figure 3.3: *Multichain protein 2f2h and its druggable cavity, provided by sc-PDB. The highlighted section corresponds to the chain where the cavity is located.*

During the execution of the outlined approach, few exceptions were spotted in several structures, revealing potential annotation errors. The reason is that in certain ligands, the chain referred to did not appear to exist in the protein. Because this represents a relatively low number of exceptions, they were discarded. Additionally, we found that the ligand can sometimes be misplaced, or annotated on the wrong chain. Figure 3.4 illustrates the point very clearly. Based on raw data from the 1srh structure, the ligand is located on the chain A, highlighted in green. However, the ligand appears to be situated on chain B, which is distant from chain A.

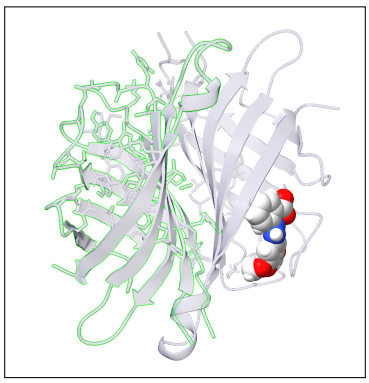


Figure 3.4: *Multichain protein 1srh and its druggable cavity. The highlighted section corresponds to the chain where the cavity should have been located according to the annotations.*

Since similar errors may occur, we calculated the minimum atomic distance between the selected chain and its respective ligand to ensure that if it exceeds a certain threshold, the data is discarded. The threshold was set based on the distance distribution of 1000 randomly selected structures. According to our analysis, we believe it was sufficient to use a sample from the data to get a sense of how the atomic distance is distributed without having to apply it to the entire dataset. Conforming to the distribution shown in Figure A.3, the threshold is set to 3 Å. Any cases that fail this condition produce an error message indicating that no cavities have been found. *All the conversion procedure can be found in scpdb_funcs.py (Appendix B).*

3.2.4.2 Fpocket benchmark

Fpocket has a number of parameters that control the detection of pockets. As part of our research to ensure the best match between the pockets in the sc-PDB and the ones generated by Fpocket, we conducted a benchmark on 500 random structures, seeking to choose the set of optimal Fpocket parameters out of 27 possible configurations that were suggested to us by an expert in this area. Different performance indicators were considered in order to determine the most suitable setting, such as the average minimum distance between the centroids of the cavities generated by Fpocket and those in the sc-PDB, the number of pockets discovered for each structure, and the number of point coordinates per pocket.

Figure A.2 shows the values obtained for each of the above-mentioned metrics, where each point represents a distinct configuration of the Fpocket parameters. With an average distance of 3.66 Å and a median value of 2.88 Å, the orange dot provides the best results among the different settings.

Using the optimal Fpocket configuration, Figure 3.5 displays the volume distribution of sc-PDB and Fpocket cavities using the convex-hull method. While the correspondence is imperfect, the distribution indicates how accurate the current Fpocket configuration is at detecting similar cavities to those from sc-PDB.

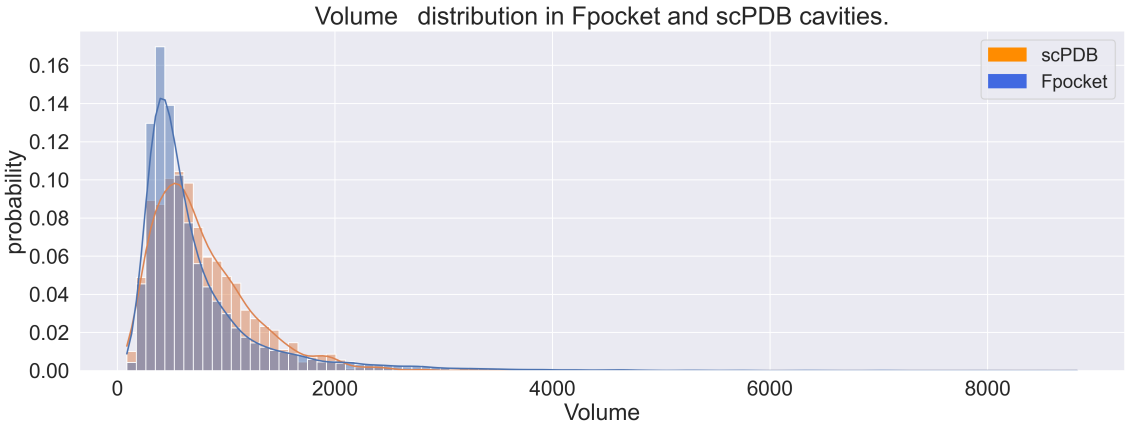


Figure 3.5: *Volume distribution of the Fpocket and sc-PDB cavities for 2000 protein structures.*

However, as illustrated in Figure 3.6, sc-PDB has more points per cavity than Fpocket. For the same structure, e.g., 1v8l, the matched pocket from Fpocket and sc-PDB has 29 and 138 points, respectively. Since the number of points can affect the quality of the model training and yield misleading results, we decided to work exclusively on Fpocket-generated data.

3.2.4.3 Data filtering

Among the identified druggable cavities computed by Fpocket and labeled by comparison to sc-PDB, 11,203 were the first based on Fpocket ranking system. In contrast, several pockets appeared to be located the closest to sc-PDB binding sites,

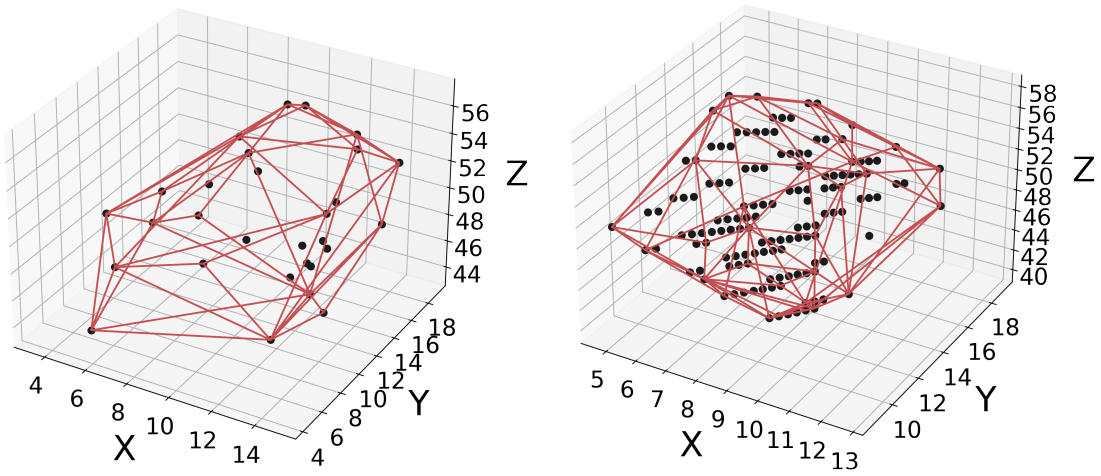


Figure 3.6: *1v8l protein's druggable cavity as detected by Fpocket (left) and sc-PDB (right).*

yet not necessarily be the best choice according to the algorithm. In this case, a potential conflict in the druggability criteria might be apparent. Such cases were dismissed for the sake of maintaining the integrity of the dataset.

Additionally, it should be noted that the distance between a detected pocket and sc-PDB binding site can occasionally be fairly large. In some situations, the best Fpocket's pocket can be found >15 Å away from its ground truth, i.e., known binding site from sc-PDB.

Therefore, considering the distance and druggability score, only pockets ranked 0 or 1 by Fpocket that are less than 5 Å away from sc-PDB centroids were retained, the other protein structures were discarded. This ensures that the detected pockets are high-quality and accurate.

3.2.4.4 Number of cavity points in Fpocket and sc-PDB

At this stage of data preprocessing, each sc-PDB binding site has a druggable Fpocket pocket matched to it. However, we observed that the number of points composing pockets from Fpocket and sc-PDB are different (see Figure 3.6). Thus, for consistency, we decided to use Fpocket point clouds for both druggable and undruggable data.

Furthermore, Figure 3.7 shows how druggable cavities tend to be larger than the undruggable ones. The discrepancy in point coordinates composing druggable and undruggable pockets may cause the network to classify based simply on the number of points instead through the learning of pocket shape. In response, undruggable pockets with fewer than 25 coordinates were removed from the dataset to approximate the number of points between the two types of pockets.
The script extract_data.py (Appendix B) contains the implementation of every step of the data preprocessing pipeline.

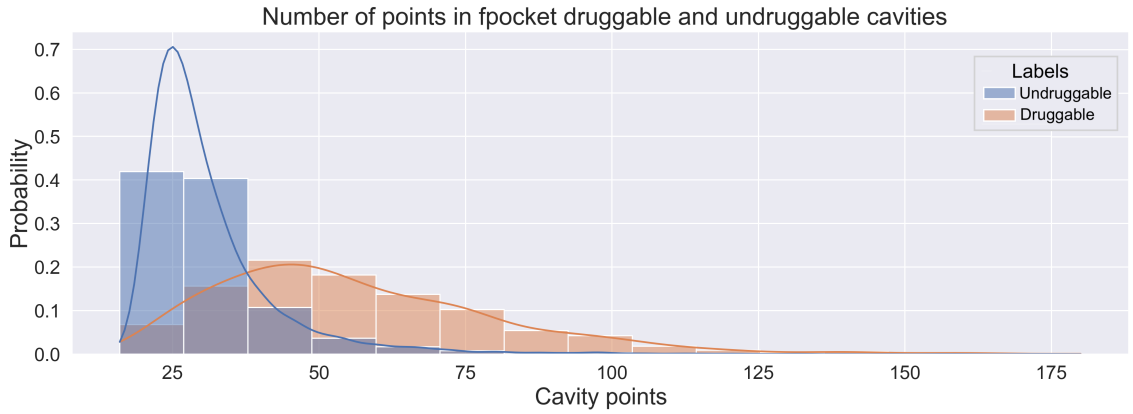


Figure 3.7: *Normalized distributions of the number of points that compose druggable and undruggable in 1000 Fpocket's cavities. Although druggable pockets are represented with a larger number, the amount of samples is generally lower compared to the undruggable type. Thus, we prevent the network from learning based on the number of points.*

3.2.5 Data splitting

For an unbiased evaluation of the model's performance, the data was split into training/test sets based on UniProt IDs (i.e., all structures of a single gene must be in the same split). This ensures evaluating the model on structures not present in the training set, as a result avoiding data leakage during testing. Indeed, when assessing the quality of a machine learning model, it is necessary to evaluate it using a distinct dataset. Both training and test samples would have nearly identical structures if different structures of the same protein are included in both sets. This leads to potential overfitting and possibly incorrect hyperparameter tuning, and as a consequence to overly optimistic assessment of the model. *The source code can be found within the GitHub repository, in the file split_data.py (Appendix B).*

3.2.6 Final dataset

Following the cavity-based approach and the uniprot IDs split, we generated a dataset of 65,662 cavities of which 63,268 were allocated in the training set and 2394 in the test set. Additionally, 10% of the training cavities were randomly retrieved to create a validation set. Thus, the final number of cavities in each split is 56,941, 6327, and 2394, representing the 87%, 9%, and 4% of the whole dataset, respectively. As for the type of the binding sites, each set contained around 86% of pockets labeled as undruggable and 14% druggable.

3.3 Our approach

3.3.1 Network architecture

Given the advantages that Euclidean neural networks offer, the framework fits the necessities of the problem. Because cavities are 3D objects and ligand binding is not affected by translation or rotation, geometric embeddings will not be sensitive

to these differences. This will allow for an easier comparison of cavities in different proteins as they translate and rotate during molecular dynamic simulations.

In this work, we present an Euclidean neural network consisting of four equivariant convolutional layers able to capture the geometric features of any 3D element.

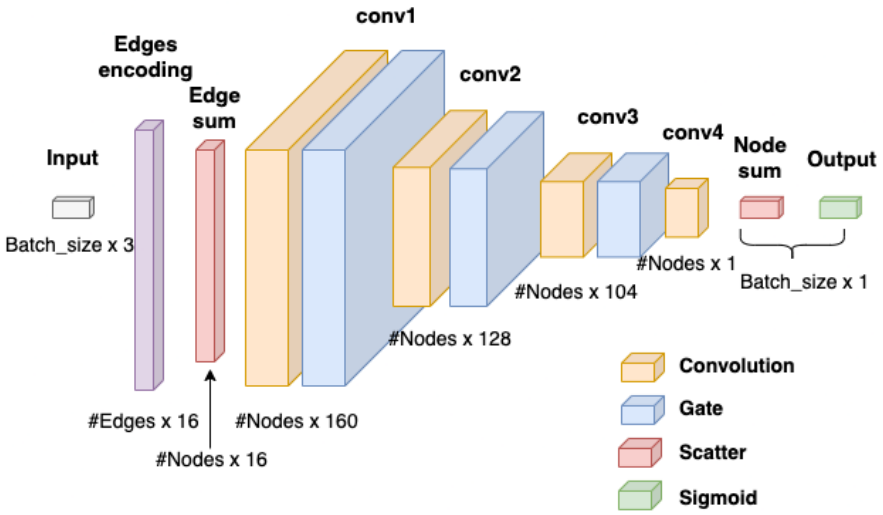


Figure 3.8: *Best model's architecture. It consists of 4 convolutional layers and 3 nonlinear gate functions. Number of edges depend on the distance cutoff established. This could eventually affect the number of nodes, that must match the sum of all the number of points contained within the batch.*

Figure 3.8 depicts the underlying structure of the assembled model. In more detail, we first transformed the point cloud into a graph according to a cutoff value of 6.2 that represents the maximum distance nodes can hold between themselves to be considered connected. Edge features were then constructed using the pairwise difference between all the nodes composing the graph that fulfilled the previous requirement. The batching was designed to ensure that only edges within the same point cloud were created. Then, we used the edge information to initialize each of the node features through a scatter operation. At this stage, the model already has an understanding of the input's geometry. This is based on a use case example provided by the framework authors and available at (`https://github.com/e3nn/e3nn`).

Next, we have the block of convolutional and gate layers. This is the part where the learning process takes place. Here, we captured the geometric features of the point clouds and learned the boundaries in the latent space between the two types of shapes. As can be seen in the figure above, the three first layers are constituted by a convolution and a nonlinear Gate layer, both of which contribute to maintaining embeddings equivariance. The gates have been designed so that the latent space is reduced from 160 to 88 before the last convolution compress it to one scalar per node. Through a new scatter operation, these were all summed to give a pocket-level scalar, which was finally transformed with a sigmoid into a label.

The most remarkable difference between the models that we trained lies in the number of convolutional layers. For instance, the model that uses three layers will

end up with a latent space of 104 features before the scatter node-scalar operation. We will later examine how this can affect its performance.

3.3.2 Hyperparameter tuning

E3nn supports creating arbitrary numbers of geometric features of different types. This directly affects the number of learnable parameters in the network, the dimension of the latent space and the model's learning capacity.

As mentioned in the previous section, in the point cloud-to-graph conversion, there is a cutoff value that establishes the maximum distance two separate nodes can hold to be considered connected. This is of utmost importance. If the value is too small, the most distant points in the cloud can be missed, causing a misleading perception of the pocket. In contrast, when the cutoff is too large, the training becomes computationally expensive, as more edges will be collected, which increases the number of edge features significantly.

Based on the models performance, we noticed that adding convolutions results in greater performance. We also experimented with larger latent spaces by modifying the input and output features of the gates. Unfortunately, this has not led to a more precise learning. Model 4CV+ — plus sign denotes the greater latent space — in Figure 4.2 corroborates this. Here, the size of the learned features at the first layer was set to 256. Yet, this assumption is subject to further testing, as we believe there is still room for improvement through the increase of the network's complexity.

Lastly, we studied how different learning rates and batches size affected the training. Small values minimize the probability of getting stuck at a local minima. However, given the size of the dataset, we had to find a trade-off between time and performance. Further details can be found in the upcoming section.

3.3.3 Training details

The training was conducted in an environment that includes Python 3.9 [24] and e3nn version 0.4.4. PyTorch [25] is also used as the deep learning framework since e3nn is a modular PyTorch framework for Euclidean neural networks.

We trained each network under a fixed seed over a maximum of 50 epochs in unbalanced batches of 256 cavities. The internal weights were updated with a variant of the gradient descent algorithm, Adam [26], and a learning rate of 0.001. The binary cross entropy was the loss function chosen given the nature of the problem. Training was carried on a Tesla V100-SXM2-32GB GPU, with a total training time of 1 hour for the e3nn model that achieves the best performance. Evaluation was conducted simultaneously to measure the capacity to predict on unseen data. To control overfitting on training data, we applied the early stopping technique with a 12-epoch tolerance under a fixed random validation split. *Implementation of the code can be found in learning_shapes_4cv.py (Appendix B).*

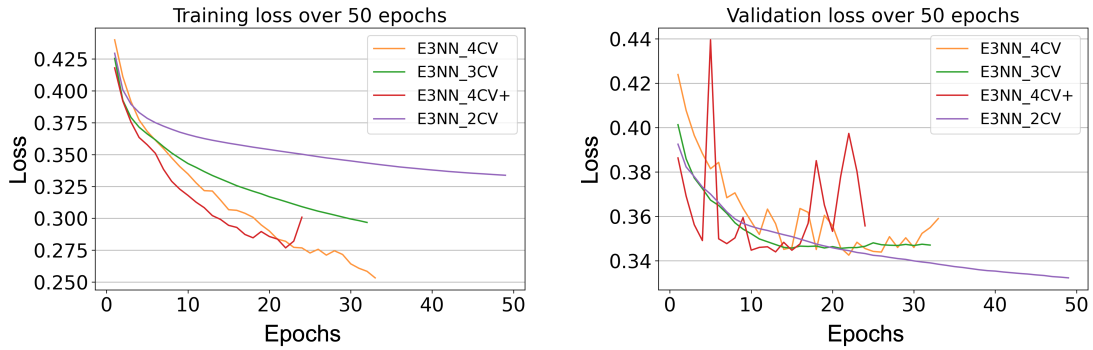


Figure 3.9: *Training and validation losses for 50 epochs. As the complexity of the models increase through the addition of new convolutional layers or the expansion of its latent space (4cv+), so does the learning capacity. However, this affects on the generalization performance, as seen on the right plot. Hence, a trade-off must be found between these two aspects.*

Besides, a logistic regression (LR) was implemented — using the Scikit-learn library [27] — for benchmarking purposes, i.e., to investigate if our network was able to learn the shape rigorously based only on point clouds compared to the LR model that uses explicit geometry-related features. These features are the number of Voronoi Vertices, mean alpha-sphere radius, mean alpha-sphere apogee area, and convex-hull computed volume. As no hyperparameter tuning was conducted here, the training data contains the validation split to evaluate the generalization capacity of the networks. *The file log_classifier.py (Appendix B) contains the code implementation.*

Model	*e3nn-2CV*	*e3nn-3CV*	**e3nn-4CV**	*e3nn-4CV+*	*Log reg*
Precision	0.6167	0.6448	**0.7077**	0.7664	0.6509
Recall	0.4091	0.5329	**0.5947**	0.4848	0.3390
F1 score	0.4919	0.5835	**0.6463**	0.5973	0.4458
AUC	0.8103	0.8645	**0.9017**	0.8855	0.8898

Table 3.1: *Metrics obtained from the models that were trained. Highlighted column corresponds to the one with the best performance on the training set according to the resulting F1 score.*

Following the training, each model was subjected to a final execution under the same split to annotate its performance. This is shown in Table 3.1. Based on this, we can observe that the model with four convolutional layers (i.e., e3nn-4CV) performed the best.

4

Results and Discussion

4.1 Dicussion

In this section, we present the results achieved on the 2394 cavities contained in our test set, in which 330 correspond to the druggable type and 2064 to the undruggable.

Considering the data shown in Table 4.1, we can state that models based on Euclidean neural networks are successfully encoding the geometry of the cavities and discriminating between both types. Although the logistic regression has a slightly superior value in the AUC metric, differences are moderate with respect to all the other approaches. Besides, the F1 score obtained by the e3nn model with four convolutional layers is the best, which denotes a better recall-precision balance in comparison to the logistic regression classifier. Thus, all e3nn models that were built can successfully classify pockets based on point clouds.

Since only crude hyperparameter optimization was conducted, it is likely that the classification performance could be improved with further tweaking. However, this is not the primary objective of this work. Instead, the main goal is to produce geometric embeddings for pockets that contains relevant physical information about the pocket shape. The ability to perform efficient classification with these models demonstrates that the learned representations do contain the desired information.

Model	*e3nn-2CV*	*e3nn-3CV*	**e3nn-4CV**	*e3nn-4CV+*	*Log reg*
Precision	0.5858	0.5623	**0.5830**	0.6027	0.6768
Recall	0.4242	0.5061	**0.5000**	0.4091	0.3364
F1 score	0.4921	0.5327	**0.5383**	0.4878	0.4494
AUC	0.8056	0.8323	**0.8404**	0.8179	0.8917

Table 4.1: *Test metrics of the models that were trained. Highlighted column corresponds to the e3nn network with the best performance on the test set according to the F1 score. Despite achieving a higher AUC in the logistic regression, the e3nn does a better job at predicting true druggable cavities.*

4.1.1 Confusion matrices

The confusion matrices for the best performing e3nn model (i.e., e3nn-4CV) and the logistic regression confirm the metrics shown previously. As illustrated in Figure 4.1, e3nn-4CV has twice as many false positives as the logistic regression, which demonstrates a better precision from the logistic regression classifier. However, e3nn-4CV achieves a better recall, with 165 true positives against the 111 of the counterpart. Considering the high degree of imbalance in our dataset, results are considered positive. Overall, e3nn-4CV finds the best trade-off between these two metrics, with a F1 score of 0.5383.

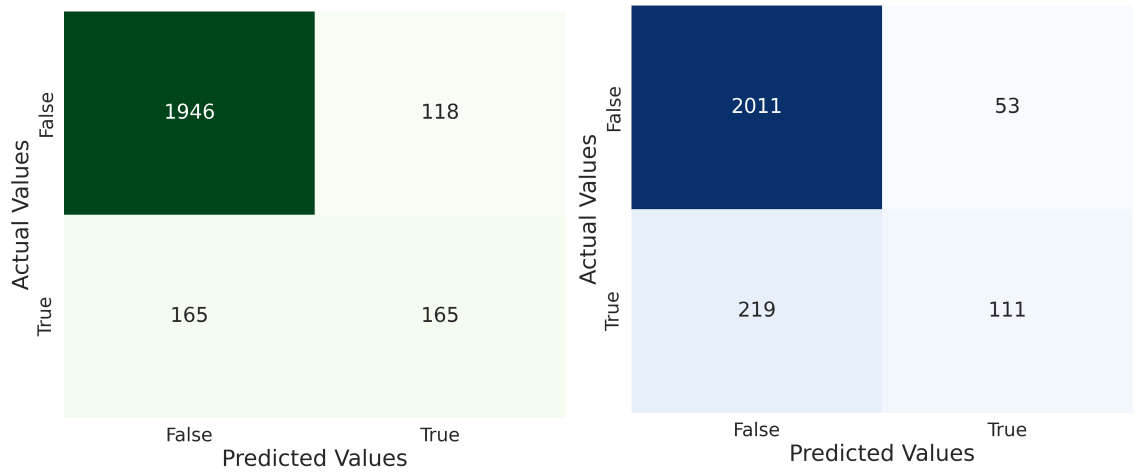


Figure 4.1: *Confusion matrices obtained for the best performing models. The left-sided shows the results for the e3nn model with four convolutional layers that uses point clouds, whereas the right one is the logistic regression that uses the scalar geometric features retrieved from Fpocket.*

Despite the fact that the e3nn matrix is not perfect, it suggests that a meaningful physical description of druggable pockets is encoded in the learned representations. However, future releases should aim to reduce the number of false negatives, as this can result in ignoring valid pockets in real-world applications.

4.1.2 ROC curves

Figure 4.2 depicts the ROC curves for each of the e3nn models and the logistic regression. Again, it is critical to emphasize that this benchmark is done with the only purpose of verifying if the geometric embeddings produced by the e3nn models contain relevant geometric information.

Based on the figure, a clear pattern can be seen between the performance of the model and the number of convolutional layers that are employed. This might suggest increasing the complexity of future models to reach more accurate embeddings. However, there are potential risks by following this reasoning. In the approach 4CV+, we considered the same number of layers as in the best model with a higher latent space, i.e., number of internal features used to represent the shape of the

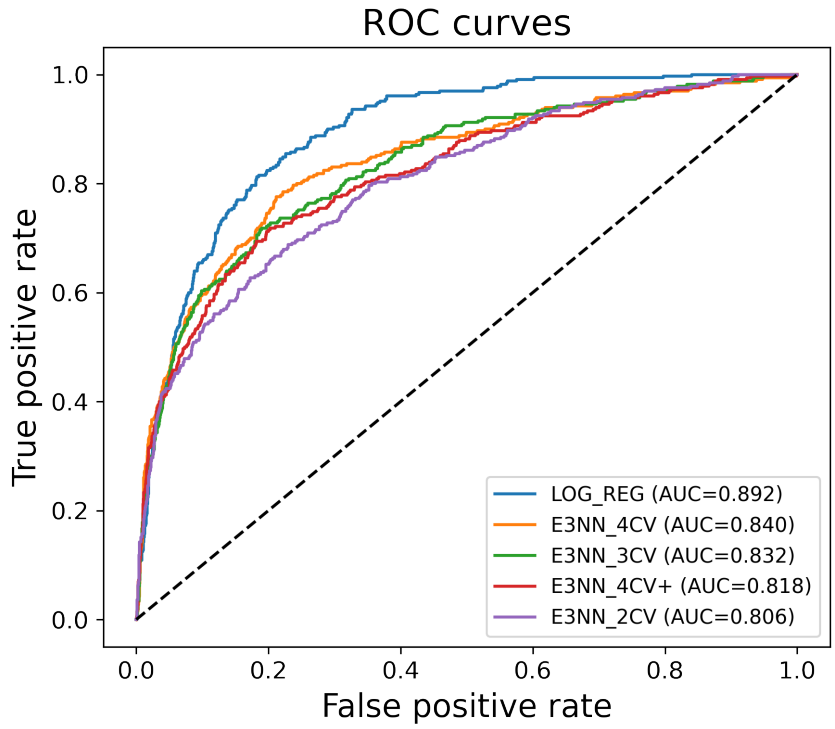


Figure 4.2: *ROC curves obtained by the 5 different models we trained on 2394 unseen cavities.*

cavity. Results were worse due to potential overfitting. In more detail, the latent space here starts with 256 features, whereas in the best model this value is set to 160. Hence, more tweaking needs to be conducted to reach further conclusions.

4.2 Trajectory prediction

One of the many applications this model can have is the possibility of identifying specific time intervals within the motion of proteins where binding sites may become receptive to drugs. Therefore, we examined how the 3MNP protein — retrieved from PDB [28] — behaved over 200 time frames, each consisting of 2.5 nanoseconds. To detect cavities in every step, we applied Fpocket with the same parameters as for generating the dataset. To track one pocket throughout the trajectory, we picked the top ranked cavity by Fpocket in the first frame, aligned all of the protein frames to the initial frame, and then calculated the centroids of all detected pockets over the trajectory, selecting the one whose distance from the previously chosen was the smallest. *File MD_prediction.py (Appendix B) contains the implementation.*

To compare the different shapes a single cavity conforms, we used the embeddings generated by the model right before they are passed to the final convolutional layer. Thus, in this example we worked with 200 cavities and the 88 geometric descriptors produced. As we wanted to visualize the latent space, we applied UMAP [29]

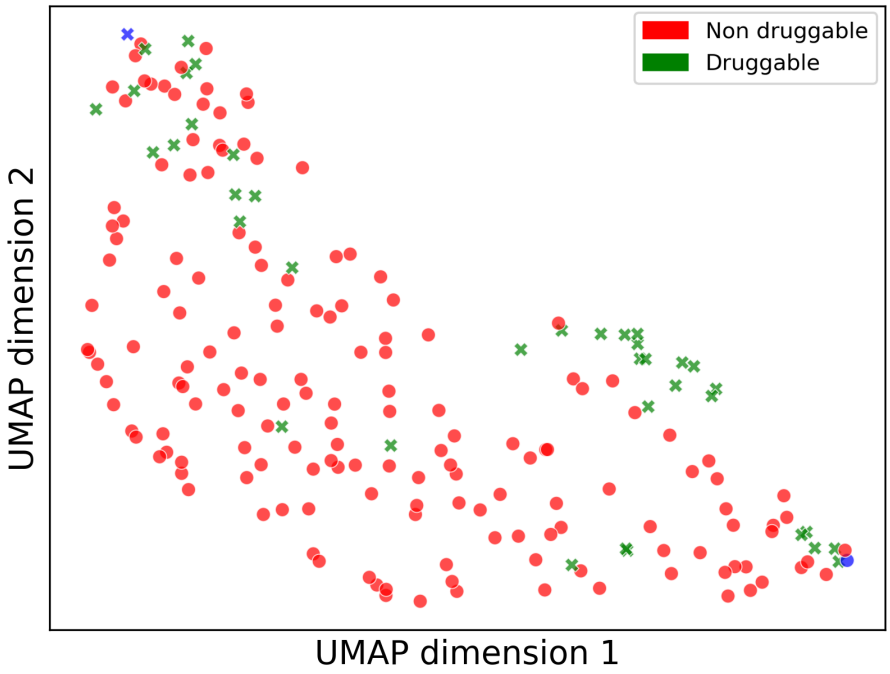


Figure 4.3: *2D view of the latent space after applying UMAP. Dots represent each shape embedded by e3nn at every time frame. Green crosses depict those which the model predicted as druggable, whereas the red circles represent the undruggable states. Blue color is for the shapes that — being each associated to a different type — differ the most from a geometrical standpoint.*

dimensionality reduction technique to it. The reasoning behind choosing UMAP over other techniques is based on the "increased speed and better preservation of the data's global structure" authors claim to achieve. We reduced the number of features to 2. The final result is shown in Figure 4.3. The plot shows that there are certain frames at which the binding site is druggable according to the model's assessment. However, cavity is most of the time in an undruggable state.

The blue points in the scatter plot represent the two most distant shapes, i.e., the shapes the model considers to be more geometrically distinct. For this example we picked one of each type. As shown in Figures 4.4 and 4.5, significant differences can be detected between the shapes selected for analysis. The druggable cavity appears more open, whereas the undruggable one seems to be the opposite. In addition to this, the druggable shape is considerably bigger in size. This is something that has been verified after calculating their volumes, which are 1727 and 533, respectively. Contrary to the belief that volume could be an indicator sufficient to describe the druggability state, figure A.1 shows that a larger volume does not always result in a druggable state of the pocket.

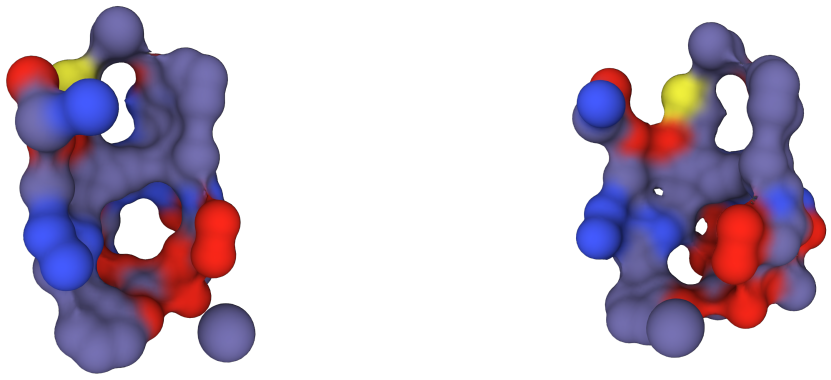


Figure 4.4: *Two views of the druggable shape in protein 3MNP at frame 50. Pocket has been detected and ranked as 0 by Fpocket.*

Figure 4.5: *Two views of the undruggable shape in protein 3MNP at frame 113. Pocket has been detected and ranked as 5 by Fpocket.*

5

Conclusions

In this work, we have proposed a deep learning-based approach for encoding the shape of protein cavities. Our solution allows the measurement of the similarity between binding sites through the calculation of the Euclidean distance between the vectors containing their e3nn-generated descriptors. By using an equivariant architecture, our model is robust to translations, rotations and inversion, allowing any protein pocket to be analyzed directly without dependence on absolute position or arbitrary number of points used to describe them. These embeddings were constrained to represent physically-relevant information through a supervised learning task, predicting druggable and undruggable pockets.

To accomplish this, a dataset of protein pockets was built from scratch through the design and implementation of a workflow where the cavity detection algorithm Fpocket was used to generate decoy cavities, while sc-PDB was used as a reference to assign labels to them, being druggable or not.

As a result, our supervised model was able to efficiently classify druggable and undruggable sites — reaching an AUC of 0.8404 — based on point cloud data only, i.e., no information related to the chemical nature of the amino acids near the pocket. We then used this model to inspect a cavity during frames of a molecular dynamic simulation, allowing us to explore how movements in the pocket can alter druggability.

Thus, a wide range of possibilities emerges where this tool can be considered. In this work, we showcased a scenario in which the trained model can detect potential druggable states on a single cavity along the protein's movement. Throughout the trajectory, we were able to not only visualize how the label flipped between the two classes but also detect small clusters where shapes with a similar geometry were sharing the same state.

Future work should address further applications. Considering the clustering we noticed on the shapes of the cavity during the protein's trajectory, a visualization of all the clusters that would result from a cavity dataset may reveal geometrical similarities between pockets from different proteins. This could provide additional knowledge about certain protein families.

A second interesting approach would be to encode the ligand structure in order to

find the most geometrically similar cavity. As a result of this, we can detect the real pocket of the protein, i.e., the space that the ligand occupies when the protein and the ligand are bound together. This can be very helpful, as there is not always a protein-ligand complex available for every protein.

With respect to the druggability prediction task, we believe that the use of geometric embeddings together with the chemical information provided by algorithms like Fpocket could result in a more robust solution. Although the shape plays an important role, other information — such as polarity or hidrophobicity — is required to unambiguously determine druggability.

Based on these findings, we are excited to see how this topic develops over the next years. With the hope that this work highlights the potential of the geometric embeddings generated by this tool and their applications within the molecular field, we expect to see more consolidated solutions that will help researchers in their daily tasks, contributing this way to a faster discovery and development of drugs.

All code generated in this project is made available on GitHub (`https://github.com/acorrochanon/Pocket-shapes`) which allows the results presented to be reproduced and extended.

Bibliography

[1] Sandhya Kortagere, Matthew Krasowski, and Sean Ekins. "The Importance of Discerning Shape in Molecular Pharmacology". In: *Trends in pharmacological sciences* 30 (Feb. 2009), pp. 138–47. DOI: 10.1016/j.tips.2008.12.001.

[2] Ryan Coleman and Kim Sharp. "Protein Pockets: Inventory, Shape, and Comparison". In: *Journal of chemical information and modeling* 50 (Mar. 2010), pp. 589–603. DOI: 10.1021/ci900397t.

[3] T Liu and Russ Altman. "Identifying Druggable Targets by Protein Microenvironments Matching: Application to Transcription Factors". In: *CPT: pharmacometrics systems pharmacology* 3 (Jan. 2014), e93. DOI: 10.1038/psp.2013.66.

[4] Peter Schmidtke and Xavier Barril. "Understanding and Predicting Druggability. A High-Throughput Method for Detection of Drug Binding Sites". In: *Journal of medicinal chemistry* 53 (Aug. 2010), pp. 5858–67. DOI: 10.1021/jm100574m.

[5] Vincent Le Guilloux, Peter Schmidtke, and Pierre Tuffery. "Fpocket: An open source platform for ligand pocket detection". In: *BMC Bioinformatics* 10 (Feb. 2009), p. 168. DOI: 10.1186/1471-2105-10-168.

[6] Noah Ollikainen, René Jong, and Tanja Kortemme. "Coupling Protein Side-Chain and Backbone Flexibility Improves the Re-design of Protein-Ligand Specificity". In: *PLoS Computational Biology* 11 (Sept. 2015), e1004335. DOI: 10.1371/journal.pcbi.1004335.

[7] José Jiménez-Luna et al. "DeepSite: Protein binding site predictor using 3D-convolutional neural networks". In: *Bioinformatics (Oxford, England)* 33 (May 2017). DOI: 10.1093/bioinformatics/btx350.

[8] Valerio Biscione and Jeffrey Bowers. "Learning translation invariance in CNNs". In: *arXiv preprint arXiv:2011.11757* (2020).

[9] Nathaniel Thomas et al. "Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds". In: (Feb. 2018).

[10] Mario Geiger et al. *Euclidean neural networks: e3nn*. Version 0.5.0. Apr. 2022. DOI: 10.5281/zenodo.6459381. URL: https://doi.org/10.5281/zenodo.6459381.

[11] Feng Gao et al. "Predicting chemical ecotoxicity by learning latent space chemical representations". In: *Environment International* 163 (2022), p. 107224. ISSN: 0160-4120. DOI: https://doi.org/10.1016/j.envint.2022.107224.

[12] John Capra et al. “Predicting Protein Ligand Binding Sites by Combining Evolutionary Sequence Conservation and 3D Structure”. In: *PLoS computational biology* 5 (Dec. 2009), e1000585. DOI: 10.1371/journal.pcbi.1000585.

[13] A. Zee. *Group Theory in a Nutshell for Physicists*. Princeton University Press, 2016.

[14] Mario Livio. “Physics: Why symmetry matters”. In: *Nature* 490 (Oct. 2012), pp. 472–3. DOI: 10.1038/490472a.

[15] Maurice Weiler et al. “3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data”. In: (July 2018).

[16] Tess Smidt. *Neural networks with Euclidean Symmetry for the Physical Sciences*. URL: https://tinyurl.com/e3nn-physics-meets-mlf. Nov. 2018.

[17] Piet Isacker. “Group Theory in a Nutshell for Physicists”. In: *Physics Today* 70 (Jan. 2017), pp. 58–58. DOI: 10.1063/PT.3.3430.

[18] Jérémy Desaphy et al. “Sc-PDB: A 3D-database of ligandable binding sites - 10 years on”. In: *Nucleic acids research* 43 (Oct. 2014). DOI: 10.1093/nar/gku928.

[19] Alexander Wlodawer et al. “Protein crystallography for aspiring crystallographers or how to avoid pitfalls and traps in macromolecular structure determination”. In: *The FEBS journal* 280 (Aug. 2013). DOI: 10.1111/febs.12495.

[20] Jérémy Desaphy et al. “Comparison and Druggability Prediction of Protein-Ligand Binding Sites from Pharmacophore-Annotated Cavity Shapes”. In: *Journal of chemical information and modeling* 52 (July 2012), pp. 2287–99. DOI: 10.1021/ci300184x.

[21] Philippe Le Mercier and Lydie Bougueleret. *The Universal Protein Resource (UniProt)*. Jan. 2007. DOI: 10.1093/nar/gkl929.

[22] Noel O’Boyle et al. “Open Babel: An Open Chemical Toolbox”. In: *Journal of cheminformatics* 3 (Oct. 2011), p. 33. DOI: 10.1186/1758-2946-3-33.

[23] Sebastian Raschka. “BioPandas: Working with molecular structures in pandas DataFrames”. In: *The Journal of Open Source Software* 2 (June 2017). DOI: 10.21105/joss.00279.

[24] Python Core Team. *Python: A dynamic, open source programming language*. Python version 3.9. Python Software Foundation. 2020. URL: https://www.python.org/.

[25] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[26] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[27] Fabian Pedregosa et al. “Scikit-Learn: Machine Learning in Python”. In: *J. Mach. Learn. Res.* 12.null (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[28] Helen Berman et al. “The Protein Data Bank”. In: *Nucleic acids research* 28 (Feb. 2000), pp. 235–42.

[29] Leland McInnes and John Healy. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: (Feb. 2018).

[30] G. Madhavi Sastry et al. “Protein and ligand preparation: parameters, protocols, and influence on virtual screening enrichments”. In: *Journal of Computer-Aided Molecular Design* 27.3 (Mar. 2013), pp. 221–234. ISSN: 1573-4951. DOI: 10.1007/s10822-013-9644-8. URL: https://doi.org/10.1007/s10822-013-9644-8.

[31] William L. Jorgensen et al. “Comparison of simple potential functions for simulating liquid water”. In: *The Journal of Chemical Physics* 79.2 (1983), pp. 926–935. DOI: 10.1063/1.445869. eprint: https://doi.org/10.1063/1.445869. URL: https://doi.org/10.1063/1.445869.

[32] Mark James Abraham et al. “GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers”. In: *SoftwareX* 1-2 (2015), pp. 19–25. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2015.06.001. URL: https://www.sciencedirect.com/science/article/pii/S2352711015000059.

[33] Yong Duan et al. “A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations”. In: *Journal of Computational Chemistry* 24.16 (2003), pp. 1999–2012. DOI: https://doi.org/10.1002/jcc.10349. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.10349. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.10349.

[34] H. J. C. Berendsen et al. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81.8 (1984), pp. 3684–3690. DOI: 10.1063/1.448118. eprint: https://doi.org/10.1063/1.448118. URL: https://doi.org/10.1063/1.448118.

A

Appendix 1

The 3NMP structure was obtained from the PDB and the ligand was removed. The system was prepared using the Protein Preparation Wizard [30] workflow implemented in Maestro to add missing hydrogens and assign likely tautomeric states at pH 7, solvated in a TIP3P [31] water box and neutralized with Na+ and/or Cl– ions. All simulations were performed using Gromacs [32] and the AMBER03 [33] force field and a timestep of 1 femtosecond. The procedure began with a minimization step, followed by 100 ps NVT and NPT equilibration steps to bring the system to 300k and 1 Bar pressure using a Berendsen thermostat and barostat [34]. The production trajectory was propagated for 50ns, with 200 evenly spaced frames written out over the trajectory.

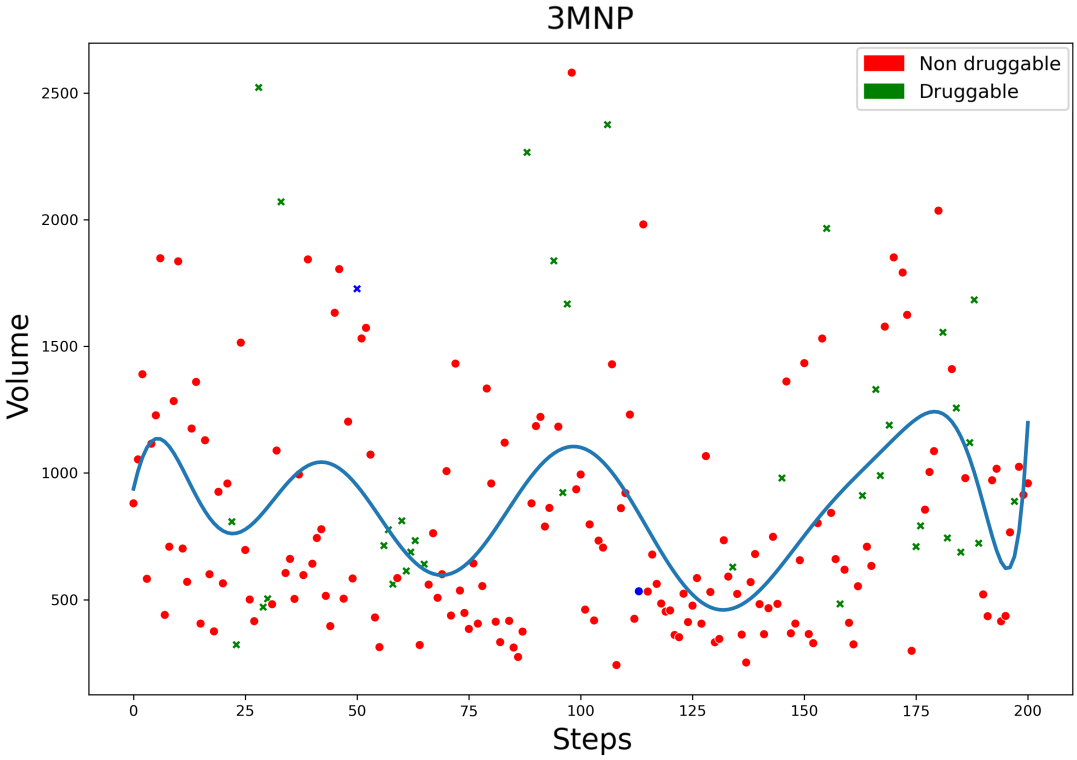


Figure A.1: *Volume of each of the shapes a single binding site conforms over 200 time frames. Volume has been calculated using the convex-hull.*

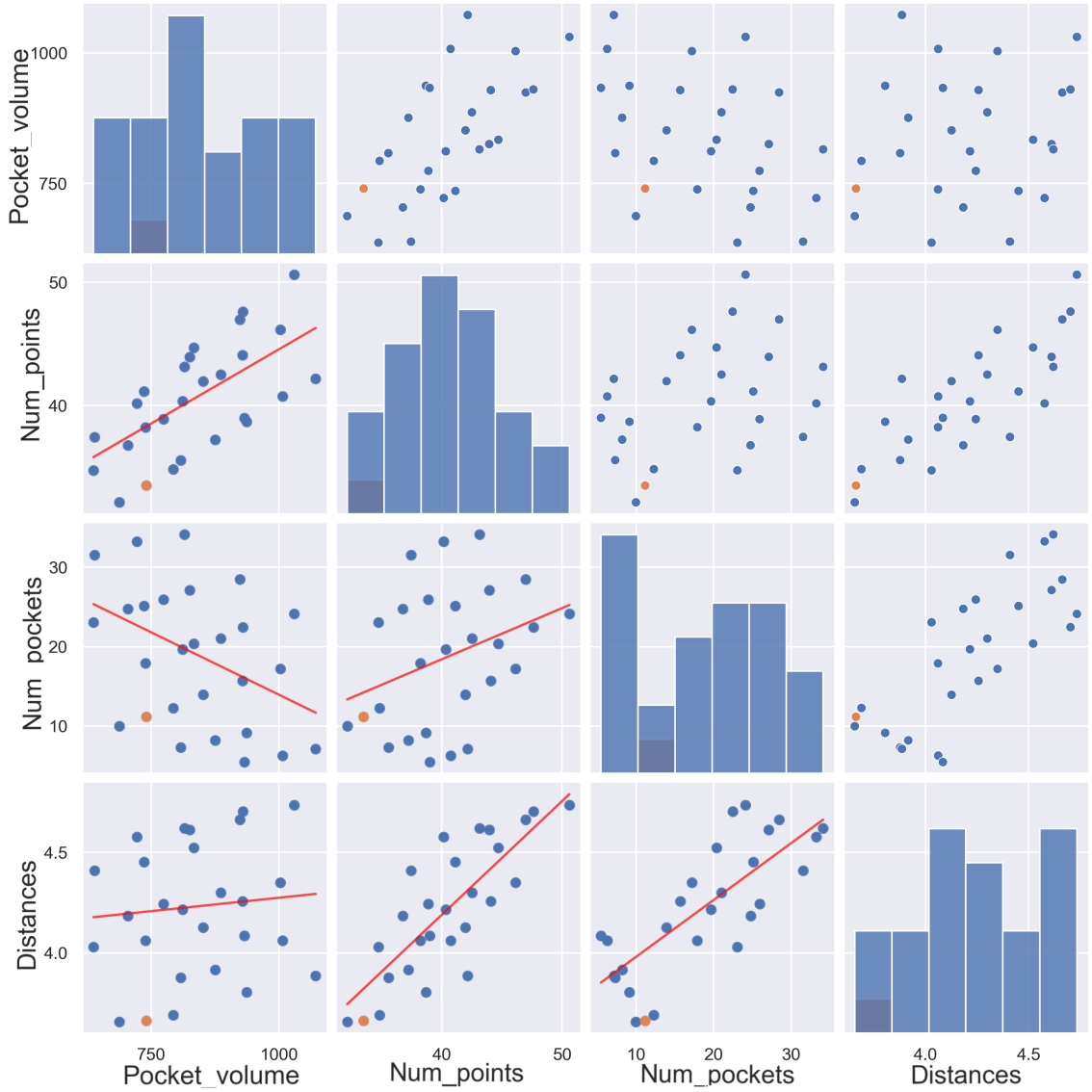


Figure A.2: *Fpocket's benchmark. Each dot represents a different configuration, where orange stands for the optimal parameters. This setting results in a small number of detected pockets per protein, pocket's alpha spheres, and low cavity volume.*

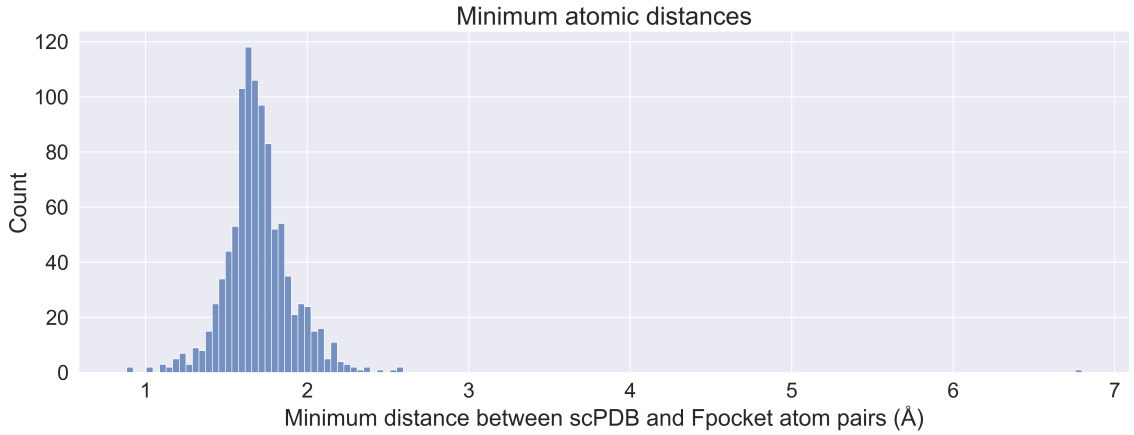


Figure A.3: *Distribution of the minimum atomic distances between the selected chain and its corresponding ligand for 1000 protein structures.*

B

Appendix 2

Figure B.1 illustrates the final structure of the repository. Here, we list some of the most important scripts and give a brief description:

- **MD_prediction.py**: Trajectory prediction. Here we execute the model against the snapshot of the protein taken at each frame. The cavity is tracked by applying Fpocket in every iteration and computing the distance against the ones obtained in the previous iteration. Protein snapshots are generated with a framework that was not developed in this work.

- **common_funcs.py**: In this file one can not only find internal methods such as the distance calculation between centroids, but also more explicit ones, like the data filtering.

- **unisplit.py**: Uniprot clustering. Here we cluster the protein structure ID's by their respective UniprotID. It returns two splits with the training and test clusters, which will be later considered for the final splitting process.

- **split_data.py**: Data splitting. Here the final set of cavities, labels, atom types, and features are split based on the resulting clusters obtained in the UniprotID clustering.

- **scpdb_funcs.py**: All the functions related to the processing of the data contained in the sc-PDB. The conversion from .mol2 to .pdb and the posterior cleaning of the pdb files depending on ligands chain location are here included.

- **log_classifier.py**: Logistic Regression classifier that uses scalar geometry-related variables obtained from Fpocket.

- **learning_shapes_xCV.py**: Training and testing of the networks occur in here. Each of these files differ in the number of convolutional layers or in the latent space size. This is explicitly pointed out in the name of file.

- **fpocket_funcs.py**: Functions that cover all the processing of the data obtained from the Fpocket algorithm. Here we extracted the coordinates of each of the detected cavities, scalar features associated with each of them, and created the data structures with which we will interact.

- **extract_data.py**: This file contains the whole data preprocessing pipeline, integrating functions from different files.
- **data_funcs.py**: Here is contained all the code required to combine the cavities and labels to construct the final dataset, create the batches according to the desired size, and some other additional internal functions.

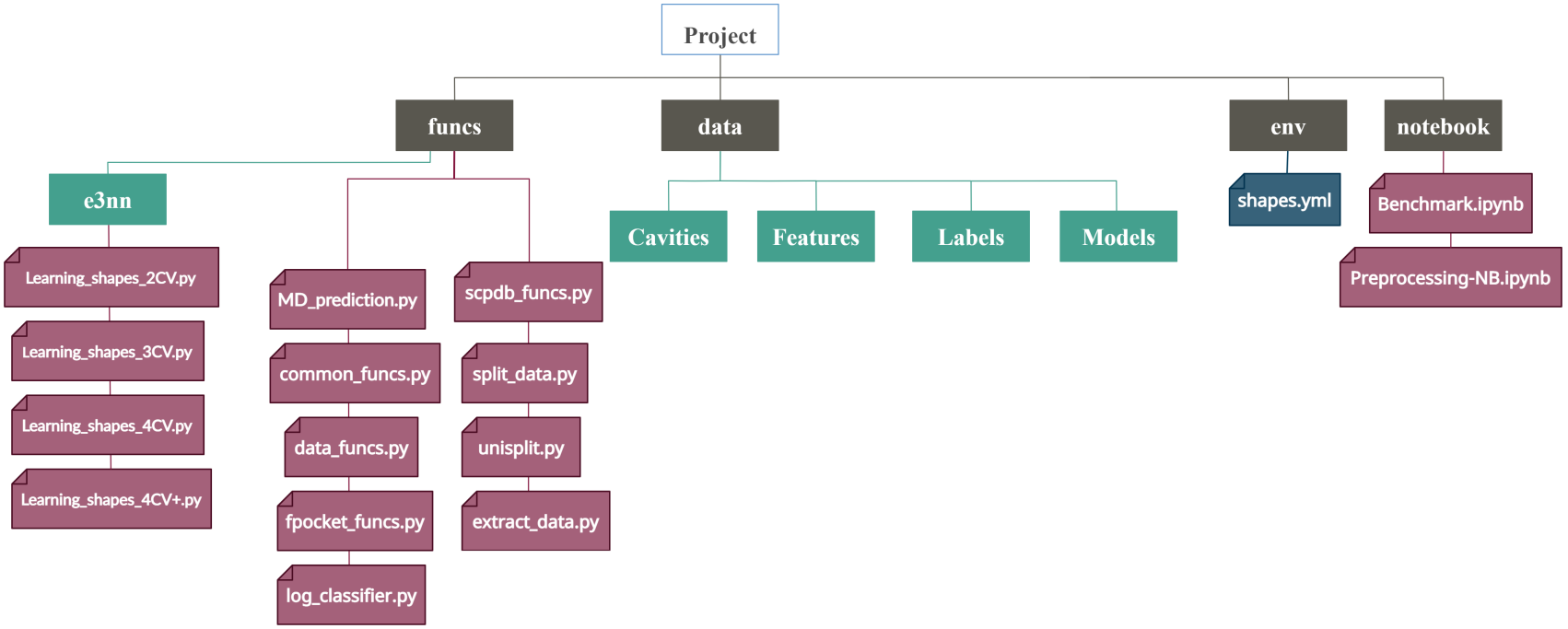


Figure B.1: *The source tree layout of the project, reflecting how files are organized.*