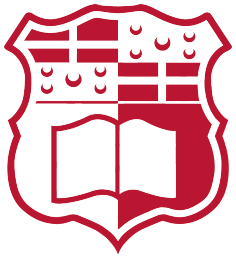

Applying Machine Learning to Ultrafast Shape Recognition in Ligand-Based Virtual Screening

Etienne Bonanno

Supervised by Dr Jean Paul Ebejer

Department of Artificial Intelligence

Faculty of ICT

University of Malta

September, 2019

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. Artificial Intelligence.

L-Università ta' Malta

FACULTY/INSTITUTE/CENTRE/SCHOOL I.C.T

DECLARATIONS BY POSTGRADUATE STUDENTS

Student's I.D. /Code 476476M

Student's Name & Surname Etienne Bonanno

Course M.Sc. (A.I.)

Title of Dissertation

Applying Machine Learning to Ultrafast Shape Recognition in Ligand Based Virtual Screening

(a) Authenticity of Dissertation

I hereby declare that I am the legitimate author of this Dissertation and that it is my original work.

No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.

I hold the University of Malta harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

(b) Research Code of Practice and Ethics Review Procedures

I declare that I have abided by the University's Research Ethics Review Procedures.

As a Master's student, as per Regulation 58 of the General Regulations for University Postgraduate Awards, I accept that should my dissertation be awarded a Grade A, it will be made publicly available on the University of Malta Institutional Repository.

Signature of Student

ETIENNE BONANNO
Name of Student (in Caps)

08/09/2019
Date

Statement of Originality

I, the undersigned, declare that this is my own work unless where otherwise acknowledged and referenced.

Candidate Etienne Bonanno

Signed _____

Date April 24, 2020

To Patricia

For putting up with my endless nights of study during our first years of marriage.

Acknowledgements

I wish to thank, first and foremost, my supervisor, Dr. Jean-Paul Ebejer, for his inimitable talent for imparting knowledge to his students, for sparking ideas for new avenues of research, for tirelessly supporting my endeavours throughout the time I spent studying and working on this dissertation and for the encouragement he gave me during the dark times when the cosmic joker seemed intent on working his mischief.

Deep gratitude and love also go to my wife Patricia, for putting up with endless study-nights, spells of dejectedness and moments of panic all during our first two years of marriage. I could not have done this without her support through the difficult times.

Endless love and gratitude go to my parents without whose personal sacrifices I would not have been the man I am today. To them I owe everything.

I would also like to thank all the excellent lecturers who taught me over these past two years, under whose tutelage I spent many enjoyable hours of learning, and the Department of Artificial Intelligence at the University of Malta that brought together such an amazing team and made available such a high quality course of studies.

My gratitude also goes to the Research Support Services Directorate at the University of Malta as well as Amazon Web Services through their AWS Educate initiative, for deeming my work worthy and funding my use of cloud services throughout the course of my work.

Abstract

Ultrafast Shape Recognition (USR), along with its derivatives, are Ligand-Based Virtual Screening (LBVS) methods that condense 3-dimensional information about molecular shape, as well as other properties, into a small set of numeric descriptors. These can be used to efficiently compute a measure of similarity between pairs of molecules using a simple inverse Manhattan Distance metric.

In this study we explore the use of suitable Machine Learning techniques that can be trained using USR descriptors, so as to improve the similarity detection of potential new leads. We use molecules from the Directory for Useful Decoys-Enhanced to construct machine learning models based on three different algorithms: Gaussian Mixture Models (GMMs), Isolation Forests and Artificial Neural Networks (ANNs). We train models based on full molecule conformer models, as well as the Lowest Energy Conformations (LECs) only. We also investigate the performance of our models when trained on smaller datasets so as to model virtual screening scenarios when only a small number of actives are known *a priori*.

Our results indicate significant performance gains over a state of the art USR-derived method, ElectroShape-5D (ES5D), with GMMs obtaining a mean performance up to 430% better than that of ES5D in terms of Enrichment Factor with a maximum improvement of up to 940%. Additionally, we demonstrate that our models are capable of maintaining their performance, in terms of enrichment factor, within 10% of the mean as the size of the training dataset is successively reduced. Furthermore, we also demonstrate that running times for retrospective screening using the machine learning models we selected are faster than standard USR, on average by a factor of 10, including the time required for training. Our results show that machine learning techniques can significantly improve the virtual screening performance and efficiency of the USR family of methods.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network.
AUC	Area Under Curve.
AWS	Amazon Web Services.
BST	Binary Search Tree.
CADD	Computer Aided Drug Discovery.
CPM	Conformers Per Molecule.
CSR	Chiral Shape Recognition.
DG	Distance Geometry.
DNA	Deoxyribose Nucleic Acid.
DUD	Directory of Useful Decoys.
DUD-E	Directory of Useful Decoys-Enhanced.
EDULISS	Edinburgh University Ligand Selection System.
EF	Enrichment Factor.
EMBL	European Molecular Biology Laboratory.
ES4D	ElectroShape-4D.
ES5D	ElectroShape-5D.
ETKDG	Experimental-Torsion Knowledge Distance Geometry.

GMM	Gaussian Mixture Model.
GPCR	G-Protein-Coupled Receptor.
HTS	High Throughput Screening.
LBVS	Ligand-Based Virtual Screening.
LDA	Linear Discriminant Analysis.
LEC	Lowest Energy Conformation.
ML	Machine Learning.
MSC	Molecular Shape Comparison.
PCA	Primary Component Analysis.
RBS	Recursive Binary Splitting.
RDD	Resilient Distributed Dataset.
RDF	Radial Distribution Function.
RMSD	Root-Mean-Square Deviation.
ROC	Receiver Operator Characteristic.
RSS	Residual Sum of Squares.
SAR	Structure-Activity Relationships.
SBVS	Structure-Based Virtual Screening.
SMILES	Simplified Molecular Input Line Entry Specification.
SOM	Self-Organising Map.
SVM	Support Vector Machine.
UFSRAT	Ultra Fast Shape Recognition with Atom Types.
USR	Ultrafast Shape Recognition.
USRCAT	Ultrafast Shape Recognition with CREDO Atom Types.
VolRAT	Volumetric Shape Recognition with Atom Types.
VS	Virtual Screening.

1

Introduction

Cheminformatics is a multidisciplinary field of study that applies techniques in statistics and computer science to the study of biochemistry, the term being defined in Brown et. al. (1998). One of the most active research areas of cheminformatics is that of discovering new drugs by computational means, a study referred to as Computer Aided Drug Discovery (CADD)

The discovery of a new drug is a long, time-consuming process that can take 14 years to complete successfully, incurring a cost of 800 million US dollars (Lavecchia and Giovanni, 2013). Virtual Screening (VS) is a systematic search approach that leverages electronic databases of chemical compounds and modern computing resources to streamline this process. The aim of this process is to computationally pre-screen compounds for those that are most likely to exhibit affinity for binding to a given target protein. In this way laboratory screening time can be drastically reduced, devoting laboratory resources to preferentially testing only the compounds that are more likely to be successful leads (Leach and Gillet, 2007). Advances in processing power and high-capacity storage as well as the development of Big-Data techniques has made this process of molecular screening feasible today, resulting in significant savings of time and cost and significantly streamlining the drug discovery cycle.

Virtual Screening can be divided into two main domains. Structure-Based Virtual Screening (SBVS) attempts to use the known shape of a drug-like small molecule to predict whether it will bind to a known *binding pocket* on the surface of a target protein. This process is known as ligand-protein *docking*. The 3D shape-alignment procedures that are required during docking are processing intensive and due to this SBVS approaches tend to be time and resource-consuming.

The second major branch of Virtual Screening is called LBVS. In LBVS, no reference is made to the properties or shape of the target protein. Instead LBVS is used when knowl-

edge of one or more compounds that bind to the target protein is already in hand. Using knowledge about the chemical, physical and spatial characteristics of these known biochemically active compounds, called *ligands* or *actives*, LBVS attempts to discover new compounds that fulfil some similarity criterion with respect to the known actives and thus have a high probability of binding to the protein and exhibiting similar drug-like behaviour.

In general, the LBVS problem is one of ranking a set of unknown compounds by the molecules' binding affinity to the target protein as determined by a similarity value calculated for each compound to the known active ligands according to some similarity function defined on some property or properties of the molecules.

Shape-based similarity searching or Molecular Shape Comparison (MSC) is a class of LBVS algorithms that attempts to use the physical shape of the atomic arrangement within the molecules as the property upon which the similarity between molecules is judged. Analogously to molecular docking, the shape comparison process between two molecules needs to find the optimal superposition of the two molecules in order to correctly compare their shapes and this is a computationally expensive procedure.

USR is a MSC technique, aiming to get around the heavy computational requirements entailed by molecule alignment. The technique was developed in 2007 (Ballester and Richards, 2007a,b) and it involves distilling the 3-D shape of a molecule into a descriptor vector made up of 12 decimals which is invariant under rotation. These descriptors are then compared directly using a modified Manhattan distance metric, obviating the need for calculating molecule superposition. This method was developed in 2007, and since then, extensions to this algorithm have been proposed that augment the purely shape-based descriptors of USR with other chemical properties of the molecule such as Electroshape 4-D that use atomic partial charges (Armstrong et al., 2010) and USRCAT, using atom types (Schreyer and Blundell, 2012), obtaining better virtual screening scores than the original USR algorithm. Electroshape 5-D is the USR variant that has obtained the best virtual screening scores to date in the literature. For this reason, we will standardise on this variant during the course of this dissertation.

1.1 | Motivation

Molecules are, in general, not rigid structures. Molecular bonds between the atoms making up a molecule can rotate, therefore any given molecule might be able to take an exceedingly large number of different shapes in 3-D space with only a small number the shape of which will match a protein binding site. An example of a rotatable atomic bond

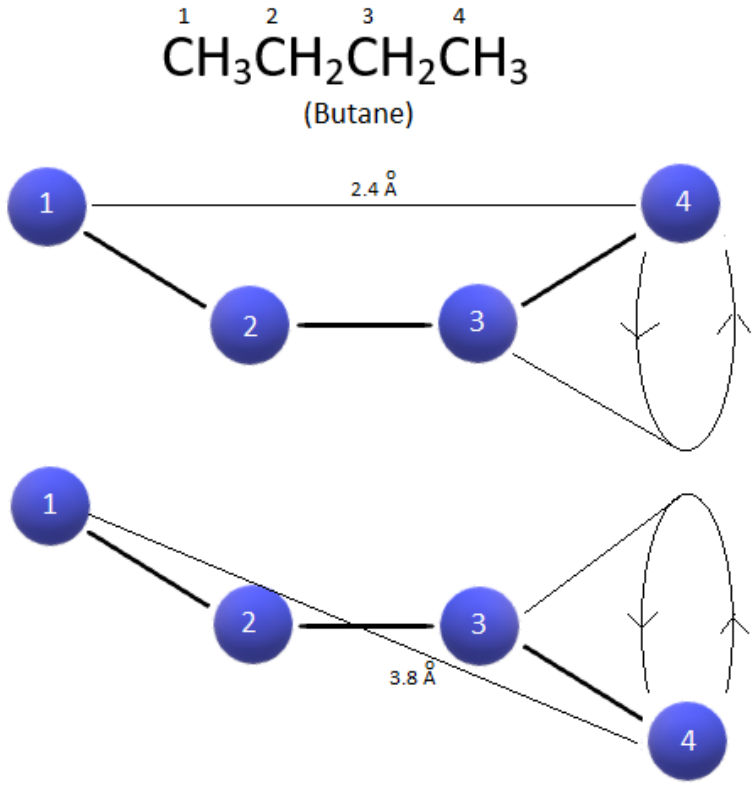

Figure 1.1: A rotatable bond determining molecular conformation

and the effect it can have on a molecule's shape can be seen in Figure 1.1. These different configurations of a molecule are referred to as *conformations*, *conformational isomers* or simply *conformers*.

Any method that makes use of molecular shape as a similarity criterion, including SBVS methods as well as USR, must take molecular conformation into account because only a small number of conformations out of thousands might happen to be shaped in such a way as to bind to a protein binding pocket.

Even though USR is considered to be an extremely fast method of molecular shape comparison, the necessity of taking thousands of molecular conformations into account means that Virtual Screening experiments against large compound databases can nevertheless take a significant amount of time and processing power to complete. In our work on this project we found that it took several hours to complete a retrospective USR run for a single protein based on our compound dataset when running on a 3-node cluster with a total of 24 cores.

Several LBVS techniques have already been augmented and improved by applying

machine learning concepts in previous research (Lavecchia and Giovanni, 2013), however machine learning has, to our knowledge, not yet been applied to USR and its related family of methods. Throughout the course of this research we will explore the effectiveness of different classes of machine learning algorithms when applied to USR descriptors in improving the virtual screening ligand ranking scores. Using a trained machine learning model for virtual screening runs would not only potentially result in better ranking performance with respect to standard USR, but would also potentially be orders of magnitude faster as the entire active dataset would, in general, not need to be processed for every compound under test.

A major objective of this research is also to determine if, by making use of machine learning, the size of the training data can be decreased with respect to that required by non-ML USR (henceforth known as Standard USR) while maintaining an adequate ligand ranking performance. In this way the storage requirements for training data would be significantly decreased, as would the training time for the machine learning model. Additionally, Ballester et al. (2009a) point out that it is possible to only use the LECs of the template actives as search queries while still preserving an acceptable performance of the algorithm and cutting down drastically on the amount of computation needed. We will explore this idea further in order to determine if this still holds when using machine learning algorithms, comparing the performance obtained by models trained on all the active conformers against that obtained by similar models trained only on LECs.

1.2 | Datasets for Virtual Screening

Datasets used to test virtual screening algorithms consist of a set of known active ligands for a given protein target, together with a set of *decoys*, i.e. compounds presumed not to bind to the target. Decoys are usually chosen to be sufficiently similar to the actives for a given target to make ligand detection a challenge for the virtual screening algorithm so as to ensure that the results are significant.

One of the most popular virtual screening datasets that has been used in the virtual screening literature in general, and in relation to USR in particular, is the Directory for Useful Decoys (DUD) (Huang et al., 2006). This free to use dataset has been compiled at the Shoichet Laboratory in the Department of Pharmaceutical Chemistry at the University of California, San Francisco and published online in 2006. This dataset provides actives and decoys for 40 target proteins with an average active-decoy ratio of 1:36 for each protein. The decoys in the dataset are selected specifically to have similar physical properties as the corresponding actives but different topology (i.e. arrangement of

atoms in 3D space). As such, the decoys are presumed not to bind to the target protein, however they have, in general, not been tested to this effect as physically testing every single decoy would be prohibitively expensive, therefore it is possible that a small portion of decoys would bind to the protein in reality.

By 2012, however, problems had been discovered in the DUD. Actives in the dataset were not diverse enough to ensure unbiased results from virtual screening algorithms. Decoy selection was also not optimal as significant imbalance existed between the net charges of actives and decoys with 42% of the actives having a net charge versus only 15% of the decoys.

These shortcomings of the DUD were addressed by the release in 2012 of a new and expanded version of the DUD, called the Directory of Useful Decoys-Enhanced (DUD-E) (Mysinger et al., 2012). DUD-E provides datasets for 102 target proteins with an active-decoy ratio of 1:50. To our knowledge, USR techniques have, to date, not been applied to the DUD-E. Evaluating, therefore, USR-like methods on the Directory of Useful Decoys-Enhanced (DUD-E) is, in itself, a valid contribution to the field of study.

1.3 | Machine Learning Aspects

The nature of similarity-based LBVS such as USR presents several challenges when it comes to the selection of suitable machine learning models. The datasets provided in the DUD-E consist of confirmed active molecules along with decoy molecules that are assumed, but not confirmed, inactive. The uncertainty in the negative training examples can be an issue for training supervised models. Additionally, DUD-E provides only decoys which are considered "challenging", i.e. which are close in size, weight and other atomic properties to the actives, while having a different topology, however, in reality the set of "inactive" molecules for any protein target is much larger in size and diversity, therefore training any supervised model on the provided decoys would not necessarily produce a robust performance when presented with unseen data that resides outside the parameters of the DUD-E decoy selection.

Despite these challenges, the problem can be tackled in several ways. Generative models are possible algorithms to use. Given a set of observations X and corresponding model parameters Y, generative models attempt to learn the joint distribution $P(X, Y)$ from which the classification for new data points can be inferred by deriving $P(Y|X)$. Generative models are normally trained separately on each label class (active and decoy in this case), learning a probabilistic distribution of points for each class. New test points

are then processed with the learned model for each class and are classified according to the class obtaining the highest probability - a process referred to as *Maximum a Posteriori estimation* (Ng and Jordan, 2002).

When conformers are generated from an active in the DUD-E, in general only several conformers will exhibit the required shape or shapes that can bind to the target protein. When all the active molecules are expanded into conformers and all of these are taken as a whole, clusters of conformers corresponding to the specific molecule shapes that can bind to the protein will be observed, as verified by Ballester et al. (2009a). In effect, every active molecule will supply, along with a large number of "random" non-binding conformers, some conformers that correspond to a small number of molecular shapes that can bind to the target protein. The superposition of all the active molecules will result in the "active" shapes being reinforced, resulting in detectable clusters. This can be leveraged by using outlier-detection models that can be trained on "noisy" data such as this and learn which training examples are "inliers" and which are "outliers".

The above two methods are both non-supervised methods that can be trained only on the actives in the dataset. Despite the reservations we outlined above, we wanted to also include a supervised method in our study, so as to be able to compare the relative performances of the different approaches. For this reason, we decided to use ANNs as our supervised method of choice. ANNs have seen extensive use in a wide variety of problems, including virtual screening, obtaining excellent results (Lavecchia, 2015).

Through the course of this project we explore representative algorithms belonging to each class of model outlined above and compare the performance obtained from each to the baseline USR performance as well as to each other. We will also train models using varying-size subsets of the DUD-E datasets so as to explore the effect of diminishing training set size on the similarity matching performance of our algorithms.

1.4 | Aims and Objectives

The aim of this study is to explore the possibilities offered by various machine learning techniques to augment the USR family of virtual screening methods in order to boost active ligand ranking as well as decrease the processing time for virtual screening essays.

The dissertation will deal with two main scientific research questions, namely:

1. Can machine-learning techniques be used instead of naïve Manhattan distance to improve Virtual Screening performance based on USR and USR-derived descriptors?

2. What is the minimal amount of data required to adequately train the machine learning model?

The application of machine learning algorithms to the problem of USR similarity matching constitutes a new approach to the problem of shape-based similarity searching in virtual screening and is therefore an avenue of research that could result in significant improvements to the field.

The second research question is important because it is always the case that the number of known active ligands for a particular target protein in a virtual screening assay is limited, therefore it is crucial to determine how the effectiveness of a machine learning model applied to a virtual screening dataset varies with severely unbalanced training data and with limited training examples.

1.4.1 | Objectives

The objectives to be reached in order to achieve the aims of this research are the following:

1. To create an implementation of USR/Electroshape 5-D to serve as the baseline against which to evaluate machine learning models (henceforth, standard USR).

2. To leverage the online chemical database known as the Directory of Useful Decoys – Enhanced (DUD-E) (Mysinger et al., 2012) which has been created especially for the purpose of evaluating VS methods.

3. To generate representative conformers for the molecules in the DUD-E database for use in the baseline USR virtual screening performance as well as for training of machine learning models.

4. To explore machine learning algorithms that can be trained only on positive examples, or which can deal with uncertainty in the negative examples.

5. To train various ML models, mapping out their performance against the baseline of standard USR with various protein targets in order to identify an optimum machine learning model

6. To run further experiments decreasing the number of known actives so as to understand how the performance of machine learning models trained on a reduced data set compares to standard USR

7. To explore the performance differences generated when training models using the full conformer model for every compound as opposed to using only the LECs.

1.5 | Approach

This project will be divided into three main tasks. First, before any experiments can be carried out, conformers have to be generated for the active and decoy molecules for the target proteins under consideration. This step is necessary because USR is a shape-based similarity matching technique and the various shapes a molecule can take have to be considered when performing the virtual screening process. In order to do this, we will make use of the freely available, open-source Cheminformatics package *RDKit*. Recommendations for doing this which we will follow are presented by Ebejer et al. (2012). Conformer generation is expected to generate several gigabytes of data and big-data techniques will be used to complete this task in a reasonable time.

Once conformer generation is complete an RDKit-based implementation of USR and its variants - CSR, ElectroShape 4D and ElectroShape 5D will be developed. These implementations will then be used to generate performance scores on the conformers generated previously. These scores will then be compared to those in the literature in order to understand how the use of DUD-E in lieu of DUD affects shape-based similarity search algorithms. In addition, the scores will also be used as benchmarks against which to compare the scores obtained through machine learning.

The next step will be to select suitable machine learning models as discussed in Section 1.3 and train them on the generated conformer data. The chosen algorithms will be Gaussion Mixture Models, Isolation Forests and Artificial Neural Networks. The performance scores obtained by these models will then be compared to those obtained by USR.

Further to this, additional experiments will be performed for each Machine Learning (ML) with varying fractions of the generated conformer data in order to explore how the performance of the ML models varies with smaller training sets. These experiments will be carried out for models trained with full conformers as well as with LECs in order to determine the significance of the performance hit when using LECs.

Due to the large volume of data created by the conformer generation step, the experiments will involve a significant element of big-data processing, requiring the use of distributed processing clusters in order to execute within a reasonable time-frame.

1.6 | Results

The results we obtained by the end of the project are significant, with average performance improvements on the order of 400% over standard ElectroShape 5D and maximum improvements on the order of 900%. Moreover, we show that the performance of the machine learning models is sustained as the size of the available training data is decreased. We therefore demonstrate that, while machine learning has proven to be of use in other virtual screening contexts, the USR family of techniques is also a promising target for the application of machine learning algorithms.

1.7 | Document Structure

This dissertation will be structured as follows.

The **Background and Literature Overview** chapter will present comprehensive background information regarding Cheminformatics, and virtual screening in general while delving in more detail into the USR family of techniques. It will present an overview of the existing literature regarding the topic and of the current state-of-the-art results.

The **Methodology** chapter will focus on the work done to achieve the aims and objectives. Here we will present a comprehensive overview of the machine learning models chosen, along with clear supporting arguments as to why such choices were made. We will also present detailed accounts of all the processes developed as well as the resources that were used during the implementation phase of the project.

In the **Evaluation and Results** chapter, we will present a detailed overview of all the experiments that were performed as well as the results obtained. In this chapter we will describe in detail the common evaluation methods used in the field and then apply them in order to formally evaluate the performance of our models.

In the **Discussion** chapter the results obtained will then be considered and contrasted within the context of the published work in the fields of virtual screening and USR. This chapter will also present an overview of how the work carried out during the course of this research project complements and improves the work previously carried out in the field.

Finally, in the **Conclusion** chapter we will revisit the fundamental aspects of the topic tackled by the dissertation, presenting an overview of our approach to the problem and of the results obtained, highlighting the contributions of this research to the field. Here we will outline the way in which the work carried out meets the goals set out in the aims and objectives and finally, will also propose future work that can be tackled in

order to further research in the field.

2

Background & Literature Overview

Cheminformatics, or chemoinformatics, is a relatively recent field of study concerned with applying results from other fields such as computer science, statistics and machine learning to problems in the field of Chemistry. The term *Chemoinformatics* was coined by Brown et. al. (1998) where it was descrbed as:

> *Chemoinformatics is the mixing of those information resources to transform data into information and information into knowledge for the intended purpose of making better decisions faster in the area of drug lead identification and optimization.*

A common task within the domain of cheminformatics is to process libraries of digital representations of chemical compounds in order to extract new information and results without the need of real-world testing in the lab.

Compound libraries used in cheminformatics can be collections of real molecules, however they can also consist of virtual molecules which are generated mathematically using various methods and by which the possible permutations of chemical space can be sampled and explored, possibly resulting in the discovery or invention of new, useful and patentable molecules. Compound libraries can therefore contain information about millions of chemical compounds, comprising terabytes of information and it is only with the recent increase in availability and affordability of cheap processing power and storage that cheminformatics became practical and cost-effective.

High Throughput Screening (HTS) is an automated, or semi-automated process, wherein a large number of chemical compounds are tested for some biological activity simultaneously, such compounds often being taken from pre-prepared, often commercially obtained, compound libraries. HTS however, apart from requiring specially equipped laboratories, is also expensive to carry out, both in financial terms as well as in terms of time.

The term *Virtual Screening* refers to a number of techniques in cheminformatics that are designed to pre-screen compound libraries via computational means, so as to identify likely *leads*, i.e. compounds that have a high probability of binding to a given target protein and therefore exhibit drug-like properties. The most likely leads can then be preferentially tested in HTS, resulting in a streamlined process which yields a higher number of hits with reduced time and cost.

2.1 | Virtual Screening

Virtual Screening techniques broadly fall into two main categories: Structure-Based Virtual Screening (SBVS) and Ligand-Based Virtual Screening (LBVS). Structure-based techniques focus on using structural information about target proteins to perform shape-based matching of molecules to known binding sites on the protein's surface, a process known as protein-ligand *docking* (Eckert and Bajorath, 2007; Lyne, 2002). Ligand-Based Virtual Screening (LBVS), on the other hand focuses on using compounds previously known to bind successfully to a given target protein as templates against which to compare other unknown compounds, the assumption being that similar molecules will bind to the same protein.

Virtual screening studies can be *retrospective* or *prospective*. A retrospective virtual screening study is performed using a dataset of compounds that has been pre-categorised into actives and decoys and is used to assess the efficacy of a particular virtual screening method or algorithm at identifying known actives. As such, a retrospective study is done completely in silico and is not expected to yield any new, previously unknown leads. A prospective study, on the other hand, is focused on using a virtual screening method on a database of test compounds in order to identify possible unknown leads which are then subsequently tested in the lab, hopefully identifying one or more active compounds.

2.1.1 | Structure-Based Virtual Screening

SBVS algorithms rank molecules by their calculated binding affinity which is approximated by evaluating some scoring function based on the shape of the molecules and the target protein. The theoretical basis underpinning SBVS is Emil Fischer's lock and key hypothesis, devised in the 19th century. Fischer postulates that a molecule will bind to a protein if it matches the shape of a binding site on the protein like a key matching a lock (Fischer, 1894) as illustrated in Figure 2.1.

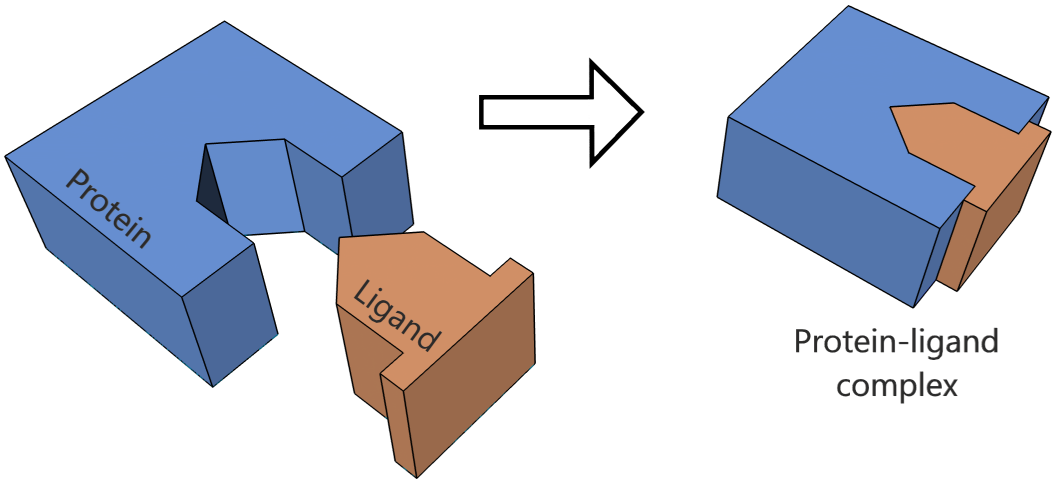

Figure 2.1: A simple illustration of Fischer's Lock-and-key principle.

SBVS is a computationally demanding process for several reasons. Molecular docking involves operations on 3-dimensional structures that are sensitive to rotation, scaling and translation. Such structures have to be docked onto the target protein subject to complex energy constraints in order to realistically simulate molecular binding. Additionally, molecules are, in general, not rigid structures. Molecular bonds between the atoms in a molecule can rotate, therefore any given molecule might be able to take an exceedingly large number of different shapes in 3-D space, only a small number of which will match the protein binding site. These different configurations of a molecule are referred to as *conformations*, *conformational isomers* or simply *conformers*.

Conformations for a given molecule are computationally generated by probabilistic sampling of the molecule's possible poses, taking into account all its rotatable bonds and minimising the resulting candidate conformers with respect to energy constraints imposed by the molecule's structure so as to approximate stable conformational shapes at the molecule's energy minima (see Section 3.2).

2.1.2 | Ligand-Based Virtual Screening

LBVS, in contrast to SBVS, assumes no prior knowledge about the properties or shape of the target protein. Instead, in LBVS, molecules already known to bind successfully to a given target protein are used as templates against which to compare other unknown molecules with the aim of finding those that are most similar to the template, and hence

likely to also bind to the protein. Such known template compounds are called *ligands* or *actives*. Compounds which exhibit the highest similarity scores to the actives will have a high probability of also binding to the same protein (Stahura and Bajorath, 2004).

The theoretical underpinning to this technique is referred to as the *similar property principle* (Johnson and Maggiora, 1990), which states that similar objects will often exhibit similar properties because changes in nature are usually gradual. When applied to LBVS the similar property principle means that molecules that are similar to a given active will probably have similar properties.

LBVS techniques can be subdivided into three main groups (Stahura and Bajorath, 2005).

- **Filtering** uses databases of known *functional groups*, i.e. molecular structures known to be related to a given molecular characteristic, in order to screen compound datasets for compounds likely to exhibit one or more desired characteristics.

- **Similarity Search** involves the extraction or calculation of molecular descriptors from various molecular properties, such as chemical composition, electrical characteristics or molecular structure. These molecular descriptors are then compared with those for the known active ligands and a similarity score inferred from the comparison. Thousands of different possible types of molecular descriptors have been devised, with applicability in differing scenarios. These techniques can be used when at least one ligand is known which binds to the target protein.

- **Compound Classification** techniques can be used when a number of related or unrelated actives are known to bind with a target protein. This allows the use of techniques such as clustering and partitioning to be performed in addition to similarity search so as to obtain better discrimination between active and non-active compounds in the unknown dataset.

A common approach in Similarity Searching is that of *Fragment-based* similarity searching, or *Molecular 2D fingerprinting* (Willett, 2006; Willett et al., 1998). This involves encoding molecular information, such as the presence or absence of given chemical fragments, structural features such as atomic rings, various ranges of molecular descriptors and other items of information into a fixed-size binary bit string, referred to as a molecular fingerprint. The similarity between two different molecules can then be estimated by comparing the respective bit strings using some standard similarity or distance met-

ric. The Tanimoto Similarity Coefficient is a popular metric that is often used, and is defined as:

$$T_s(X, Y) = \frac{\sum_i (X_i \wedge Y_i)}{\sum_i (X_i \vee Y_i)} \tag{2.1}$$

where X and Y are bitmaps and X_i is the ith bit of X. Other similarity indexes apart from Tanimoto are also possible, however (Stahura and Bajorath, 2004).

An alternative class of similarity searching methods is that of *Shape-Based Similarity Searching*, or *Molecular Shape Comparison*. Using this class of techniques, a similarity coefficient between pairs of molecules is calculated based on their shape in 3D space. In a similar manner to SBVS, these methods also leverage Fischer's lock and key hypothesis in order to posit that similarly shaped molecules are likely to bind to the same binding pocket in the target protein. Molecular shape comparison techniques are known to yield fewer matches than 2D fingerprint methods (Venkatraman et al., 2010), however the matches they identify tend to be different to those found by 2D fingerprints and are therefore, nevertheless useful (Finn and Morris, 2013). Additionally, in contrast to 2D fingerprinting, shape-based methods are capable of performing *scaffold hopping*, meaning that molecules having a similar shape to an active, but composed of different elements can still be identified as highly similar. This can result in the discovery of compounds that show similar biological activity to a known active, while circumventing issues such as patent restrictions (Ballester and Richards, 2007b).

Similarly to SBVS, and in contrast to 2D fingerprinting, Molecular Shape Comparison is also sensitive to molecular conformation and therefore the use of this class of techniques normally necessitates the generation of a large number of conformers in order to search through the possible shapes that a molecule might take.

MSC can be broadly divided into two classes of technique. *Superposition approaches* attempt to find the optimal superposition of the two molecules being compared through a process of three-dimensional rotation so as to achieve the best possible similarity score. An example of a superpositional approach is the ROCS method (Rush et al., 2005) which iteratively maximises volume overlap between two molecules by incremental rotation in order to quantify a similarity score between them.

Non-superposition approaches, on the other hand, avoid this process by using some similarity measure that does not depend on molecule superposition (Ballester and Richards, 2007b). For example, in the Shape Signatures method (Zauhar et al., 2003), a ray reflecting within the volume of the molecule is simulated. The length of each segment of the simulated ray between two reflections is collected into a probability distribution

histogram. The histograms thus derived for two molecules can be compared using a simple distance metric such as the L_1 norm, or Manhattan Distance:

$$L_1 = \Sigma_i (H_i^1 - H_i^2)$$

where H^1 and H^2 are vectors and H_i^J is the ith element of vector H^j.

In general, superposition approaches tend to yield better results than non superposition methods (Good and Richards, 1998), they are, however, much more processing-intensive, and hence slower, due to the requirement of finding the optimum alignment of the molecules being compared.

Ultrafast Shape Recognition (USR) is a non-superpositional technique developed in Ballester and Richards (2007a,b) that involves distilling molecular shape into a rotation-invariant descriptor vector made up of 12 real numbers. These descriptors are then compared directly using a modified Manhattan distance metric, obviating the need for molecule alignment.

The greatest advantage of this method is the exceedingly concise way in which the shape of a molecule is condensed into a small 12-element descriptor. Comparisons between such small descriptors are computationally inexpensive to perform and this feature of the method makes it orders of magnitude faster than any other shape-based similarity measure that existed at the time.

This method was developed in 2007, however, extensions to this algorithm have since been proposed that augment the purely shape-based descriptors of USR with other physico-chemical properties of the molecule, examples of which are Electroshape (Armstrong et al., 2010), and USRCAT (Schreyer and Blundell, 2012), which add atomic partial charges and atomic types to pure USR descriptors respectively, obtaining better virtual screening scores than the original USR algorithm.

2.2 | Ultrafast Shape Recognition Family of Methods

The USR technique was first described by Ballester and Richards (2007a,b) wherein the authors proposed a novel non-superpositional shape-based virtual screening technique meant to preserve the screening performance of superpositional techniques while obtaining the speed benefits of non-superpositional methods.

In their research, the authors point out that the shape of a molecule, or, more precisely, a conformer, can be encoded by taking the Euclidean distance of each atom to a predetermined number of centroids located within the space occupied by the molecule. The number and position of the centroids can be arbitrary, however, while pointing out

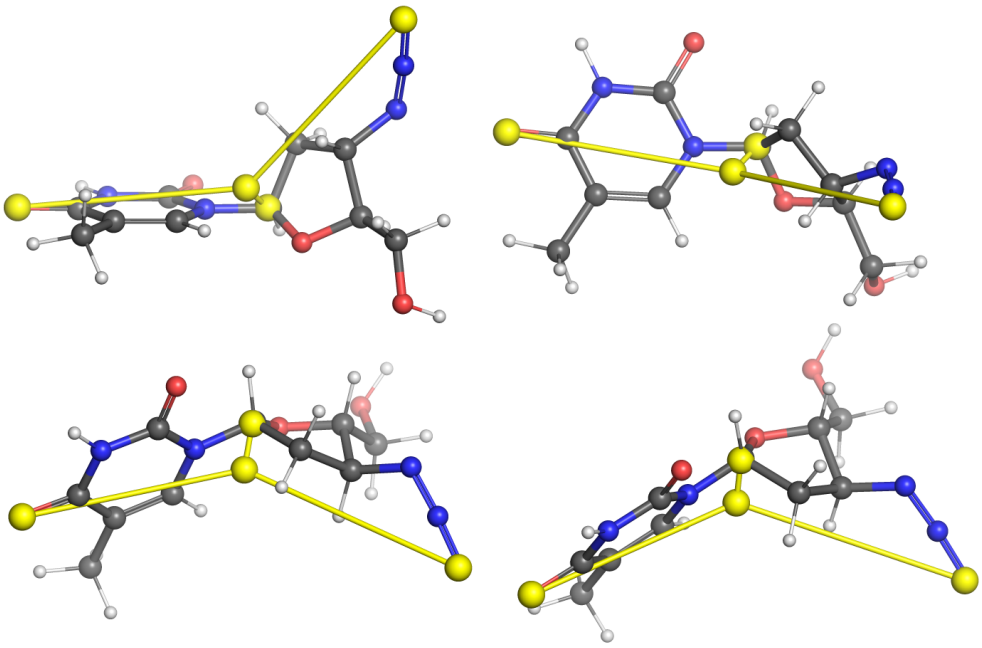

Figure 2.2: Illustration of USR centroids computed for a sample conformer of the Zidovoudine molecule. Centroids are indicated with yellow spheres. Lines between every centroid and the molecular centre are displayed for clarity. Four rotations of the molecule are illustrated.

that their selection had not been validated to be the optimal one, the authors chose four well-defined centroids as follows:

1. The molecular centroid (*ctd*)
2. The closest atom to the centroid (*cst*)
3. The furthest atom from the centroid (*fct*)
4. The furthest atom to *fct* (*ftf*)

Centroids computed for an example molecule are shown in Figure 2.2. Computing the Euclidean distances of all the atoms in the conformer to each of these four centroids yields four separate distance distributions of size proportional to the number of atoms making up the molecule.

As the authors indicate, however, for several reasons these distributions are not ideal to work with for the purposes of similarity searching. Most importantly, making use of these distributions as-is, it would not be possible to compare molecules having differing numbers of atoms because the distributions yielded by molecules of different sizes

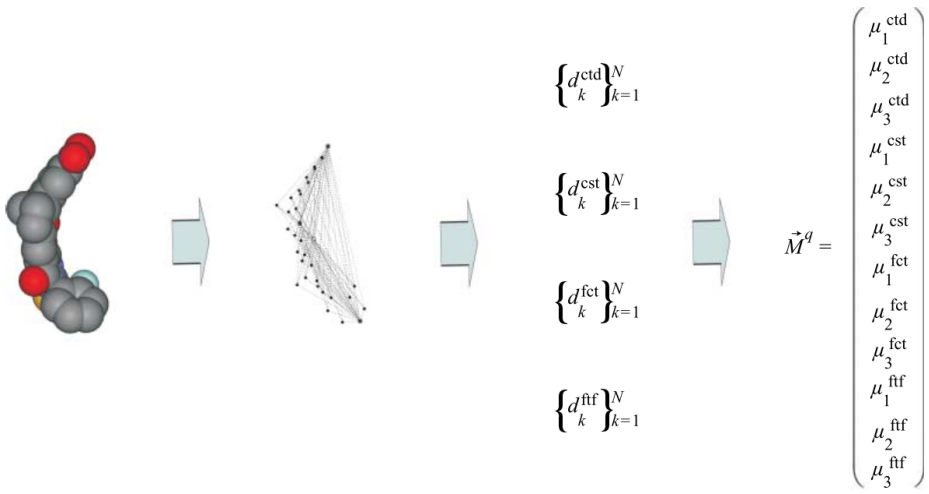

Figure 2.3: Illustration of the construction of shape descriptors using the Ultrafast Shape Recognition algorithm. The first three moments of the atomic distance distributions from 4 centroids are combined into a 12-element vector descriptor. Reproduced from Ballester and Richards (2007b)

would also be of different sizes. In addition to this, distributions are normally represented as histograms, however this would still leave open the question of finding an optimal bin size given distributions of wildly differing sizes and characteristics generated from a database of molecules, not to mention the storage volume and processing power required for their processing.

Ballester and Richards (2007a) solve these problems by pointing out that a distribution is completely determined by its statistical moments (Hall, 1983). Making use of this result from statistics, they condense the four distributions into their respective first three moments, corresponding to the mean, the variance and the skewness of the distribution. This results in a vector of 12 decimal values making up a descriptor encapsulating shape information for a given conformer. The authors propose using this vector as a stand-in for the molecule's 3D structure in similarity comparisons. This process is illustrated in Figure 2.3. Subsequently, Ballester et al. (2009a) modify this process by taking the square root and cube root of the second and third moments respectively, thus normalising them to a scale comparable to that of the first moment and resulting in better similarity matching performance.

The resulting descriptors could, in theory, be compared to each other using any similarity measure, however Ballester et. al chose to use a metric based on the Manhattan

distance according to Equation 2.2.

$$S_{qi} = (1 + \frac{1}{12} \sum_{l=1}^{12} |M_l^q - M_l^i|)^{-1} \tag{2.2}$$

where S_{qi} gives a similarity value between the query conformer q and the conformer being screened i and $\vec{M}^q$ and $\vec{M}^i$ are the descriptor vectors for the query conformer and the conformer being screened, respectively. Here the sum is normalised by dividing it by the number of elements in the USR descriptor.

In Ballester et al. (2009a) the USR method is evaluated and compared to *EShape3D*, an existing commercially available non-superpositional method and is found to offer, on average, significantly better ranking performance. In this paper, the authors perform their evaluation of the USR algorithm basing themselves on the *Enrichment Factor* (see Section 2.6).

In Ballester et al. (2009a), the authors point out that when a molecule binds to a protein, it is possible for profound conformational changes to take place in the molecule in order to allow it to bind into the binding pocket. Additionally to this, multiple shapes could successfully bind to the protein for several reasons – the protein binding pockets themselves could exhibit a degree of flexibility allowing them to accept a range of molecule shapes, resulting in different binding modes. It is also possible for a ligand to bind to a binding pocket without being completely surrounded by it, meaning that only part of the conformer shape would be relevant to the similarity matching. Furthermore, a molecule having the right shape to bind to the protein might not, in fact, do so due to unfavourable interactions between the atoms and the interface between itself and the protein's binding site.

Due to these complicating factors, the only definitive way of determining the 3-D shape of a conformer to use as a search template is to obtain it experimentally through X-Ray crystallography or Nuclear Magnetic Resonance while the molecule is actively bound to the protein. In absence of such experimentally determined conformers, assumptions have to be made that could impact the ranking performance of the algorithm.

While cognizant of the above problems, in cases where experimentally determined active bound conformations are not known, as an approximation to the bound active conformation the authors propose the use of the LEC - the conformation with the lowest energy - and therefore the most stable. As the authors point out, this introduces errors in the process, however their experiments indicate that using the LEC as the search template results in an acceptable ranking performance compared to the maximum possible performance.

In order to verify the method's efficacy in representing molecular shape, the authors take the LECs of all the actives for a given protein and perform hierarchical clustering on their USR descriptors. This reveals a number of similar clusters corresponding to the binding modes associated with the target protein, indicating that a) the USR descriptors can, in fact, be used as stand-ins for a molecule's shape and b) use of the LEC is a valid alternative to use in virtual screening.

Additionally, selecting the cluster holding the highest number of conformers effectively gives the most common active shape. The authors also propose using the closest conformer to the centre of the largest cluster as a query template, pointing out that on average it yields better enrichment factor than taking a conformer at random to use as a query template and it also gives an enrichment factor that is not much lower than the maximum possible enrichment factor for the target protein.

In the same year, Ballester et al. further validated their method by a prospective screening study in which they managed to identify novel inhibitors for arylamine N-acetyltransferases through USR (Ballester et al., 2009b). Using the identified leads, they purchased and tested a number of the predicted actives, obtaining a hit rate of 40%, considered a very good result for virtual screening.

2.2.1 | Extensions to USR

The first extension to the USR method was proposed in Cannon et al. (2008) where the authors combined USR descriptors with MACCS fingerprints, a 2-D fingerprint method which generates bitmask-like molecular descriptors with each bit encoding for some physico-chemical feature, such as the presence of rings, presence of less than 3 oxygen atoms, presence of halogen atoms etc. (Brown and Martin, 1996). The authors combine the two descriptors by simple concatenation and train a Random Forest multi-class classifier on the generated descriptors. The results of the study show that the combination of USR and MACCS descriptors on average outperforms the results obtained by either MACCS or USR alone. It is interesting to note that the authors, here, also experiment with extending the number of moments calculated when generating the USR descriptors, using three moments (as per the standard USR), four and five moments. Their findings indicate that the best performance occurred when using four moments, slightly outperforming standard USR descriptors with three moments.

An interesting extension to USR was proposed in Armstrong et al. (2009) wherein the authors propose a technique they name Chiral Shape Recognition (CSR). In their paper, the authors point out that the standard USR method does not have the possibility of distinguishing between conformers that are *enantiomers*. An enantiomer is one of a pair

of molecules that are mirror images of each other and therefore cannot be superimposed over each other. Enantiomers possess the property of *Chirality*, defined as the property of an object of being distinguishable from its mirror image. These types of objects are characterised by the absence of a plane of symmetry or a centre of symmetry.

The inability of USR to distinguish between chiral molecules means that the algorithm would classify enantiomers as similar to each other, however the fact that one enantiomer is an active does not imply that its mirror image is also active as its shape would not necessarily conform to the target protein's binding pocket. This introduces errors in the rankings produced by USR.

In order to remedy this problem, the authors propose a modification to the selection of USR centroids. They point out that the cross product is an operation that does not vary under translation and rotation but does under reflection. Formally, if $\vec{a}$ and $\vec{b}$ are vectors and ρ is any reflection, then

$$\rho(\vec{a}) \times \rho(\vec{b}) = -\rho(\vec{a} \times \vec{b})$$

In view of this property, the authors assign the first three centroids as per the original USR algorithm, as follows:

1. The molecular centroid ($C1$)
2. The furthest atom from the centroid ($C2$)
3. The furthest atom from $C2$ ($C3$)

The remaining centroid, $C4$, however is calculated differently. Two vectors are defined such that:

$$\vec{a} = C2 - C1$$

and

$$\vec{b} = C3 - C1$$

The normalised cross product $\vec{c}$ of vectors $\vec{a}$ and $\vec{b}$ is then given by:

$$\vec{c} = \left(\frac{||\vec{a}||}{2}\right) \frac{\vec{a} \times \vec{b}}{||\vec{a} \times \vec{b}||} \tag{2.3}$$

The fourth centroid is then defined as:

$$C4 = C1 + \vec{c}$$

This definition means that the first three centroids for mirror image enantiomers are all flipped around the centre, i.e. $C1' = -C1$, $C2' = -C2$ and $C3' = -C3$. However, due to the property of the cross product, in contrast to the first three centroids, the fourth centroid is the same in both enantiomers. This ensures that the algorithm successfully distinguishes between conformers that are mirror images of each other.

The evaluations performed by the authors on the CSR algorithm showed a maximum improvement of 20% at the 0.25% enrichment level which is a significant improvement in performance over USR.

Also in 2009 Steven R. Shave proposed three novel USR-based techniques in his D.Phil thesis (Shave, 2010) - Ultra Fast Shape Recognition with Atom Types (UFSRAT), Volumetric Shape Recognition with Atom Types (VolRAT) and UFSRGraph. In UFSRAT the author uses the same concepts as USR, i.e. taking the moments of distance distributions to four centroids, however, the method first partitions the atoms of the molecule into three types - Hydrophobic, Hydrogen-bond donors and Hydrogen-bond acceptor. The USR process is then performed on all the molecules, as per the standard USR, and again on each of these partitions separately, resulting in a 48-element descriptor incorporating information about the three atom types and their spatial distribution.

VolRAT is a variation on UFSRAT in which not only the atom types and distance distributions are encoded, but also the volume occupied by the atoms in the molecule. This is done by generating a grid of points centred around each molecule up to a definable radius and grid spacing. These points are then added to the appropriate atom type partition and used to generate the USR descriptors. The intention behind this idea is that using a blurred distribution in the similarity calculation might lead to molecules having a greater chemical diversity being picked out as similar to the query molecule and hence finding more leads and a greater scope for scaffold hopping.

UFSRGraph is another variation of UFSRAT which aims to get around the need for pre-generation of conformers before carrying out the virtual screening query. This is achieved by generating a graph representation of the molecule with vertices representing atoms and edges the bonds between the connected atoms. The edges are weighted according to an ideal bond length calculated using the Tripos force field. Three centroids are used rather than four as for the other algorithms. The first centroid is taken to be the atom closest to the middle of the longest path along the graph. The second centroid is defined as the furthest atom from the first centroid. The third centroid is the furthest atom from the second centroid. Distance distributions are compiled for each of the three centroids by taking the shortest path from each centroid to all the other atoms in the molecule and descriptors are calculated in the usual manner.

Shave does not perform retrospective screening studies using UFSRAT, VolRAT and

UFSRGraph for evaluation purposes and instead performs prospective screening for a small selection of target proteins using compounds from the Edinburgh University Ligand Selection System (EDULISS) dataset (Hsin et al., 2010). Due to this, standard performance metrics were not available for these methods.

Subsequently, the CSR method was extended by Armstrong et al. (2010) method, wherein the authors pointed out that the USR descriptors can be extended by quantities that are not spatial in nature, but instead pertain to physico-chemical properties of the atoms in the molecule themselves. In following with this idea, the authors extend their previous CSR descriptors by augmenting them with information about the *electrostatic complementarity* or *partial charge* of the atoms within the molecule, producing an algorithm they called ElectroShape.

The physical quantity of charge is the property of matter that causes it to experience a force when placed in an electromagnetic field. There are two types of charge, positive and negative, carried by protons and electrons respectively. According to the quantum theory of physics, the property of charge is quantized and can only occur as multiples of a fundamental unit of charge of approximately $1.602 \times 10e^{-19}$ coulombs, called the *elementary charge*, and written as e. This is also the smallest amount of charge that can exist in a free state.

When atoms form a chemical bond one or more electrons are shared between atoms, oscillating between them. While the bonded atoms as a whole always have a charge that is an integer multiple of the elementary charge, zones between and around the atoms themselves can be considered to have a fractional or partial charge. Atoms also have varying tendencies to draw shared electrons when bonded chemically to other atoms - a property called *electronegativity*. When a neutral atom bonds to another neutral atom that is more electronegative, electrons are drawn away from its nucleus towards the other atom's nucleus, causing it to have a positive partial charge and the other atom a negative one.

Armstrong et al. point out that electrostatic complementarity is an important feature when considering protein-ligand interactions which however, is completely ignored in purely shape-based similarity approaches. As a simple example, if one considers the two molecules carbon dioxide (CO_2) and hydrogen cyanide (HCN_3), as shown in Figure 2.4, their shape is highly similar and could easily be recognised as such by a purely shape-based similarity method such as USR, however their partial charge distribution is very different, and a method that incorporates this information would be more capable of distinguishing molecules in these kinds of situations.

The insight that allows the authors to incorporate partial charges into the USR scheme is the realisation that, similarly to the 3D coordinates used by USR, partial charge is sim-

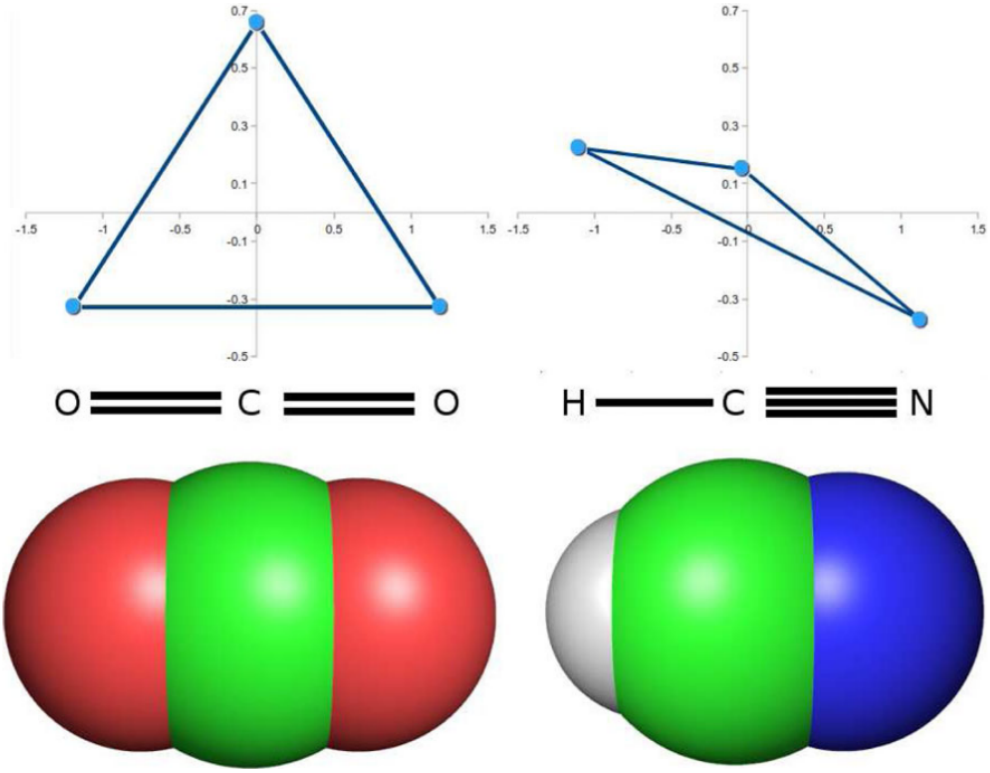

Figure 2.4: An illustration of the different partial charge distribution in two structurally similar molecules. Reproduced from Armstrong et al. (2010).

ply a number, and can be treated as a fourth coordinate along with the other three spatial coordinates of USR.

Consequently, in the ElectroShape algorithm, an atom is considered to have a coordinate (x, y, z, q) in four dimensional Euclidean space, with x, y and z being the spatial coordinates of the atom and q being the coordinate representing the partial charge. In general, however, there is a problem with this scheme in that the units of partial charge (electron charges) are different from the units of the spatial coordinates (Angstroms). Because of this disparity, the algorithm makes use of a scaling factor μ with units of Angstroms per electron charge in order to scale q accordingly. This gives atomic coordinates of the form $(x, y, z, \mu q)$

Naturally, selection of centroids is also slightly modified from plain USR. Five centroids are defined for ElectroShape. The first three centroids c_1-c_3 are defined in the same way as for CSR, except that this time the vector arithmetic is performed in four dimensions. In order to generate the fourth and fifth centroids, the equivalent vector to

$\vec{c}$ in equation 2.3 is first calculated and then used to generate the centroids as follows:

$$c_4 = (c_1)_s + \vec{c} + (0, 0, 0, \mu q_+)$$

$$c_4 = (c_1)_s + \vec{c} + (0, 0, 0, \mu q_-)$$

where $(c_1)_s$ is the spatial component of c_1 and q_+/q_- are, respectively, the highest and lowest partial charges in the entire molecule. Since, in this case five centroids are being used, as opposed to the four for USR, the ElectroShape descriptors consist of 15 elements rather than the 12 elements of USR.

For ElectroShape, Equation 2.2 is modified to give Equation 2.4

$$S_{qi} = (1 + \frac{1}{15}\sum_{l=1}^{12} |M_l^q - M_l^i|)^{-1} \tag{2.4}$$

where the sum is again normalised by dividing it by the number of elements in the descriptor, which is 15 in the case of ElectroShape. The authors point out that this method of extending USR descriptors is not restricted to partial charge and, on the contrary, is quite general, making it possible to extend the descriptors using any numeric quantity distributed among the atoms making up the molecule.

ElectroShape achieved significant performance improvements over USR and CSR, scoring an enrichment factor of 13.3 at 1%, compared to 7.4 and 7.7 for USR and CSR respectively.

Schreyer and Blundell (2012) proposed another method called Ultrafast Shape Recognition with CREDO Atom Types (USRCAT). This technique is based on UFSRAT, however in this case the similarity search algorithm was implemented as part of the CREDO Structural Interatomics Database (Schreyer and Blundell, 2009) and uses the atom type classification maintained therein rather than the atom classification in UFSRAT. The CREDO atom types used in this algorithm are hydrophobic, aromatic, hydrogen bond donor and hydrogen bond acceptor. Additionally, the authors point out that in UFSRAT, since centroids are selected separately for every atom type distribution, it is possible that there are not enough atoms of a given type to generate a distribution and/or to generate statistical moments. In USRCAT, this problem is solved by taking the same centroids for generating the distributions for all the atom types. Additionally, re-using the same centroids has been found to result in better performance because the distributions resulting from the scheme encode the different atom types with respect to the overall shape of the molecule. Using this scheme, USRCAT generates descriptors with 60 elements. USRCAT obtained a slightly higher average performance score than Elec-

troShape in retrospective screening on the DUD-E database with an $EF_{0.25\%}$ of 15.64 as opposed to 8.84 for USR and 14.48 for ElectroShape.

Armstrong et al. subsequently continued their work in 2011 (Armstrong et al., 2011) by proposing a further extension to the ElectroShape method which they termed ES5D. In this method they extended the descriptor proposed in their earlier paper with the quantity of *lipophilicity* as a fifth dimension. Lipophilicity is a measure of the solubility of a compound in fats and oils. The lipophilicty of a compound is expressed as a *Partition Coefficient P* - the ratio of the concentrations of a compound in a mixture of two immiscible solvents. In the case of lipophilicity, the solvents used are usually water and 1-octanol. This quantity is usually expressed as the logarithm of the partition coefficient $logP$.

Experimentally, $logP$ is measured by dissolving the compound under test in a mixture of water and 1-octanol and then measuring the concentrations of the compounds in each of the two solvents. The lipophilicity can, however, also be estimated computationally by considering the contribution to the final $logP$ value made by each atom in the molecule, such contributions being computed from a training set of compounds with experimentally measured $logP$. This approximation is referred to as *Atomic Lipophilicity* or $aLogP$.

In ElectroShape5D, the atomic contributions to the $aLogP$ are considered as a fifth dimension along with the fourth dimension provided by the partial charges in the original ElectroShape method.

The authors demonstrated that adding lipophilicity to the spatial components and the partial charge introduced in their 2009 work further improves similarity matching in benchmarking studies, obtaining an average $EF_{1\%}$ of 14.6 compared to 13.3 in ElectroShape. ElectroShape5D is currently the latest member of the USR family of methods and the one which obtains on average the best benchmarking scores.

In this section we have presented an overview of USR and the USR family of techniques, giving an account of how the USR technique works and how successive improvements have been applied to the basic technique to improve its predictive power. In the next section we will examine the techniques used in generating the conformers that are required by shape-based LBVS techniques, including USR.

2.3 | Molecular Conformations

As discussed in the previous sections, molecules are, in general, not rigid structures. The bonds between atoms can rotate, meaning that a molecule can potentially take a

large number of different shapes. The different shapes a molecule can take are referred to as molecular *conformers*.

Conformer generation is the process of using a computational representation of a molecule, together with prior knowledge about the forces between atoms in order to generate possible 3-D shapes that the molecule could conceivably take. These 3-D representations are then used directly in SBVS or shape-based LBVS for docking and similarity matching respectively.

As an example, some sample conformers of the Zidovudine molecule, an antiretroviral medication used to treat HIV/AIDS, are shown in Figure. 2.5. This molecule binds to the Human Immunodeficiency Virus type 1 Protease protein. As can be seen from the figure, and more clearly from the chemical structure of the same compound in Figure. 2.6, the molecule is made up of four groups connected by 3 rotatable bonds.

Appendix A contains further detailed background information about atomic forces within a molecule and about conformer generation.

2.4 | Molecular Representation

In the DUD-E datasets we used for this project, molecules were supplied in Simplified Molecular Input Line Entry Specification (SMILES) format. SMILES provides a manner of describing the chemical composition and structure of a molecule as a simple string. A SMILES string consists of a sequence of chemical symbols in the order in which they are bonded in the molecule. For example, the SMILES string for ethanol (C_2H_5OH, Figure. 2.7) is **CCO**. Equally correct would be **OCC** and **C(O)C**, where parenthesis indicate branching of the structure.

Ring structures in SMILES are indicated by splitting the ring at an arbitrary point and labelling the split with numbers indicating adjacent atoms. For example the Dioxane molecule ($C_4H_8O_2$, Figure 2.8) is represented by the SMILES string **O1CCOCC1**. *Aromatic* ring structures with alternating single and double bonds are common in organic chemistry. In order to indicate aromaticity in a SMILES string, the atoms inside the ring are written as lower-case. An example of this is Benzene (C_6H_6, Figure. 2.9).

A possible problem with SMILES is that, in general, there are multiple ways of encoding any one particular molecule. This is undesirable as detecting or searching for identical molecules can be a problem when different representations are being used. In order to mitigate this problem, several algorithms have been developed which, given a molecule, generate exclusively one SMILES representation, termed the *Canonical* SMILES. There is no one standard canonicalization algorithm, however, and every toolkit typi-

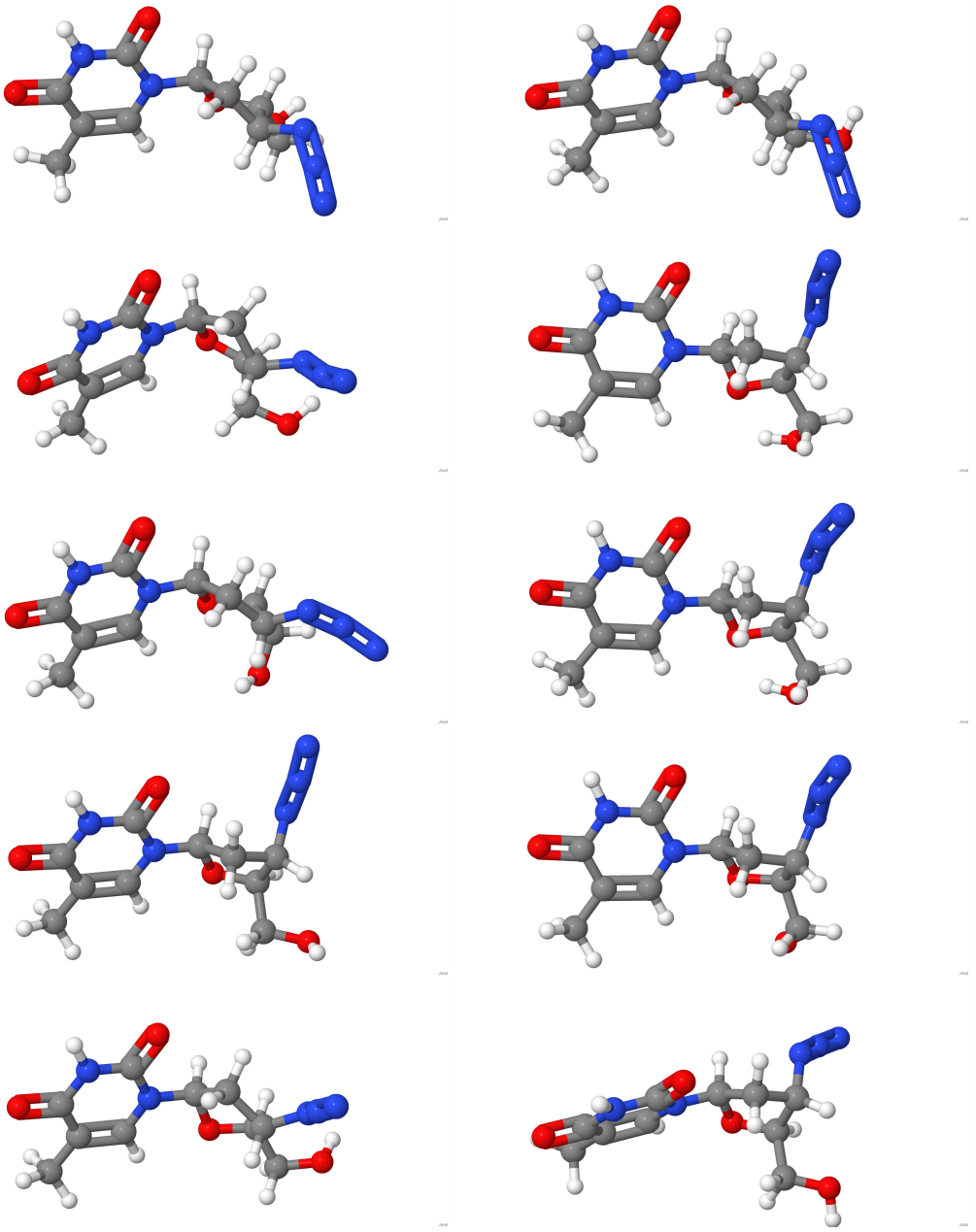

Figure 2.5: Sample conformers for compound CHEMBL129(Zidovudine), an active compound for Human Immunodeficiency Virus type 1 Protease. These renderings illustrate the manner in which a molecule can take different shapes due to rotatable bonds between selected atoms. In this case, this compound has 3 rotatable bonds. Carbon atoms shown in grey, Hydrogen in white, Oxygen in red and Nitrogen in blue.

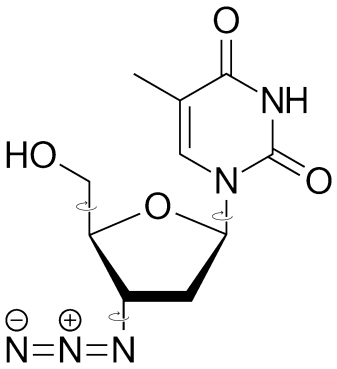

Figure 2.6: Chemical structure of Zidovudine.

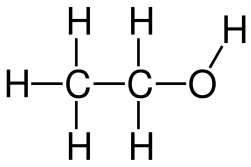

Figure 2.7: Chemical structure of Ethanol.

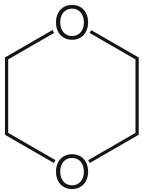

Figure 2.8: Chemical structure of Dioxane (Hydrogens ommitted as per convention).

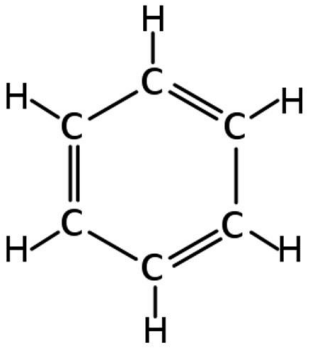

Figure 2.9: Chemical structure of Benzene.

cally uses its own. For the purposes of this research project, however, we did not need to compare SMILES strings, therefore this limitation did not affect us.

2.5 | Machine Learning Aspects

In line with the declared aims and objectives for this research project, we have explored the use of several machine learning techniques as alternatives to the similarity metric in the USR algorithm in an attempt to obtain better similarity ranking scores. In this section we present a detailed overview of the machine learning techniques we have deemed to be most suitable for this task. For every technique we describe, we present our motivation for choosing such a technique and how the technique is suited to this particular problem along with a detailed description of each technique.

2.5.1 | Gaussian Mixture Models

A Gaussian Mixture Model is a generative machine-learning model that models a distribution of data points using a combination of Gaussian distributions. It can be considered to be a clustering algorithm similar to k-means (Hartigan and Wong, 1979), however in a GMM, cluster membership of a data point is not absolute but instead can be influenced probabilistically by several centroids.

2.5.1.1 | The Gaussian Distribution

The GMM (Reynolds, 2015) is an unsupervised algorithm that models data points as mixtures of weighted Gaussian distributions.

The Gaussian Distribution (also known as the Normal Distribution) is a probability distribution governed by two parameters - the mean μ and the standard deviation $\sigma = \sqrt{variance}$ and is given by Equation 2.5

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{2.5}$$

Some example Gaussian curves can be seen in figure 2.10. As can be seen from this figure, the positioning of the curve along the x-axis is governed by the mean (μ) parameter, while the sharpness of the peak is influenced by the standard deviation (σ) parameter.

The single-variable Gaussian Equation 2.5 can be generalised to N dimensions by making x and μ into vectors of size N and the standard deviation into the NxN covariance matrix Σ. This is shown in Equation 2.6

$$\mathcal{N}(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big) \tag{2.6}$$

where T is the number of data points collected.

2.5.1.2 | Combining Gaussians

In general, a given distribution of data points will not necessarily be fitted satisfactorily by a single Gaussian, however we can attempt to approximate the arbitrary distribution by taking the weighted sum of a number of Gaussians as the probability of a point being a member of the distribution. This is termed a Gaussian Mixture Model and is described by Equation 2.7

$$f(x|\mu,\Sigma) = \sum_{k=1}^{M} c_k \frac{1}{\sqrt{2\pi|\Sigma_k|}} exp[(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)] \tag{2.7}$$

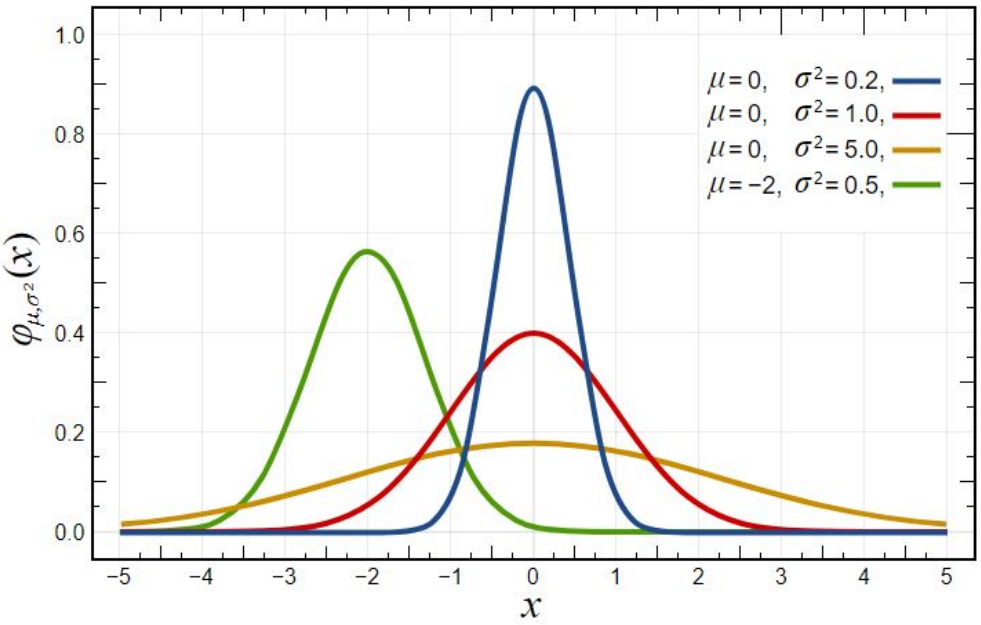

Figure 2.10: Some example Gaussian distributions with various means and standard deviations.

where M is the number of Gaussians, also known as *components*, making up the GMM. The parameters of the GMM are therefore:

- the number of Gaussian components making up the model(M)
- the Gaussian weights c_k such that $\sum_k c_k = 1$
- the set of mean vectors $\vec{\mu}$, one per gaussian
- the set of covariance matrices Σ, one per gaussian

A graphical representation of a simple 1-D GMM can be seen in Figure 2.11 and that for a 2-D GMM in Figure 2.12.

A GMM is trained using the **Expectation Maximization** algorithm (Dempster et al., 1977). This algorithm initially assigns tentative initial values to the model parameters. These could be random, but they are generally assigned using the k-means clustering algorithm on the data points. Once initial values are set, two steps, analogous to the k-means algorithm are repeated until convergence:

- **E-Step**: Evaluate the responsibilities of every component using the current parameter values (the latent variables):

$$\gamma(x) = p(k|x) = \frac{p(k)p(x|k)}{p(x)} = \frac{c_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{M} c_j N(x|\mu_j, \Sigma_j} \tag{2.8}$$

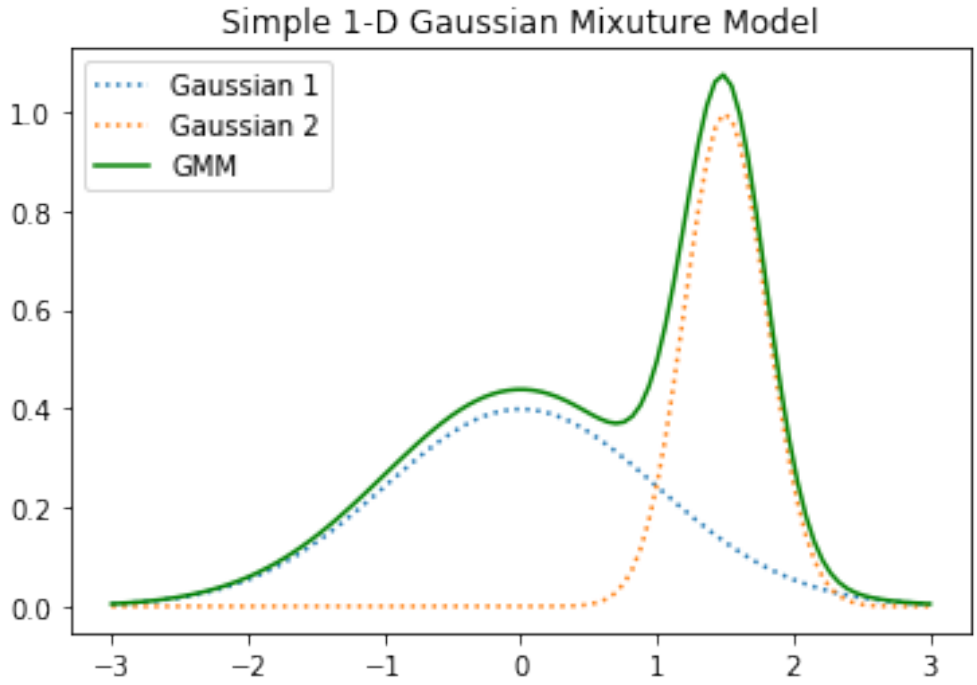

Figure 2.11: A plot of a simple example 1-D Gaussian Mixture Model with two components.

- **M-Step**: Re-estimate the parameters using the current responsibilities using the below equations to update the means, weights and covariance respectively:

$$\mu_j = \frac{\sum_{n=1}^{N} \lambda_j(x_n) x_n}{\sum_{n=1}^{N} \lambda_j(x_n)} \tag{2.9}$$

$$c_j = \frac{1}{N} \sum_{n=1}^{N} \lambda_j(x_n) \tag{2.10}$$

$$\Sigma_j = \frac{\frac{1}{N} \sum_{n=1}^{N} \lambda_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\frac{1}{N} \sum_{n=1}^{N} \lambda_j(x_n)} \tag{2.11}$$

By repeating the above steps until convergence, i.e. until the subsequent updates to the parameters are smaller than a preset threshold value, the optimal parameters for the GMM can be obtained.

The most important hyper-parameter that has to be tuned in a GMM is the number of Gaussian components to be used in the model. This can be expensive to determine, and is usually found empirically by a grid-search process.

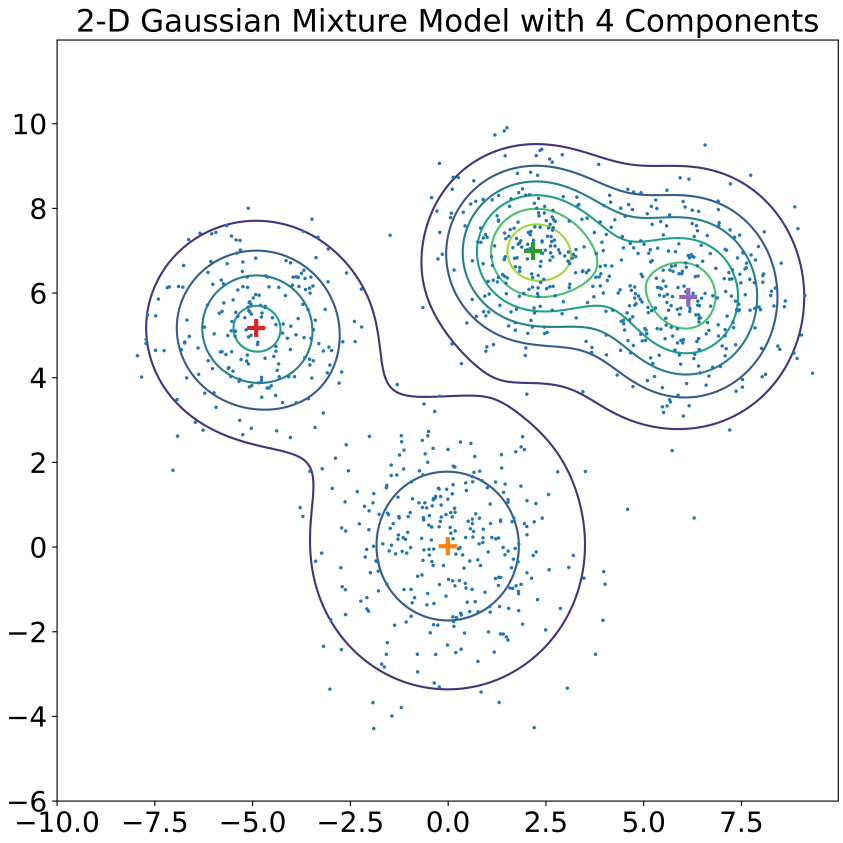

Figure 2.12: A contour plot of an example 2-D Gaussian Mixture Model with four components using dummy data.

The effectiveness of a GMM also depends on the constraints placed upon Σ, the covariance matrix. Putting no constraints on Σ maximises the GMM's expressive power, however also involves heavy computation. In most cases, Σ is constrained to be a diagonal matrix, resulting in lighter computational requirements while sacrificing some accuracy in the model. This can be compensated for, however, by using more components and for this reason, much of the results present in the literature assume diagonal Σ.

2.5.1.3 | Application in USR Similarity Matching

GMMs have wide-ranging applications in machine learning. They have been used in speech recognition (Stuttle, 2003), audio speech classification (Siegler et al., 1997), for language and speaker identification (Reynolds, 1995; Reynolds and Rose, 1995), as well as in visual object tracking (Santosh et al., 2013) and image enhancement applications (Celik and Tjahjadi, 2011).

They have also already been used in virtual screening and protein-ligand docking for example in (Grant and Pickup, 1995; Grant et al., 1996; Jahn et al., 2010, 2011).

The rationale behind the use of GMMs in USR similarity matching is the expected clustering of active conformers in USR descriptor-space, corresponding to the proteins' binding sites. In other words, if we plot the descriptors of all the active conformers for a given target protein we would observe dense clusters marking the preferred conformer shapes that bind successfully to the protein. This property of active conformer shapes has previously been pointed out by Ballester et al. (Ballester et al., 2009a). This is a type of clustering problem that is well suited for treatment using GMMs.

In this research project we trained GMMs for every protein target based on the active compounds provided in DUD-E. We then used the trained GMM to rank the other compounds by similarity. As a measure of similarity of a compound to the active template we used the per-sample average log likelihood of all the conformer USR descriptors for each compound being evaluated.

The *Likelihood* or *likelihood function* is defined as the probability that a set of statistical parameters fit a set of given data. In this case, the likelihood is returning the probability that the set of learned Gaussians in the GMM describe the given descriptor. The log of the likelihood is often used as a method of avoiding underflow errors in which probability values are too small to be represented adequately by the numerical precision of the processor. It also simplifies and speeds up the related calculations.

The per-sample average log likelihood is used to give an overall measure of the fit of the entire conformer ensemble for a given compound relative to the trained GMM of a template active compound. This scheme seems to work well (see Section 4), however it is not the only one that could be used. For example, one could take the maximum probability over the conformer ensemble. This, scheme, however was not evaluated and should be considered as a candidate for future research.

2.5.2 | Isolation Forest

The third machine learning algorithm that we have explored in the course of this project is that of *Isolation Forests* (Liu et al., 2008). Isolation forests are a class of machine learning models known as *ensemble* models. Ensemble models make use of a collection of simpler models to improve their predictions over those that would have been obtained by any single one model. Isolation Forests are similar to the Random Forest algorithm (Ho, 1995) in that they create a number of *Decision Trees* based on the training data and averages the predictions from each decision tree to arrive at a final result. While Random Forests are a supervised algorithm used to perform classification tasks, Isolation Forests are unsupervised and are meant to be used to perform *anomaly detection*. Decision Trees are at the foundation of several other machine learning algorithms, including Isolation Forest. Due to the importance of Decision Trees, the following section will give an overview of the algorithm as a foundation for our subsequent discussion of Isolation Forests.

2.5.2.1 | Decision Trees

A Decision Tree is a machine learning model that infers rules from a training set and makes predictions according to those rules and it can perform classification as well as regression predictions (Breiman, 2017; Myles et al., 2004). The advantage of decision trees is that predictions made using them are explainable by considering the path through the tree that was followed to arrive at the prediction. This is not the case when using many other supervised algorithms such as Neural Networks.

A decision tree is constructed by recursively partitioning the feature space of the training dataset, producing a set of recursive if/else rules which, at the leaf nodes, lead to a prediction.

In general the main challenge to constructing a decision tree is the procedure of choosing the feature to partition on at each recursive split within the algorithm. There are several different algorithms that can be used, different ones being needed in regression and classification cases. As the problem of USR ranking is a regression problem, we will detail the construction of a regression tree in this section.

The goal of constructing a decision tree is to partition the m-dimensional feature space into p non-overlapping portions. The prediction yielded by the decision tree will then be the mean of all the training observations within the partition under which the test observation falls. The partitions are constructed recursively using an algorithm called Recursive Binary Splitting (RBS).

At every recursive step RBS seeks to find the split that minimises the Residual Sum of Squares (RSS). The RSS is the square sum of the deviations of the predictions of an estimation model and the actual data and in the case of a decision tree is can be written as:

$$RSS = \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})$$

where M is the number of non-overlapping partitions in feature-space, R_m is partition m, $i \in R_m$ are the test observations is partition R_m, y_i is the prediction for test observation i and $\hat{y}_{R_m}$ is the average regression value for the training observations in partition m.

The problem of partitioning the feature space into M partitions in this way is extremely computationally intensive (NP-complete). The RBS algorithm mitigates this problem by splitting the feature space recursively in a greedy fashion, i.e. it only considers the minimisation of the RSS for each split and not for the feature space as a whole. This makes it much more computationally feasible.

At each recursive step, the algorithm randomly permutes the features, evaluating a number of potential splits and selects the one minimising the RSS at that point. The algorithm is then applied to each of the two children nodes thus created and continues recursively until a stopping condition is fulfilled, normally when a minimum RSS is reached, a maximum tree depth or a set minimum number of training samples remain within a leaf node.

2.5.2.2 | Ensemble Learning - Combining Trees into Forests

A single Decision Tree, being random in the way it generates partition splits, may not necessarily give the best prediction performance on a given set of test observations. Indeed, different decision trees trained on the same training data might yield different predictions.

In order to mitigate this problem and improve the predictive performance of the model, *ensemble* models have been proposed, which train several different decision trees and arrive at a final prediction by combining all their outputs into a single, more accurate, prediction.

One such ensemble algorithm is the Isolation Forest(Liu et al., 2008). Isolation Forests are used to identify anomalies in a set of observations. Contrary to other *clustering* algorithms which attempt to identify similar samples within the input dataset, Isolation Forests explicitly identify anomalies within the data.

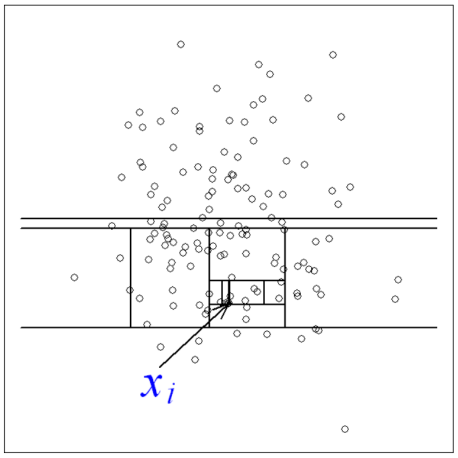

Figure 2.13: Isolating inlier point x_i requires 12 partitions Reproduced from Liu et al. (2008)

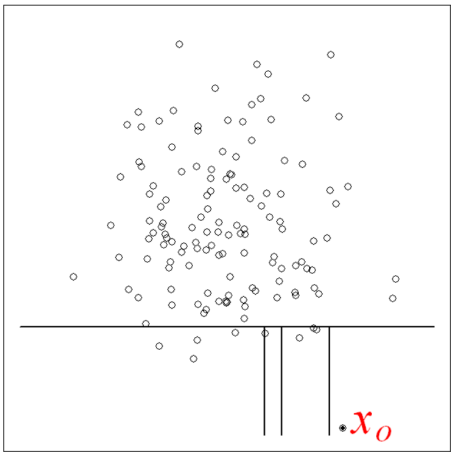

Figure 2.14: Isolating outlier point x_o requires only 4 partitions Reproduced from Liu et al. (2008)

It does so by exploiting the fact that, averaged over a number of Decision Trees, the path length that will be needed to generate a prediction for an outlier will be, on average, significantly shorter than that required for an inlier observation. An example of this is shown in Figures 2.13 and 2.14.

In practice an anomaly score is calculated from the average path length. Since a decision tree can also be considered a Binary Search Tree (BST), we can make use of the result that the average path length of an unsuccessful search in a set of n BST instances is given by:

$$c(n) = 2H(n-1) - (2(n-1)/n)$$

where $H(i)$ is the Harmonic number, estimated by $ln(i) + 0.5772156649$. Now, if $h(x)$ is the path length of a point x, $c(n)$ is the average of $h(x)$ and can be used to normalise $h(x)$. The anomaly score of a point x is therefore defined as:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $E(h(x))$ is the mean $h(x)$ from the collection of decision trees. Using this measure as an anomaly score ensures that if s is close to 1, then the point is almost certainly an anomaly and if $s < 0.5$, then the point is safe to be regarded as a normal instance.

2.5.2.3 | Application in USR Similarity Matching

Notwithstanding its relatively recent publication, the Isolation Forest algorithm has already been used as an anomaly detection method in a variety of applications. It has been used to detect anomalies in streaming data (Ding and Fei, 2013), detecting faults in semiconductor manufacturing processes (Susto et al., 2017), as an aid in identifying and mapping targets for mineral prospecting (Chen and Wu, 2019) and predicting faults in software code changes (He et al., 2017), among other applications. To our knowledge, however, this is the first time the algorithm has been applied to the problem of virtual screening.

The rationale for using Isolation Forests as an algorithm for ranking USR descriptors is by extension from Ballester et al. (2009a) wherein the authors showed that clustering the conformers of the active molecules for a given protein, several cluster centroids emerge, corresponding to the shapes matching the binding modes of the target protein.

By definition, a large number of actives will fall on, or close to a given centroid, forming high-density zones around the centroids. Non-binding conformers will fall outside these high-density zones, making them into outliers or anomalies. Training an Isolation Forest using the descriptors for the active compounds and ranking test points by their anomaly score should yield results with good predictive power.

2.5.3 | Artificial Neural Networks

Artificial Neural Networks are machine learning models loosely inspired by the structure of the brain. Brain tissue is made out of cells called *neurons* whose job is to transmit electrical signals (Kandel et al., 2000). A neuron is connected to other neurons through a network of dendrites - tree-like branching extensions of the cell. The neuron receives input signals from other neurons through its dendrites. Depending on the inputs it receives, the neuron generates an output signal along a long filament called the *axon*. The axon terminates at a network of synapses which connect to the dendrites of other neurons by secreting neurotransmitter molecules which bind to receptors in the dendrites, generating, in turn, another input signal.

The brain consists of a mesh of billions of neurons allowing signals flowing between them with information processing being performed in a massively parallel fashion. Given some sensory input, the brain is capable of modifying the functioning of the neurons to emit the correct outputs in order to synthesise appropriate behaviour/effects as the signals propagate through the network of neurons. This is, in essence, the process known as learning.

It is to be emphasised, however, that while being inspired by the structure of the brain, ANNs do not attempt to model all its biological complexities, and is not meant to mimic the brain in any meaningful way.

The initial algorithm that was inspired by the cellular structure of the brain, was described by Rosenblatt (1957) and was called the *Perceptron*.

A perceptron is a binary classifier, i.e. function taking multiple real-valued inputs and outputting a binary result and it is defined as follows:

$$f(x) = \begin{cases} 1, & \text{if } b + \sum_{i=1}^{m} w_i x_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

where x_i are the set of inputs, w_i are a set of associated weights and b is a bias value. These are usually represented in vector form as $\vec{x}$ and $\vec{w}$ so that equation 2.12 can be rewritten as:

$$f(x) = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

In effect, the perceptron computes a weighted sum of its inputs and outputs a binary value based on a decision boundary set by the perceptron's bias and an inbuilt *activation function*. It is easy to see how this algorithm approximates the function of a biological neuron.

The algorithm to train a perceptron is simple:

1. Define a training rate r. This value will regulate the speed with which parameters will be adjusted at every training iteration. If r is too small, training will be very slow, but if r is too large, the system might never converge to a solution.

2. Define a maximum number of iterations to perform during training.

3. Initialise weights and biases to 0 or random values

4. For every training example $(\vec{X}_i, Y_i)$, obtain prediction

   $$P_i(t) = f[\vec{w}(t) \cdot \vec{X}_i]$$

   where $P_i(t)$ is the prediction and $\vec{w}(t)$ is the weights vector at time t.

5. Update the weights to improve the prediction:

   $$\vec{w}(t+1) = \vec{w}(t) + r(Y_i - P_i(t))\vec{X}_i$$

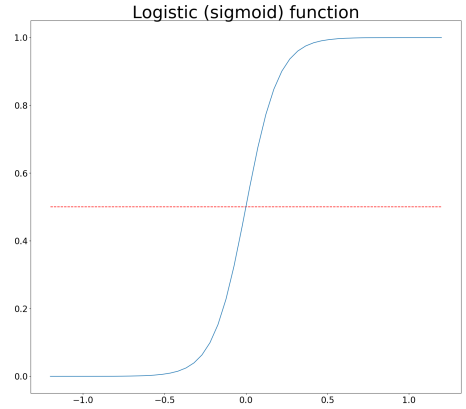

Figure 2.15: Example plot of the Sigmoid Logistic activation function

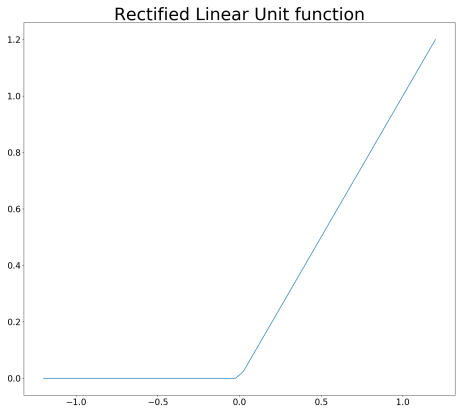

Figure 2.16: Plot of the Rectified Linear Unit function

6. Repeat above steps until convergence or maximum iterations are reached.

The activation function shown in Eqaution 2.12 is known as the *Heaviside Step Function* and is the simplest activation function possible, however others are possible. In particular the *logistic sigmoid* function is often used:

$$h_W(x) = \frac{1}{1 + e^{-W^T x}}$$

where W is a column vector of weights and x is the column vector of input values. A sample plot of the sigmoid function is shown in Figure 2.15.

Another activation function in common use is the *Rectified Linear Unit* or *RelU*. It is defined as $max(0, x)$ and is shown in Figure 2.16.

Initial interest in perceptron research dwindled when it was shown that they were limited in the functions they could represent. The perceptron would only be able to classify linearly separable patterns. For example, a single perceptron can simulate AND and OR operations but not XOR (Mitchell, 1997). These limitations led to a decline in the initial interest in neural network research, however interest resurged with increases in processing power and the realisation that arranging perceptrons in layers with the output of one layer feeding into the input of the next would permit the synthesis of a far greater set of functions.

This arrangement became known as a *multi-layer perceptron* or a *feed-forward* neural network, i.e. one in which the connections between neurons do not form a cycle. An example of this is shown in Figure 2.17.

A neural network might have any number of hidden layers. A network with more than two hidden layers is normally termed a *deep neural network*. While in theory, ac-

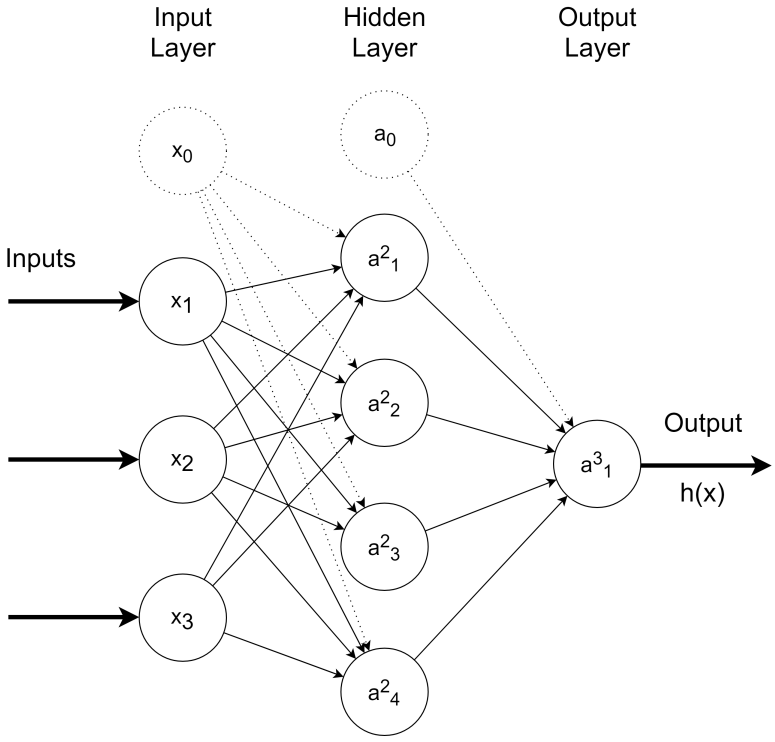

Figure 2.17: Topology of a simple feed-forward neural network with 3 input neurons, 2 output neurons and 5 neurons in a single hidden layer.

cording to the *Universal Approximation Theorem*, a shallow neural network with one hidden layer can approximate any function (Csáji, 2001), a deep network will require fewer neurons to approximate the same function, and hence is more efficient.

2.5.3.1 | Training a Neural Network: Back-Propagation

While the multi-layer neural network is much more powerful than the single-layer perceptron, it cannot be trained using the simple algorithm used for the single-layer perceptron as described above.

The procedure shown above for optimising the output of a perceptron is a special case of an algorithm named *Gradient Descent*. In general the Gradient Descent algorithm is intended to find the minimum in a convex function and it does so by iteratively moving from point to point along the curve in steps which size is governed by a learning rate and by the gradient of the function at the current point. The closer to the minimum the algorithm gets, the smaller the gradient and the smaller the steps it takes. The algorithm terminates when the step size has reduced below a pre-set minimum (Mitchell, 1997).

Concretely, given a function $f(x)$ and given a starting point x_1, the gradient of $f(x)$ is computed at x_1, given by $f'(x_1)$ where $f'(x) = \frac{d}{dx}f(x)$. The next point in the progression is given by:

$$x^{(2)} = x^{(1)} - af'(x_1) \tag{2.14}$$

where a is the value for the learning rate. This it repeated iteratively until

$$|x_{n+1} - x_n| < \text{some value } \delta$$

where δ is a pre-set minimum step size cutoff value.

This algorithm can be generalised to multivariate functions of the form $f(x_1...x_M)$. This is done by updating every term $x_1...x_M$ as shown in Equation 2.14 using the partial derivative of $f()$ with respect to each term, i.e. for all m in 1...m:

$$x_m^{(n+1)} = x_m^{(n)} - a\frac{d}{dx_m}f(x_1...x_m)$$

Now, the performance of any supervised machine learning model can be described by defining a *Cost Function* over the expected output y of the model given by the training data and the actual output of the model $\hat{y}$, parameterised by the parameters of the model. The cost function serves to quantify the distance of a model's predictions from a truth value. The objective of the learning algorithm is to modify the model parameters such as to minimise the value of the cost function, i.e. to bring the predicted values as close to the training values as possible. It is possible to define various cost functions. A common cost function that is widely used is the *Mean Square Error*:

$$J_{MSE}(\vec{\theta}) = \sum \frac{1}{n}(\hat{y} - y)^2 \tag{2.15}$$

$$= \sum \frac{1}{n}(h(\vec{\theta}) - y)^2 \tag{2.16}$$

where the vector $\vec{\theta}$ gives the parameters of the models and $h(\vec{\theta})$ is the prediction of the model, also known as the *Hypothesis Function*.

The cross-entropy cost function is also commonly used when the hypothesis function takes the form of the sigmoid (logistic) function:

$$J_{ce}(\vec{\theta}) = -\sum [yln(\hat{y}) + (1-y)ln(1-\hat{y})])$$

Gradient Descent is then commonly used to minimise the cost function with respect to the parameters of the model $\vec{\theta}$.

Consider the neural network shown in Figure 2.17. This network consists of three layers - an input layer of three neurons $x_1 \dots x_3$, a hidden layer of five neurons $a_1 \dots a_5$ and a single-neuron output layer. Let a_i^j be the activation of neuron i in layer j and W^j be the matrix of weights controlling the mapping from layer j to layer j+1 such that $w_{i,k}^j$ is the weight applied at neuron i in layer $j+1$ to the input coming from neuron k in layer j. By convention we define a bias unit x_0 (or a_0) in each layer, which always outputs a value of 1. The weight associated with this bias unit, $w_{i,0}^j$ will be equivalent to the bias of neuron i in layer j. The function $h(\boldsymbol{x})$ is referred to as the *Hypothesis function* - the function governing the prediction of the neural network given the input vector $\boldsymbol{x}$

Referring to the neural network in Figure 2.17, we can write:

$$a_1^2 = g(w_{1,0}^1 x_0 + w_{1,1}^1 x_1 + w_{1,2}^1 x_2 + w_{1,3}^1 x_3)$$

$$a_2^2 = g(w_{2,0}^1 x_0 + w_{2,1}^1 x_1 + w_{2,2}^1 x_2 + w_{2,3}^1 x_3)$$

... etc., where $g(x)$ is the chosen activation function for the layer. Similarity, for layer 3:

$$a_1^3 = g(w_{1,0}^3 a_0^2 + w_{1,1}^3 a_1^2 + w_{1,2}^3 a_2^2 + w_{1,3}^3 a_3^2 + w_{1,4}^3 a_4^2)$$

Generalising and using vector notation we can write:

$$\boldsymbol{A}^j = g(\boldsymbol{z}^j) = g(\boldsymbol{W}^{jT} \boldsymbol{A}^{j-1})$$

where $g(x)$ is the activation function.

Generalising further, we can consider the hypothesis function of the entire network to be a function of all the weights in the network, therefore if a suitable cost function is used this can be minimised via gradient descent and in the process, the weights of the network adjusted to optimise the predictive performance of the model.

In order to do this, however, the partial derivative of the cost function with respect to all the weights of the network are needed. In order to calculate these partial derivatives, an algorithm known as *Back-propagation* is used.

The back-propagation algorithm was developed in the 1960s by several researchers independently of the context of neural networks (Schmidhuber, 2015) and it was applied for the first time to neural networks by Werbos (1974).

In essence, the back-propagation algorithm involves several iterative steps (Mitchell, 1997):

1. Initialise network with random weights
2. Define a cost-function $J(W)$ which returns an error value given the hypothesis value $h(x)$ and the expected value y.

3. Apply training example to input neurons and forward-propagate the neuron activations to the output layer.

4. For every layer l in $(L..1)$ where L is the number of layers, calculate the vector error terms starting from the last layer:

$$\boldsymbol{\delta^L} = h(x) - (y) = \boldsymbol{a^L} - \boldsymbol{y}$$

and

$$\boldsymbol{\delta^l} = ((\boldsymbol{w^l})^T \boldsymbol{\delta^{l+1}}) * g'(\boldsymbol{z^j})$$

where $g'(z^j)$ is the derivative of the activation function applied to the weighted sums of the inputs in layer j. This is the "back-propagation" part of the algorithm.

5. Calculate the partial derivatives of the cost function with respect to the weights as:

$$\frac{\partial J(w)}{\partial w^l_{i,j}} = a^l_j \delta^{l+1}_i$$

6. Repeat the above for all training examples, accumulating the partial derivatives for each weight into a matrix $\Delta^l_{i,j}$

7. Optimise the weights using Gradient Descent (or other algorithm) using the average derivatives over all the samples:

$$\frac{\partial J(w)}{\partial w^l_{i,j}} = \frac{1}{m} \Delta^l_{i,j}$$

The above steps are repeated until the loss of the neural network, i.e. the value of the loss function, converges to within a pre-set threshold.

A neural network can be used for both *classification* tasks as well as *regression* tasks.

Classification involves predicting a class or label for a given input. There are, broadly speaking, three types of classification tasks:

- **Binary Classification**. The network outputs a single True/False binary value depending on the input. For example, a network might be trained to process a picture and output a value of 1 when the picture depicts a cat and 0 otherwise.

- **Multi-class Classification**. The network has multiple boolean outputs, each output signifying a different condition. For example, a network might have two outputs, one of which is True when the input picture contains a cat and the other being True when the input picture contains a dog.

- **Multi-class, multi-label Classification**. In this configuration, the network is capable of recognising multiple conditions simultaneously. In keeping with the above example, the network would be able to recognise when the input picture contains multiple types of animals.

Regression tasks, on the other hand, involve the prediction outputting a continuous value, rather than boolean ones. A neural network set up as a regressor might be used, for example, to predict real-estate prices.

These different types of tasks require the use of different activation functions at the output layer as well as different cost functions. Binary classifiers usually use the logistic activation function in the output layer and the cross-entropy loss function. Regressors, on the other hand, use the Mean Square Error as the cost function and a linear activation function in the output layer. Note, however, that a neural network is not limited to using a single activation function throughout the network, but different layers can be set up with different activation functions(Mitchell, 1997).

2.5.3.2 | Application in USR Simialarity Matching

ANNs have been an extremely successful, general purpose, class of machine learning algorithms that have found extensive use in such a wide variety of fields that a complete review of their applications would be out of scope for this project. They have also, however, been extensively used within the field of virtual screening, for prospective as well as retrospective studies and structure-based as well as ligand-based virtual screening (Lavecchia, 2015; Lavecchia and Giovanni, 2013; Molnár and Keserű, 2002; Omata et al., 2007; Selzer and Ertl, 2006; Winkler and Burden, 2002).

As part of our experiments, we have applied neural networks to the problem of ranking USR descriptors by similarity. Our motivation for exploring this algorithm was that, in contrast to the previous two algorithms, ANNs are supervised models, requiring both positive and negative examples during training. In selecting such an algorithm, we could compare the similarity matching performance of a ubiquitous 2-class supervised model with that of our two other chosen 1-class models.

Note, however, that in a real-world, prospective scenario, the scope for using ANNs in a virtual screening application would be dictated by the number, and quality of negative training examples that are available. The previous 1-class models that we explored would have no such limitation.

We initially set up feed-forward neural networks in regression-mode, with a single hidden layer using the RelU activation function with 100 neurons and a single output neuron using the Mean Square Error cost function. These were parameters, obtained

empirically by preliminary experiments on reduced-size datasets based on our available data.

During the course of our experiments we expected models trained on full conformers to achieve better results than corresponding models trained on LECs. In contrast to the previous two models, however, we failed to observe this behaviour in our ANNs. In view of this, we subsequently re-trained alternative ANN models using 500-node hidden layers in order to determine if the mode complex networks could achieve better predictive performance for full conformer models. More details regarding this can be found in Section 4.2.3.

It is essential to point out that the network topologies we chose for this project are only a reasonable starting point determined by preliminary exploration of the available data. Given enough time and processing resources, the ideal scenario would have been to more fully explore the hyper-parameter space for these models, varying the topology of the network so as to find the better-performing configurations. In particular, we would have liked to perform tuning of the networks with respect to cost function and activation functions, as well as the number of hidden layers and their size and other hyper-parameters that are available in the model. The aim of this project, however, was to determine if USR descriptors were amenable to be successfully modelled by using machine learning algorithms. The fine tuning of such models to achieve the best possible performance must necessarily be left to future work, due to the inherent time and resource constraints that we had to abide by.

2.6 | Evaluation Criteria

Within the Ligand-Based Virtual Screening literature two main evaluation criteria are commonly used: Receiver Operator Characteristic (ROC) Curves/Area Under Curve (AUC) and Enrichment Factor (EF).

ROC Curves are a method of measuring the performance of a machine learning model. In general, a classification model will output a positive or a negative result by first calculating some continuous value and then deciding upon the positive/negative output by applying some threshold. The choice of such threshold will determine which values of the output will be considered positive and which negative.

Now if we define the True Positive Rate (TPR) as the fraction of positive test instances predicted correctly to be positive and the False Positive Rate (FPR) as the fraction of negative test instances predicted wrongly to be positive, we can plot the TPR vs. the FPR as a function of the threshold. An example ROC Curve generated as part of this

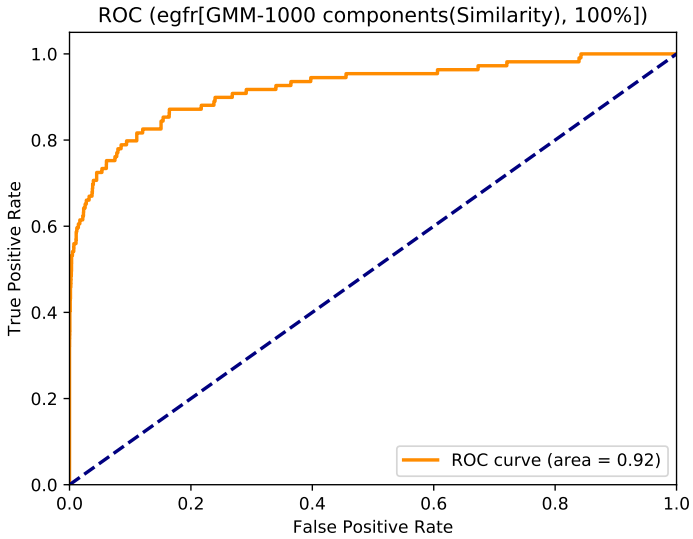

Figure 2.18: Example ROC Curve.

research project can be seen in Figure 2.18.

For a totally random classifier, the curve will be a diagonal line with origin at (0,0) and unit gradient. A graph curving above this line indicates a good (above random) classifier and one below this line signifies a bad (below random) classifier. Note that inverting the label of a bad classifier will turn it into a good classifier. Consequently, the greater the area under the ROC curve, the better the classifier and the AUC metric is used to characterise the power of classification model.

In the context of similarity ranking in USR-like methods, producing ROC curves involves treating each of the N actives $A_{0..N}$ as a template molecule, calculating its similarity with every other molecule in the dataset. This produces N lists, $L_{0..N}$ of similarity scores corresponding to each of the N active templates. There are then two ways of producing ROC curves from these lists, as described in Armstrong et al. (2010).

One way is to use the similarity scores themselves together with the truth labels to generate the curve by varying the similarity threshold, i.e. the similarity threshold is swept through values between 1 and 0, at each step counting the true positives and false positives found at that similarity level.

The second way is to use the *rank* of the ordered molecules within each list L_i as the threshold quantity. In this case the rank threshold starts from 0 and is incrementally increased, counting the true and false positives at each step, until all the molecules are accounted for. In our work we generate both types of ROC curves so as to compare

results.

Armstrong et al. (Armstrong et al., 2010) point out that, in general, it is not possible to predict a priori whether one particular VS method will produce better similarity ROC curves or rank ROC curves, however they point out that ElectroShape methods tend to produce better scores with rank ROC curves.

Note that these procedures are valid when performing traditional, non-machine learning, retrospective virtual screening evaluations. When using machine-learning models, however, in general, the similarity scores for the molecules under test are generated against a learnt statistical model incorporating information from all the active molecules as an ensemble. It is therefore not possible to make a distinction between similarity ROC and rank ROC in this case, because separate similarity lists for each active are not obtainable. When evaluating ML models, therefore, it is only possible to calculate ROC curves based on the output quantity of the respective model, it being a similarity score, probability score or similar.

The Enrichment Factor is a measure used specifically in retrospective LBVS studies, such as those we perform in this dissertation. The Enrichment Factor at a given percentage of a dataset is defined as the ratio of the fraction of actives correctly found within the first x% of the ranked dataset to the fraction of actives that would be found by chance. More formally:

$$EF_{i,j,x\%} = \frac{a_{i,j,x\%}/c_{x\%}}{a_{i,j,100\%}/c_{100\%}}$$

where $EF_{1,j,x\%}$ is the enrichment factor at x% for active j in target i, $a_{i,j,x\%}$ is the number of actives found in the top x% of the sorted dataset for active j in target i and $c_{i,j,x\%}$ is the total number of compounds in x% of the dataset.

Summing this formula over the actives we obtain the average enrichment factor for a given target and summing again over all the targets gives the average enrichment factor for the method.

The difference between these two evaluation criteria is that while ROC Curves/AUC gives a picture of the performance of the method across the entire dataset, the EF emphasises the *Early enrichment*, i.e. the performance of the model in detecting actives at the top ranking positions. This makes sense in the context of virtual screening, where only the top-ranked compounds will go ahead to be physically tested in the laboratory. The limitation in the use of EF is that it is not comparable between studies using different compound databases because it depends on the active/compound ratio present in the dataset. This means that while EF gives a useful metric for comparing models within

the same study, using the same dataset of compounds sampled in a similar manner, the ROC/AUC metric is more suited when comparing results from different studies.

2.7 | Related Work

As discussed in the previous sections, there has been relatively intense development related to the technique of USR, with several researchers iteratively improving its performance by augmenting the basic technique with more information. In spite of this, however, to date there has not been a study that applied machine learning concepts to USR and USR-related descriptors.

Machine learning, has however, been applied extensively to all branches of Virtual Screening, including SBVS, as well as fingerprint-based LBVS.

In the area of SBVS, the focus is on the development of scoring functions for docking based on machine learning. Docking is the process of computationally matching a molecule to a protein binding pocket as described in Section 2.1. Machine learning techniques applied to docking scoring functions have proven to yield much better results than classical, non-ML scoring functions (Ain et al., 2015) One of the more recent examples of this is Wójcikowski et al. (2017) where the authors develop a new machine learning-based scoring function based on the Random Forest algorithm which outperforms classical scoring functions.

Artificial Neural Networks have also been used extensively in virtual screening. In Betzi et al. (2006), an ANN was trained using a combination of several classical scoring functions and was shown to yield superior results than a linear combination of the same scoring functions. In Pereira et al. (2016), DeepVS, a deep learning approach to docking is proposed which automatically learns the relevant features from ligand-protein complexes and, being evaluated using the Directory of Useful Decoys (DUD), obtained the best docking score to date in terms of ROC AUC.

In the realm of LBVS, the molecular fingerprints generated by 2D similarity fingerprinting techniques can easily act as training features for machine learning models (Lavecchia, 2015). Several classes of features can be used as training data for machine learning models. Machine learning methods have been applied successfully to 2D descriptors in a large number of studies (Geppert et al., 2010; Lavecchia, 2015; Stahura and Bajorath, 2004). As an example, in Chen et al. (2007) the authors apply kernel discrimination and naive Bayesian classifier methods to a variety of 2D descriptors, managing to enrich the top 1% with up to 90% of the dataset actives, demonstrating the effectiveness of such methods with a particular focus on the method performance on reduced

training sets.

In Warmuth et al. (2003) the Support Vector Machine (SVM) algorithm is used in conjunction with active learning, wherein an iterative approach to training the ML model is taken, using the actives identified from previous iterations as new training data for subsequent ones.

Hert et al. (2006) explores the use of data fusion and machine learning techniques to enhance the performance of similarity searching of molecules when multiple reference structures are available. This is in contrast to traditional ligand-based screening where only one reference structure at a time is used. In addition to this, the authors propose *turbo* similarity searching as an alternative when only one reference structure is available. Using this method, machine learning models are trained using the known active, as well as other similar molecules as active training examples. The authors show that using this idea, significant improvements in performance are achieved.

In Kurczab et al. (2011) 60 different machine learning methods from the WEKA data mining software (Hall et al., 2009) are applied to various types of fingerprints and training set sizes, concluding that there is no method that consistently outperforms every other one, however there are methods that universally produce consistently good performance.

When it comes to Molecular Shape Comparison (MSC) LBVS techniques, there has been much less research in the application of machine learning techniques than there has been for fingerprint techniques. It is a known fact that fingerprint techniques yield a higher number of hits than 3D structural approaches (Venkatraman et al., 2010) and therefore it is not surprising that a large effort has gone into improving the methods that are known to work best.

It is also realised, however, that the inability of shape-based LBVS to perform as well as 2D approaches stems from the fact that in traditional approaches only one conformer at a time can be taken as a template molecule for the similarity search (Venkatraman et al., 2010). This makes it difficult or impossible to detect deeper patterns that would be evident if multiple conformers could be considered en-bloc. This possibility of achieving just that, however, exists in the application of machine-learning methods which, in general, take examples of valid data points and generalise them so as to extract statistically significant patterns against which new, unclassified data points can be compared.

Jahn et al. (2010, 2011) remark on this point and in their research propose a method of generalising the entire conformer-space of a molecule in terms of Gaussian Mixture Models. This is done by sampling the pairwise atom-distance distribution for flexible atom pairs, i.e atoms connected by one or more rotatable bonds in the molecular graph,

and from these train a GMM for each flexible atom pair. These pre-processing steps are carried out for every molecule in the database. Molecule similarity is then calculated using an overlap metric between GMMs in order to quantify the similarity between the Gaussians in each GMM, and hence between the pairs of GMMs. The authors report significant performance improvements compared to 15 different 2D and 3D methods including USR over which an average performance increase of 4 times was achieved in terms of $EF_{1\%}$.

A more conventional application of machine learning was proposed in Sato et al. (2012) wherein the authors applied SVM-based models to Molecular Shape Overlays. In their paper, the authors point out that applying machine learning techniques to 3D LBVS methods is not as straightforward as doing so for 2D techniques because 3D methods do not generally result in descriptors or features suited to the task. Due to this, the authors develop Molecular Shape Overlays, a novel technique in which the molecules in their compound database are superimposed on all the available actives for a given target using ROCS so that a similarity profile for the molecules can be calculated. These profiles, in the form of vectors of features were then used to train the SVM.

Support Vector Machines are a powerful class of supervised models that are capable of discovering non-linear separation boundaries between labelled data points (Cortes and Vapnik, 1995). The power of SVMs is the ability of using custom Kernel functions to encode similarity between different types of data points. Due to this flexibility, SVMs have found wide-ranging use in a plethora of domains where machine learning is applied such as text classification (Pradhan et al., 2004), image classification (Barghout, 2015) and optical character recognition (Decoste and Schölkopf, 2002).

A variety of custom kernel functions specific to virtual screening have been developed for use with SVMs (Lavecchia, 2015), including a Pharmacophore kernel (Mahé et al., 2006) which enables SVM classification to be performed based on pharmacophore-based similarity. Pharmacophores are groups of atoms in a molecule that have an active role when binding to a target protein. The discovery of pharmacophores has an important role in the determination of Structure-Activity Relationships (SAR) - the relationship between chemical structures and specific biological activity.

A variety of neural network called Self-Organising Map (SOM) was applied in Selzer and Ertl (2005) in order to identify ligands to the G-Protein-Coupled Receptor (GPCR) protein target using the Radial Distribution Function (RDF) to generate training data from 3-D conformers. The RDF is based on a histogram of atomic distances and atomic properties such as partial charge. SOMs are a type of neural network in which neurons are connected in a grid pattern and are trained so that similar inputs are mapped to adjacent neurons. They are generally used for clustering and for visualisation purposes.

Clustering actives for a class of proteins called G protein-coupled receptors (GPCRs) using a SOM, the model managed to identify 71% of the GPCR ligands in a data set of which only 5.9% were actives

As stated previously, there is much work still to be carried out when it comes to the application of machine learning to the field of shape-based similarity searching in LBVS. The USR family of methods, however, are singularly well-suited to such a treatment, by virtue of their use of numeric descriptor vectors, similar in structure to those produced in 2D methods. These descriptors are easy to use unchanged as training instances for machine learning models and doing so affords the advantage of combining the information inherent in the descriptors for all the actives into a single, comprehensive search which has a greater likelihood of identifying significant patterns than the traditional method of searching compound databases one active at a time.

2.8 | Summary

In this chapter we have given a comprehensive overview of the field of Virtual Screening and related concepts. We have discussed the different types of Virtual Screening techniques with particular reference to Molecular Shape Comparison in Ligand-Based Virtual Screening.

We have then given the history of the USR technique and the related family of methods, detailing the concepts inherent in the methods and going into detail with the specific techniques that we have used in the course of this project.

We then discussed concepts relating to conformer generation which is a necessary step in data pre-processing for shape-based virtual screening techniques, following which we gave detailed overviews of the three machine learning methods we have explored in the dissertation as well as the metrics we used to evaluate our methods.

The chapter then concludes with an overview of the literature related to our project.

3

Methodology

This chapter details the methods and processes employed to achieve the stated aims and objectives. The knowledge presented in the Background and Literature Overview chapter forms the foundation for the work presented in this chapter.

3.1 | Approach Overview

The implementation of the project is broadly divided into several major functional parts as detailed below. An overview of our approach can be seen in Figure 3.1.

- **Conformer Generation**. This process takes the DUD-E SMILES molecule datasets and generates the necessary number of conformers for every molecule, saving them to file for use by subsequent tasks.
- **Descriptor Generation**. This process takes the conformers generated in the previous process and uses them to generate corresponding descriptors for USR, CSR, ElectroShape-4D (ES4D) and ES5D.
- **Standard USR, CSR, ElectroShape-4D and ElectroShape-5D Algorithms**. This process implements the USR family of algorithms as specified by Ballester and Armstrong in Ballester (2011), Armstrong et al. (2009), Armstrong et al. (2010) and Armstrong et al. (2011) in order to be able to generate baseline performance scores to which we can then compare the performance scores obtained by our machine-learning models.
- **Machine learning model building, tuning and evaluation**. This task is the implementation of a model selection, tuning and evaluation pipeline for training the

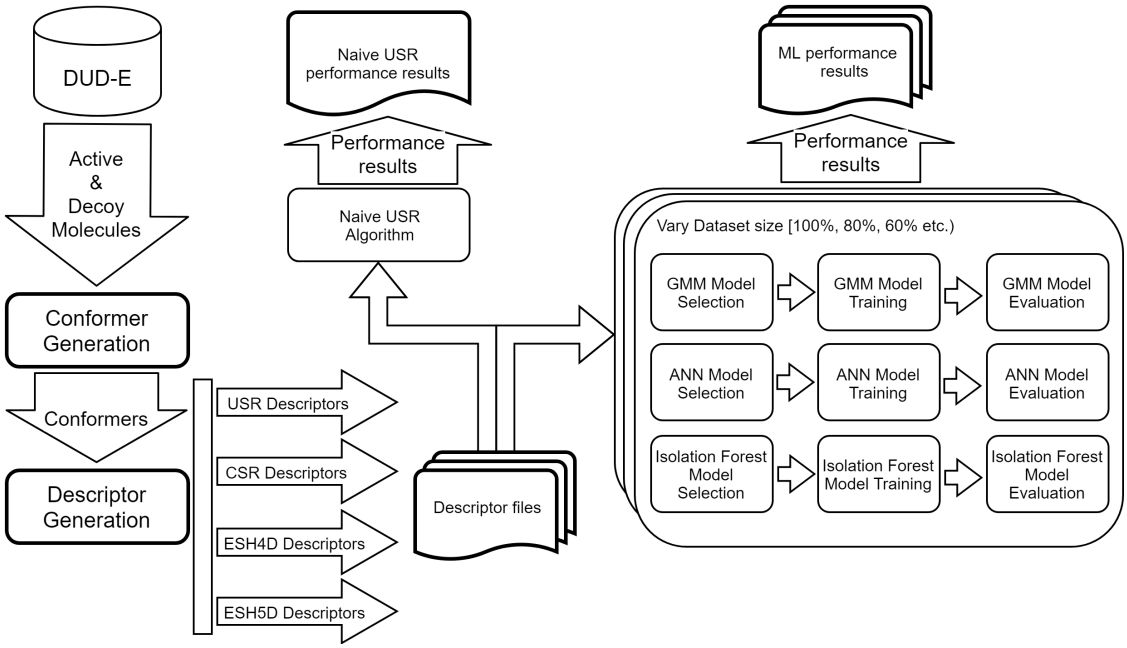

Figure 3.1: This schematic represents the major functional blocks comprising our approach to meeting the objectives stated in Section 1.4. Molecule entries from DUD-E are processed by the conformer generation module which expands the molecule definitions into individual conformers. These conformers are then used to generate USR, CSR, ES4D and ES5D descriptors. The descriptors are then used in our implementation of these algorithms to generate baseline performance metrics. The same descriptors are then also used to construct, tune, train and evaluate Machine-Learning models based on Gaussian Mixture Models, Artificial Neural Networks and Isolation Forests. These models are constructed and evaluated over several fractions of the entire dataset in order to gauge how well the models adapt to decreasing dataset size.

machine learning algorithms that we determined to be suitable for modelling the problem of similarity searching based on USR descriptors. In this project, we explore the use of Gaussian Mixture Models, Aritifical Neural Networks and Isolation Forests. Using the tuned and trained models we generate performance scores on the same target proteins as the previous step. One of the aims of this research project is to determine how the performance of machine learning algorithms applied to USR similarity searching degrade with decreasing training dataset size. These experiments are performed by systematically selecting decreasing-size subsets of the available training data and producing separate evaluation results for each subset.

All development was done in Python 3.6 and all processes were run on Ubuntu 18.04

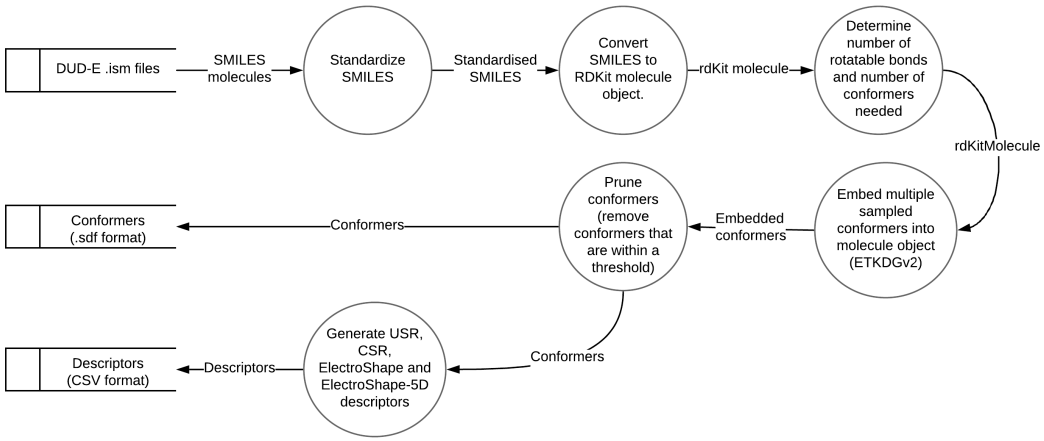

Figure 3.2: Conformer generation overview

instances. The following sections describe every functional block in more detail.

3.2 | Conformer Generation

We implemented two versions of the conformer generation process. One version was designed to run on a single machine, parallelising the load among the available cores (`genConformersParallel.py`) and the other was designed to make use of the Spark platform so as to be scalable over multiple clustered machines (`genConformersSpark.py`). The conformers used in this project were generated using the Spark version of the code.

A data-flow diagram depicting the Conformer Generation process can be seen in Figure 3.2.

For the purposes of this research project, all Cheminformatics-related processing was performed using the 2018.09.1 version of the RDKit library, a widely-used open-source Cheminformatics library (Landrum and Others, 2013). RDKit is written natively in C++, however also has Python bindings, allowing it to be easily used from Python.

The input datasets to the conformer generation process are `*.ism` files downloaded from DUD-E. For every target protein, DUD-E provides two datasets - `actives_final.ism` and `decoys_final.ism`. Each `.ism` file contains a list of molecule definitions specified in the standard SMILES format (Weininger, 1988), one molecule per line together with the ChEMBL code for each molecule. ChEMBL is *"... a manually curated database of bioactive molecules with drug-like properties"*, maintained by the European Molec-

Target	Description	Active Mols.	Decoy Mols.	Active Confs.	Decoy Confs.	Confs./mol (Actives)	Confs./mol (Decoys)
ACE	Angiotensin-converting enzyme	282	16,900	31,947	1,266,730	113	74
ACES	Acetylcholinesterase	453	26,250	55,549	2,153,887	122	82
ADA	Adenosine deaminase	93	5,450	7,786	332,177	83	60
ALDR	Aldose reductase	159	9,000	4,797	375,355	30	41
AMPC	Beta-lactamase	48	2,850	1,351	99,431	28	34
ANDR	Androgen Receptor	269	14,350	12,068	543,761	44	37
CDK2	Cyclin-dependent kinase 2	474	27,850	21,273	1,371,687	44	49
COMT	Catechol O-methyltransferase	41	3,850	1,262	147,125	30	38
DYR	Dihydrofolate reductase	231	17,200	16,679	873,009	72	50
EGFR	Epidermal growth factor receptor erbB1	542	35,050	41,580	2,405,525	76	68
ESR1	Estrogen receptor alpha	383	20,685	21,024	1,212,349	54	58
FA10	Coagulation factor X	537	28,325	38,757	2,087,845	72	73
FGFR1	Fibroblast growth factor receptor 1	139	8,700	9,232	535,529	66	61
GCR	Glucocorticoid receptor	258	15,000	12,111	652,595	46	43
HIVPR	Human immunodeficiency virus type 1 protease	536	35,750	67,552	3,436,686	126	96
HIVRT	Human immunodeficiency virus type 1 reverse transcriptase	338	18,891	16,576	836,334	49	44
HMDH	HMG-CoA reductase	170	8,750	22,037	827,459	129	94
HS90A	Heat shock protein HSP 90-alpha	88	4,850	4,918	235,367	55	48
INHA	Enoyl-[acyl-carrier-protein] reductase	43	2,300	3,900	118,362	90	51
KITH	Thymidine kinase	57	2,850	3,168	150,295	55	52
MCR	Mineralocorticoid receptor	94	5,150	3,960	215,697	42	41
MK14	MAP kinase p38 alpha	578	35,850	34,310	2,096,198	59	58
NRAM	Neuraminidase	98	6,200	6,030	325,337	61	52
PARP1	Poly [ADP-ribose] polymerase-1	508	30,050	18,925	1,242,760	37	41
PDE5A	Phosphodiesterase 5A	398	27,550	32,657	1,876,746	82	68
PGH1	Cyclooxygenase-1	195	10,800	8,123	410,263	41	37
PGH2	Cyclooxygenase-2	435	23,150	19,598	960,837	45	41
PNPH	Purine nucleoside phosphorylase	103	6,950	3,277	284,801	31	40
PPARG	Peroxisome proliferator-activated receptor gamma	484	25,300	71,166	2,527,881	147	99
PRGR	Progesterone receptor	293	15,650	13,041	578,492	44	36
PUR2	GAR transformylase	50	2,700	7,931	195,987	158	72
PYGM	Muscle glycogen phosphorylase	77	3,950	3,300	212,652	42	53
RXRA	Retinoid X receptor alpha	131	6,950	8,008	316,919	61	45
SAHH	Adenosylhomocysteinase	63	3,450	1,883	118,691	29	34
SRC	Tyrosine-protein kinase SRC	524	34,500	39,561	2,313,655	75	67
THRB	Thrombin	461	27,004	57,028	2,131,048	123	78
TRY1	Trypsin I	449	25,980	47,961	1,933,063	106	74
VGFR2	Vascular endothelial growth factor receptor 2	409	24,950	25,349	1,518,622	61	60

Table 3.1: The list of 38 protein targets that we considered in the course of the project along with the number of active and decoy molecules that were available for each protein target and the respective number of active and decoy conformers that we generated for each target. These targets correspond to the "Dud38" subset in DUD-E.

ular Biology Laboratory (EMBL) (Gaulton et al., 2012).

For reasons of resource availability, we considered a subset of the protein targets in DUD-E for this research project. The subset of proteins that we chose is shown in Table 3.1 and consists of the molecules also present in DUD. This was done because most USR-related research papers we considered used DUD for evaluation purposes. Choosing the same molecules allowed us to directly compare (i) the performance obtained by the USR family of methods when applied to DUD-E as opposed to DUD and (ii) the performance of machine learning methods compared to the non-machine learning USR algorithms.

Each SMILES definition in the input `.ism` files is passed through standardisation functions provided by RDKit. These functions, previously provided by the independent MolVS project[1] and now integrated into RDKit, perform sanitisation procedures on the molecule, checking for problems in the definition and fixing them, yielding a standardised SMILES string. The reason for which we perform this standardisation step is that, using SMILES, it is possible to specify the same molecule in several different ways. For example Hydrogen molecules may be included or omitted, charges may be included or omitted, aromatic rings may be broken in different places, etc. There are also other conditions that can cause confusion. For example *tautomers* are isomers of a compound that differ from each other only in the positioning of their Hydrogen atoms. Different tautomers give rise to different SMILES strings, but chemically, they readily convert between each other, therefore they are usually regarded to be the same chemical compound. The standardisation functions provided by RDKit take a large number of special cases such as the above into account and ensure that any SMILES string representing a given compound is converted to a canonical form that is invariant for that compound. All further operations are performed on the standardised molecules.

During the implementation of this module, we discovered an issue in RDKit which was causing it to hang when generating conformers for some molecules that had been standardised, while this did not happen for the same molecules when we ommitted the standardisation step. We contacted the RDKit maintainers who confirmed that it was a new bug related to the way RDKit interprets SMILES strings when performing a round-trip conversion from SMILES to the internal RDKit molecule representation and back to SMILES. We have reported this issue in the RDKit issue tracker[2], however it has not been fixed in time for the completion of this project. This resulted in a failure generating conformers for a small number of molecules from each protein target as seen in table.... This problem, however was not severe enough to alter our results.

As discussed in Section 2.3, the number of conformers to generate in order to adequately sample a molecule's conformational space is dependent on the number of rotatable bonds present in the structure of the molecule. For each of the input molecules, therefore, we use RDKit to determine the number of rotatable bonds and based on this, we compute the number of conformers n needed. Ebejer et al. propose a protocol for conformer generation with RDKit in Ebejer et al. (2012). Their publication has been widely cited and the procedure has been used in a variety of contexts, eg. Gerhardt et al. (2017) and Li et al. (2016). We therefore followed Ebejer at al.'s guidelines as follows:

[1] https://github.com/mcs07/MolVS [Last Accessed 5 May 2019]
[2] https://github.com/rdkit/rdkit/issues/2169 [Last Accessed 26th May 2019]

$$n = \begin{cases} 50, & \text{if } n_{rot} <= 7, \\ 200, & \text{if } n_{rot} <= 12, \\ 300, & \text{otherwise.} \end{cases}$$

where n_{rot} represents the number of rotatable bonds in the molecule.

Conformer generation is performed using open-source code by Steven Kearnes[3] which we modified in two ways:

- **Use of ETKDG**. We modified the code to use Experimental-Torsion Knowledge Distance Geometry (ETKDG) as the conformer generation algorithm. As described in Section 2.3, ETKDG is a newer stochastic conformer generation method which builds upon the existing Distance Geometry (DG) algorithm by using experimental knowledge about preferential torsional-angles and "prior" basic knowledge about molecular structure. The major advantage in using ETKDG as opposed to DG is that the output of DG is not optimal and the resulting conformers can be generated in a distorted state (e.g. aromatic rings not flat, torsional angles not conforming to experimental values). In order to remedy this, a second `energy minimisation` step is usually performed on these conformers in which interatomic force-field calculations are used to relax the molecule into a stable, energy-minimised state. This computationally expensive step is avoided by ETKDG as the embedded knowledge in the algorithm produces conformers that are already energy-minimised. We considered the upgrade of the existing code to use ETKDG as a desirable change to make for the above reasons.

- **Maximum energy cutoff**. In order to align our conformer generation process with Ebejer et al. (2012), we added a maximum energy cutoff value of 5 kJ/mol such that any conformer that has a total energy above this value is discarded. This is important because excessively high-energy molecules are unstable and unlikely to occur naturally.

Unfortunately, the ETKDG algorithm does not enable the computation of energy values for conformers. This means that it is necessary to use a forefield calculation to determine the energy for every conformer. This implies that energy minimisation based on the forefield has to be performed anyway before the energy calculation, notwithstanding the use of ETKDG. This is because the parameters used by the forefield will be different from those used by ETKDG and therefore energy calculations derived from

[3]https://github.com/pandegroup/vs-utils [Last Accessed 4 May 2019]

ETKDG-generated conformers based on the different forefield would generate wrong results. It is nevertheless, still advantageous to use ETKDG because the conformers generated will be closer to the energy minima than those generated by DG and therefore the energy minimisations step will be faster.

There are two basic steps that occur during conformer generation:

1. **Conformer generation**. The first step is the generation of a set number of conformers, depending on the number of rotatable bonds as discussed above, using the conformer generation method that was chosen - ETKDG in our case. This results in an unsorted list of randomly generated conformers.

2. **Conformer Similarity Pruning**. In this step, the conformers that are generated above are processed so as to eliminate similar ones. The similarity between conformers here is quantified using the Root-Mean-Square Deviation (RMSD). The RMSD is defined as the average root-mean-square distance between corresponding atoms in the two conformers being compared:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2}$$

   where δ_i is the distance between the ith atoms of the two molecules and N is the number of molecules. Conformers having energies above a set threshold are also removed. This process is illustrated in Figure 3.3. This process is in-line with that suggested in Ebejer et al. (2012)

3. **Conformer Energy Cutoff**. The remaining conformers after pruning are then sorted by increasing energy ensuring that the first conformer is the LEC

When the process is complete, the conformer generator then embeds the generated conformers into the Molecule object and returns it to the caller. The returned conformers are then written to file in the standard .sdf format (Dalby et al., 1992).

3.3 | Descriptor Generation

We have provided custom implementations for generation of descriptors for a selection of USR family of algorithms as follows:

- USR
- CSR

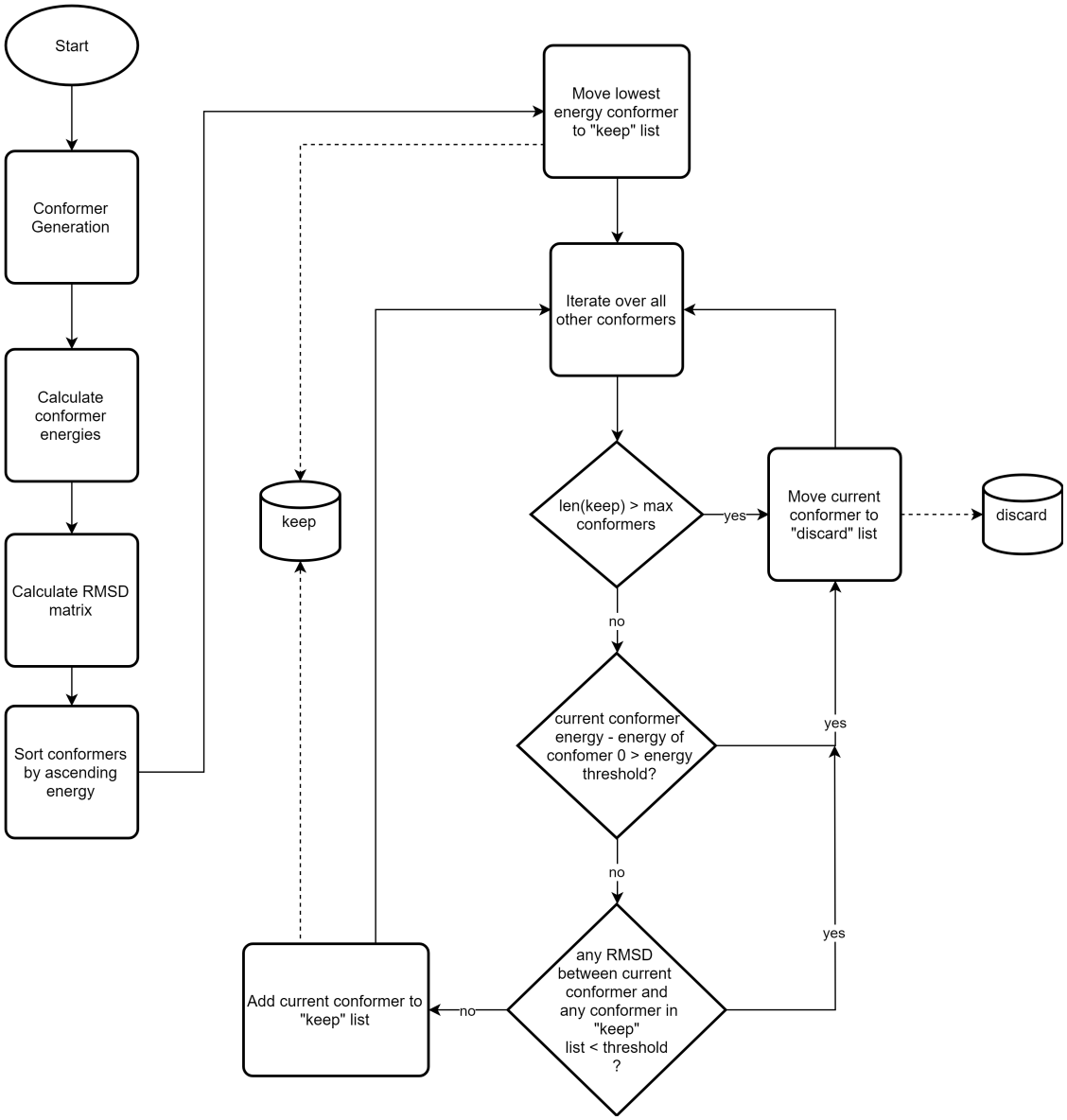

Figure 3.3: This flowchart illustrates the conformer pruning algorithm that we used in our conformer generation process. Conformers generated by ETKDG are first sorted by their energies. The Lowest Energy Conformer is always kept. Any conformer with a RMSD value against any other kept conformer is discarded. Any conformers with an energy exceeding the LEC by a set threshold are also discarded.

- ES4D (RDKit-based implementation)
- ES5D (RDKit-based implementation)

In contrast to USR and CSR which are based exclusively on spatial information, ES4D and ES5D incorporate physicochemical information into the descriptor, therefore in these cases RDKit was needed to calculate the relevant values. ES4D incorporates the atomic partial charges and we extracted these using the `getMMFFPartialCharge()` function. ES5D along with partial charges, also makes use of ALogP values which we extracted using the RDKit `_CalcCrippenContribs()` function from the `Crippen.rdMolDescriptors` class.

Note that, while conformer generation and descriptor generation are conceptually different processes, in the interest of efficiency we have combined them into a single Python process that generates both conformers and corresponding descriptors in a single run. Note also, that since ES5D is the highest scoring method among the four we implemented (USR, CSR, ES4D and ES5D) and in the interest of optimal resource utilisation, we chose to use only USR and ES5D in our machine learning experiments going forward.

The conformer and descriptor generation processes generated in excess of 300GB of data.

3.4 | Evaluation of Standard USR Family of Methods

In order to serve as a baseline performance measure against which to compare the performance of our machine learning algorithms, we have implemented our own version of the USR algorithm, using it to generate performance metrics for similarity searches on the DUD-E dataset for USRand ES5D descriptors.

Our version of the USR algorithm is distributed in nature, making use of the Spark platform to scale to the available number of machine instances.

A data-flow diagram depicting the USR Virtual Screening process is shown in Figure 3.4. This process is implemented by the class `USRMoleculeSim` in `USRRun.py`

As can be seen from the diagram, the process starts by concatenating the descriptors of the conformers of the n_a active molecules. The descriptors for the active conformers are broadcast to the Spark workers so that they can be accessed from the entire cluster. The conformer descriptors for the n_c candidate molecules for the search are then parallelised in Spark, creating a Resilient Distributed Dataset (RDD). The conformer similarity function `USRMoleculeSim.doSim()` is then mapped onto the conformer RDD,

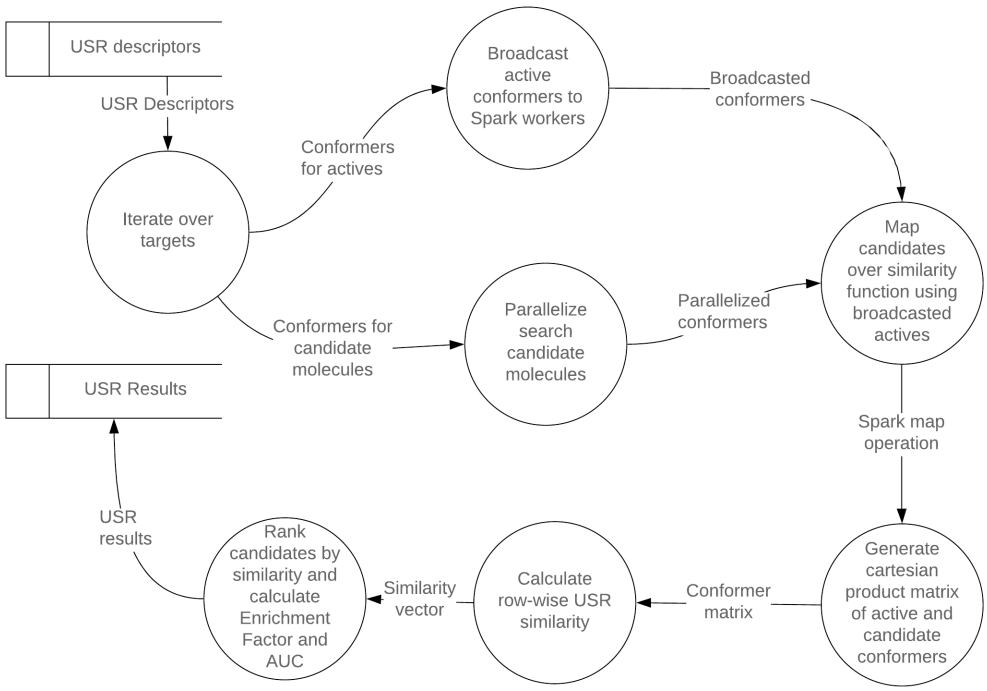

Figure 3.4: An illustration of the Spark-based implementation of the USR similarity evaluation process.

resulting in the similarity function being called once for every search molecule while being passed the molecule's conformer descriptors.

The similarity function is required to return a maximum similarity value for each search conformer with respect to the conformers of each active molecule. This could be done by iterating over each conformer of each active molecule and each search conformer and calculating the similarity value individually, however this would result in an exceedingly slow implementation in Python.

Instead of this naïve approach, we optimise the process by leveraging the fast native mathematical matrix operations provided by the `NumPy` library. We do this by using the `NumPy repeat()` and `tile()` functions to generate, for each active molecule, expanded active conformer and search conformer matrices in such a way that when combined in a row-wise manner, the two matrices results in the Cartesian product of the $n_a c$ active

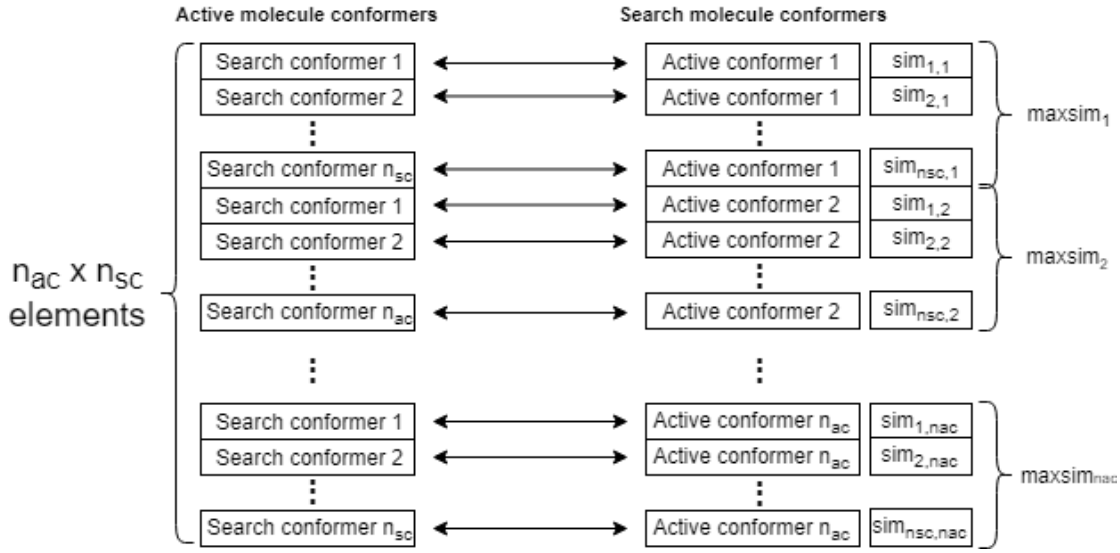

Figure 3.5: Matrix-based maximum similarity calculation between search conformers and active conformers.

conformer descriptors and the n_sc search conformer descriptors. The Manhattan distance can then be calculated in a row-wise manner between these two matrices giving a a $n_ac \times n_cc$-size matrix of similarity scores. The function then, for each active molecule, selects the maximum similarity score over the candidate conformers, resulting in one n_a-size vector containing the maximum similarity score of the search molecule per active molecule. This process is illustrated in Figure 3.5.

As per the Spark map operation, the above process is performed once for every search molecule, ultimately returning a $n_a \times n_c$ matrix of similarity scores, with one maximum similarity score per search molecule per active.

This matrix is then used to calculate

- Mean Enrichment Factor at 1% and 5%. This is the mean of the Enrichment Factor calculated over the active templates.
- The similarity-based ROC curve and corresponding AUC
- The rank-based ROC curve and corresponding AUC

as described in Section 2.6.

At a later stage, we converted this code to run independently of Spark, using the Python `mupltiprocessing` package to distribute processing among the available processors using a `map` paradigm similar to that we used in Spark. This was because later

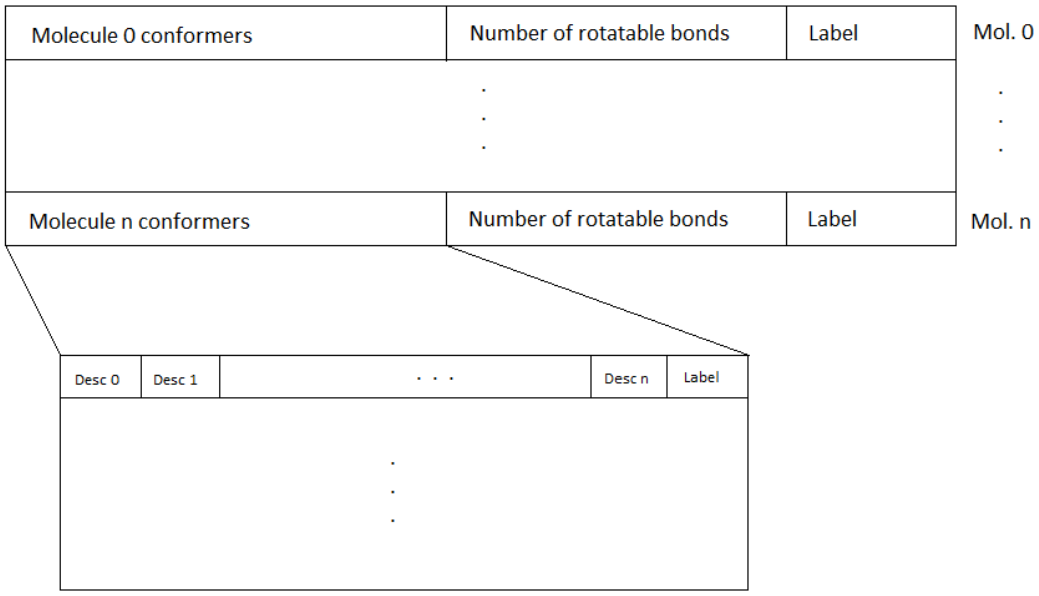

Figure 3.6: The "SEPARATE" record format is used when conformer descriptors belonging to different molecules need to be kept separate from each other when loading them into memory. Each molecule is represented by a record. The first field of each molecule records contains an array holding all the conformers of the molecule.

on during the project, new hardware was made available that had enough processors not to necessitate the use of a cluster.

3.5 | Dataset Loading and Partitioning

We have implemented custom dataset loading and partitioning code to deal with the USR family of methods descriptor files created by the conformation generation step in Section 3.2. We chose to do this because the input data records (the molecule descriptors) consist themselves of multiple records (the conformers for each molecule) and this format of data was not well handled by the standard Python machine learning libraries. The `conformer_utils.py` Python module contains our custom dataset loading and partitioning library.

The `loadDescriptors()` function is used to load the conformer descriptors from the files pre-generated in the descriptor generation process. This function takes several parameters to tailor which data files are loaded and how the data is returned. These

parameters are the following:

- `num_actives`: the number of actives to load. If <= 1 it is taken as a percentage of the total number of actives
- `active_decoy_ratio`: The number of decoys to load as a fraction of the number of actives. -1 to maintain the population ratio of decoys to actives
- `dtype`: The descriptor type ("usr", "csr", "esh", "es5")
- `selection_policy`: "SEQUENTIAL" - returns molecules in the order in which they appear in the dataset, "RANDOM": returns molecules in random order.
- `return_type`: `"SEPARATE"` - conformers for each molecule are returned as separate DataFrames, `"LUMPED"` - all the molecule's conformers are returned lumped in a single DataFrame.
- `exclusion_list`: A list of files to exclude from the loading. This is used during partitioning of the dataset to ensure that no overlaps occur with previously loaded portions of the data when loading subsequent portions.

The `"LUMPED"` and `"SEPARATE"` return types are used as needed in varying situations, depending on which format is most convenient. While the `"LUMPED"` format is simple, consisting of a single DataFrame with one conformer per line, the `"SEPARATE"` is slightly more complex and is illustrated in Figure. 3.6.

The `conformer_utils.py` module also supplies the following functions:

- **split()**: Split a record set into n folds using either "SEQUENTIAL" or "RANDOM" policy, returning separate record sets for each fold. This is used for n-fold cross-validation.
- **lumpRecords()**: Converts a `"SEPARATE"` format record set into a `"LUMPED"` format record set
- **joinDataframes()**: joins multiple `"SEPARATE"` format record sets into a single record set.

3.6 | Machine Learning Experiments

In general, in order to train and evaluate a machine learning model, the available dataset has to be partitioned into a training dataset and a testing dataset. The model is trained

on the training dataset and evaluated on the testing dataset. The result of the evaluation serves to guide the tuning of the model's hyperparameter values and the process of model training and evaluation is repeated until the optimum hyperparameter values are determined.

It is important that the performance metrics for model evaluation are taken on unseen data, i.e. a dataset disjoint from the testing dataset. This ensures that the resulting model does not exhibit high *variance*. A model with high variance implies that the model predictions model too closely the training data and do not generalise well to unseen data. This requirement, however implies that not all the available dataset is used in the training of the model, as a portion must be reserved for evaluation. This is clearly not ideal.

N-Fold cross-validation is an algorithm that enables all the available data to be used in the training of the model. N-fold cross validation is performed by partitioning the dataset set into N folds. Each fold is then used in turn as a test set, while the rest of the dataset is used as a training set. At every iteration, a different fold is used as the test set and evaluation criteria are calculated. At the end of the N iterations the mean and standard deviation of the given evaluation criterion can be determined from the results of the N iterations. This process reduces variability in the result and also ensures that all the available data contributes to evaluation of the model. This would not be the case if the available dataset had to be simply split into test set and training set and the model evaluation performed only once as explained above.

In all our machine learning experiments, we have adhered to a common dataset handling and cross validation scheme.For every machine learning experiment that we carried out, 20% of the molecules were loaded as a hold-out set and the rest were loaded in "`SEPARATE`" mode as a validation set. Using the validation set, we carried out 5-fold cross validation in order to determine the optimum hyperparameter values for the machine learning algorithm being considered in the given experiment.

Once the optimum hyper-parameters are determined, the 80% of the dataset not allocated to the hold-out set is reloaded in "`LUMPED`" mode and used to train a final version of the model using the optimum hyper-parameters. This model is then evaluated against the hold-out set and a final set of performance values is calculated and saved. An activity diagram illustrating this process is shown in Figure 3.7.

The decision to implement a 20/80 split in our datasets was an arbitrary one, which, however is a common one used in machine learning model training scenarios (Alqahtani and Whyte, 2016; Shahin et al., 2004). Training/validation dataset split can be a sensitive parameter when dealing with small datasets because high variance can result if the training dataset is too small with not enough examples to train a generalised

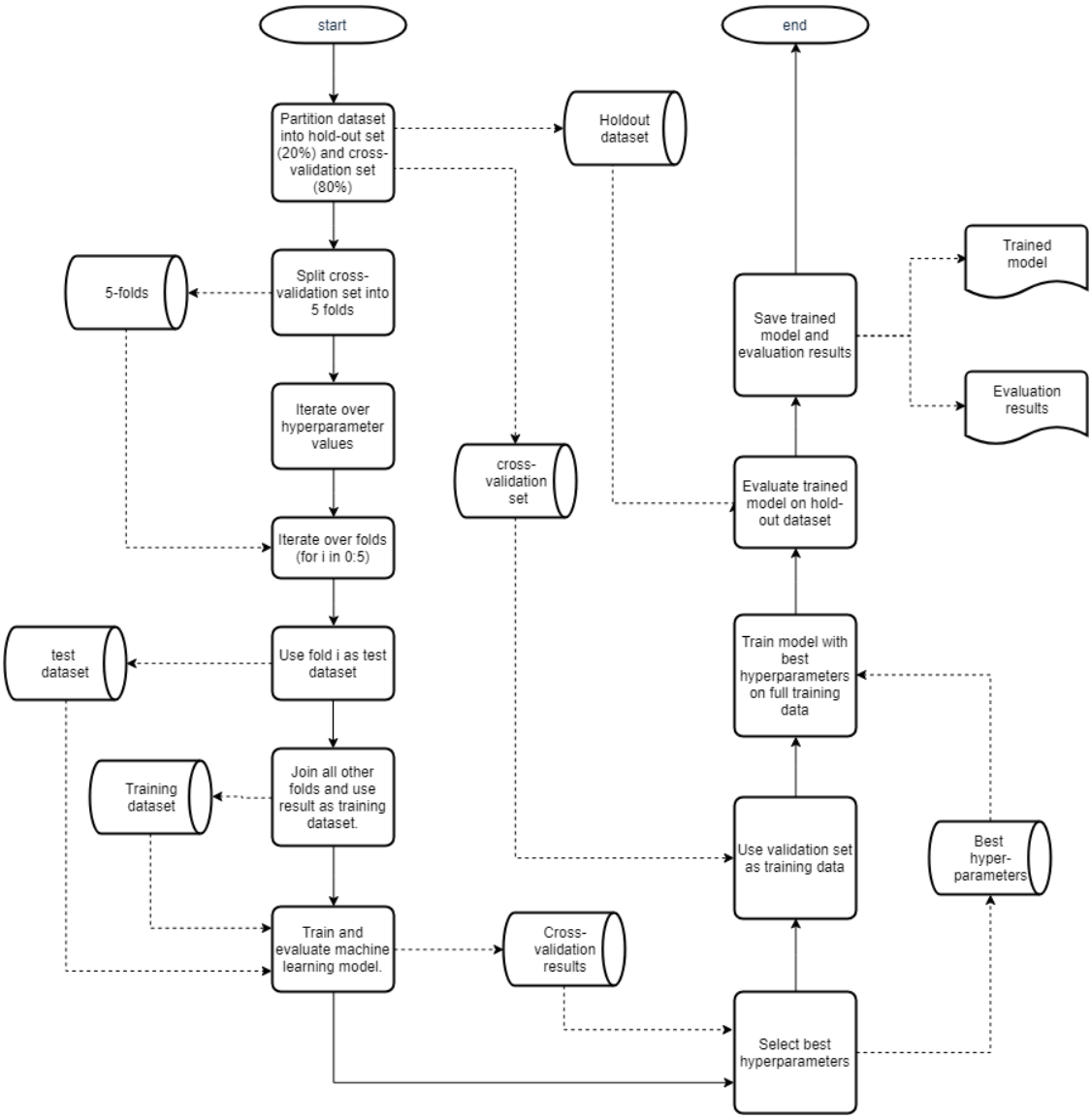

Figure 3.7: Flowchart illustrating the pipeline we used to train and evaluate models in our machine learning experiments.

model. In our case, however, our datasets are large and therefore performance will not be as sensitive to the split ratio.

As discussed in previous chapters, we have chosen three machine learning models to explore within this research project - Gaussian Mixture Models, Artificial Neural Networks and Isolation Forest. In the following subsections the implementation details for each of these algorithms are discussed.

3.6.1 | Gaussian Mixture Models

In our GMM implementation, we have used the `sklearn.mixture.GaussianMixture` class in the Python Scikit-learn library. The only hyperparameter that was tuned for the purposes of this research was the number of components, which was varied between the values of 1, 10, 50, 100 and 1000. The covariance type was always set to "full" as this is known to yield the best fit to the input data, albeit at the cost of some performance.

The GMM is trained exclusively on active molecule conformers. The candidate molecules are then ranked using the *score()* method provided by the `GaussianMixture` class. The score method takes a set of input records and returns their average log-likelihood with respect to the trained GMM. This is a measure of how well the GMM represents the input data and so can be taken as a measure of similarity.

The candidate molecules are finally ordered by their score, giving a similarity-based ranking that is then used to calculate the Enrichment Factor as well as generate similarity-based and rank-based ROC curves, all of which are saved to file.

As a further experiment with GMMs we implemented a scheme inspired by Jahn et al. (2010) and Jahn et al. (2011). Here, the authors used GMMs to encode bond rotation in molecules so as to generate a probabilistic model of a molecule's conformational space, i.e. use GMMs to describe all the ways a molecule's shape might change taking into account all its rotatable bonds. Evaluating the similarity of two molecules would then involve computing the overlap of the Gaussian components making up the GMM representing each molecule, obviating the need to generate and evaluate separate conformers.

In our implementation we attempted to use a similar idea - we encoded each molecule as a GMM generated from its conformer descriptors. We then computed the similarity between two molecules as the overlap of the Gaussian components making up their respective GMMs. We experimented with several GMM overlap metrics such as:

- Bhattacharyya affinity (Jebara et al., 2004)
- Expected Likelihood (Jebara et al., 2004)

- Normalized Matching Probability (Zhou et al., 2017)

Preliminary tests, however showed that results obtained using this scheme were not satisfactory and this avenue of research was put on hold in favour of other approaches that proved to be more promising.

3.6.2 | Artificial Neural Networks

In order to explore the performance of ANNs in the context of our research, we have built a Keras[4] v.2.2.4 with a Tensorflow v.1.12.2 backend to implement a simple ANN. We configured the input layer of the ANN to consist of one neuron per descriptor vector element, i.e. 12 neurons for USR, 15 neurons for ES4D and 18 for ES5D. We also configured a single hidden 100-neuron layer using the *relu* activation function as well as a linear single-neuron output.

The ANN was trained using both active and decoy data. The labels used during training are a value of 100 for conformers of active molecules and 0 for conformers of decoys. This is to allow the network to generate a variety of output values between 0 and 100 which are then used as similarity scores.

During evaluation, we compute a similarity score for every conformer in every search molecule and we take the maximum similarity score per molecule as the overall similarity score for that search molecule.

Again, in common with the other machine learning methods, we order the molecules by similarity score and we generate the Enrichment Factor and ROC curves.

3.6.3 | Isolation Forest

In our evaluation of the Isolation Forest algorithm we used the `sklearn.ensemble.IsolationForest` class in the Scikit-learn library to implement an Isolation Forest-based classifier on the conformer datasets, tuning the number of estimators hyperparameter for the Isolation Forest model between the values of 500, 100 and 10 during the cross-validation process.

The Scikit-learn implementation of Isolation Forest exposes a `decision_function()` method which, given one or more input records, returns an average anomaly score for each record, computed over the base classifiers of the trained Isolation Forest model. Conveniently, this function returns a positive score for inliers and a negative score for outliers, therefore the average anomaly score can also be used to rank candidate

[4] http://keras.io [last accessed 7th May 2019]

molecules by similarity with respect to the active molecules on which the model was trained.

Using this function, we compute the average anomaly score for every conformer in a search molecule and we select the maximum score as the overall score for the search molecule. We then sort the search molecules by similarity score and we calculate the Enrichment Factor and ROC Curves.

3.6.4 | Implementation Details

In this sub-section we detail the software packages that we used during the course of the research project as well as the hardware on which we generated our results. Each step in our process pipeline presented different requirements, therefore we have listed each process' details in a separate sub-section.

3.6.4.1 | Conformer and Descriptor Generation

The conformer generation and descriptor generation processes were combined into a single Python 3.6 script which was run on a spark cluster of 3 Amazon Web Services (AWS) compute-optimised c5.2xlarge instances having 8 cores and 16GB each.

The conformer generation process makes use of RDKit 2008.09.1 and was run on Spark 2.3.0. The descriptor generation process uses no external dependencies.

Subsequently during the project we transitioned to a single server with a 64 core Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz cpu with 125Gb of RAM.

3.6.4.2 | USR/ElectroShape-4D/ElectroShape-5D Implementation

Our implementation of the USR family of algorithms was run on a Spark v.2.3.0 cluster made up of of 3 AWS r5.2xlarge instances with 8 cores and 64 GB of memory each.

Subsequently during the project we transitioned to a single server with a 64 core Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz cpu with 125Gb of RAM.

3.6.4.3 | Gaussian Mixture Model and Isolation Forest Evaluations

For both our GMM and Isolation Forest evaluations, we made use of the implementations provided by the Scikit-learn v.0.19.1 library. They were both trained on a server having twin Intel Xeon E5-2660 v.4 CPUs, giving a total of 28 physical cores and 56 threads, and 128Gb of RAM.

Subsequently during the project we transitioned to a single server with a 64 core Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz cpu with 125Gb of RAM.

3.6.4.4 | Artificial Nerual Network Evaluation

The ANN evaluation code is based on Keras v.2.2.4 with a Tensorflow v.1.12.2 backend. The ANN was implemented in Python and was trained on an Amazon EC2 p2.xlarge instance with 4 cores, 61GB of memory and a single NVIDIA K80 GPU with 2,486 parallel processing cores and 12GB of on-board memory.

Subsequently during the project we transitioned to a single server with a 64 core Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz cpu with 125Gb of RAM.

3.7 | Summary

In this chapter we have presented a detailed description of the code that was required to meet the objectives of this research project. We have presented an overview of the process pipeline which allowed us to generate the results required to answer the research questions posed in the Introduction chapter.

Additionally, we have also detailed our method of conformer and USR descriptor generation which is the foundation of the USR family of methods. Following this, we have described our implementation of several methods in the USR family which we used to generate a baseline performance measure against which we could compare the performance of our machine learning models.

We have described briefly our custom dataset loading and partitioning code, which was necessary to implement as the out of the box functions provided with most Python libraries were not suited to handling the conformer data efficiently.

Finally we have described our machine learning training and evaluation pipeline along with the implementations specific to the three machine learning Models that we selected to explore within the scope of the project.

In the following chapter, we present the results yielded by our experiments which we evaluate and examine in later chapters.

4

Results & Evaluation

In accordance with the aims and objectives stated in Section 1.4, we have implemented several experiments based on the datasets in DUD-E as follows:

- Baseline implementations of USR and ES5D and retrospective screening on the 38 targets chosen from DUD-E using full conformer models for the molecule datasets. These targets were chosen because much previous work done on LBVS is based on the DUD database. In particular Armstrong et al. (2010, 2011) evaluate their results basing themselves on the compounds provided in DUD.
- USR and ES5D retrospective screening on the same targets using only the Lowest Energy Conformers from the active templates instead of the full conformer model. This is suggested in Ballester et al. (2009a).
- **Gaussian Mixture Model** retrospective screening based on ES5D descriptors.
- **Isolation Forest** retrospective screening based on ES5D descriptors.
- **Neural Network** retrospective screening based on ES5D descriptors.

GMMs and Isolation Forests can be trained exclusively using positive examples (active compounds in our case), thus mimicking the traditional virtual screening process. Our Neural Networks, on the other hand, were trained in a standard supervised manner, using both actives and decoys as training data. This is to be enable comparison of the performance given by both types of approaches.

Due to time and resource availability, we chose to forego performing experiments using CSR and ElectroShape-4D because previous work shows that the performance of ES5D matches or improves on both (Armstrong et al., 2011). For similar reasons the ML-based experiments were performed on ES5D descriptors.

4.1 | Benchmark USR Results

The first step in our research was to replicate the results of Ballester et al. (2009a) and Armstrong et al. (2011) by implementing the USR, ES5D algorithms and using them to run retrospective screening experiments on the DUD-E targets. This allowed us to compare the performance obtained by ES5D compared to plain USR and verify the results of Armstrong et al. (2011). These experiments were done using the full conformer model for all the actives and repeated using only the active LECs. The resulting comparative performance in terms of Enrichment Factor as well as AUC can be seen in Figures 4.1 and 4.2.

As can be seen from Figure 4.1, the enrichment at 1% is significantly better that that at 5%. This is an expected outcome because, as pointed out in Ballester (2011), USR is most effective at the high similarity range and therefore naturally obtains better enrichment at lower threshold values corresponding to higher similarity.

The performance in terms of AUC, shown in Figure 4.2 also shows that on average ES5D outperforms plain USR except for 11 targets.

Figures 4.3 and 4.4 show the corresponding plots for the LEC-based implementations. As expected, the enrichment factor obtained by the experiments using the LECs is lower than those using the full conformer model. The AUC is also lower for the LEC experiments. Interestingly, however, the performance hit seems to be lower for ES5D than for USR. This is likely due to the fact that the ElectroShape methods use atomic properties that are not exclusively spatial and are therefore less affected by conformer configuration.

A comparison of the enrichment factor at 1% of USR and ES5D is shown in Figures 4.6 and 4.7 for full conformers and LEC respectively. It can be seen from these figures that the ES5D improves the enrichment over USR in all targets to varying degrees, up to a maximum improvement of almost 8 times that obtained by USR. It is interesting to note that the improvement of ES5D over USR is similar for both full conformers as well as LEC.

The evaluation of USR in Ballester et al. (2009a) did not make use of DUD and only involved 8 targets, therefore a comprehensive performance comparison to our implementation is not possible. In Armstrong et al. (2011), however, the entire DUD is used, therefore a comparison is more meaningful.

In order to facilitate comparison of the performance of our implementation of ES5D to that in Armstrong et al. (2011), we have reproduced a visualisation of the Enrichment Factor performance obtained therein in Figure 4.5

Unfortunately, it is not possible to directly compare the enrichment factor across

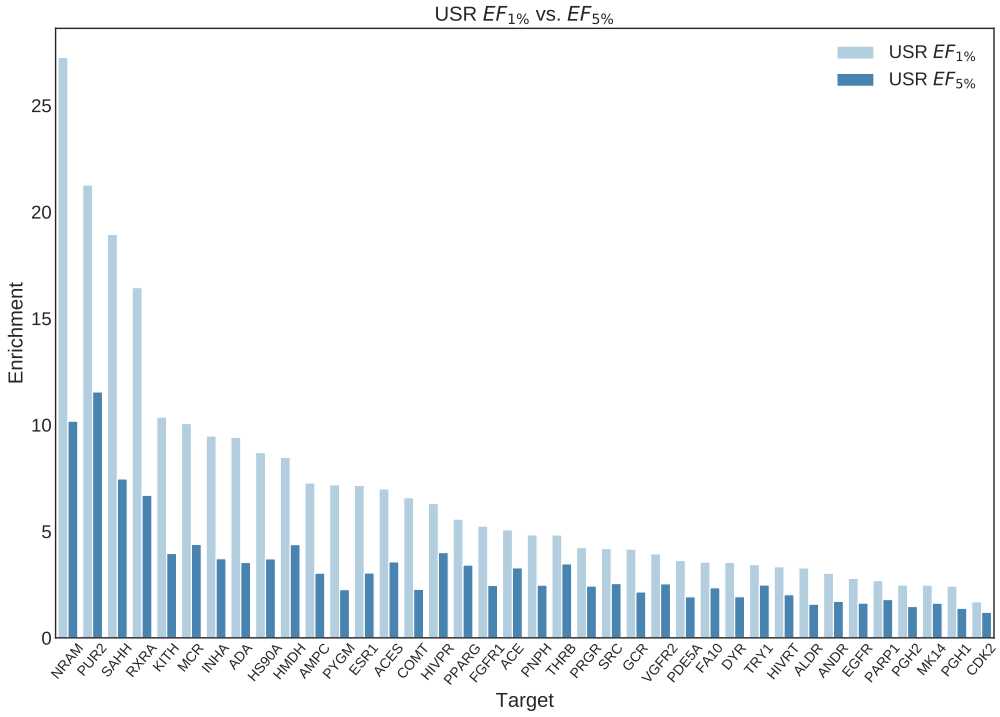

Figure 4.1: USR comparative performance of Enrichment Factor at 1% and 5% obtained from retrospective screening using full conformer models.

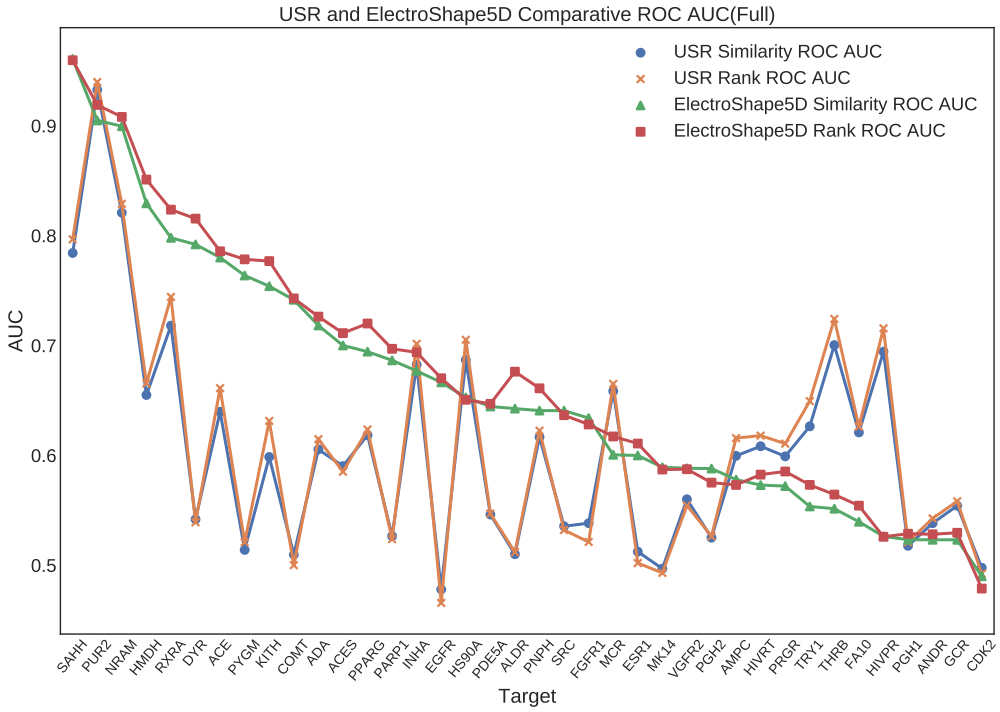

Figure 4.2: USR and ES5D comparative AUC performance obtained from retrospective screening using full conformer model.

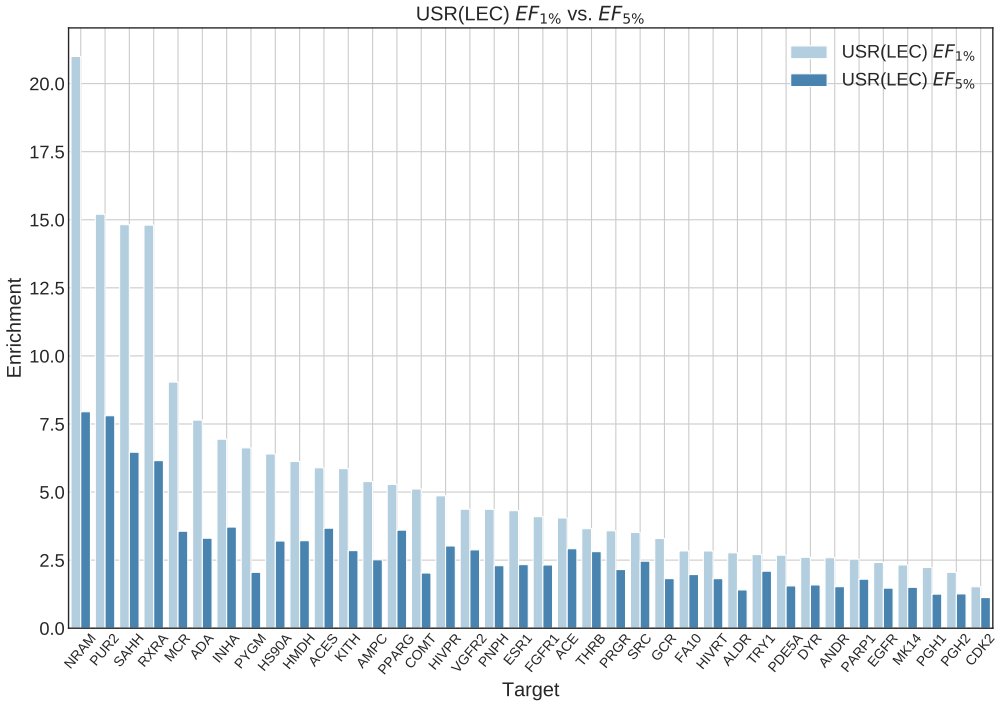

Figure 4.3: USR comparative performance of Enrichment Factor at 1% and 5% obtained using Lowest Energy Conformers.

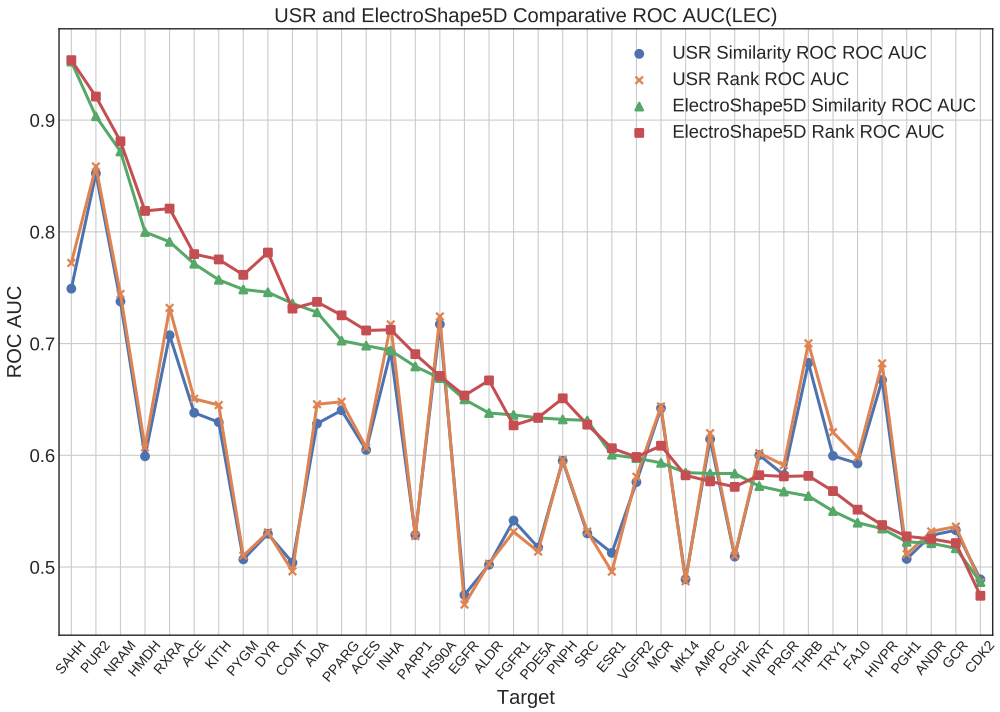

Figure 4.4: USR and ES5D comparative AUC performance using Lowest Energy Conformers.

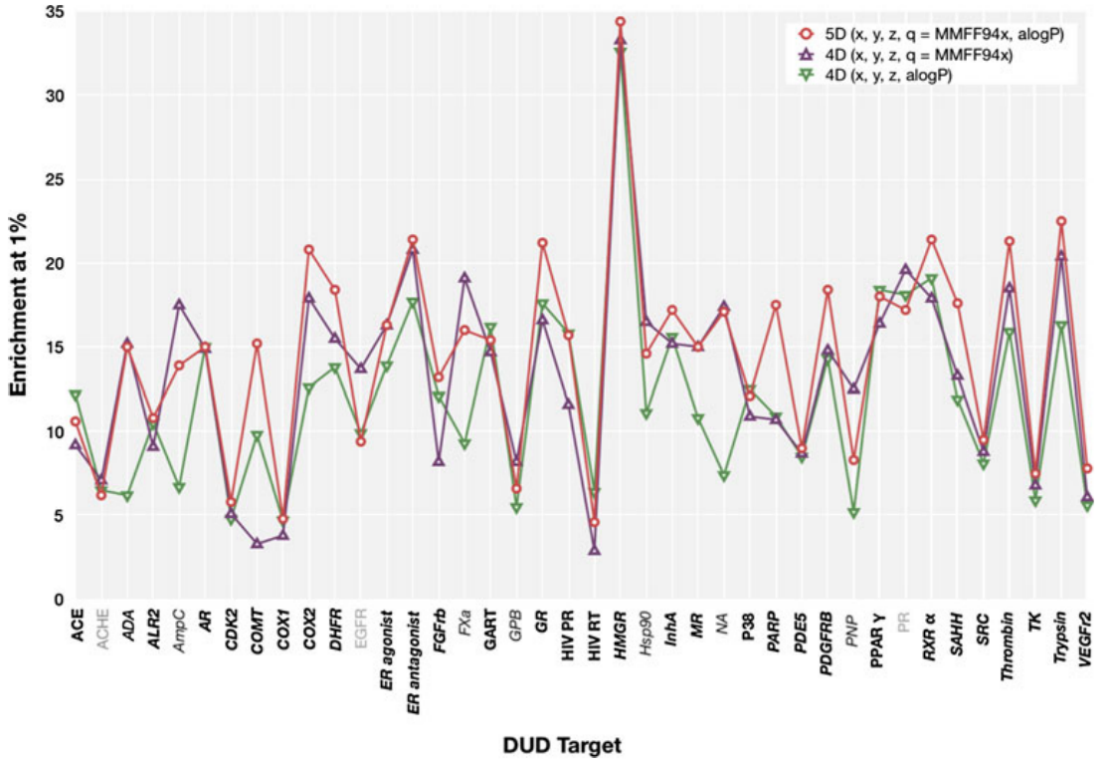

Figure 4.5: $EF_{1\%}$ of ES5D calculated on the DUD database as reported in Armstrong et al. (2011). Reproduced from Armstrong et al. (2011)

datasets, possibly having different selection criteria for decoys and well as different ratios of actives to decoys. This is because the maximum possible enrichment factor at the top n% of a dataset of which x% are actives is $min(100/n, 100/x)$. This means that the enrichment factor depends on the ratio of actives to decoys in the given dataset. Given that DUD and DUD-E have different active to decoy ratios, the comparison is problematic. Apart from this, as pointed out in Ballester et al. (2009a), the Conformers Per Molecule (CPM) can also have an effect on the enrichment achieved.

We can observe, however, that in general, targets in Figure 4.5 (marked "5D(x, y, z, q = MMFF94x, alogP)") from the original paper show a similar trend to those generated by us in Figure 4.7, i.e. most targets that show a high enrichment in one also show a high enrichment in the other and vice versa, but not all. The Pearson product-moment correlation coefficient for the two sets of data is 0.35, indicating a mild positive correlation. Given the differences in decoy selection in DUD-E in comparison with DUD Mysinger et al. (2012), it is not surprising that our results differ somewhat from those obtained by Armstrong. This relatively low, albeit positive, correlation coefficient, indicates that

differences in dataset selection can have a significant impact on virtual screening results.

In order to illustrate better the relative performance impact of using the LEC instead of the full conformer model, in Figure 4.8 we have plotted the $EF_{1\%}$ of ES5D using LECs vs. using full conformers.

It can be seen from this plot that for most targets, using LECs instead of full conformers only yields a small performance penalty, with the maximum being approximately 32% decrease, but with an average performance degradation of 9.4%.

4.2 | Machine Learning Results and Comparison with USR

In this section we present the results we obtained in our retrospective screening experiments using machine learning techniques and contrast them with the non-machine learning results presented in the previous section. Since ES5D consistently performs better than USR, we have used ES5D descriptors as training data for our machine learning models in an attempt to obtain the best possible scores.

In all our machine learning experiments, we have trained two versions of each type of model - one using full conformer models and one using only LECs so as to be able to compare their performance. It is clear that using exclusively the LECs to train our models requires much less memory and processing power because the amount of data involved in the process is much less (on average 61 times smaller) than that for using full conformers. As will be demonstrated in the following sections, we have managed to obtain significantly better performance using only the LECs compared to standard ES5D and this is in itself a significant result in line with our second research question (see Section 1.4).

4.2.1 | Gaussian Mixture Models

The performance for our GMM models in terms of EF can be seen in Figures 4.9 and 4.10 for full conformers and LECs respectively while that in terms of AUC is shown in Figures 4.11 and 4.12.

As can be seen from the above figures, while the use of the full conformer model to train the GMM yields better results than training with LECs, the LEC results still outperform ES5D significantly, with a mean improvement of 290% and a maximum of 829% in terns of $EF_{1\%}$ as well as 133% mean improvement and 171% maximum improvement in terms of AUC.

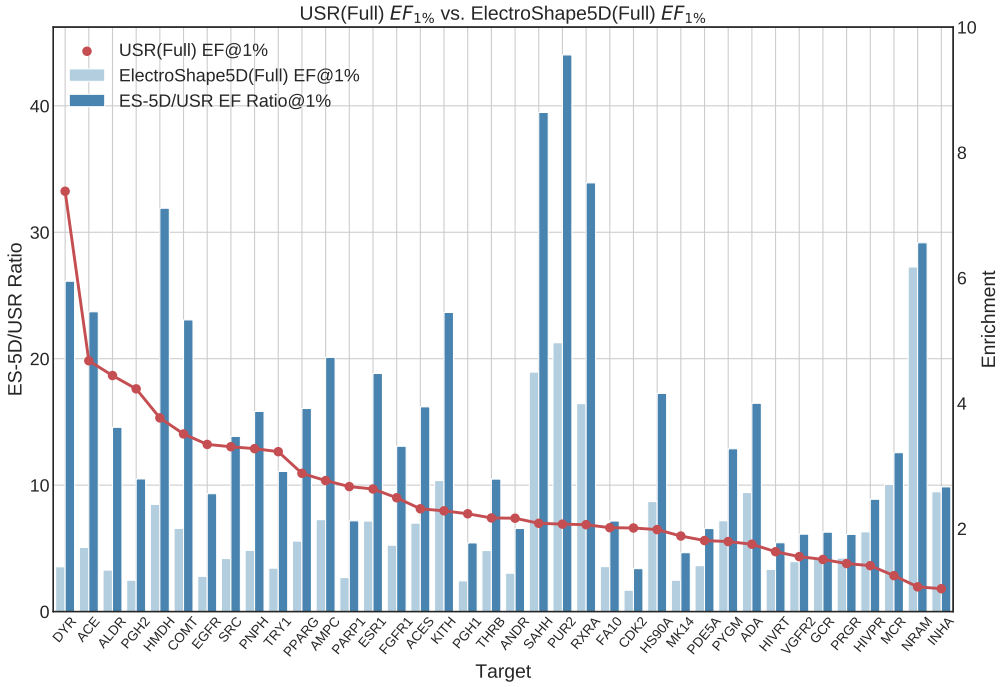

Figure 4.6: Comparison of USR enrichment at 1% and ES5D enrichment at 1% (full conformers) along with the ratio between them. Mean ratio=2.529±1.219, max=7.381, min=1.041

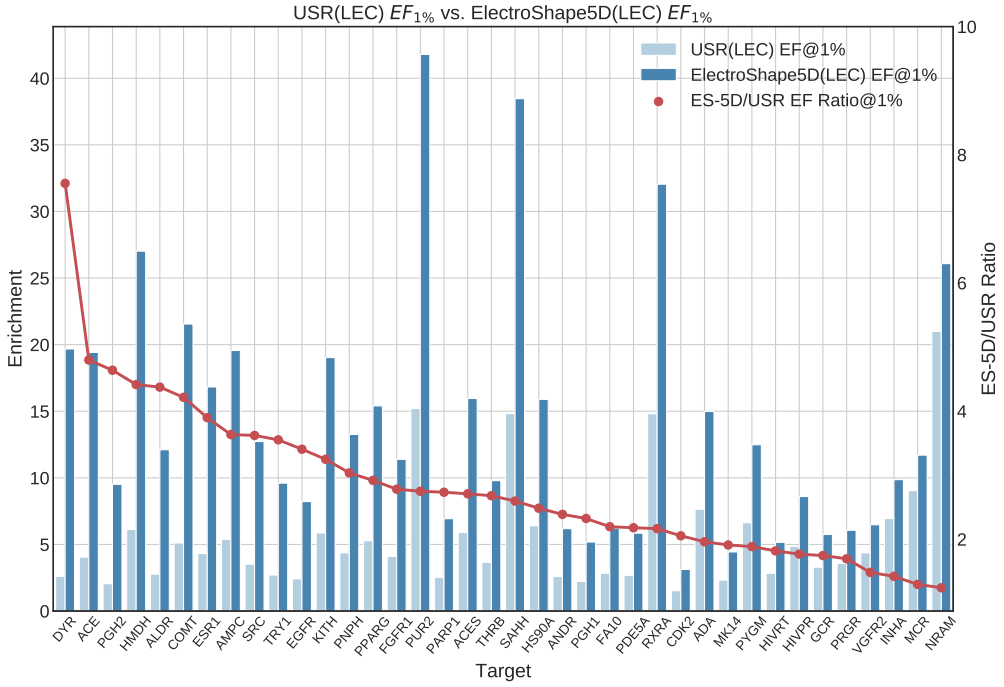

Figure 4.7: Comparison of USR enrichment at 1% and ES5D enrichment at 1% (LEC) along with the ratio between them. Mean ratio=2.830±1.253, max=7.553, min=1.242

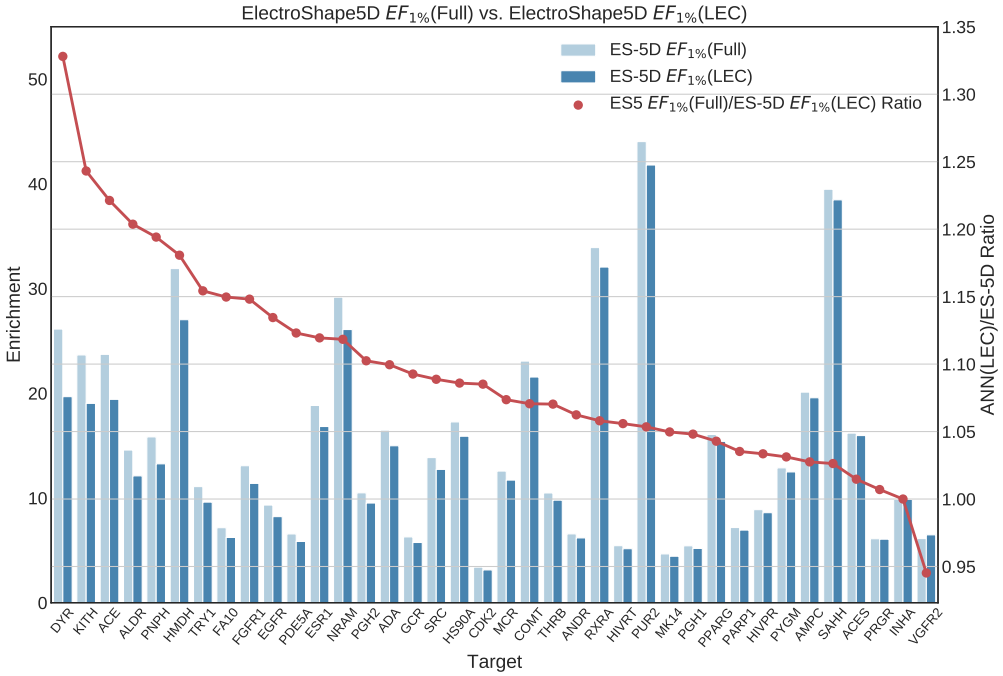

Figure 4.8: Comparison of ES5D enrichment @ 1% using full conformer model vs. using LECs. Mean performance degradation: 9.4%

Using full conformers, on the other hand, we obtained a mean improvement of 432% over ES5D in terms of $EF_{1\%}$ with a maximum improvement of 941%. In terms of AUC, the mean improvement is 138% and maximum improvement 173%.

4.2.2 | Isolation Forest

The $EF_{1\%}$ performance for our Isolation Forest models can be seen in Figures 4.13 and 4.14 for full conformers and LECs respectively while that in terms of AUC is shown in Figures 4.15 and 4.16.

Similarly to the results observed for GMMs, we get better performance using full conformers than using LECs. For most targets, the LEC performance improves on ES5D, giving a mean improvement of 190% and a maximum improvement of 460% in terms of EF as well as 123% mean and 152% maximum improvement in terms of AUC.

The use of full conformers obtains a mean improvement of 211% over ES5D in terms of $EF_{1\%}$ with a maximum improvement of 403%. In terms of AUC, the mean improvement is 123% and maximum improvement 152%.

Isolation Forest did not register performance increases across all targets however.

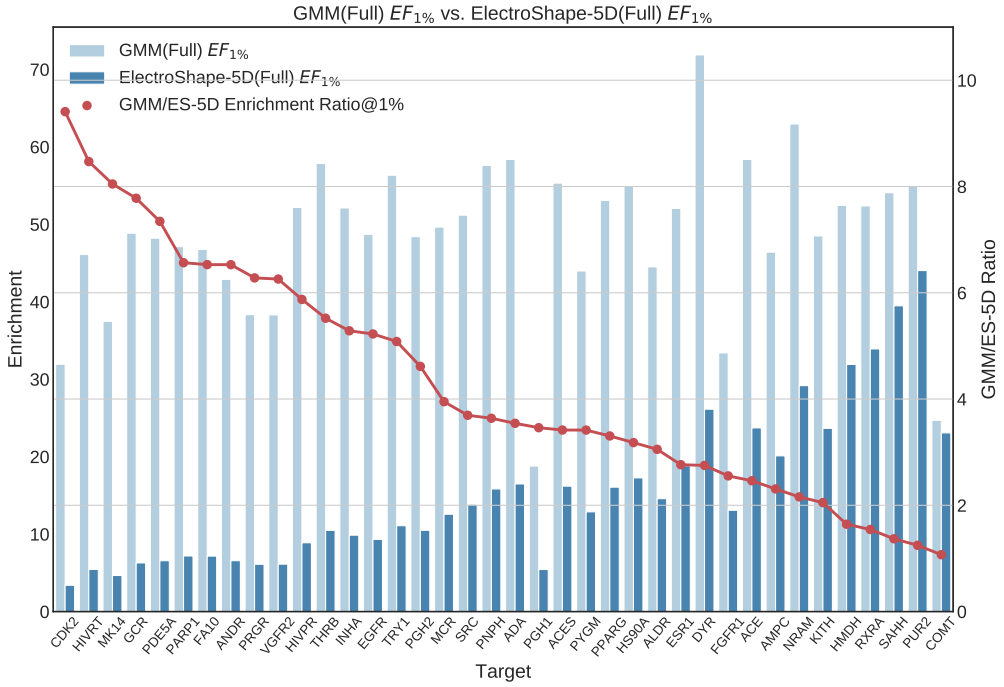

Figure 4.9: Comparative $EF_{1\%}$ of GMM vs.ES5D using full conformer model. Mean improvement 432%. Maximum improvement 941%

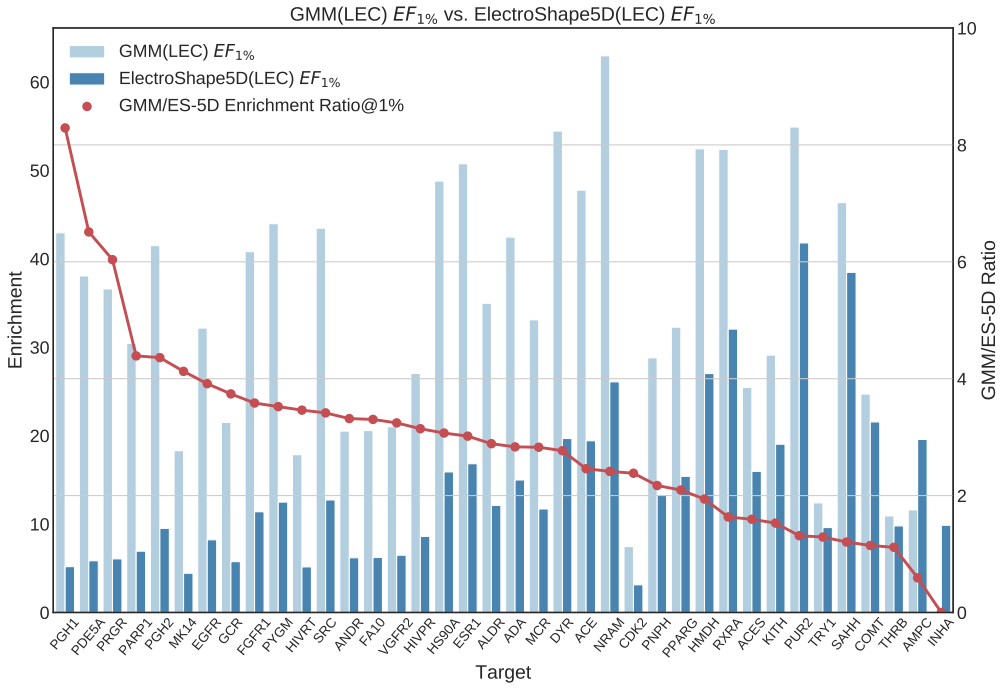

Figure 4.10: Comparative $EF_{1\%}$ of GMM vs.ES5D using Lowest Energy Conformers. Mean improvement 290%. Maximum improvement 829%.

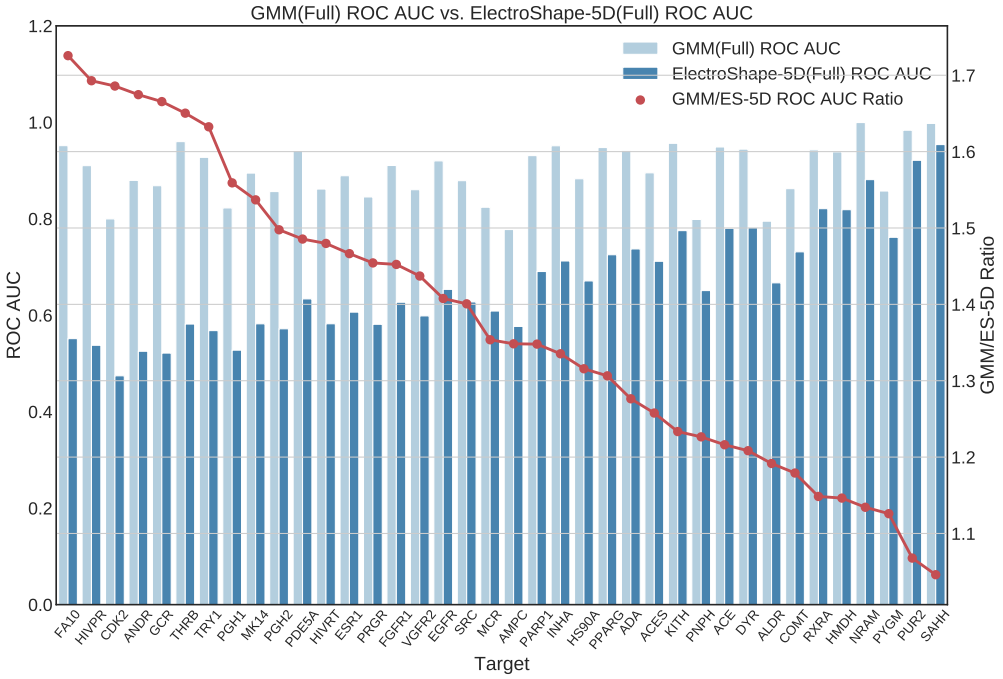

Figure 4.11: Comparative AUC of GMM vs.ES5D using full conformer model. Mean improvement 138%. Maximum improvement 173%

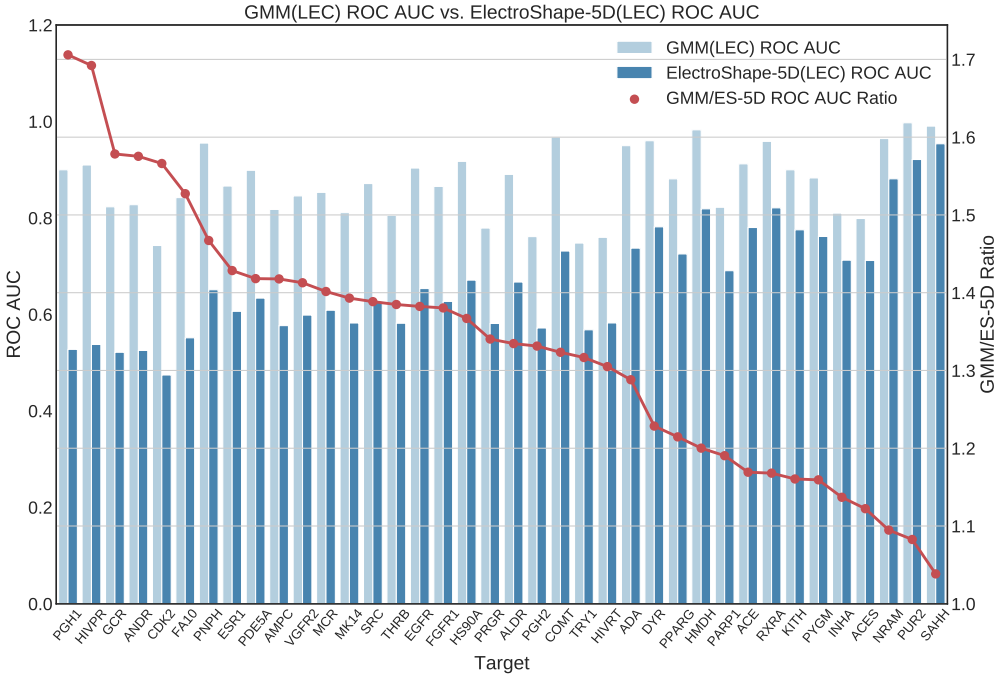

Figure 4.12: Comparative AUC of GMM vs ES5D using full conformer model. Mean improvement 133%. Maximum improvement 171%.

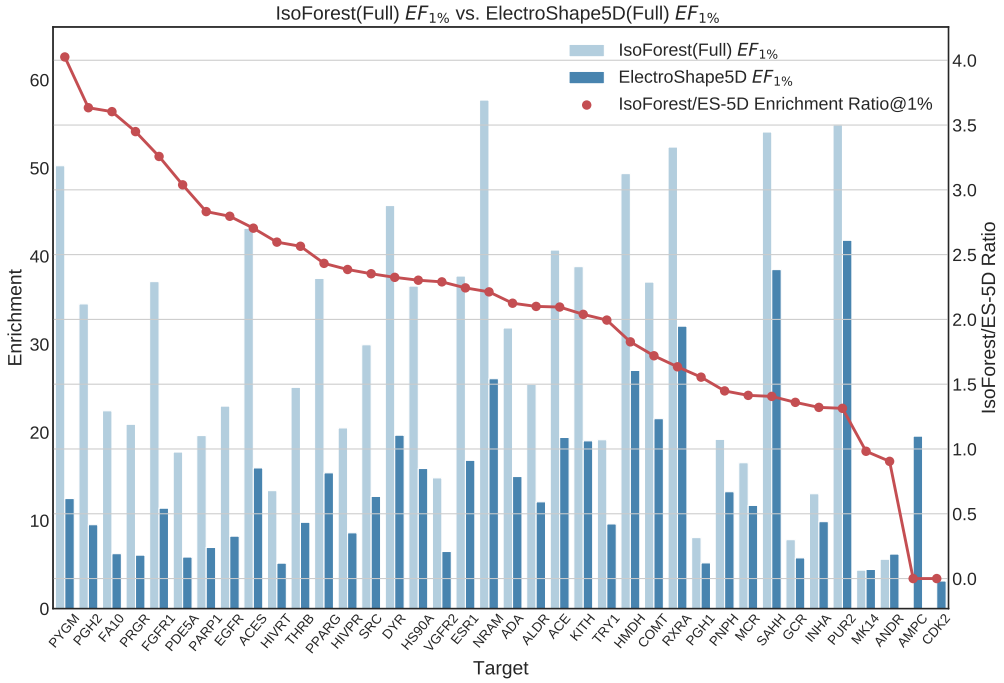

Figure 4.13: Comparative $EF_{1\%}$ of IsoForest vs.ES5D using full conformer model. Mean improvement 211%. Maximum improvement 403%

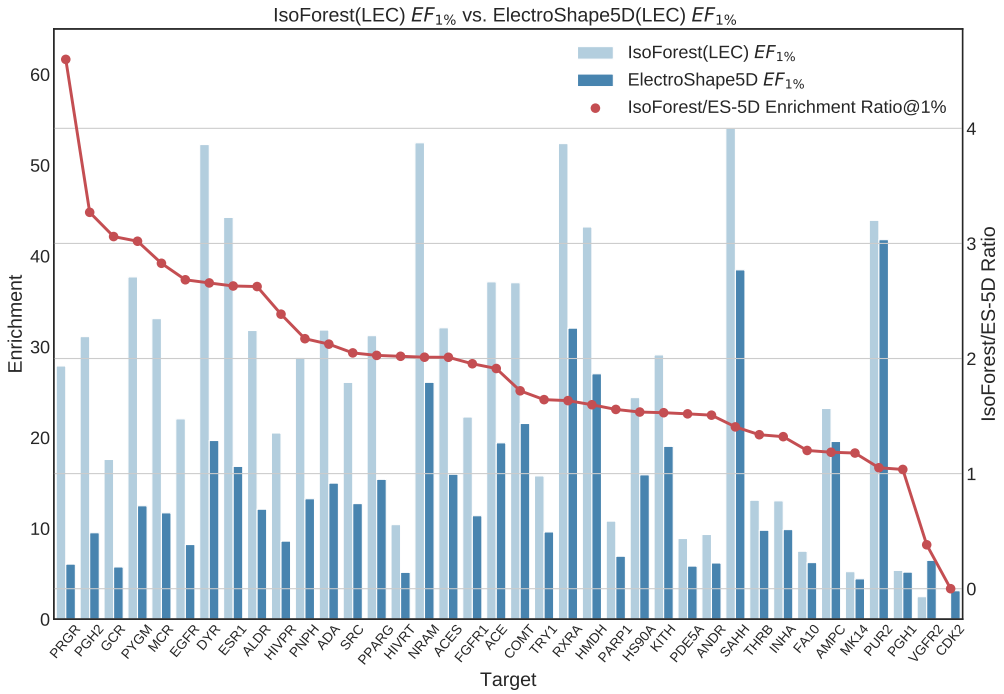

Figure 4.14: Comparative $EF_{1\%}$ of IsoForest vs.ES5D using full conformer model. Mean improvement 190%. Maximum improvement 460%

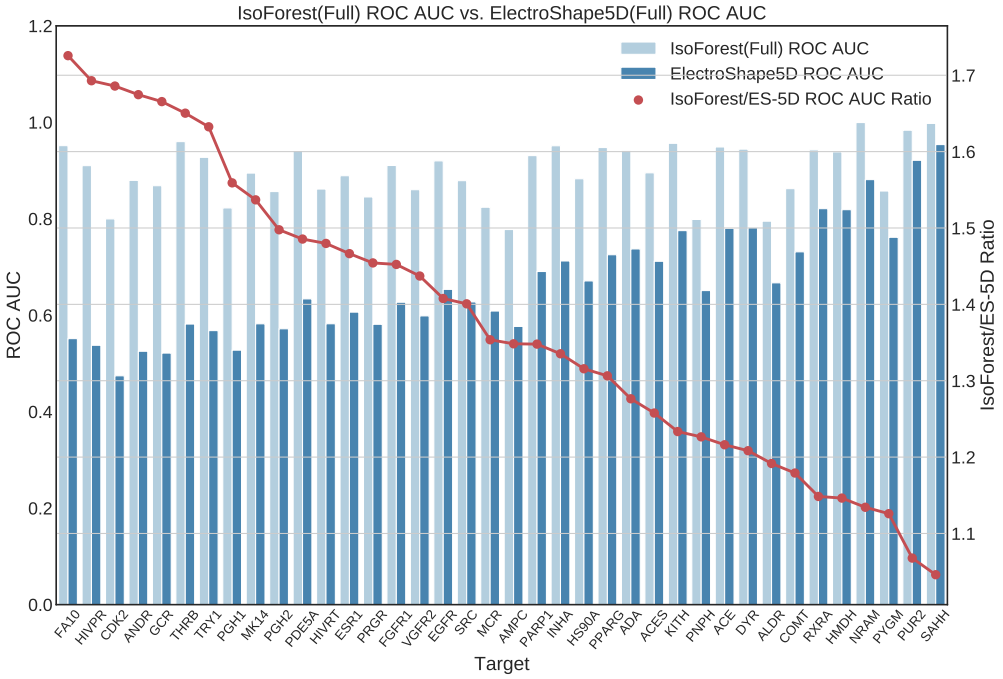

Figure 4.15: Comparative AUC of IsoForest vs.ES5D using full conformer models. Mean improvement 123%. Maximum improvement 152%.

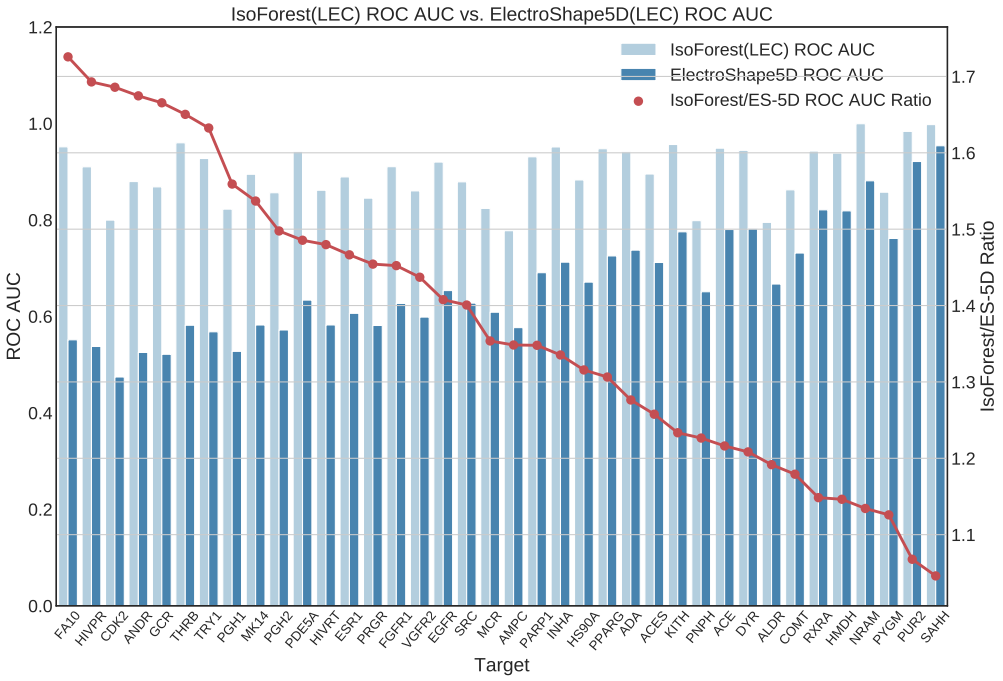

Figure 4.16: Comparative AUC of IsoForest vs ES5D using full conformer model. Mean improvement 126%. Maximum improvement 155%

Cdk2 scored 0 $EF_{1\%}$ for both full conformers as well as LECs, while **ampc** scored 0 only on the full conformers model and only registering a modest improvement when trained on LECs. Several other targets registered a decrease in EF performance compared to ES5D. In terms of AUC, however, all targets performed better for both full conformer as well as LEC models. This indicates that the early enrichment characteristics of Isolation Forest may not be ideal for the task of virtual screening. Examining, for example, the ROC curve for **cdk2**, seen in Figures 4.17 and 4.18, we can see that indeed, the first part of the curve dips below the 0.5 diagonal, indicating poor early enrichment. Indeed the ROC curve for the full conformer model is significantly worse than that for the LEC model. Both, however score 0 at the 1% level enrichment.

4.2.3 | Artificial Neural Networks

As for GMMs and Isolation Forests, we trained two versions of ANN - full conformer versions and LEC versions. As expected, the major advantage of using LECs for training data is that a much smaller volume of data is used to train the model resulting in shorter training time requirements. We expected to see a performance drop in the LEC model with respect to the full conformer-trained model, as for the other models, however, training both using a hidden layer size of 100 nodes, this did not materialise and the performance obtained for the LEC-trained model was virtually the same as that for full-conformer-trained models, for the same hidden layer size (2.558 ± 1.067 vs. 2.563 ± 1.292).

Upon increasing the hidden layer size to 500 nodes, this situation did not change appreciably (3.332 ± 1.289 vs. 3.276 ± 1.485). It is also interesting that the ANN performance did not surpass that of the full-conformer GMM. From these results, it seems that the ANN model does not cope with full conformer data as well as GMMs.

The $EF_{1\%}$ performance for our ANN models can be seen in Figures 4.19, 4.20, 4.21 for full conformers with hidden layer sizes of 500 and 100 nodes and LECs respectively while that in terms of AUC is shown in Figures 4.22, 4.23 and 4.24.

Nevertheless, even for the lower-performing, 100-node ANN models, for most targets, the ANN performance improves on ES5D. Both 100-node ANNs give a mean improvement of 256% and maximum improvements of 491% for LECs and 565% for full conformers, in terms of $EF_{1\%}$ as well as almost 140% mean and 175% maximum improvement in terms of AUC. The 500-node version of the full conformer model yields a mean improvement of 328% and a maximum of 613% while the LEC version obtains a mean of 333% with a maximum of 618%.

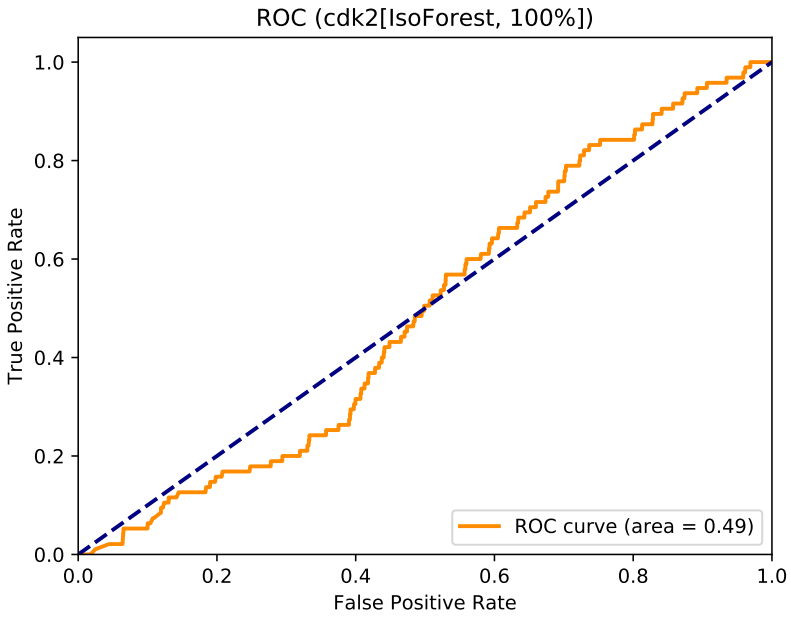

Figure 4.17: ROC Curve for target CDK2 for Isolation Forest trained with full conformers.

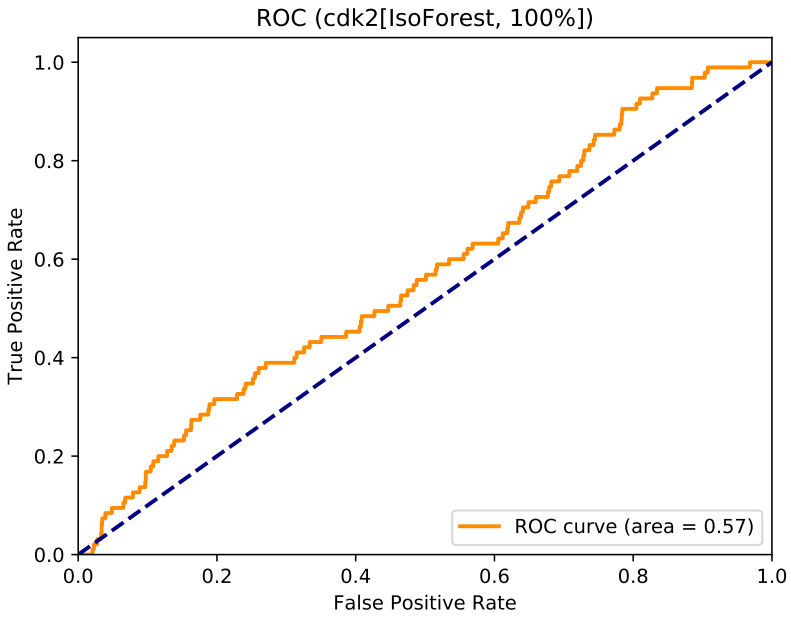

Figure 4.18: ROC Curve for target CDK2 for Isolation Forest trained with Lowest Energy Conformers.

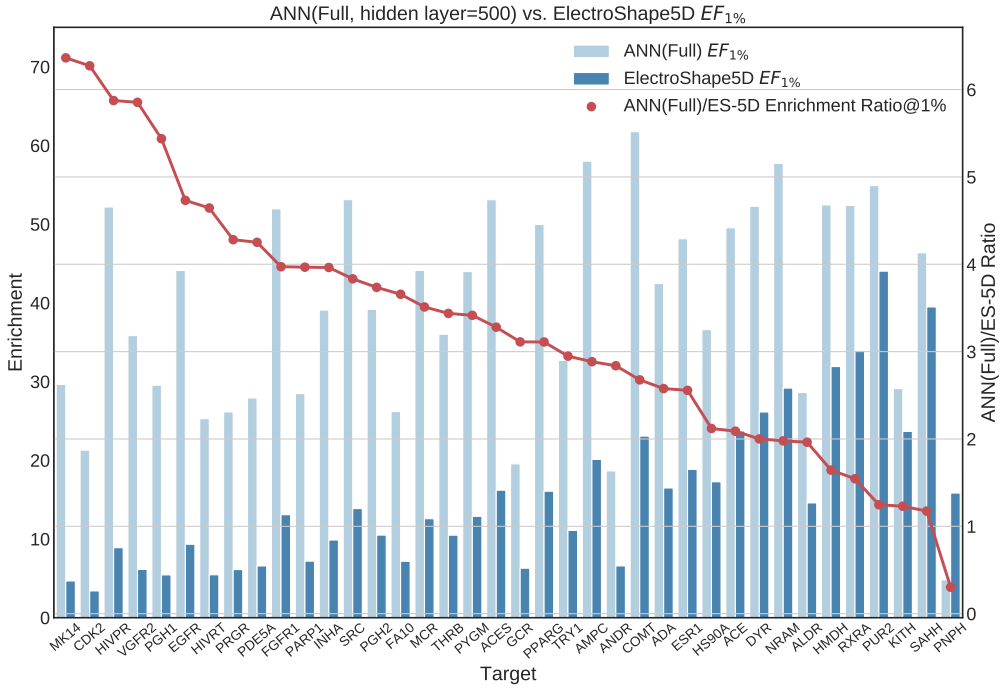

Figure 4.19: Comparative $EF_{1\%}$ of Neural Network vs. ES5D using full conformer model (hidden layer size=500). Mean improvement 328%. Maximum improvement 636%

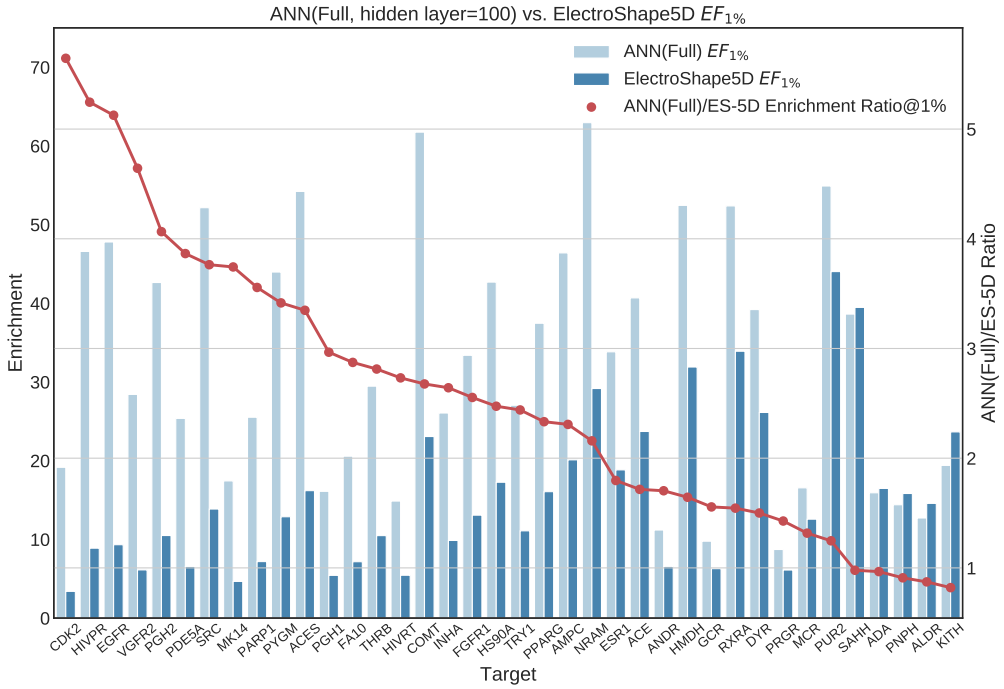

Figure 4.20: Comparative $EF_{1\%}$ of Neural Network vs. ES5D using full conformer model (hidden layer size=100). Mean improvement 256%. Maximum improvement 564%

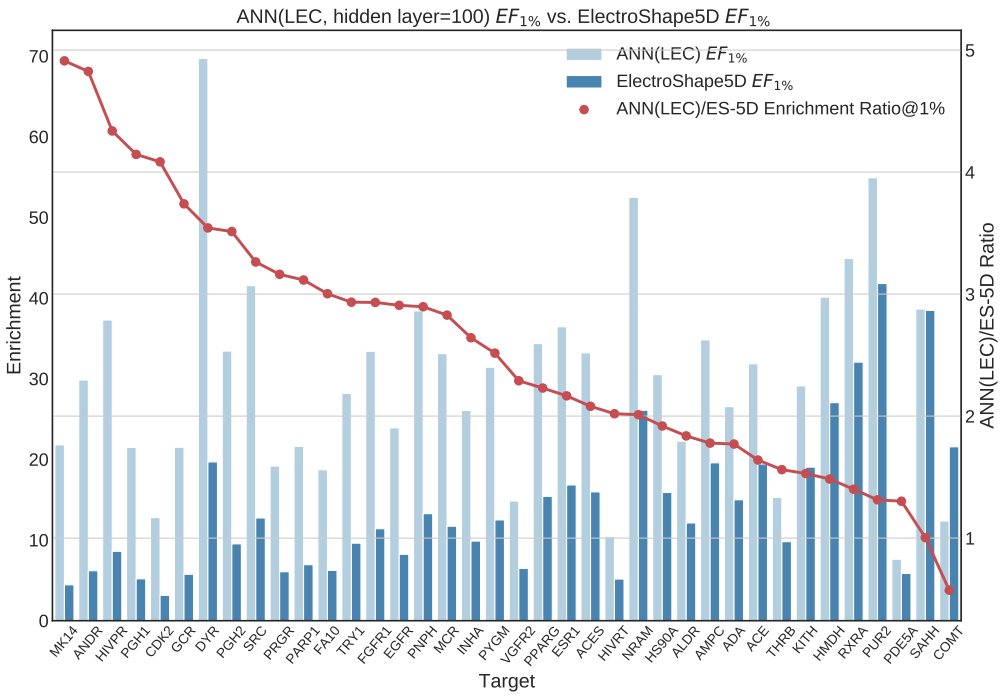

Figure 4.21: Comparative $EF_{1\%}$ of Neural Network vs. ES5D using LECs (hidden layer size=100). Mean improvement 256%. Maximum improvement 491%.

4.3 | Full Dataset Performance Summary

In order to facilitate the comparison of our results, we have summarised the basic statistics related to the enrichment factor at 1% and the AUC that we have obtained across the evaluated models. These can be seen in Table 4.1. These are shown in terms of improvement ratio of ES5D, such that a value of 1 indicates that the same performance as ES5D was obtained.

4.4 | Variation with Dataset Size

We have repeated our experiments for every machine learning algorithm multiple times using successively smaller portions of the available dataset so as to explore the manner in which the performance given by each model degrades with dataset size. This is in keeping with our second research question and fulfils objective 6 as described in Section 1.4, i.e. to run further experiments decreasing the number of known actives so as to understand how the performance of machine learning models trained on a reduced data set compares to standard USR. We have illustrated the respective performance change

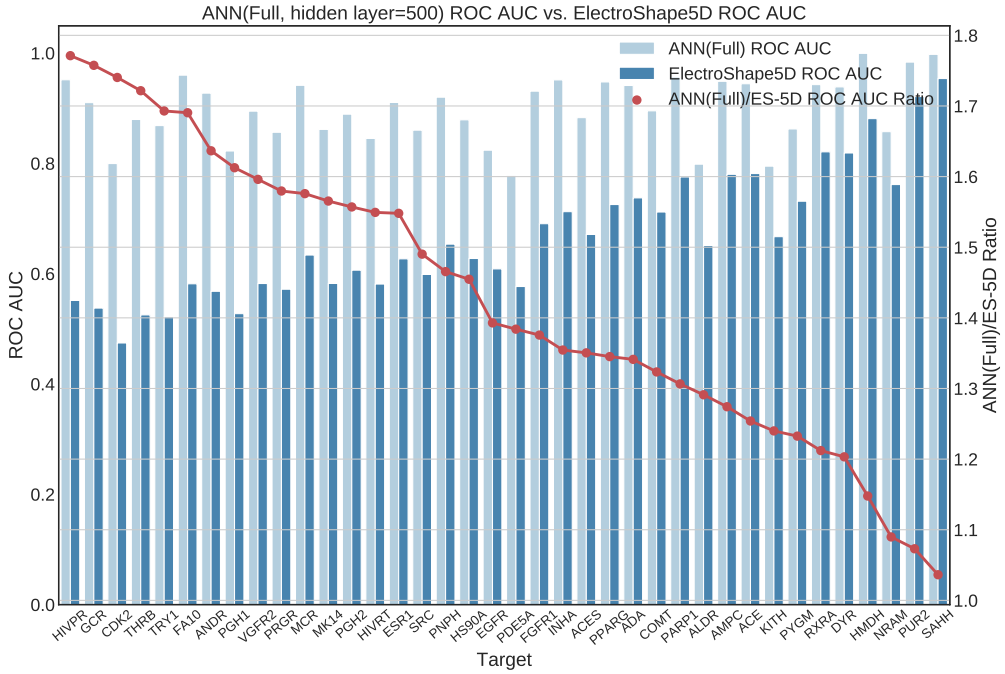

Figure 4.22: Comparative AUC of Neural Network vs. ES5D using full conformer model(hidden layer size=500). Mean improvement 143%. Maximum improvement 177%.

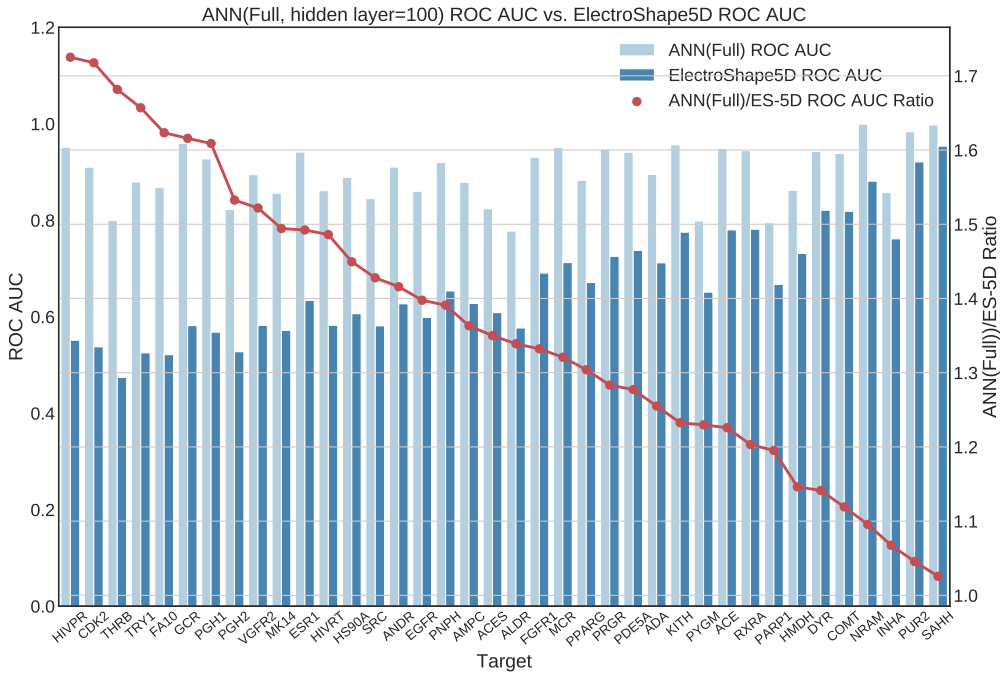

Figure 4.23: Comparative AUC of Neural Network vs. ES5D using full conformer model(hidden layer size=100). Mean improvement 136%. Maximum improvement 173%.

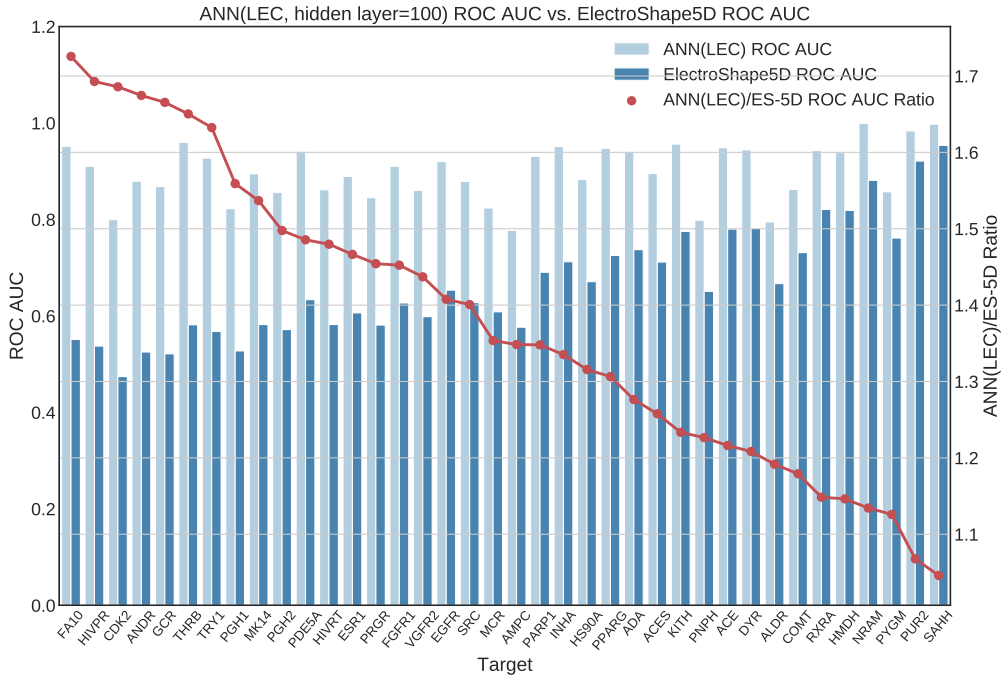

Figure 4.24: Comparative AUC of Neural Network vs. ES5D using LECs(hidden layer size=100). Mean improvement 139%. Maximum improvement 175%

LEC	GMM	IsoForest	ANN (Hidden layer=100)	ANN (Hidden layer=500)
Mean(±std)	2.912 (±1.623)	1.905(±0.839)	2.558(±1.067)	3.332(±1.289)
Max	8.290	4.599	4.912	6.180
Min	0.000	0.000	0.573	1.205
Mean AUC(±std)	1.334(±0.166)	1.258(±0.152)	1.394(±0.165)	1.443(±0.210)
Max AUC	1.706	1.545	1.752	1.785
Min AUC	1.038	0.992	1.045	1.046
Full Conformers				
Mean(±std)	4.300(±2.227)	2.113(±0.901)	2.563(±1.292)	3.276(±1.485)
Max	9.409	4.025	5.645	6.363
Min	1.071	0.000	0.820	0.303
Mean AUC(±std)	1.378(±0.194)	1.236(±0.142)	1.363(±0.196)	1.427(±0.204)
Max AUC	1.726	1.529	1.725	1.771
Min AUC	1.046	0.986	1.026	1.036

Table 4.1: Summary of machine-learning results expressed as improvement ratios over ES5D. A value of one indicates that the same performance as ES5D was obtained.

with the fraction of dataset used for training and evaluation for all our models, both in full conformer as well as LEC versions in Figures 4.25-4.32.

We then caluclated the actual number of actives involved in the training for each target and each dataset fraction and binned the result into 9 bins so as to get an understanding how the performance of the models varies with number of actives used for training. These plots can be seen in Figures 4.33-4.40

The statistical significance annotations were computed using the Wilcoxon rank-sum test (Mann and Whitney, 1947). This is a non-parametric test and therefore does not assume normality in the data. We have visually checked the distribution for each bin using histograms and found that they were not normal. It also assumed that the groups being compared are independent and not paired, which is the case with our box plots. The Wilcocon rank-sum test tests the null hypothesis that for any two observations a and b drawn from group A and group B respectively, $P(a > b) = P(b > a)$. The alternative hypothesis rejects this, i.e. the distributions are not equal.

It is apparent from these figures that performance is better maintained for low number of actives by using full conformer models than by LECs. This conclusion is borne out by both the figures plotting performance against the dataset fraction used as well as by those against number of actives. Nevertheless, even for small active training sets for which the mean performance is low, outliers are apparent with high enrichment factors. This shows that the performance of the methods we have explored is highly dependent on the protein target that is being considered and it is difficult to know a-priori, how well a method will perform given the number of available actives.

For LEC models a performance peak is apparent at around 25-49 actives, beyond which performance degrades again. We observed this effect on GMMs and Isolation Forest models, but not on Neural Networks. It is possible that implementing a more comprehensive parameter sweep during the tuning of these models could eliminate or reduce this effect. For example, in the case of GMMs, allowing a larger number of Gaussian components would probably resolve the active clusters better and improve performance for larger numbers of actives.

4.5 | Retrospective Screening Running-time of Machine Learning Models

In order to understand how the time required to train and perform a retrospective virtual screening run varies with dataset size, we plotted the time taken to perform our experiments against the corresponding dataset portion used as training set using box

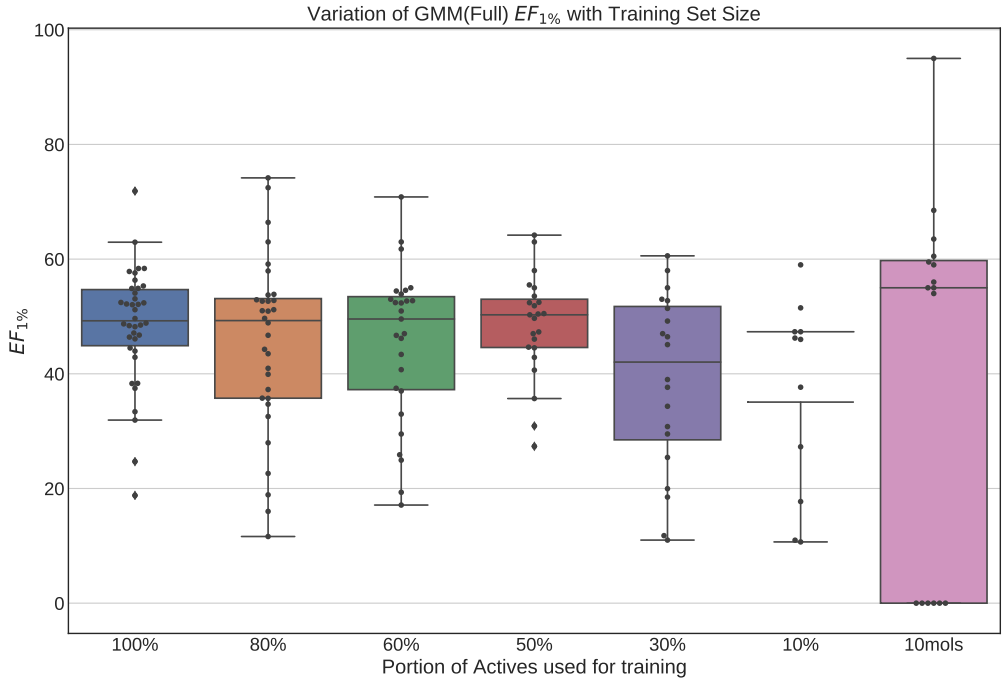

Figure 4.25: Performance variation of full-conformer model GMM with dataset fraction used for training.

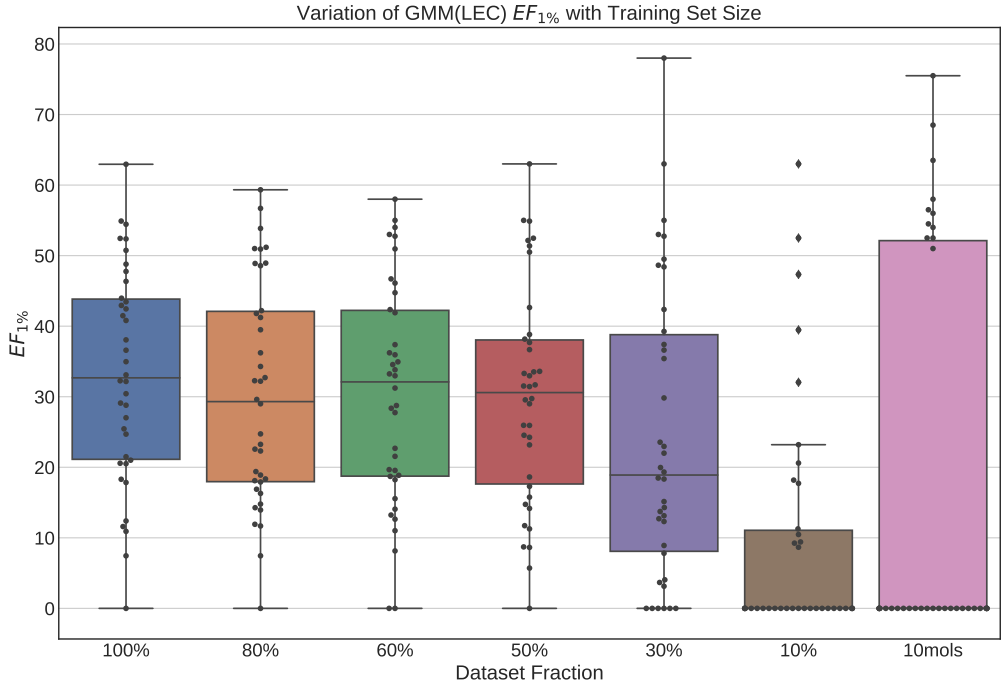

Figure 4.26: Performance variation of LEC model GMM with dataset fraction used for training.

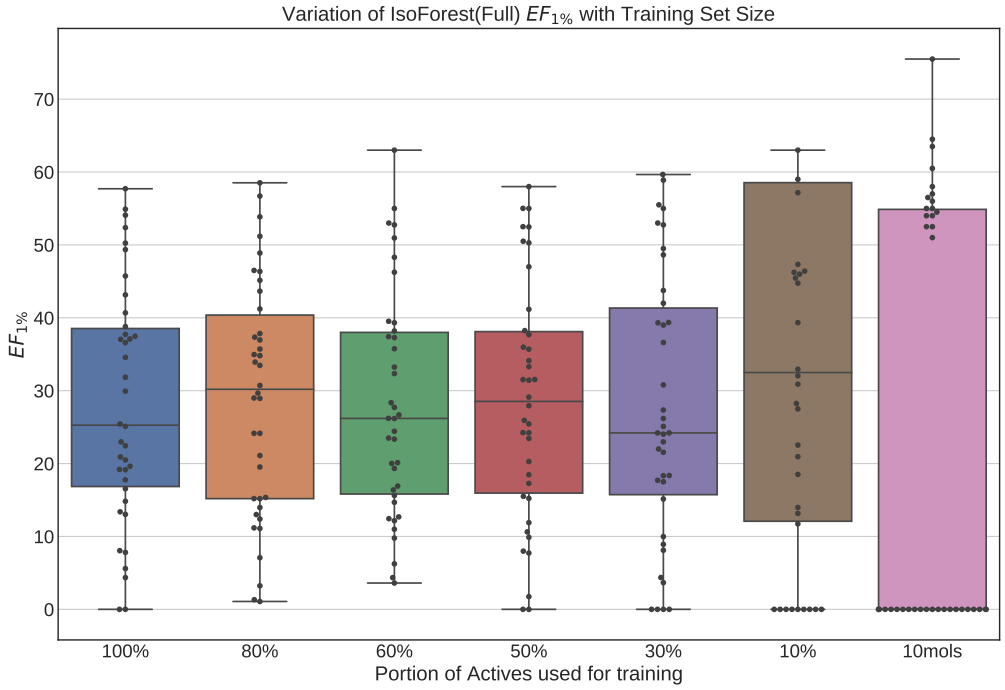

Figure 4.27: Performance variation of full-conformer model Isolation Forest with dataset fraction used for training.

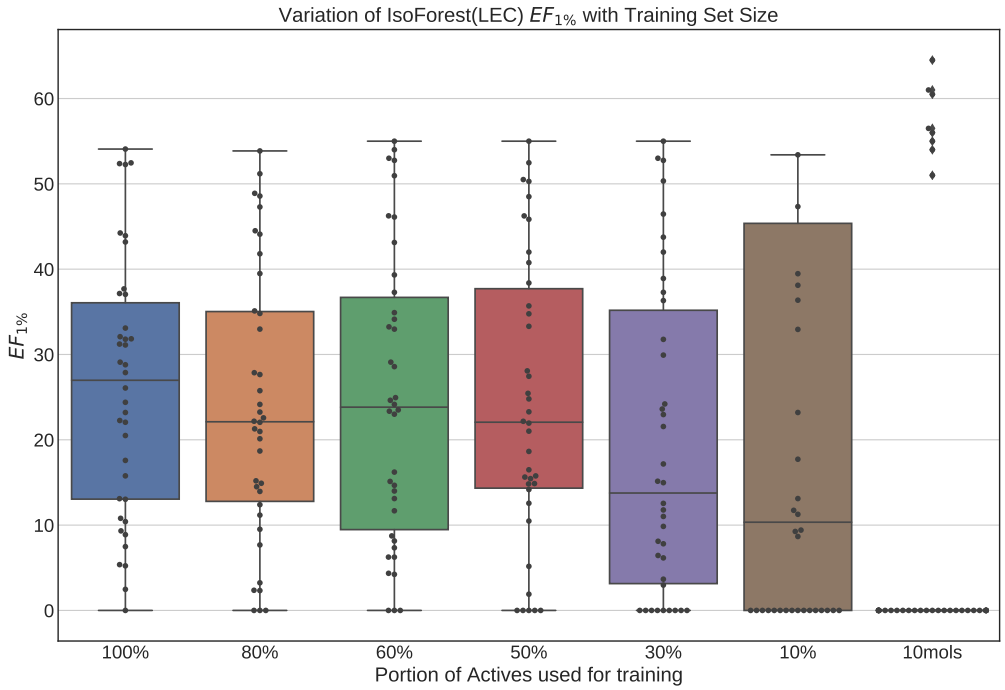

Figure 4.28: Performance variation of LEC model Isolation Forest with dataset fraction used for training.

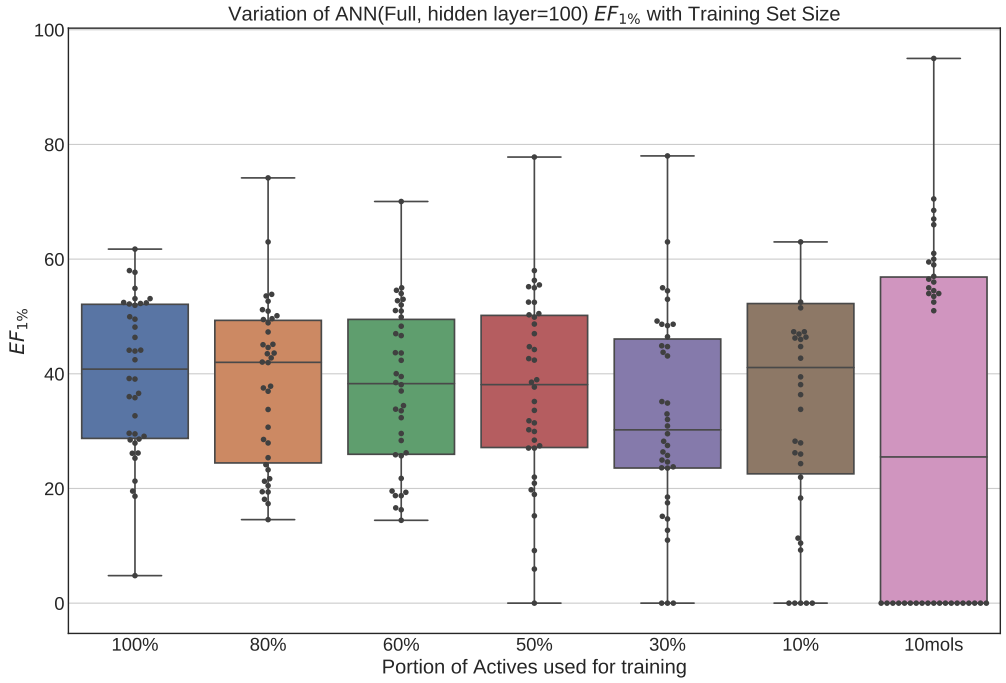

Figure 4.29: Performance variation of full-conformer model neural network (hidden layer size=100) with dataset fraction used for training.

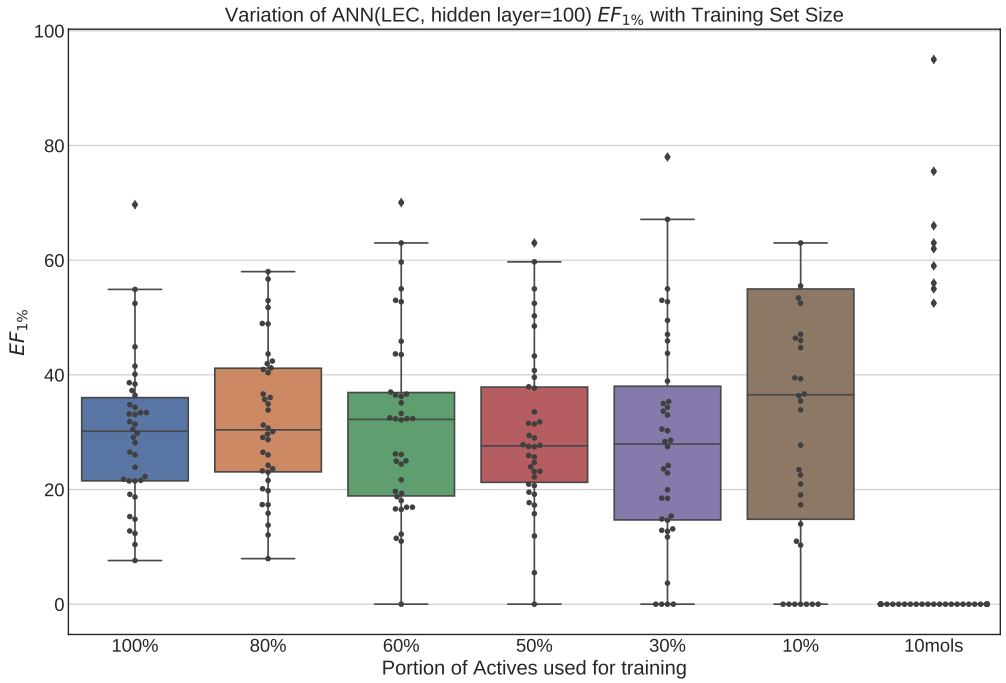

Figure 4.30: Performance variation of LEC model neural netowork (hidden layer size=100) with dataset fraction used for training.

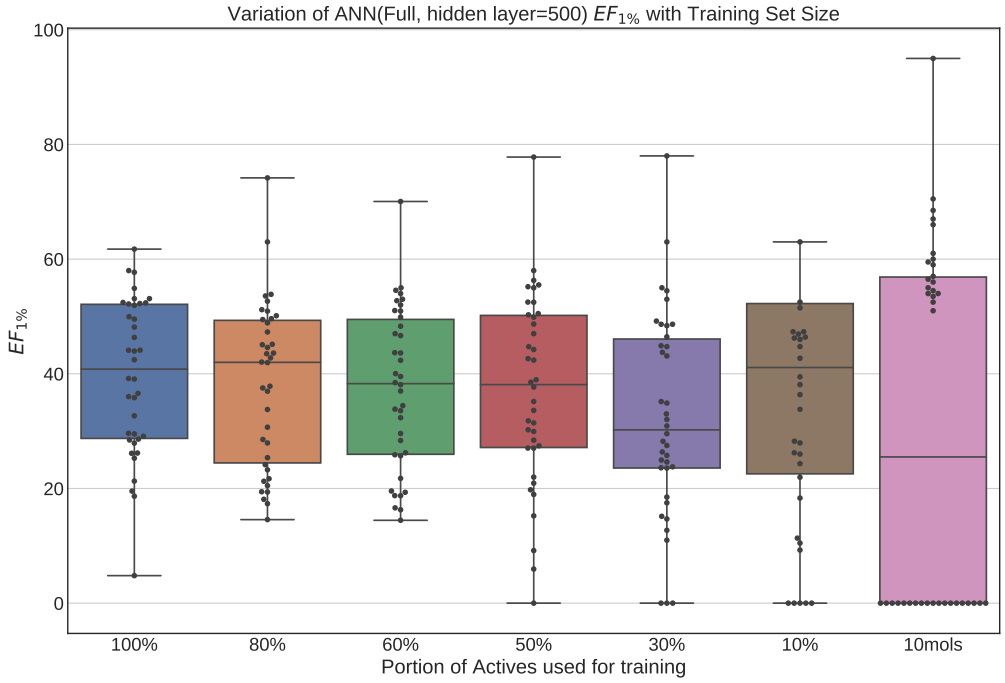

Figure 4.31: Performance variation of full-conformer model neural network (hidden layer size=500) with dataset fraction used for training.

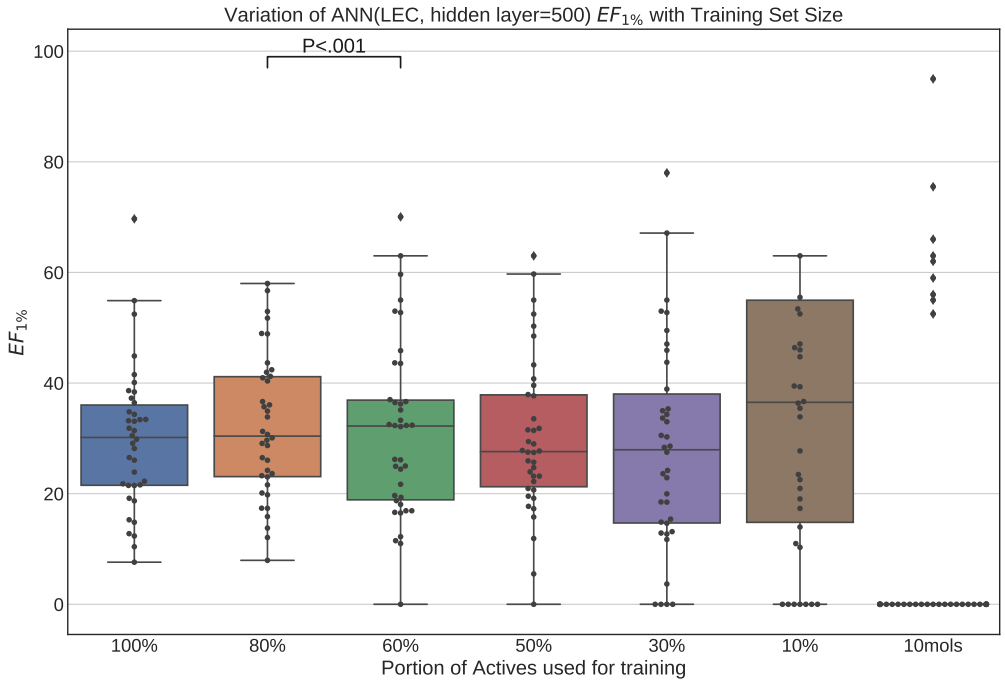

Figure 4.32: Performance variation of LEC model neural netowork (hidden layer size=500) with dataset fraction used for training.

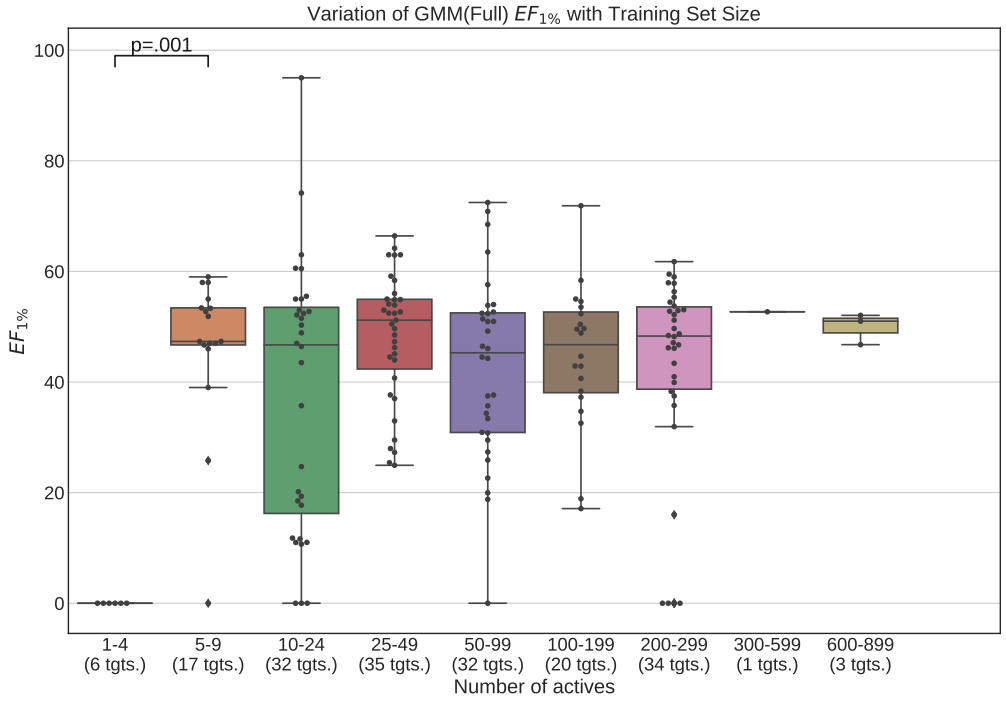

Figure 4.33: Performance variation of full-conformer model GMM with number of actives.

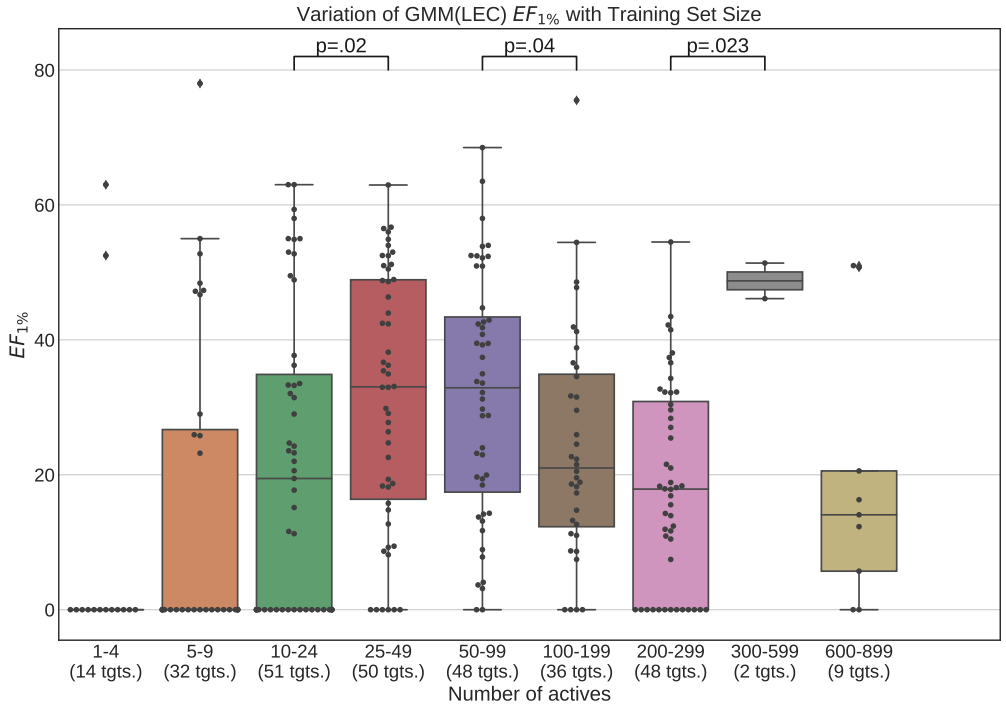

Figure 4.34: Performance variation of LEC model GMM with number of actives.

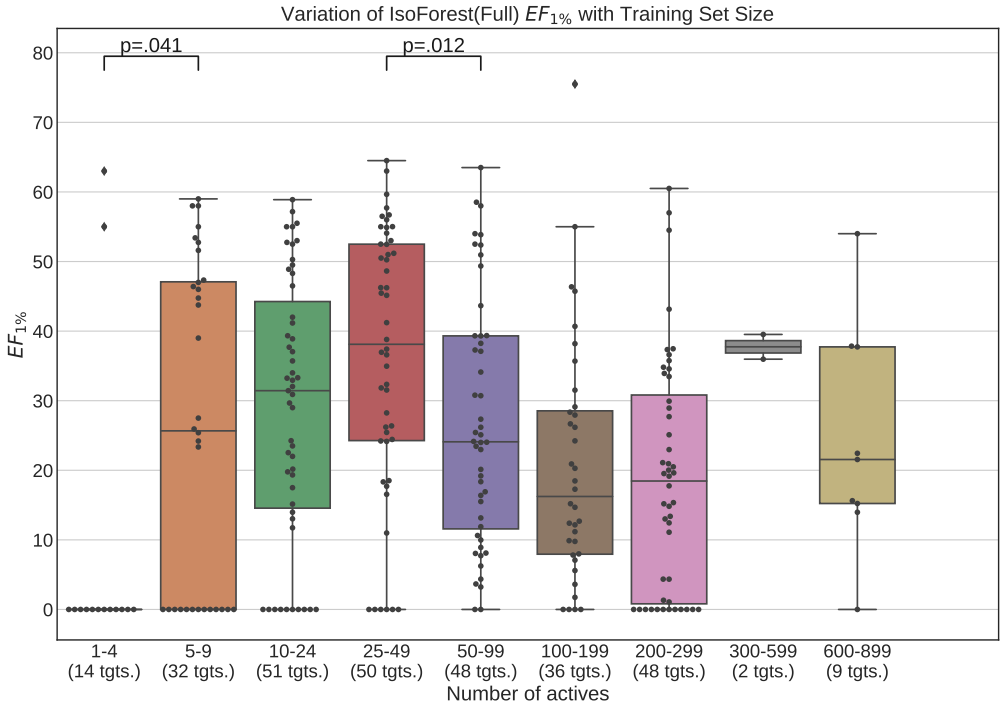

Figure 4.35: Performance variation of full-conformer model Isolation Forest with number of actives.

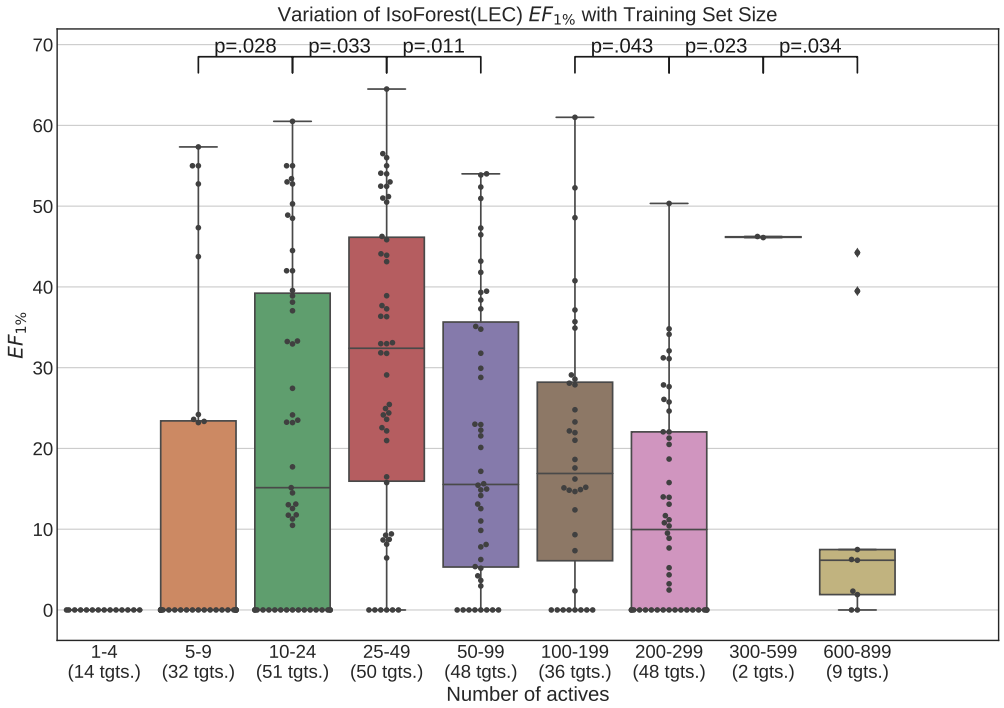

Figure 4.36: Performance variation of LEC model Isolation Forest with number of actives.

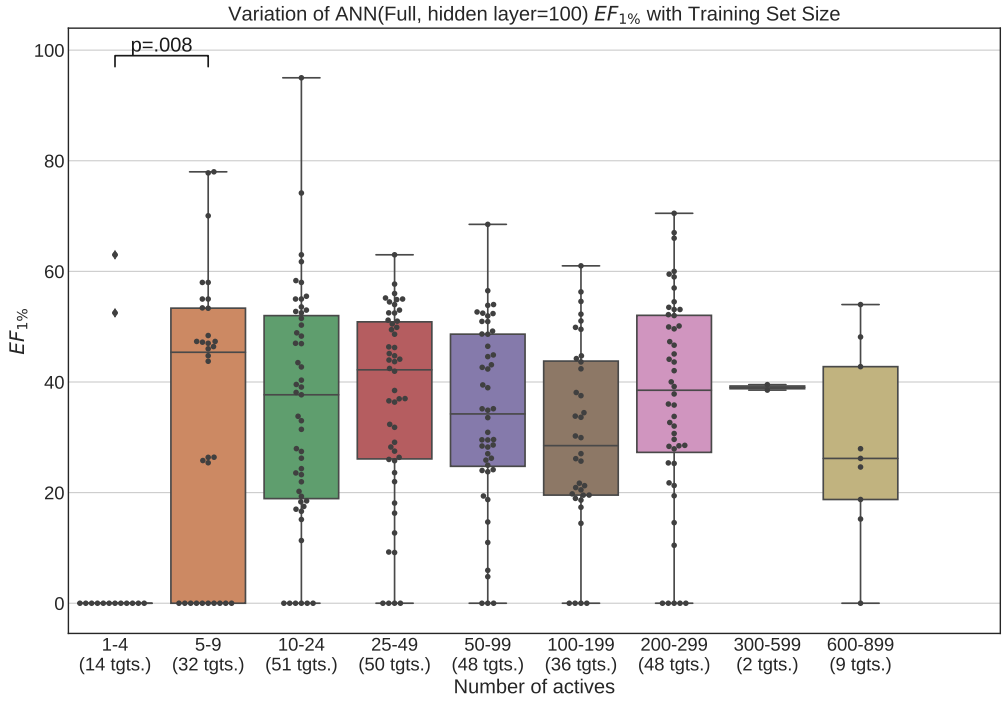

Figure 4.37: Performance variation of full-conformer model neural network (hidden layer size=100) with number of actives.

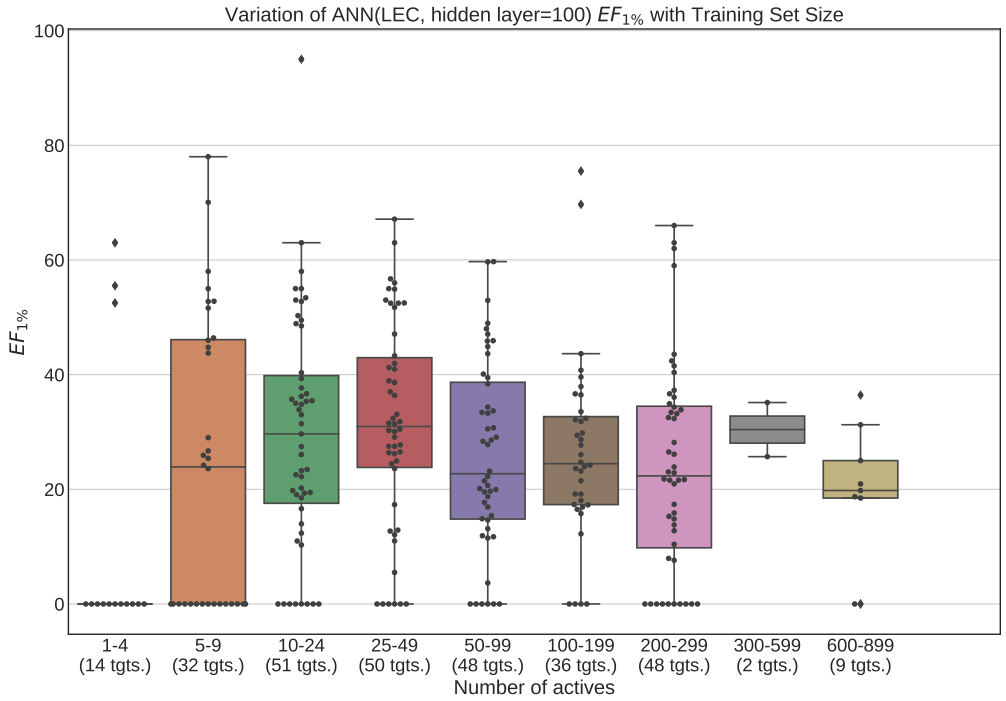

Figure 4.38: Performance variation of LEC model neural netowork (hidden layer size=100) with number of actives.

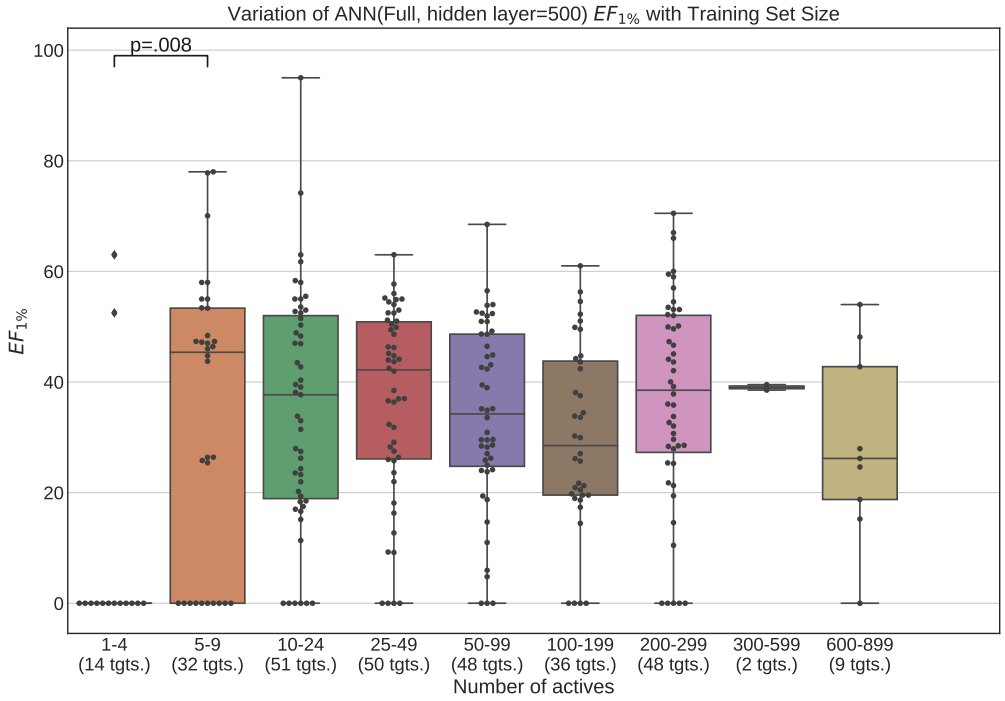

Figure 4.39: Performance variation of full-conformer model neural network (hidden layer size=500) with number of actives.

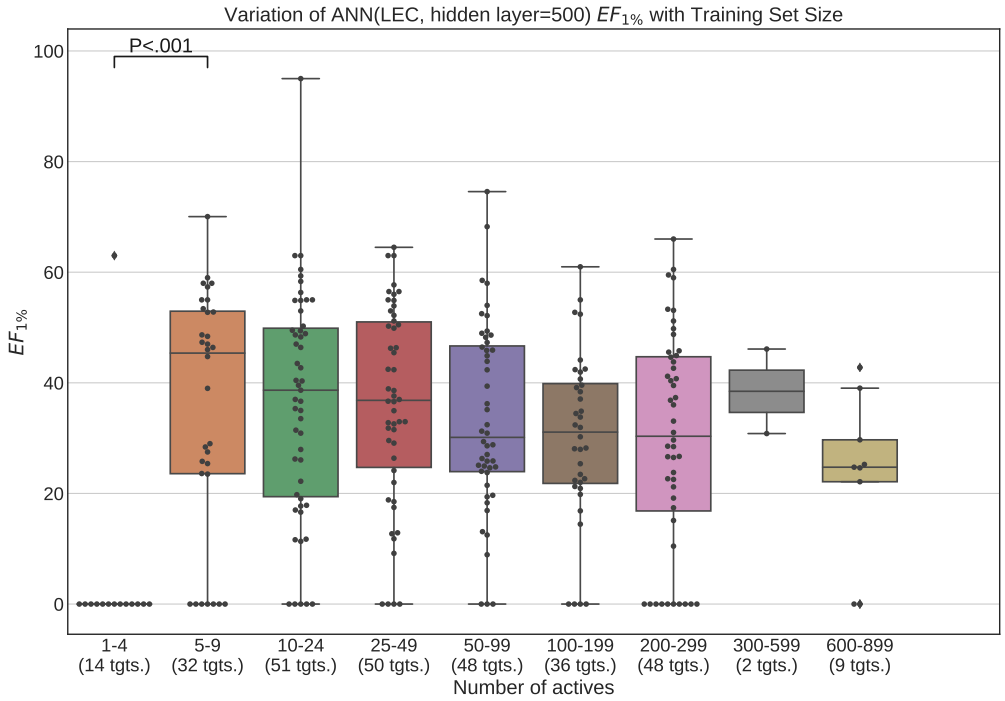

Figure 4.40: Performance variation of LEC model neural netowork (hidden layer size=500) with number of actives.

plots, with separate boxes representing the run-time for each machine learning algorithm. The timings include the time taken to train the final, tuned model and evaluate the molecules under test. These plots are shown in Figures 4.41 and 4.42. Note that, if used in a prospective screening scenario, a machine learning model would have been pre-trained from the available training data, therefore the time required for training would not be a factor when measuring the running time for such a study. In this case, however, since a retrospective experiment was being carried out we considered the total time required to be an important consideration.

It is apparent from these plots that GMMs were the quickest models overall for LEC models (8s$\pm$11s mean time) and the second quickest for the full conformer models (787s$\pm$868s mean time). For full conformer-trained models, GMMs were quicker for dataset fractions up to 60% of the full dataset, however were slower than Isolation Forest for dataset fractions larger than 60%. At the 30% fraction the GMM speed appeared to dip. This dataset fraction, however, appears to also have an unusual number of outlier data points and could have been caused by transient resource contention on the machine on which the experiments were being run.

Isolation Forest speed performance compared favourably to GMMs for large datasets when using full conformer models (397s$\pm$373s mean time for isolation forest vs. 787s for GMMs), however for smaller datasets using LECs it was considerably slower than the other algorithms, including ANNs (453s$\pm$423s for Isolation Forest vs. 131s$\pm$89s for ANNs). This is quite surprising and is likely due to the fact that no matter the size of the training data, an ensemble of decision trees of comparable size need to be created by the algorithm. Tweaking the hyper-parameters to use smaller ensembles for LECs would probably make the this model faster, however, this was not attempted in this study.

Neural Networks appears to be the algorithm that is the most consistent with respect to time performance. In general it is the slowest algorithm (1855s$\pm$1659s mean time for full conformers and 131s$\pm$89s mean time for LECs), except for Isolation Forest in the LEC scenario.

Finally, we compared the required run-time for the machine learning models with that required for USR and ES5D. These timings for the non-machine learning algorithms are shown in Figure 4.43 and 4.44 for full conformer models and LECs respectively.

Comparing the timings for USR and ES5D screening runs, it is immediately apparent that significant time-savings are obtained by the use of LECs over the full conformer models. With a mean run-time of 5,021s for USR and 7,409s for ES5D for full conformers, LEC is clearly much faster with a mean of 804s for USR and 883s for ES5D.

In order to directly compare the timings for all three machine learning algorithms, we have provided the mean, maximum and minimum run-times averaged over all the

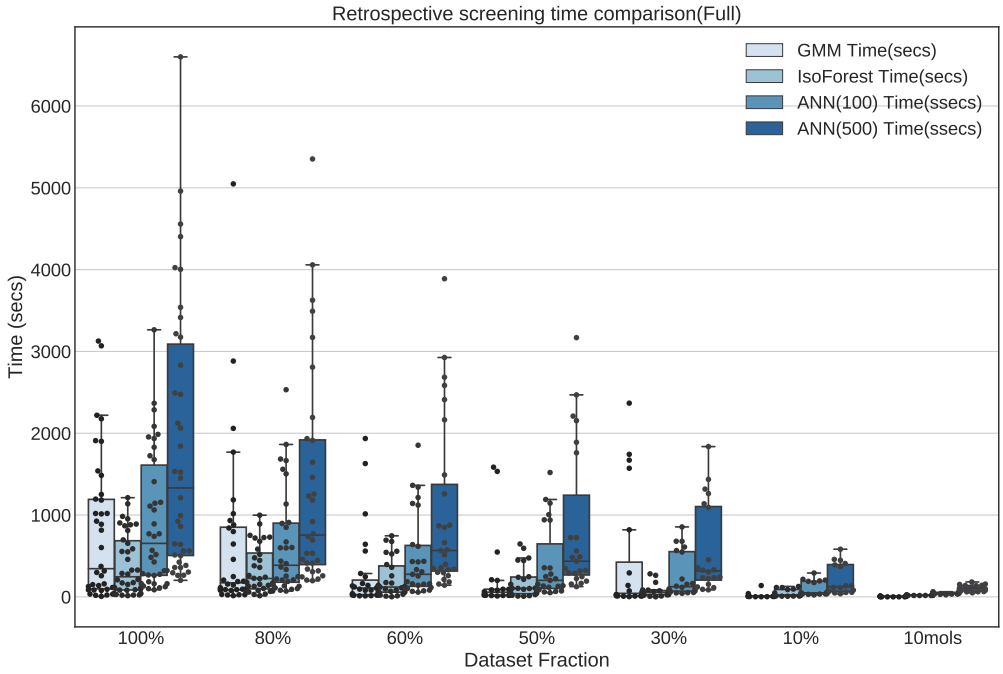

Figure 4.41: Run-time for training and retrospective screening for full conformer models.

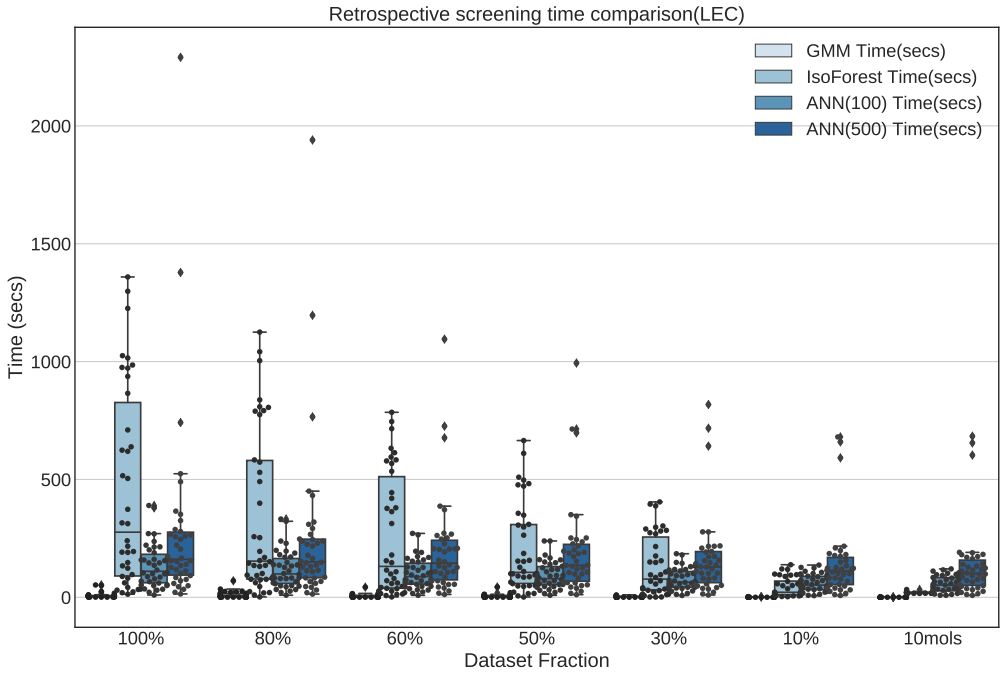

Figure 4.42: Run-time for training and retrospective screening forLEC models.

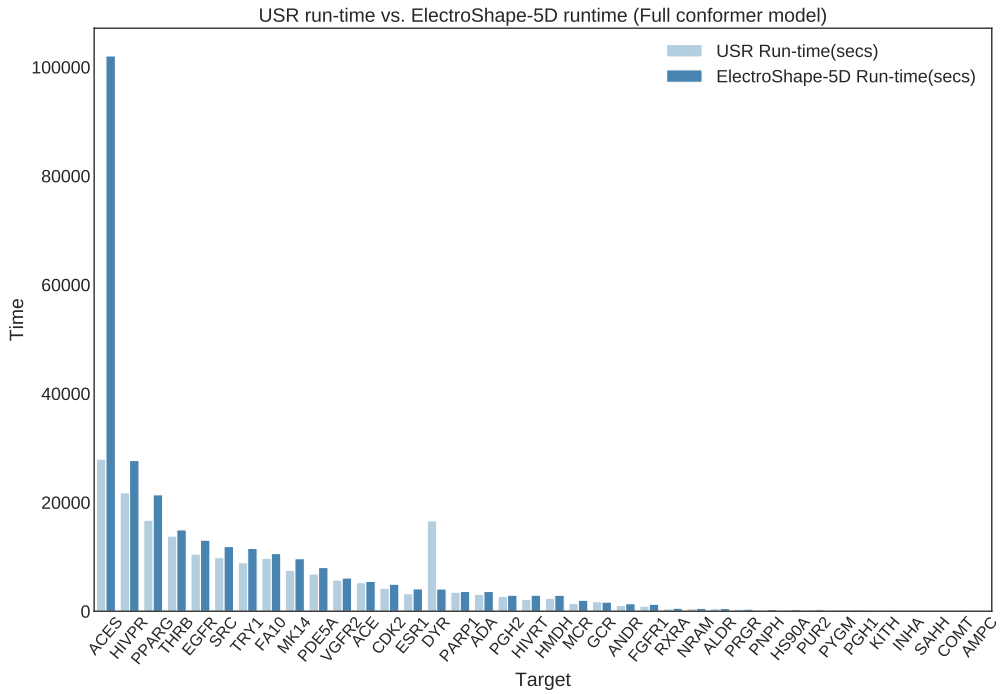

Figure 4.43: Run-time in seconds for USR and ES5D retrospective screening using full conformer models.

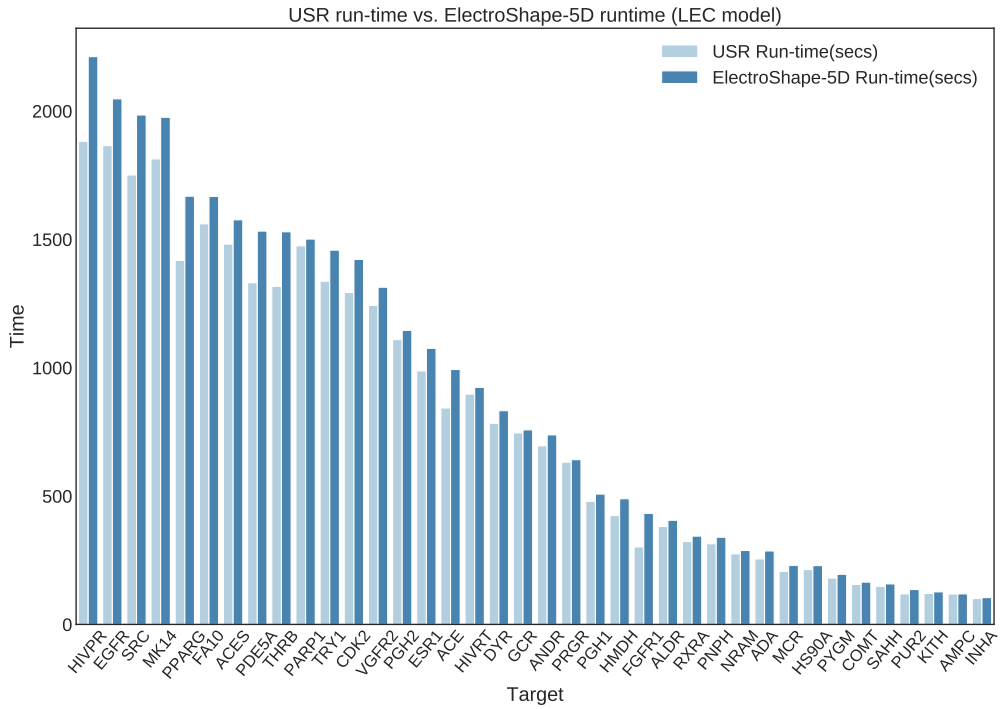

Figure 4.44: Run-time in seconds for USR and ES5D retrospective screening using LECs.

LEC	USR	ES5	GMM	IsoForest	ANN(100)	ANN(500)
Mean(s) (±s.d.)	804(±594)	883(±661)	8(±11)	453(±423)	134(±92)	285(±413)
Max(s)	1,882	2,212	52	1,359.00	390	2,290
Min(s)	100	104	0.7	13	10	14
Full Conformers	USR	ES5	GMM	IsoForest	ANN(100)	ANN(500)
Mean(s) (±s.d.)	5021(±6699)	7409(±16985)	789(±868)	397(±373)	934(±825)	1855(±1649)
Max(s)	27,940	102,034	3,126	1,212	3,264	6,600
Min(s)	59	61	6	14	87	202

Table 4.2: Tabulated runnimg-time statistics for all LEC and full conformer models. Timings are shown in seconds and include training and testing.

targets in Table. 4.2, for the full dataset virtual screening experiments. We did not consider the sub-100% dataset fractions because we only ran USR and ES5D experiments on the full datasets, therefore we wanted to compare like with like. Additionally, reduced-dataset runs, will, of course, always take less time to run than using the full dataset.

From these tables it is clear that the timings for all three machine learning algorithms are considerably better than those for USR and ES5D in terms of all three statistics.

As observed previously, GMM takes by far the least amount of time on average than any other method when trained with LECs, and is slightly slower that Isolation Forests for the very largest datasets. The time needed is, however, still far smaller that that for either USR or ES5D. When trained on smaller datasets, however, the speed of GMMs is impressive, being orders of magnitude faster on LEC training data than any other method.

In general, it appears that Isolation Forest is the slowest method on average, except, as mentioned above, in the case of the full conformer model training.

4.6 | Discussion

As outlined in Section 1.4, the primary aim of this study is to investigate if machine learning techniques can be used in lieu of the standard Manhattan distance-based similarity score used by the USR family of methods and if in this way, the virtual screening performance of the technique could be improved.

The results we obtained demonstrate unambiguously that this aim has been fulfilled. We have considered three different machine learning methods we selected due to their applicability to the problem of ranking molecules by similarity, and doing so we obtained mean improvements over ES5D in terms of Enrichment Factor at 1% rang-

ing from 190% to 293% mean improvement with LEC-trained models, to 196% to 432% improvement using full conformer training. In terms of AUC we have obtained mean improvements of 133% to 136% on LEC models and 123% to 138% when training on full conformers.

These improvements over ES5D are of a similar magnitude to the performance increase afforded by ES5D itself over USR (maximum improvement 738%, mean 253% for full conformers and maximum 755%, mean 283% for LECs) and are, therefore, highly significant. Machine learning algorithms assimilate the features of all the active molecules into a single model, in contrast to the naïve USR-based algorithms which can only consider one molecule at a time as a search query. This feature of machine-learning algorithms appears to make a large difference to the similarity matching performance in the LBVS context when compared with the non-machine learning USR family of methods.

In addition to the above encouraging results, it is also significant that, notwithstanding the necessity to train the machine learning models before running the virtual screening procedure, the total time required to perform our retrospective screening runs on each target took, on average, a much shorter time to complete than the standard USR algorithms. Part of this discrepancy is likely the efficiency of our Python implementation of USR, which must necessarily be slower than the c-based implementations of the algorithms in the Scikit-learn library. The magnitude of the difference however, makes it unlikely for this to be the entire explanation. A large part of the discrepancy also comes from the fact that, in USR, all the conformers in the test set of molecules must be compared to every conformer of the active molecule, and this must be done for every active search template molecule. Over the course of an entire retrospective screening run, this adds up to a large amount of computation.

With machine learning algorithms, however, this is not necessary. The bulk of the running-time when using machine learning methods is the training of the model, however this, in general, does not require the repeated comparison of all the data points with all the active data points in a Cartesian product fashion. Additionally, once a model is trained, classifying new data points is generally a fast process because it does not involve comparing the new point with the training data directly, but only requires that the new data be evaluated according to the statistical model built during training. All this, clearly depending on which particular machine learning algorithm is being used, implies a much smaller amount of computation than the "brute force" approach inherent in standard USR.

Our second aim in the pursuit of this project is to explore the way in which machine learning models respond to decreasing dataset sizes. Algorithms that maintain good early enrichment with small datasets are desirable in the LBVS context because,

often, only a small number of active molecules are known in advance, and hence a large training dataset will not be available.

Our results, presented in the previous sections, show that, on average, the machine learning algorithms trained on full conformer models tend to preserve early enrichment performance better than those trained on LECs when trained on a small number of actives. It would appear, therefore, that when less than 25 actives are available, it would be advantageous to train using full conformers in order to obtain a significantly better performance than the equivalent training using LECs. In these cases, the performance-speed trade-off caused by the increased volume of data inherent in using full conformers is not so onerous by virtue of the small training set size that is implied.

When taking into consideration, both the early enrichment performance of our models, as well as the mean running time required to complete a retrospective screening run, our conclusion is that, considering the three machine learning algorithms explored in this study, the GMM algorithm is the one that is recommended in most cases. This is because it produces, on average, the best early enrichment scores and does so in the smallest amount of time compared to the other algorithms.

4.7 | Summary

In this chapter we present the results we obtained in pursuit of our stated research questions. We have initially implemented the USR and ES5D algorithms, so as to perform a set of retrospective virtual screening runs based on the DUD-E datasets we selected as described in the Methodology chapter. The DUD-E targets that we chose were the same targets available in the DUD database so as to make comparison with earlier work mostly performed using DUD possible. By doing this, we obtained a set of baseline performance measures against which we could then compare the results obtained by our machine learning models.

We next set up machine learning pipelines for three representative algorithms which we deemed to be conducive to obtaining relevant results in the LBVS problem, namely, Gaussian Mixture Models, Isolation Forests and Neural Networks.

We ran retrospective screening experiments on the same DUD-E targets so as to obtain corresponding performance scores, and additionally we also ran screening experiments on successively smaller fractions of the available datasets so as to chart the performance degradation of our models with dataset size.

From the results thus obtained, we calculated improvement ratios, comparing the performance of our models with that of standard ES5D, thus confirming a significant

performance increase obtained through machine learning. We also determined that our models were also good at preserving screening performance with decreasing dataset size, especially those trained with full conformer molecule models. These are both important results which we believe will be of significant future value to researchers in the field.

During the course of our retrospective screening experiments, we stored timing information, keeping a record of the required time to complete the given experiment. Through this we showed that machine learning algorithms can complete retrospective screening in a much shorter time-frame than the standard USR-based algorithms.

Finally, from the screening performance results as well as timing results that we obtained, we determined that the GMM is, in general, the most effective algorithm to use out of the three we considered in the study, both for its superior screening performance compared to the other two, as well as to its fast training and evaluation time, giving significant time savings over the other algorithms.

5

Conclusions

In this dissertation we explore the potential that exists in applying machine-learning to the USR family of methods. The standard USR method condenses the three dimensional shapes of molecules into a small numeric descriptor vector that can be directly compared to other USR descriptor vectors using a Manhattan distance-based metric. In doing this a measure of shape-similarity between different molecules is obtained. In this study we replace this naïve distance measure that is used in the standard USR family of algorithms by instead, training several machine learning models directly using the USR and ES5D descriptors themselves.

In Chapter 1 we give a brief introduction to the topic of cheminformatics and virtual screening and describe the motivation behind our research. We give an overview of the machine learning methods that we used as well as of the datasets on which we based our research. We also state the aims and objectives of the project and outline the approach followed by our research.

We give comprehensive background information about the domain of virtual screening in Chapter 2 and Appendix A. We give an account of the USR family of methods delving in detail into the workings of the most important ones for this project. We also give detailed information about molecules and conformers and about their generation and processing *in silico*. Additionally, in this chapter we also give detailed information about the machine learning methods we used in the course of the project as well as about the evaluation criteria we use to evaluate our results. Finally, we also give an overview of the existing research relating to our project.

In Chapter 3 we give implementation details about the processes we built for the purposes of the project. During the course of the project we implemented the generation of conformers from the 2D SMILES representations of molecules, our versions of several USR methods as well as machine learning pipelines to tune, train and evaluate our

machine learning models.

We present the results that we obtained in Chapter 4. Here we first evaluate our benchmark implementation of the USR methods we chose. We subsequently also present the evaluations of our machine learning results in absolute terms and also in terms of improvement from our benchmark, non-USR results. In this chapter we present results related to machine learning models trained on the full dataset as well as models trained on successively smaller fractions of the available data, so as to be able to assess the robustness of a model with decreasing training set size. We also report results relating to the time required to run retrospective virtual screening using each of our trained models.

5.1 | Revisiting Aims and Objectives

Through this research project we sought to answer two research questions, namely:

1. Can machine-learning techniques be used instead of naïve Manhattan distance to improve Virtual Screening performance based on USR and USR-derived descriptors?

2. What is the minimal amount of data required to adequately train the machine learning model?

In order to answer the first research question, the first task was the implementation of the conformer generation process, which takes the 2D SMILES molecule definitions as provided in the DUD-E database, and from them generates a number of conformers for each molecule that are sufficient to adequately sample the molecule's conformational space.

Once we had generated conformers for every selected protein target from DUD-E, we implemented the basic USR algorithm as well as ES5D, one of the highest performing, state of the art, variants of USR. Performing retrospective virtual screening experiments using these algorithms gave us baseline performance figures with which we could directly compare the performance of our trained machine learning models. In order to verify that our implementation was working correctly, we compared our baseline results to those in the literature, ascertaining that the performance obtained across the common protein targets was similar.

We selected three machine learning algorithms that were suitable for applying to the scenario of LBVS using USR descriptors, namely Gaussian Mixture Models, Isolation Forests and Artificial Neural Networks and we trained and evaluated these models

using chosen compound datasets from the DUD-E database. In doing so, we obtained results that significantly outperformed USR as well as ES5D when using both the full conformer models of the active compounds as training data, as well as when using only the Lowest Energy Conformations (LECs). Concretely, in terms of $EF_{1\%}$ the best mean improvement over ES5D was that of 430% obtained using GMMs trained on full conformers, the same models having obtained a maximum improvement of 941% over ES5D. This was followed by a mean improvement of 328% with a maximum of 636%, obtained by ANNs, again trained on full conformer models. When using LECs as training data, GMMs obtained a mean performance improvement of 291% and a maximum of 829%, outperforming ANNs which obtained a mean improvement of 257% with a maximum of 613%. It is clear, however, that some targets are more responsive to screening by USR descriptors, there being a relatively large variance in the mean performance figures. This is also reflected in the literature (Armstrong et al., 2009, 2010, 2011; Ballester et al., 2009a) and is therefore expected.

The results obtained from these experiments demonstrate that the answer to the first research question is in the affirmative. We have indeed found ample potential in the use of machine learning techniques for improving the yield of virtual screening processes making use of USR and USR-derived descriptors.

Furthermore, we also compared the running times of the machine learning retrospective virtual screening experiments with the time required for the non-machine learning ones and showed that even considering the time required to train the models, the machine learning experiments took an order of magnitude less time to run than the plain USR ones.

In order to explore our second research question, we trained the machine learning models on progressively smaller fractions of the chosen DUD-E datasets so as to explore the manner in which the performance of the models varied with decreasing training dataset size. Through our results we demonstrated that when using full conformers to train the models, better performance is obtained when the number of actives is low. In general a performance peak is observed when training with 25-49 actives. With the LEC models, this peak is more pronounced, indicating that for small active training sets it is more advantageous to train with full conformers than LECs.

A general observation in our results is that, across the machine learning models that we trained, those trained on full conformers preserve good performance when trained with as little as 5-9 actives, while with those trained on LECs, the cutoff is in the 10-24 active range, further confirming our conclusion that for small datasets, models should be trained using full conformer models.

Taking into account all the results obtained, we came to the conclusion that GMMs

were, overall, the most efficient models that we tested, achieving excellent performance in the shortest time (except for the largest datasets) and while also exhibiting good stability with decreasing dataset size.

5.2 | Critique and Limitations

While the aims of the research project have been reached, as evidenced by our results, the broad exploration that the research questions required, meant that the available time and resources put a limit as to how exhaustively every machine learning algorithm could be explored.

For all three machine learning models we only performed tuning on the major hyper-parameters, accepting default values for others, however it is not to be excluded that better performance could have been obtained by including other hyper-parameters in the parameter sweep during model tuning. Also due to resource limitations, the search space for the hyper-parameters that were tuned might not have been chosen to be wide enough in scope. A possible corroboration of this is that some models exhibited a performance drop with a larger number of training actives indicating that a higher model complexity was required to successfully model the training data.

Much of the older work related to USR was evaluated relative to the DUD database which provides ligand datasets for 40 protein targets. DUD, however, has been shown to be problematic in its choice of decoys, possibly resulting in better performance in retrospective screening experiments than would occur in corresponding prospective studies. Due to this, we chose to use the improved DUD-E database instead to evaluate our models. Due to resource limitations, however, we were forced to select a subset of the 120 protein targets provided in DUD-E. In order to be able to facilitate comparisons with older work, we chose to use a 38 molecule subset of the DUD-E with targets corresponding to targets provided in DUD. Ideally, however, the study should be performed on all the targets in DUD-E.

Finally, even though we have obtained promising results in our retrospective studies, the proof of any virtual screening method is its success in discovering new leads in a *prospective* study. Of course, this is beyond the scope of this project, however the techniques employed in this dissertation or ones inspired by them can easily be employed in future prospective studies and offer good promise for future research to be carried out.

5.3 | Future Work

Through this study we have demonstrated that applying machine learning to USR and USR-derived descriptors has the potential to improve virtual screening performance over the Manhattan-distance measure that is used in the standard USR family of methods.

In view of this, the focus of any future work related to the research presented in this dissertation should be to identify optimal machine learning techniques to use in the USR-based virtual screening context.

One major limitation of LBVS processes is the difficulty of obtaining good results when only a small number of active compounds are known. Our results clearly show that the algorithms are the least effective when less that 9 actives are known. Efforts have already been made in improving results when only one or a small number of reference compounds are available, for example using group fusion (Hert et al., 2005, 2006), however this problem is also a shared one in other domains where machine learning has proven useful. The problem of *One-Shot Learning* is that of categorisation of objects when only one or a small number of training examples are available. This problem is an important one in computer vision and several algorithms have been proposed in this direction (Fei-Fei, 2006; Fei-Fei et al., 2006) and similar ideas have recently also been applied to drug discovery (Altae-Tran et al., 2017).

The successful application of one-shot learning techniques to USR-derived descriptors would be a significant improvement to this work, potentially making the process more useful for the common problem of similarity searching with a small number of search templates. One-shot learning should be a major avenue to be explored in future research.

While we have made use of a simple neural network in our study, more advanced neural network architectures, collectively known as deep neural networks or deep learning, have been proposed and used with success in a wide variety of domains such as semantic parsing (Bordes et al., 2012), natural language processing (Mikolov et al., 2013), computer vision (Cireşan et al., 2012a), transfer learning (Cireşan et al., 2012b) as well as in virtual screening (Carpenter et al., 2018; Nasser et al., 2018; Xiao et al., 2018) where excellent results have been obtained. The application of Deep Learning to USR-based descriptors would be an interesting and potentially fruitful avenue to explore in future studies.

Additionally, there is promising research that can be done in the direction of minimising the size of the USR descriptors, such as, for example, carrying out feature reduction using dimensionality reduction algorithms such as Primary Component Anal-

ysis (PCA) or Linear Discriminant Analysis (LDA) as well as through deep learning approaches such as in Nasser et al. (2018). Reducing the size of the descriptors would have large implications on the efficiency of the screening process, due to the large number of conformers that need to be processed. Even a small reduction in descriptor size could translate to a significant reduction in the size of the dataset that needs to be processed by the algorithm.

5.4 | Final Remarks

To the best of our knowledge, this research project constitutes the first study to explore the viability of several machine language algorithms in their application to the problem of LBVS using USR and USR-derived descriptors.

We have demonstrated that USR-like descriptors are well suited to be used as training examples for machine learning algorithms, managing to obtain performance figures in terms of Enrichment Factor and AUC well above the figures obtained by the standard Manhattan Distance-based USR-based algorithms.

We have also explored the performance variations in the use of full conformer models as opposed to LECs as training input to our models, indicating that the use of LECs can be a viable alternative that vastly reduces the volume of data that needs to be processed, and hence the running time required to produce results. Nevertheless, our findings indicate that the use of the full conformer models is more indicated when small training sets are available, as better performance is preserved.

Additionally, we have explored the performance variation of the machine learning models when trained on successively smaller active compound datasets so as to understand the usefulness of the methods when applied to prospective screening scenarios where only a small number of actives are known. In doing so, we have demonstrated that the performance of the trained models is relatively stable at varying dataset sizes, but that those trained with full conformers preserve better performance at low training set sizes, therefore indicating that the use of full conformer models is indicated when only a small number of actives is known.

This work, due to the sheer magnitude of options available when it comes to machine learning methods, must necessarily be considered to be simply a starting point for further research into the topic of machine learning on USR descriptors. We believe, however, that it makes a valid contribution to the field, as it demonstrates significant performance improvements over the current state-of-the-art methods.

The fields of CADD and Virtual Screening are becoming increasingly important in

today's pharmacological industry as new, potentially life-saving drugs are developed at an ever increasing pace. We are hopeful that our efforts will be extended further by future researchers, so as to achieve further improved virtual screening performance and make a concrete contribution not only within the academic sphere, but also to peoples' lives.

A

Further Background Information

A.1 | Atoms and Molecules

The atom is the smallest particle into which a chemical *element* can be divided while preserving its chemical and physical properties. An element is defined as a substance that is made up of only one type of atom.

Most of the mass of an atom is comprised of a central nucleus possessing a positive electrical charge. The nucleus is made up of two types of sub-atomic particles - the positively charged *Proton* and the *Neutron* which is electrically neutral. These particles are known as *nucleons* and they have a mass of approximately 1 atomic mass unit or $1.66053906660 \times 10^{-27}$ kg. The nucleus is surrounded by a cloud of negatively charged particles called *Electrons* and which are much less massive than the nucleons (5.489×10^{-4} atomic mass units). In general, the number of protons in an atom is equal to the number of electrons, making the atom as a whole electrically neutral. It is possible to have atoms with the same number of protons and electrons but with a different number of neutrons. These atoms have a different mass, however retain the same chemical properties and they are referred to as *isotopes*. An atom can also lose or gain electrons making it respectively positively or negatively charged. Atoms that have an unbalance between electrons and protons are termed positive or negative *Ions*.

Various models have been proposed to describe the structure of the atom, however the currently accepted one is largely based on that proposed by Neils Bohr in 1913 in which he used Quantum Theory to describe the behaviour of electrons around the nucleus. In the currently accepted model, the energy of the electrons is *quantized*, i.e. it is not continuous in nature and can only take several well-defined values. This causes the electrons around the nucleus to be arranged in shells or layers at differing distances from the nucleus, the lower the energy, the smaller the distance. Each electron shell can

only hold a fixed number of electrons and, in general, electrons move into more energetic shells only if less energetic shells are already full. The number of electrons in the outer shell determine the chemical properties of the element. The outer shell is known as the *valence shell* and the number of electrons it contains is known as the *valency* of the element and it determines the number of bonds to other atoms that the atom can sustain.

Atoms can bind electrostatically to other atoms forming a new substance called a *compound* having chemical properties distinct from any of the atoms making it up. There are various types of chemical bonds that can occur between atoms, each having different properties and different strengths and consisting of different mechanisms.

Some types of inter-atomic bonds occurring within a molecule are include the following:

- **Covalent bonds**. Two atoms can share one or more pairs of valence electrons giving rise to a very strong bond. This is the most common type of bond that occurs in organic chemistry.

- **ionic bonds**. In an ionic bond, one or more electrons moves from one atom to another giving rise to two ions of opposite charge and resulting in an electromagnetic attraction between them. This type of bond is less strong than a covalent bond and occurs predominantly with metal or halogen atoms.

- **metallic bonds**. Due to the characteristics of heat and electrical conductivity possessed by metals, the bonds between atoms in a metal are characterised as being akin to positive ions in a sea of electrons. The electrons act as "glue", keeping the atoms together, however are able to move freely within the structure, giving rise to the properties of conductivity in metals.

- **Van der Waals forces**. These are a set of weak forces that occur between atoms as well as between molecules caused by permanent or temporary uneven distribution of electrons around the atom or molecule. This causes the formation of an electrical dipole, i.e. a separation of positive and negative charge over the atom or molecule, allowing an attractive force to occur between atoms.

- **Hydrogen bonds**. These bonds occur between Hydrogen atoms that are already bonded within a molecule and atoms in other molecules. They occur because a bonded Hydrogen atom has a slight positive charge due to its single electron being pulled away towards the bonded atom. This causes an attractive force between it and other bonded atoms that are negatively charged. This force in weak

in isolation, however it is important in organic chemistry as it occurs in water, giving water some of its unique properties, and it is also important in stabilising the structures of protein and DNA.

The strongest and most important bonds when it comes to the study of drug-like small molecules are covalent and ionic bonds as they are the strongest. The shape of a molecule is dependent on the pattern of bonds that occur between the atoms forming it.

The "backbone" of organic chemistry is the Carbon atom. Carbon has a valency of 4, meaning that it can sustain 4 bonds to other atoms simultaneously. Crucially, apart from being able to bond to other elements such as Hydrogen, Oxygen, Nitrogen etc., it can also bond to other carbon atoms, enabling the formation of structures such as chains of arbitrary length, rings, tetrahedrons, and ball structures. This gives rise to an almost infinite variety of molecules ranging from the simple carbon dioxide molecule, where a single carbon atom binds to two Oxygen molecules (valency 2) forming two bonds with each Oxygen atom, to the Deoxyribose Nucleic Acid (DNA) molecule, which encodes all the information necessary to build a living organism.

The basic building blocks of life are compounds called amino-acids. About 500 different amino acids are known, out of which 20 are coded for in DNA. Amino-acids can form long chains called *polypeptide chains*. Depending on the exact composition of a polypeptide chain, it will fold into a definite shape forming a *protein*. Proteins are made up of one or more polypeptide chains and perform a vast array of functions within a living organism. Proteins can be regarded as catalysts - compounds that act as external agents enabling or speeding up a chemical reaction. The specific shape of a given protein will catalyse a specific reaction within the organism, thus performing a given job. Proteins are involved in DNA replication, transportation of molecules within a between cells, and also as structural material for cells.

The process of building proteins in a living organism is governed by the sequence of *Nucleotides* in the DNA molecule. Nucleotides are the building blocks of DNA. Four nucleotides are used in DNA - Guanine, Thymine, Guanine and Cytosine. These nucleotides are grouped in threes within DNA, each group of three called a *codon*, each codon coding for a specific amino-acid. This scheme permits coding of a total of $4^3 = 64$ amino-acids, however only 20 amino-acids are, in fact, produced as several codons encode for the same amino-acid.

The function of a protein within the organism depends almost exclusively on its shape. The field of Drug Discovery deals with finding small molecules that are able to bind with a given protein, in the process altering its shape and hence its function. Proteins are capable of binding to other molecules through specific binding sites on

their surface. The shape of the binding site dictates the shape of the molecule that is capable of binding to it. This is Emil Fischer's lock-and-key principle, as discussed in Section. 2.1.

The challenge in finding such compounds is that molecules are not necessarily rigid structures. Any single atomic bond within a molecule can allow rotation causing the spatial relationship of the structures on either side of it to change, therefore changing the shape of the entire molecule. This process also changes the molecule's total energy due to a different distribution of interatomic forces within the molecule having been created.

Such configurations of a molecule that are stable are called *conformers* or *conformational isomers*. These stable conformations of the molecule will only occur at energy minimums as a higher energy conformation will always relax into a lower energy one when possible.

A.2 | Conformer Generation

In general a molecule can take a variety of stable shapes called conformations. This variety can be characterised as an n-dimensional *conformational space* where n is the number of rotatable bonds in the molecule. Conformer generation is a process of searching this conformational space in order to find stable conformers.

There are three major approaches to achieving this, which are:

- **Systematic Search**. This involves systematically varying each rotatable bond in order to consider all possible shapes of the molecule. This is, however only feasible for small molecules. An example of this approach is provided by the *Confab* tool, part of the Open-Babel cheminformatics toolbox [1].
- **Stochastic Sampling**. The conformational space can be sampled randomly or through directed stochastic techniques such as Monte Carlo approaches or genetic algorithms. The Open-Babel toolbox supplies a genetic algorithm-based conformer generation tool apart from the Confab tool mentioned above. The conformer generation routines provided by RDKit are also stochastic in nature, randomly generating atom positions between lower and upper bounds determined by the atom connections and a set of predetermined rules.
- **Knowledge-based approaches**. This kind of approach uses experimentally-determined rules about spatial relationships between different types of atoms in order to gen-

[1] http://openbabel.org [Last accessed 12/06/2019]

erate conformers. RDKit provides a knowledge-based method of conformer generation called ETKDG. This uses experimentally determined rules in order to generate valid conformers.

A.2.0.1 | Energy Minimisation

With most conformer generation methods, the resulting conformers may not be perfect and may not be located at an energy minimum, i.e. the net force on the atoms in their generated positions might not be zero. In order to correct this, a procedure known as *energy minimisation* is carried out. Defining $E(r)$ as the energy of the molecule as a function of the atom positions, energy minimisation iteratively minimises this function by:

1. Calculate the force on each atom, i.e. $-\frac{dE}{dr}$
2. If the force is zero or within a pre-set threshold, stop
3. If not incrementally move the atoms by some amount Δr in order to decrease the force exerted on the atom.
4. repeat

A function minimisation algorithm such as gradient descent can be used to perform this procedure, however in practice other algorithms specifically adapted to the task are used as they give superior performance.

There are several methods of modelling $E(r)$, however a common way of doing so is by the use of a *force field*. A force field is a parameterised model of the molecule energy expressed in terms of the bonded energy, i.e. the energy inherent in covalent and ionic bonds between atoms, and of the non-bonded energy, due to forces such as the Van der Waals force. These functions are used in conjunction with sets of parameters for each type of atom and chemical bond, usually experimentally determined, in order to calculate the energy of the molecule.

Performing an energy minimisation step will ensure the stability of the generated conformer by effectively removing stresses introduced by the imperfect conformer generation algorithm.

Note that some conformer generation methods, such as ETKDG, do not need an energy minimisation step, however if conformer energies are needed, these have to be calculated based on a force field. In this case, energy minimisation based on the same forcefield needs to be performed as, unless this is done, since the conformer generation parameters will, in general, not match those of the force field, abnormally high energy

values will result. In general however, these kinds of conformer generation algorithms produce configurations that are very close to energy minima, and so the energy minimisation step will tend to be relatively quick.

References

Qurrat Ul Ain, Antoniya Aleksandrova, Florian D. Roessler, and Pedro J. Ballester. Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 5(6):405–424, 2015.

Ayedh Alqahtani and Andrew Whyte. Estimation of life-cycle costs of buildings: regression vs artificial neural network. *Built Environment Project and Asset Management*, 2016.

Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

Stuart M. Armstrong, Garrett M. Morris, Paul W. Finn, Raman Sharma, and W. Graham Richards. Molecular similarity including chirality. *J. Mol. Graph. Model.*, 28(4):368–370, 2009.

Stuart M. Armstrong, Garrett M. Morris, Paul W. Finn, Raman Sharma, Loris Moretti, Richard I. Cooper, and W. Graham Richards. ElectroShape: Fast molecular similarity calculations incorporating shape, chirality and electrostatics. *J. Comput. Aided. Mol. Des.*, 24(9):789–801, 2010.

Stuart M. Armstrong, Paul W. Finn, Garrett M. Morris, and W. Graham Richards. Improving the accuracy of ultrafast ligand-based screening: Incorporating lipophilicity into ElectroShape as an extra dimension. *J. Comput. Aided. Mol. Des.*, 25(8):785–790, 2011.

Pedro J. Ballester. Ultrafast shape recognition: method and applications. *Future Med. Chem.*, 3(1):65–78, 2011.

Pedro J. Ballester and W. Graham Richards. Ultrafast shape recognition for similarity search in molecular databases. *Proc. R. Soc. A Math. Phys. Eng. Sci.*, 463(2081):1307–1321, 2007a.

Pedro J. Ballester and W. Graham Richards. Ultrafast shape recognition to search compound databases for similar molecular shapes. *J. Comput. Chem.*, 28(10):1711–23, jul 2007b.

Pedro J. Ballester, Paul W. Finn, and W. Graham Richards. Ultrafast shape recognition: Evaluating a new ligand-based virtual screening technology. *J. Mol. Graph. Model.*, 27(7):836–845, 2009a.

Pedro J Ballester, Isaac Westwood, Nicola Laurieri, Edith Sim, and W Graham Richards. Prospective virtual screening with Ultrafast Shape Recognition: the identification of novel inhibitors of arylamine N-acetyltransferases. *J. R. S Ioc. Interface*, 7(43):335–342, 2009b.

Lauren Barghout. Spatial-taxon information granules as used in iterative fuzzy-decision-making for image segmentation. In *Granular Computing and Decision-Making*, pages 285–318. Springer, 2015.

Stéphane Betzi, Karsten Suhre, Bernard Chétrit, Françoise Guerlesquin, and Xavier Morelli. GFscore: A General Nonlinear Consensus Scoring Function for High-Throughput Docking. *Journal of Chemical Information and Modeling*, 46(4):1704–1712, jul 2006. ISSN 1549-9596. doi: 10.1021/ci0600758.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial Intelligence and Statistics*, pages 127–135, 2012.

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

Robert D Brown and Yvonne C Martin. Use of structure- activity data to compare structure-based clustering methods and descriptors for use in compound selection. *J. Chem. Inf. Comput. Sci.*, 36(3):572–584, 1996.

Frank K Brown et. al. Chemoinformatics: what is it and how does it impact drug discovery. *Annu. Rep. Med. Chem.*, 33:375–384, 1998.

Edward O. Cannon, Florian Nigsch, and John Bo Mitchell. A novel hybrid ultrafast shape descriptor method for use in virtual screening. *Chem. Cent. J.*, 2(1):1–9, 2008.

Kristy A. Carpenter, David S. Cohen, Juliet T. Jarrell, and Xudong Huang. Deep learning and virtual drug screening. *Future medicinal chemistry*, 10(21):2557–2567, 2018.

Turgay Celik and Tardi Tjahjadi. Automatic image equalization and contrast enhancement using Gaussian mixture modeling. *IEEE Transactions on Image Processing*, 21(1):145–156, 2011.

Beining Chen, Robert F. Harrison, George Papadatos, Peter Willett, David J. Wood, Xiao Qing Lewell, Paulette Greenidge, and Nikolaus Stiefl. Evaluation of machine-learning methods for ligand-based virtual screening. *J. Comput. Aided. Mol. Des.*, 21(1):53–62, jan 2007. ISSN 1573-4951.

Yongliang Chen and Wei Wu. Isolation forest as an alternative data-driven mineral prospectivity mapping method with a higher data-processing efficiency. *Natural Resources Research*, 28(1):31–46, 2019.

Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012a.

Dan C Cireşan, Ueli Meier, and Jürgen Schmidhuber. Transfer learning for latin and chinese characters with deep neural networks. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2012b.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Balázs Csanád Csáji. Approximation with artificial neural networks. *Fac. Sci. Etvs Lornd Univ. Hungary*, 24: 48, 2001.

Arthur Dalby, James G. Nourse, W. Douglas Hounshell, Ann K. I. Gushurst, David L. Grier, Burton A. Leland, and John Laufer. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.*, 32(3):244–255, 1992.

Dennis Decoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine learning*, 46 (1-3):161–190, 2002.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013.

Jean Paul Ebejer, Garrett M. Morris, and Charlotte M. Deane. Freely available conformer generation methods: How good are they? *J. Chem. Inf. Model.*, 52(5):1146–1158, 2012.

Hanna Eckert and Jürgen Bajorath. Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches. *Drug Discov. Today*, 12(5-6):225–233, 2007.

Li Fei-Fei. Knowledge transfer in learning to recognize visual objects classes. In *Int. Conf. Dev. Learn.*, pages 1–8, 2006.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.

Paul W. Finn and Garrett M. Morris. Shape-based similarity searching in chemical databases. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 3(3):226–241, 2013.

Emil Fischer. Einfluss der configuration auf die wirkung der enzyme. *Berichte der deuts chemischen Gesellschaft*, 27(3):2985–2993, 1894.

Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Res.*, 40(D1):1100–1107, 2012.

Hanna Geppert, Martin Vogt, and Jürgen Bajorath. Current Trends in Ligand-Based Virtual Screening: Molecular Representations, Data Mining Methods, New Application Areas, and Performance Evaluation. *J. Chem. Inf. Model.*, 50(2):205–216, 2010.

Michael R Gerhardt, Liuchuan Tong, Rafael Gómez-Bombarelli, Qing Chen, Michael P Marshak, Cooper J Galvin, Alán Aspuru-Guzik, Roy G Gordon, and Michael J Aziz. Anthraquinone derivatives in aqueous flow batteries. *Advanced Energy Materials*, 7(8):1601488, 2017.

Andrew C. Good and W. Graham Richards. Explicit calculation of 3D molecular similarity. *Perspect. drug Discov. Des.*, 9:321–338, 1998.

J. A. Grant and B. T. Pickup. A Gaussian description of molecular shape. *J. Phys. Chem.*, 99(11):3503–3510, 1995.

J. A. Grant, M. A. Gallardo, and B. T. Pickup. A fast method of molecular shape comaprison: a simple application of a Gaussian description of molecular shape. *J. Comput. Chem.*, 17(14):1653–1666, 1996.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

Peter Hall. A distribution is completely determined by its translated moments. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 62(3):355–359, sep 1983.

John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

Yueyang He, Xiaoyan Zhu, Guangtao Wang, Heli Sun, and Yong Wang. Predicting bugs in software code changes using isolation forest. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 296–305. IEEE, 2017.

Jérôme Hert, Peter Willett, David J. Wilton, Pierre Acklin, Kamal Azzaoui, Edgar Jacoby, and Ansgar Schuffenhauer. Enhancing the effectiveness of similarity-based virtual screening using nearest-neighbor information. *J. Med. Chem.*, 48(22):7049–7054, 2005.

Jérôme Hert, Peter Willett, David J. Wilton, Pierre Acklin, Kamal Azzaoui, Edgar Jacoby, and Ansgar Schuffenhauer. New methods for ligand-based virtual screening: Use of data fusion and machine learning to enhance the effectiveness of similarity searching. *J. Chem. Inf. Model.*, 46(2):462–470, 2006.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

Kun-Yi Hsin, Hugh P. Morgan, Steven R. Shave, Andrew C. Hinton, Paul Taylor, and Malcolm D. Walkinshaw. EDULISS: a small-molecule database with data-mining and pharmacophore searching capabilities. *Nucleic Acids Res.*, 39(suppl_1):D1042—-D1048, 2010.

Niu Huang, Brian K. Shoichet, and John J. Irwin. Benchmarking sets for molecular docking. *J. Med. Chem.*, 49(23):6789–6801, 2006.

Andreas Jahn, Georg Hinselmann, Nikolas Fechner, Carsten Henneges, and Andreas Zell. Probabilistic modeling of conformational space for 3D machine learning approaches. *Mol. Inform.*, 29(5):441–455, 2010.

Andreas Jahn, Lars Rosenbaum, Georg Hinselmann, and Andreas Zell. 4D flexible atom-pairs: An efficient probabilistic conformational space comparison for ligandbased virtual screening. *J. Cheminform.*, 3(7): 23, 2011.

Tony Jebara, Risi Kondor, Andrew Howard, Kristin Bennett, and Nicoì O Cesa-Bianchi. Probability Product Kernels. Technical report, 2004.

Mark A. Johnson and Gerald M. Maggiora. Concepts and applications of molecular similarity, 1990.

Eric R Kandel, James H Schwartz, and Thomas M Jessell. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Rafal Kurczab, Sabina Smusz, and Andrzej Bojarski. Evaluation of different machine learning methods for ligand-based virtual screening. *J. Cheminform.*, 3(1):P41, apr 2011. ISSN 1758-2946.

Greg Landrum and Others. RDKit: cheminformatics and machine learning software. *RDKIT. ORG*, 2013.

Antonio Lavecchia. Machine-learning approaches in drug discovery: methods and applications. *Drug Discov Today*, 20(3):318–331, 2015.

Antonio. Lavecchia and Carmen Di Giovanni. Virtual screening strategies in drug discovery: A critical review. *Curr. Med. Chem.*, 20(23):2839–2860, 2013.

Andrew R. Leach and Valerie J. Gillet. *An Introduction to Cheminformatics*. Springer, Sheffield, revised ed edition, 2007. ISBN 9781402062902.

Hongjian Li, Kwong-S Leung, Man-H Wong, and Pedro J Ballester. USR-VS: a web server for large-scale prospective virtual screening using ultrafast shape recognition techniques. *Nucleic acids research*, 44(W1): W436—-W441, 2016.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE Int. Conf. Data Min.*, pages 413–422. IEEE, 2008.

Paul D. Lyne. Structure-based virtual screening: An overview. *Drug Discov. Today*, 7(20):1047–1055, 2002.

Pierre Mahé, Liva Ralaivola, Véronique Stoven, and Jean-Philippe Vert. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model.*, 46(5):2003–2014, 2006.

Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.*, pages 50–60, 1947.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Adv. Neur. In.*, pages 3111–3119, 2013.

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1997.

László Molnár and György M Keserű. A neural network based virtual screening of cytochrome p450 3a4 inhibitors. *Bioorganic & medicinal chemistry letters*, 12(3):419–421, 2002.

Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6): 275–285, 2004.

Michael M. Mysinger, Michael Carchia, John J. Irwin, and Brian K. Shoichet. Directory of useful decoys, enhanced (DUD-E): Better ligands and decoys for better benchmarking. *J. Med. Chem.*, 55(14):6582–6594, 2012.

Maged Nasser, Naomie Salim, Hentabli Hamza, and Faisal Saeed. Deep belief network for molecular feature selection in ligand-based virtual screening. In *International Conference of Reliable Information and Communication Technology*, pages 3–14. Springer, 2018.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

K Omata, Y Kobayashi, and M Yamada. Artificial neural network aided virtual screening of additives to a co/srco3 catalyst for preferential oxidation of co in excess hydrogen. *Catalysis Communications*, 8(1):1–5, 2007.

Janaina Cruz Pereira, Ernesto Raul Caffarena, and Cicero Nogueira dos Santos. Boosting docking-based virtual screening with deep learning. *Journal of chemical information and modeling*, 56(12):2495–2506, 2016.

Sameer S. Pradhan, Wayne H. Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 233–240, 2004.

Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

Douglas A. Reynolds. Speaker identification and verification using Gaussian mixture speaker models. *Speech communication*, 17(1-2):91–108, 1995.

Douglas A. Reynolds and Richard C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83, 1995.

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Thomas S. Rush, J. Andrew Grant, Lidia Mosyak, and Anthony Nicholls. A shape-based 3-D scaffold hopping method and its application to a bacterial protein- protein interaction. *J. Med. Chem.*, 48(5): 1489–1495, 2005.

D. Hari Hara Santosh, Poornesh Venkatesh, P. Poornesh, L. Narayana Rao, and N. Arun Kumar. Tracking multiple moving objects using gaussian mixture model. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(2):114–119, 2013.

Tomohiro Sato, Hitomi Yuki, Daisuke Takaya, Shunta Sasaki, Akiko Tanaka, and Teruki Honma. Application of support vector machine to three-dimensional shape-based virtual screening using comprehensive three-dimensional molecular shape overlay with known inhibitors. *J. Chem. Inf. Model.*, 52(4):1015–1026, 2012.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Adrian Schreyer and Tom Blundell. CREDO: a protein–ligand interaction database for drug discovery. *Chem. Biol. Drug Des.*, 73(2):157–167, 2009.

Adrian M. Schreyer and Tom Blundell. USRCAT: Real-time ultrafast shape recognition with pharmacophoric constraints. *J. Cheminform.*, 4(11), 2012.

Paul Selzer and Peter Ertl. Identification and Classification of GPCR Ligands Using Self-Organizing Neural Networks. *QSAR Comb. Sci.*, 24(2):270–276, 2005.

Paul Selzer and Peter Ertl. Applications of self-organizing neural networks in virtual screening and diversity selection. *Journal of chemical information and modeling*, 46(6):2319–2323, 2006.

Mohamed A Shahin, Holger R Maier, and Mark B Jaksa. Data division for developing neural networks applied to geotechnical engineering. *Journal of Computing in Civil Engineering*, 18(2):105–114, 2004.

Steven R. Shave. Development of high performance structure and ligand based virtual screening techniques. 2010.

Matthew A. Siegler, Uday Jain, Bhiksha Raj, and Richard M. Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA speech recognition workshop*, volume 1997, 1997.

Florence Stahura and Jurgen Bajorath. Virtual Screening Methods that Complement HTS. *Comb. Chem. High Throughput Screen.*, 7(4):259–269, 2004.

Florence L Stahura and Jürgen Bajorath. New methodologies for ligand-based virtual screening. *Curr. Pharm. Des.*, 11(9):1189–1202, 2005.

Matthew Nicholas Stuttle. *A Gaussian mixture model spectral representation for speech recognition*. PhD thesis, University of Cambridge, 2003.

Gian Antonio Susto, Alessandro Beghi, and Seán McLoone. Anomaly detection through on-line isolation forest: An application to plasma etching. In *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 89–94. IEEE, 2017.

Vishwesh Venkatraman, Violeta I. Pérez-Nueno, Lazaros Mavridis, and David W. Ritchie. Comprehensive comparison of ligand-based virtual screening tools against the DUD data set reveals limitations of current 3D methods. *J. Chem. Inf. Model.*, 50(12):2079–2093, 2010.

Manfred K. Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active Learning with Support Vector Machines in the Drug Discovery Process. *J. Chem. Inf. Comput. Sci.*, 43(2):667–673, mar 2003. ISSN 0095-2338.

David Weininger. SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, 1988.

Paul Werbos. Beyond Regression:" New Tools for Prediction and Analysis in the Behavioral Sciences. *Ph. D. Diss. Harvard Univ.*, 1974.

Peter Willett. Similarity-based virtual screening using 2D fingerprints. *Drug Discov. Today*, 11(23-24):1046–1053, 2006.

Peter Willett, John M. Barnard, and Geoffrey M. Downs. Chemical similarity searching. *J. Chem. Inf. Comput. Sci.*, 1998.

David A Winkler and Frank R Burden. Application of neural networks to large dataset qsar, virtual screening, and library design. In *Combinatorial Library*, pages 325–367. Springer, 2002.

Maciej Wójcikowski, Pedro J. Ballester, and Pawel Siedlecki. Performance of machine-learning scoring functions in structure-based virtual screening. *Sci. Rep.*, 2017.

Tao Xiao, Xingxing Qi, Yuzong Chen, and Yuyang Jiang. Development of ligand-based big data deep neural network models for virtual screening of large compound libraries. *Mol. Inform.*, 37(11):1800031, 2018.

Randy J. Zauhar, Guillermo Moyna, LiFeng Tian, ZhiJian Li, and William J. Welsh. Shape signatures: a new approach to computer-aided ligand-and receptor-based drug design. *J. Med. Chem.*, 46(26):5674–5690, 2003.

Linfei Zhou, Wei Ye, Bianca Wackersreuther, Claudia Plant, and Christian Bohm. A Pseudometric for Gaussian Mixture Models. In *DBKDA 2017 Ninth Int. COngerence Adv. Databases, Knowl. Data Appl.*, pages 37–42, 2017.