UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Image-based chemical property descriptors for antimalarial drug screening

Author:
Núria CAMÍ

Supervisor:
Dr. Miquel DURAN-FRIGOLA
Dr. Gemma TURON
Dr. Jordi VITRIÀ

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2022

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Image-based chemical property descriptors for antimalarial drug screening

by Núria CAMÍ

Nowadays there is still an undeniable inequality in biomedical research for Low and Middle Income Countries, which produce less than 5% of the world's research and their population relies on healthcare solutions devised abroad. Particularly, Malaria remains one of the deadliest infectious diseases worldwide, in fact worsened by the COVID-19 pandemic. In order to speed up research and reduce the cost of experiments for drug discovery, Artificial Intelligence has an important role to play.

MolMap is a powerful Deep Learning tool capable of capturing important molecular representations of compounds (image-based) for enhanced prediction of pharmaceutically relevant properties. In this project we will explore the possibilities of this method for its application to a particular set of compounds and their corresponding activity against the Malaria pathogen. The resulting performance of the MolMap Convolutional Neural Network once trained on this data will be crucial for later making predictions and inferring on those molecules most suitable for developing new drugs.

Acknowledgements

First of all, I would like to sincerely express my gratitude to Dr. Gemma Turon and Dr. Miquel Duran-Frigola, who have guided me through the whole process of understanding the problem and building a feasible solution. They both have supported and helped me whenever I needed it, and the knowledge gained is noticeable. For me it has been such a pleasure to work hand in hand with them and being, during these months, part of Ersilia.

Secondly, I would also like to thank my university supervisor Dr. Jordi Vitrià for his guidance, inspiration and advice in critical moments of the process. His contribution has undoubtedly been indispensable for a correct execution of the project.

And finally, big thanks to my family, friends and master colleagues who have accompanied and supported me throughout the entire process.

Contents

Glossary of Abbreviations and Acronyms

AI Artificial Intelligence

CE Cross-Entropy

CNN Convolutional Neural Network

DL Deep Learning

EOSI Ersilia Open Source Initiative

FFs Fingerprints Features

Fmaps Feature Maps

GCN Graph Convolutional Network

k-**NN** *k* Nearest Neighbors

ML Machine Learning

MMV Medicines for Malaria Venture

MolDs Molecular Descriptors

MolRs Molecular Representations

MSE Mean Squared Error

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SMILES Simplified Molecular Input Line Entry System

SOTA State of the Art

tSNE tdistributed Stochastic Neighbor Embedding

UMAP Uniform Manifold Approximation and Projection

Chapter 1

Introduction

1.1 Project statement

The current project focuses on the exploitation of the Medicines for Malaria Venture (MMV) dataset on the activity of putative antimalarial compounds targeting asexual blood stages of the *Plasmodium falciparum* parasite. The aim of this project is to use an appropriate Deep Learning tool on this data and deliver it as an open-source asset to the whole scientific community.

To tackle this problem, we capitalize on the **MolMap library** (Shen et al., 2021). This method is capable of generating images (or maps) from molecular features of compounds that can be used to train a Convolutional Neural Network for regression and classification tasks. We formulated our problem as a binary classification of activity/inactivity against the Malaria pathogen.

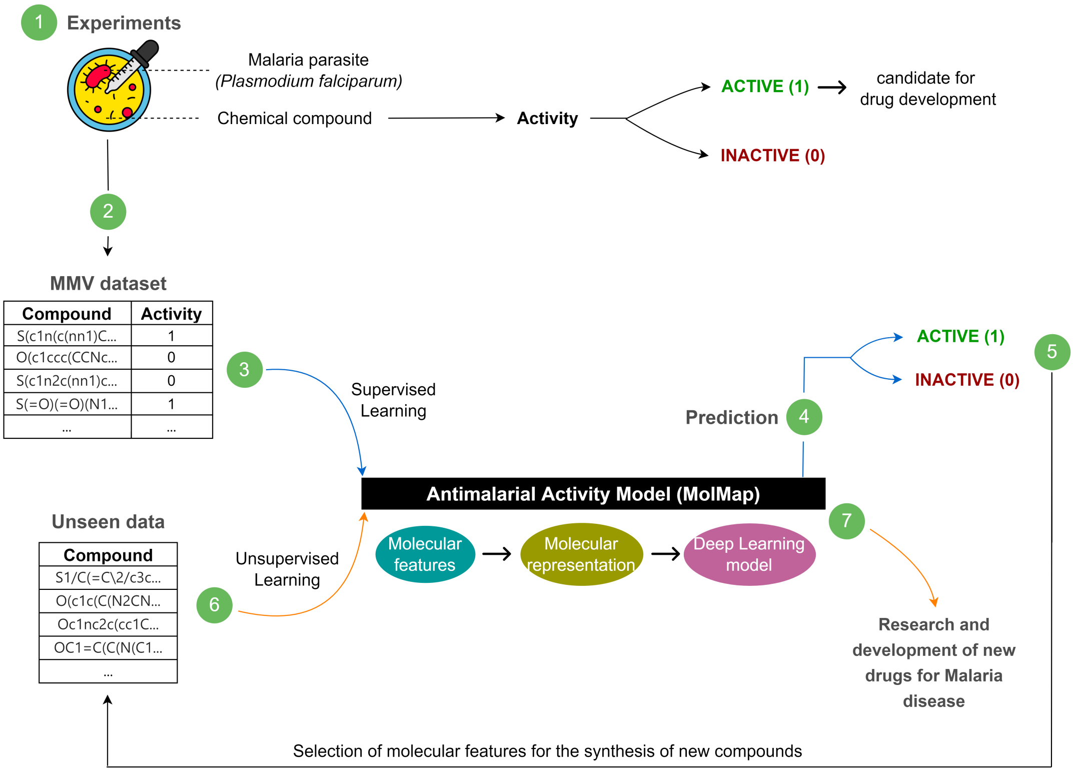
1.1.1 Motivation, goals and methodology

Malaria is a disease caused by a parasite (*Plasmodium falciparum*) that is transferred via the bite of infected mosquitoes. Global South countries are actually the most affected ones by this disease, where around 240 million people are infected each year and more than 600.000 die (*World malaria report 2021* 2021).

Although approved treatments exist, the parasite tends to develop resistance to the current drugs. For this reason, there is a need for novel therapeutic strategies involving new chemical entities. In a context of limited funding and low expected return on investment, the final objective for research institutions working on Malaria **drug discovery** is to reduce the number of experiments (development cycle) needed for finding new effective drugs that overcome this disease.

Within this context, we had at our disposal a dataset containing several experiments performed on a list of **molecule compounds** and their corresponding **activity** when the Malaria pathogen had been exposed to them. By training an appropriate Deep Learning model, we will potentially be able to predict and infer on those molecules most suitable for developing new drugs.

Accordingly, our main goals have been understanding, analyzing and training a competitive **antimalarial activity model** (supervised learning) capable of predicting the activity, but also efficient for distinguishing relevant **image-based Molecular Representations (MolRs)**. These MolRs will allow synthetic chemists to understand the key **molecular features** and move the most promising candidates forward in the drug discovery pipeline. Figure 1.1 summarizes the main ideas just described.



Compound	Activity
S(c1n(c(nn1)C...	1
O(c1ccc(CCNc...	0
S(c1n2c(nn1)c...	0
S(=O)(=O)(N1...	1
...	...

Compound
S1/C(=C\2/c3c...
O(c1c(C(N2CN...
Oc1nc2c(cc1C...
OC1=C(C(N(C1...
...

FIGURE 1.1: The diagram shows the key steps for the development process of antimalarial drug screening applied to MolMap method.

At the beginning of this chapter we already pointed to MolMap, a powerful tool that has made it possible to fulfill our goals. However, before applying MolMap to the provided dataset, we first checked how the method worked with a smaller dataset and in comparison to other baseline models. In addition, we studied the impact of changing the parameters of the model and analyzed its robustness regarding the image reconstruction. The implementation of the default version of MolMap turned out not to be suitable for large datasets like the MMV activity data. Thus, one of the most important achievements of the project it has been presenting a solution for this particular problem.

1.2 Ersilia Open Source Initiative

The Ersilia Open Source Initiative (EOSI) is a non-profit organization aimed at strengthening the research capacity against infectious and neglected diseases in Low and Middle Income Countries. They tackle their mission by leveraging the potential of artificial intelligence and Machine Learning (AI/ML) to speed up research and decrease experimental costs. EOSI's overarching goal is to achieve a reference open-source online repository, named the Ersilia Model Hub, that serves pre-trained, ready-to-use AI/ML tools to the scientific community. The Hub contains curated AI/ML assets published in the literature in addition to in-house assets developed by EOSI in coordination with research institutions that are aligned with their mission.

Ersilia's team is mainly managed by Dr. Gemma Turon (Co-Founder & CEO) and Dr. Miquel Duran-Frigola (Co-Founder, CSO & Trustee), both of them experts in chemistry, molecular biology and computational pharmacology with an interest for global health.

Chapter 2

Background

2.1 State of the art

Deep learning (DL) is beginning to impact biomedical research and, particularly, in the different stages of drug discovery. This has been possible as a result of the ability of DL to integrate large datasets, learn arbitrarily complex relationships and incorporate existing biomedical knowledge (Vamathevan et al., 2019; Wainberg et al., 2018). Primary examples of this progress cover target validation, identification of prognostic biomarkers and analysis of digital pathology data in clinical trials.

In particular, the applications of Deep Neural Networks architecture in drug discovery have been numerous and include bioactivity and synthesis path prediction, *de novo* molecular design, biological image analysis and, the most important for our purpose: **learning of pharmaceutical properties through feature maps with a Convolutional Neural Network structure**.

Convolutional Neural Networks (CNNs) are one of the most powerful structures in the DL field. They achieve an outstanding predictive performance in areas such as speech and image recognition by hierarchically composing simple local features into complex models. There exist other variants such as Graph Convolutional Networks (GCNs), that can be applied to structured data in the form of graphs or networks, or Recurrent Neural Networks (RNNs), which takes the form of a chain of repeating modules of neural networks in which connections between nodes form a directed graph along a sequence (Vamathevan et al., 2019).

Learning of pharmaceutical properties has been mainly conducted by four different types of MolR classes (Shen et al., 2021):

- Graph-based feature representations: GCNs that have been explored for learning directly from the underlying graphs of molecules.
- String-based representations: CNNs and RNNs that have been employed for learning from the embeddings of the string representations of chemical structures (such as SMILES, see Chapter 2.2.1).
- Image representations: CNNs that have been used for learning from the rule-based renderings of a 2D chemical digital grid.
- Knowledge-based representations: DL models that have been developed to learn from chemical properties (see Chapter 2.2.2) derived from human knowledge.

Indeed, the broader learning of the expert human knowledge bases have facilitated an extensive learning of the pharmaceutical properties used for drug discovery by selecting appropriate MolRs. Although notable progress has been made lately in knowledge-based representations, Shen et al., 2021 emphasize that the combined potential of human expert knowledge of molecular representations and CNNs has not been adequately explored for enhanced learning of pharmaceutical properties. For this reason, new tools such as **MolMap** have been developed as a state-of-the-art (SOTA) software for **knowledge-based representations**.

In brief, MolMap facilitates the generation of maps that capture the intrinsic correlations of molecular features, which can be used for training a CNN. Thus, the shared-weight CNN architectures can be exploited for enhanced learning and prediction of pharmaceutical properties.

To reduce the technical barrier and support wider applications, the CNN used in MolMap (named **MolMapNet**) has demonstrated to be highly competitive against established models on most of the 26 benchmark datasets. Of note, the benchmark datasets included the MoleculeNet (Wu et al., 2017).

MoleculeNet is the most popular benchmark specially designed for testing ML methods of molecular properties, belonging to a diverse set of tasks, ranging from biophysical to physiological predictions. This work curates a number of public dataset collections, establishes metrics for evaluation, and offers high quality open-source implementations of multiple previously proposed molecular featurization and learning algorithms. Having MolMap software benchmarked on datasets such as MoleculeNet guarantees to be a powerful and qualified tool for molecular Machine Learning.

2.2 Basics of molecular chemistry

2.2.1 SMILES

Designing and discovering molecules in an efficient way is a critical step for optimization in drug discovery (McNaughton et al., 2022). At the same time, the first step in the formalization of chemistry is to name a chemical compound, which requires an unambiguous and reproducible notation for the simplest atom to the most complicated structure.

SMILES (Simplified Molecular Input Line Entry System) is a chemical notation system designed for chemical information processing (Weininger, 1988). Figure 2.1 shows an example that may help its understanding.

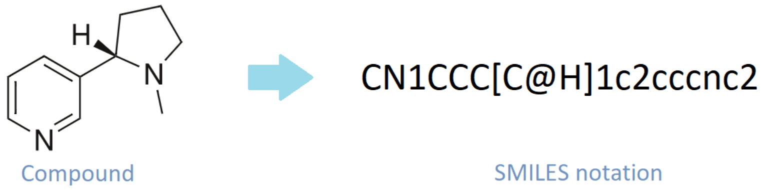


FIGURE 2.1: Compound and SMILES notation of Nicotine. Source: adapted from https://es.wikipedia.org/wiki/SMILES.

SMILES format turned out to be a universal standard used in many cheminformatics software. In fact, the most important thing is that each SMILES string contains structural information that can be used to calculate complex molecular features (Goh et al., 2017). These molecular features are also known as chemical property descriptors, and will be presented in the next section.

2.2.2 Chemical property descriptors

Chemical property descriptors (or simply, **chemical descriptors**) are widely employed to represent molecular characteristics in cheminformatics. The intuition behind these properties is that molecules that have similar molecular descriptors, will be also similar to each other and behave similarly in biological systems. Besides, they are used to predict the activity and other properties resulting from the chemical structures of compounds.

A unique characteristic of the MolMap approach is that it tries to capture most of the existing, well-accepted chemical descriptors in a unified format (an image or a map). Broadly speaking, there are two families of property descriptors: the **Molecular Descriptors (MolDs)** and the **Fingerprints Features (FFs)**. In order to avoid confusing the reader, it is worth stressing that, although the general concept to refer to these properties has the term 'descriptors', it doesn't just refer to MolDs, but to both MolDs and FFs.

The main trait for distinguishing between MolDs and FFs is that the first one typically represent continuous properties, such as the number of atoms of the molecule, the molecular weight or the number of rings. In contrast, the second one stands for binary (structural) features, such as if the molecule has a chloride atom or not, or a double bond or not, etc.

Although it is preferable to explore the lower-level representations without relying on human intuitions, the extensive knowledge bases of MolDs and FFs are highly useful for learning MolRs and pharmaceutical properties from human-knowledge perspectives (Shen et al., 2021). One thing that has been explored is the correlation between pairs of MolDs and FFs, respectively. For example, it is easy to understand that the molecular weight is directly proportional to the number of atoms, so these two properties are correlated. This is a very simple example, but the idea is that there are some other 'unrelated' properties that have shown high degrees of intrinsic correlation, leading to meaningful discoveries. To give an example, it is known that the polar surface area correlates with the counts of hydrogen bond acceptors and donors (Shen et al., 2021, Clemons et al., 2011).

Referring to the code part, the open-source library RDKit (Landrum et al., 2022) makes the computation of these properties possible from molecules represented as SMILES. This computation outputs a matrix with molecules in the rows and descriptors (features) in the columns. For its part, MolMap has been designed to transform these matrices into robust 2-D maps used as input for training the MolMapNet CNN. In order to embed all these amount of descriptors (that can be on the order of 10^3 or higher) in 2-dimensional images, MolMap uses a dimensionality reduction technique called UMAP.

2.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction (McInnes et al., 2018). In short, UMAP assumes that the provided data is equally (**uniformly**) distributed across a topological space (**manifold**), which can be **approximated** from these finite data samples and mapped (**projected**) to a lower-dimensional space (Dobilas, 2021). Apart from performing supervised dimensionality reduction, it is also commonly used for unsupervised learning.

The first step of the UMAP procedure consists in learning the manifold structure in the high-dimensional space, which is possible by finding the nearest neighbors using the **Nearest-Neighbor-Descent algorithm** of Dong, Charikar, and Li, 2011. The number of neighbors plays an important role here: a small value for neighbors will imply a very local interpretation that accurately captures the fine detail of the structure, while a large neighbors value will mean that our estimates will be based on larger regions, thus more broadly accurate across the manifold as a whole (Dobilas, 2021).

At the same time, the hypothesis of assuming uniform distribution of points across the manifold suggests that there may be regions more dispersed and others more condensed, which implies a variance in the distance metric. A simple way to visualize it is by drawing circles of different size around each data point, as it is shown in Figure 2.2.

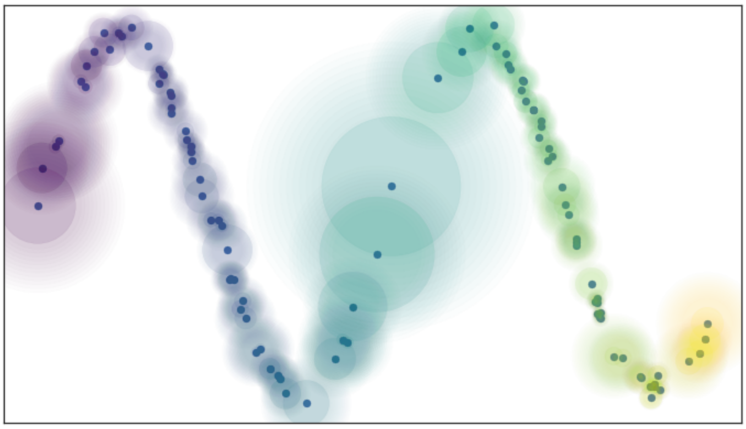


FIGURE 2.2: Local connectivity and fuzzy open sets. Source: UMAP documentation.

These fuzzy circles that are extended beyond the closest neighbor represent the certainty of connection with other points, which indeed decreases as we get farther away from the point of interest. This connection certainty is expressed through **edge weights** (w). In addition, in order to avoid having unconnected points, it is possible to indicate to the algorithm the minimum number of points to which each point must be connected.

At last, UMAP ends up constructing a graph by connecting the previously identified nearest neighbors (Figure 2.3).

The second step consists in finding a low-dimensional space in which this connected neighborhood graph can be represented with the **Standard Euclidean distance**. Therefore, one of the first things to specify is the minimum distance between the embedded points (by default, 0.1). The new representation is found by minimizing the **Cross-Entropy (CE)** as a cost function:

$$CE = \sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right),$$

where e refers to a particular edge from the set of edges E found in step 1, $w_h(e)$ to the known weights of edges from high-dimensional manifold approximation (step 1) and $w_h(e)$ to the weights to be discovered for low-dimensional representation (step 2).

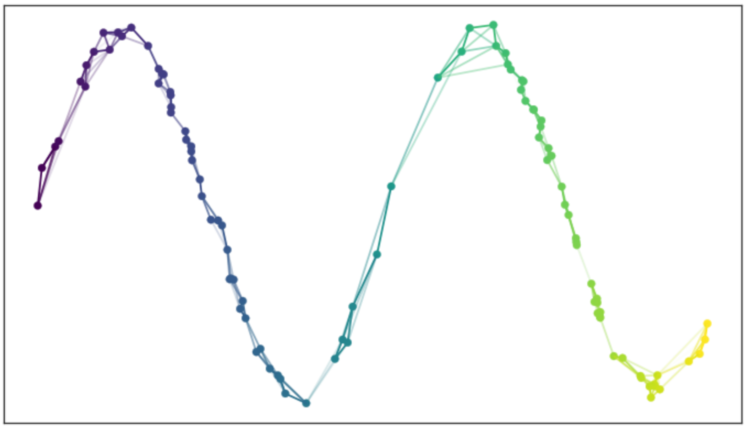


FIGURE 2.3: Graph with combined edge weights. Source: UMAP documentation.

Note that the ultimate goal is finding the optimal weights of edges in the low-dimensional representation, which arise as the above Cross-Entropy cost function is minimized following an iterative **Stochastic Gradient Descent (SGD)** process (Dobilas, 2021).

To conclude, we'll briefly discuss some advantages of UMAP in front of **tSNE (tdistributed Stochastic Neighbor Embedding)**, which is a popular technique used for similar purposes.

The major difference is that UMAP is capable of capturing the **global data structure**, while tSNE can only model locally. This means that for tSNE only within cluster distances are meaningful, while between cluster similarities are not guaranteed. The reason behind this is that tSNE uses the KL-divergence as a cost function, while UMAP uses the CE (Oskolkov, 2019).

Another remarkable difference is that UMAP uses the SGD instead of the regular Gradient Descent as tSNE does. In practice, this implies a speed up in computation and less memory consumption for UMAP.

UMAP overall follows the philosophy of tSNE, but introduces a number of improvements that have made us decide to use it in one of the intermediate steps of MolMap procedure (Chapter 3.3).

2.4 Convolutional Neural Networks

In this section we'll briefly introduce the main ideas for a good understanding of Convolutional Neural Networks. In particular, we'll empathize with those concepts that configure the MolMapNet architecture.

In simple words, a **Convolutional Neural Network (CNN)** is a Deep Learning algorithm that, given an input image (or a feature vector), is capable of reducing it into a form which is easier to process without losing features that are critical for getting a good prediction. This can be achieved by assigning learnable **weights and biases** to various aspects/objects in the image (that is, giving relative importance), in order to later differentiate one image from another (Saha, 2018).

As an example, recall the MNIST dataset, composed of images of 28 x 28 x 1 pixels. To clarify, the first two dimensions correspond to the height and width, respectively, and the third one to the number of color channels (black & white image contains only 1 channel). In this case, the total number of neurons in the input layer will be 28 x 28 = 784, which is manageable enough. But, what if the size of the images are 1.000 x 1.000 instead? Actually, having 10^6 or more neurons in the input layer is definitely not a problem for CNN, and this is what makes them powerful (Shahid, 2019). To get an idea, let's first take a glimpse on how the architecture of a CNN looks like (Figure 2.4).

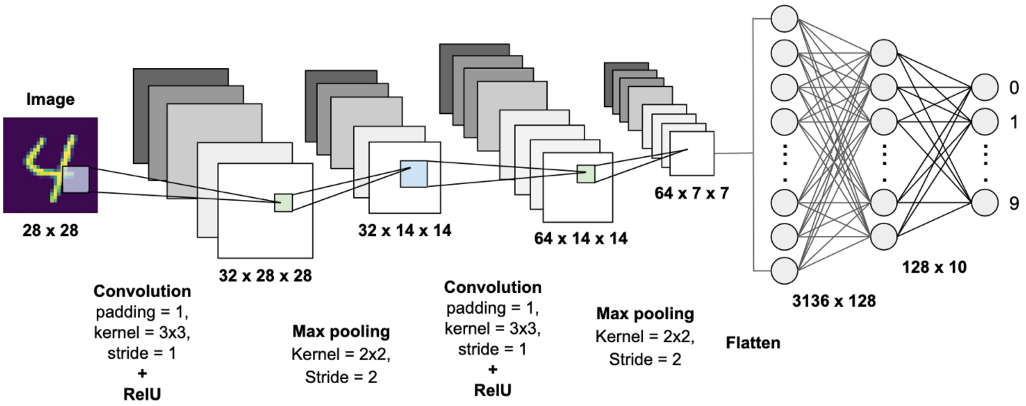


FIGURE 2.4: Scheme of a Convolutional Neural Network. Source: https://becominghuman.ai/building-a-convolutional-neural-network-cnn-model-for-image-classification-116f77a7a236.

The first layer corresponds to the **Convolutional Layer** and is the result of a dot product between the filter matrix values and the pixel matrix values. Here is where the learning takes place by extracting the image features. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel** or filter (Saha, 2018). In this process, an amount of pixels (**Padding**) are added to the images. Besides, the **Stride** denotes how many steps we are moving in each step in convolution (by default, 1)(Shahid, 2019).

After the image is passed through the first filter, it then goes on to the **Pooling Layer**. This layer reduces the size of the filter layer, which allows us to train the model faster. Also, it prevents overfitting by dropping unwanted values in the filter tensor. A popular pooling method is called **Max Pooling**, which is the same process as filtering but taking the maximum value from the portion of the image covered by the Kernel instead of taking the dot product (Ferdinand, 2020).

Before entering our convoluted input into a dense (fully-connected) layer, we must **flatten** it, which means turning from multi-dimensional to a 1-D input tensor. The flattened input is then passed through the **Dense Layers**. Here, each neuron receives input from all the neurons of the previous layer and, when the final layer is reached, ends up giving a predicted outcome.

In most cases, the **dropout** regularization method is also applied. Dropping out neurons is randomly setting some of the output neurons to zero, just to speed up the training and backpropagation process, and to prevent overfitting our network (Ferdinand, 2020). An alternative could be to apply **weight decay**, which works by adding a penalty term to the cost function of a neural network. To recap, the **cost function** is the one that computes the distance between the current output of the algorithm and the expected output (Pere, 2020).

At this point, if we didn't do any other transformation in our network, it would be a simple linear regression algorithm. In order to be able to solve much more complex problems than a linear one, some non-linearity has to be added. Here is where the **activation function** takes place. The most common one is the **Re-Lu (Rectified**

Linear Unit), which outputs the input directly if it is positive, and zero otherwise (Figure 2.5).

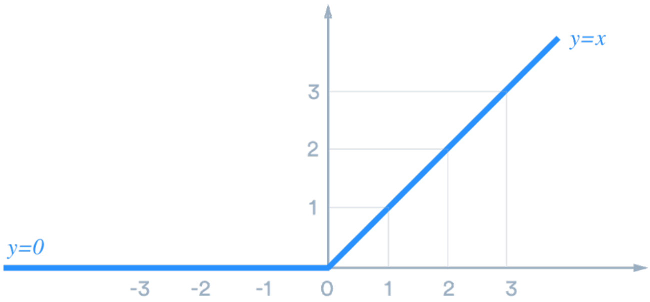


FIGURE 2.5: The Re-Lu Activation function. Source: https://towardsdatascience.com/a-simple-guide-to-convolutional-neural-networks-751789e7bd88.

Now, by training the CNN on some number of images, the parameters of the model will be also determined. Note that these parameters are precisely the weights and biases mentioned at the beginning of this section, and are decisive for making predictions.

The most common loss functions for evaluating the accuracy of these predictions are Cross-Entropy and Mean Squared Error (MSE). **Cross-Entropy** calculates the difference between the probability distribution of the network's output and the probability distribution of the labels. On the other hand, **Mean Squared Error** is computed as the average of the squared differences between the predicted and actual values (Ferdinand, 2020). Besides, the goal of any optimization problem is to minimize the cost function, i.e., finding the best weights/biases which will return the lowest cost (loss). The most common **optimizer** for CNN is **Adam**.

The **MolMapNet** model provided by MolMap is a CNN specifically created for learning pharmaceutical properties. A peculiarity of this network is its **bimodality**, meaning that it is capable of learning from feature vectors of both MolDs and FFs at once. Further explanation of MolMapNet can be found in Chapter 3.3.5.

Chapter 3

Methodology

3.1 MMV dataset

First of all, we should mention that we signed an agreement with MMV, the provider entity of the dataset, which commits us not to disclose any confidential information related to the data content (for example, the chemical formulas of the molecule compounds). For the same reason, the data itself and the corresponding experiments have not been published on the project's GitHub Repository, although all code to train the models and test them is available (check Appendix A for more information).

The provided dataset contains information for up to **226.928 molecules**. In particular, the columns correspond to **14 different experiments** indicating a specific type of experiment where the parasite has been exposed to the molecules. There is a primary screening performed for the full data and then a collection of more expensive and laborious secondary experiments performed only on a subset of the molecules, typically corresponding to active compounds in the primary screening.

The molecule compounds were given in SMILES format and the target variables (i.e. the experiments measurements) in continuous values. In terms of applicability, in a primary screening a scientist wants to know if the compounds are capable of inhibiting the parasite (**active**) and which ones are not (**inactive**), and thus progress to a secondary screening and beyond. For this reason, only **classification tasks** have been considered by doing a proper binarization of the target variable.

Concretely, taking advantage of the magnitude of samples for the primary screening, we just take this one into consideration. In short, this experiment is an asexual blood stage assay that measures the % growth of the parasite after 72h of incubation with the compound (concentration: 2 micromolar).

One standard cutoff used for binarizing this kind of variables is considering the compounds with percentage of growth inhibition greater than or equal to 50% as active, and the remaining ones as inactive. This division resulted in a total of **226.054 molecules** labeled as **inactive (class 0)** and **867 molecules** as **active (class 1)**. Besides, 7 of them were discarded due to a wrong SMILES codification format.

On one hand, we had to deal with a severe class (active/inactive) imbalance. On the other hand, due to the size of the dataset, we also had memory issues. The adopted solutions to these problems plus the full implementation of MolMap on this dataset is explained in Chapter 4.2.

3.2 MolMap library

MolMap is a new feature-generation method that has been developed in order to map molecular descriptors (MolDs) and fingerprint features (FFs) into robust two-dimensional feature maps (Fmaps). These maps, that can be conceived as MolRs, besides capturing the intrinsic correlations of molecular features, are used to train a deep learning model (Convolution-Neural-Network-based MolMapNet) for predicting pharmaceutical properties.

MolMap was trained by broadly profiling 1.456 MolDs and 16.204 FFs of 8.506.205 molecules. The obtained Fmaps were assessed for deep learning of 13 pharmaceutical and 3 physicochemical properties[1] on 26 public benchmark datasets, including MoleculeNet. Finally, the performances of the MolMapNet models were evaluated with respect to those of the SOTA deep learning models on the same benchmark datasets and data splits (Shen et al., 2021). The results obtained suggested that MolMap Fmaps are powerful MolRs, and MolMapNet is useful for learning pharmaceutical properties competitively with respect to the SOTA.

3.3 MolMap Procedure

In the following sections the consecutive steps of MolMap procedure will be detailed. Although not every step is directly displayed when running the MolMap library out-of-the-box, understanding each part of the process is equally interesting and important to comprehend how the final result is achieved.

3.3.1 Distance matrix

The first step consists in building a **distance matrix** of both MolDs and FFs according to some metric. This matrix ensures a proper learning from these chemical properties and enables to capture the intrinsic correlations between them.

MolMap allows to choose between the cosine, the correlation or the Jaccard distance, which are computed as follows:

Cosine distance	Correlation distance	Jaccard distance
$\mathrm{d}_{(x,y)} = 1 - \dfrac{x \cdot y}{\lVert x \rVert \lVert y \rVert}$	$\mathrm{d}_{(x,y)} = 1 - \dfrac{(x-\bar{x})(y-\bar{y})}{\lVert (x-\bar{x}) \rVert \lVert (y-\bar{y}) \rVert}$	$\mathrm{d}_{(x,y)} = 1 - \dfrac{x \cap^{y}}{x \cup^{y}}$

The resultant matrix of MolDs and FFs has as many rows and columns as MolDs and FFs we have computed, respectively (and is, therefore, squared and symmetric). For instance, for the 1.456 MolDs and 16.204 FFs computed in Shen et al., 2021, their corresponding distance matrices had shapes 1.456×1.456 and 16.204×16.204, respectively (Figure 3.1).

3.3.2 Feature 2D embedding (UMAP)

In order to learn from MolDs and FFs with CNNs, we need to represent them on 2D maps in a way such that during the mapping we lose as little information as possible. This can be achieved by a manifold learning algorithm that approximates and properly projects the chemical properties.

The recently developed tool **UMAP** is based on the Riemannian geometry and algebraic topology algorithms, and it has demonstrated excellent capability for this task.

[1] Concretely, 13 different classes of MolDs and 3 different sets of FFs, which coincide precisely with the number of channels of the MolMapNet.

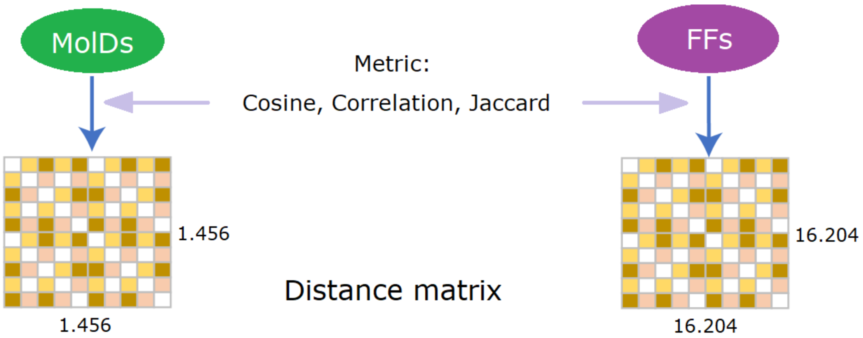


FIGURE 3.1: Scheme of the distance matrix step out of 1.456 MolDs and 16.204 FFs. Source: adapted from Shen et al., 2021.

UMAP is able to project the pairwise distances of MolDs and FFs extracted from the MolMap distance matrices onto a 2D feature space as feature points. What really matters from this transformation is that these feature points embed the broadly learned correlation relationships of the selected MolD classes or FF sets. Figure 3.2 shows the corresponding UMAP projections for the 1.456 MolDs and 16.204 FFs.

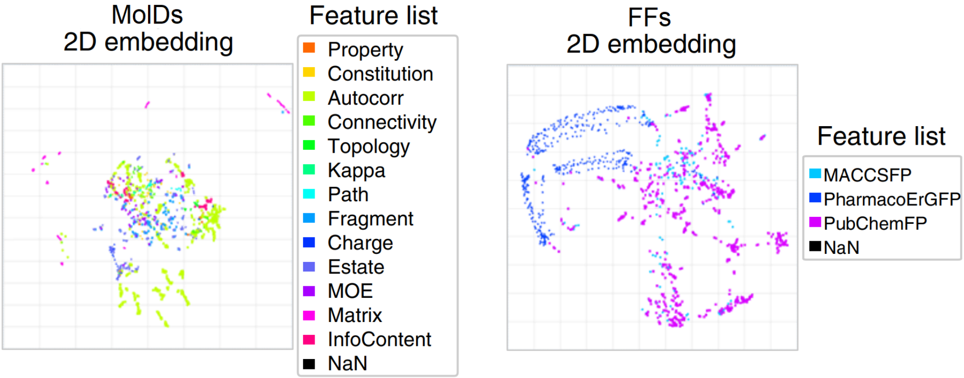


FIGURE 3.2: UMAP projections of 13 classes of MolDs and 3 sets of FFs. Each coloured point in the map represents a particular MolD or FF. Source: adapted from Shen et al., 2021.

Apart from UMAP, the MolMap library also allows using other known statistical methods for visualizing high-dimensional data such as tSNE and Multidimensional Scaling, which can properly be set as a parameter when calling the fit function of MolMap.

3.3.3 Grid assignment

The feature points displayed in UMAP are then assigned to the regular grids of a **2D-grid map** by using the Jonker–Volgenant algorithm for linear assignment (Jonker and Volgenant, 1987). This algorithm is used for minimizing the cost squared distance matrix given by $d_{(x,y)} = \|x^{\text{embedding}} - y^{\text{grid}}\|$ in order to preserve as much as possible the broadly learned correlation relationships of the MolDs and FFs (Figure 3.3).

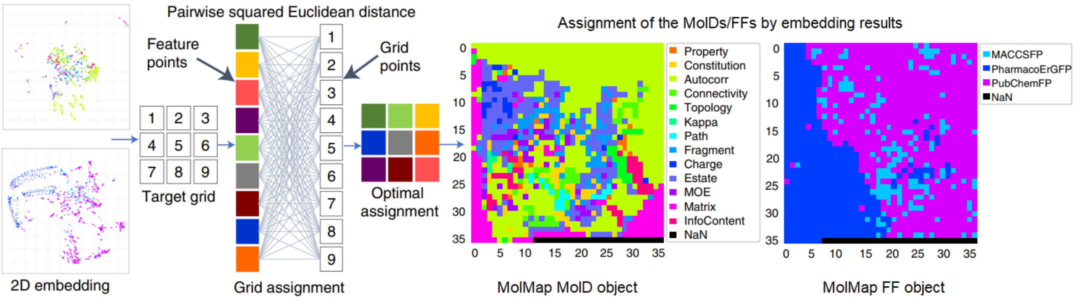


FIGURE 3.3: Starting from the UMAP projection, the Jonker–Volgenant algorithm maps each chemical descriptor until covering the full grid. Source: adapted from Shen et al., 2021.

3.3.4 Feature maps (Fmaps)

Now, given the SMILES representation of a molecule, we can obtain particular **feature maps (Fmaps)** of both MolDs and FFs, which preserve fairly well the relationships between the chemical properties. See, for example, the resulting Fmaps of Aspirin (acetylsalicylic acid) in Figure 3.4. The figure displays each descriptor class or fingerprint set in a distinctive color (channel) and the corresponding feature values indicated by the color intensities[2] (higher values are with higher intensities, 0 value is in white).

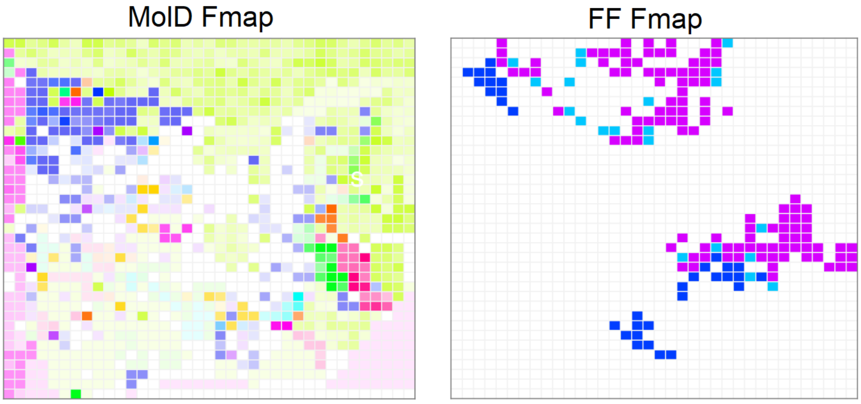


FIGURE 3.4: Generation of the MolMap Fmaps of 13 classes of MolDs and 3 sets of FFs for aspirin, upon inputting its SMILES string. Source: adapted from Shen et al., 2021.

At this point we have converted one-dimensional unordered vectors (the computed MolDs and FFs) to two-dimensional clustered Fmaps. These 2D Fmaps would be the input for training a CNN structure described in the next section.

3.3.5 MolMapNet

The **MolMapNet** model is a CNN deep learning architecture able to capture important MolRs for enhanced prediction of pharmaceutically relevant properties.

This network was constructed for simultaneous learning from both MolDs and FFs (**dual-path model**) but, depending on the user's purpose, it can also be used with only one type of chemical features (**single-path model**).

[2] Note that the color intensities are only perceived in MolD Fmap as they represent continuous properties. Otherwise, colors look completely opaque in FF Fmap since they are binary features.

The single path with descriptors approach tends to perform better for continuous predictors such as regression tasks, while the single-path with fingerprints is better for categorical predictors such as classification tasks. Overall, the dual-path approach takes advantage of both input types (MolD and FF Fmaps), thereby becoming highly competitive in both regression and classification tasks. For classification tasks, the weighted CE loss was used, while for the regression tasks, the loss function was set to MSE. As well, the Re-Lu activation function was used for both tasks.

For the dual-path approach, it was proven that MolMap-based deep learning of pharmaceutically relevant properties could also be improved by learning each of the 13 MolD classes or the 3 FF set in separate channels of multichannel networks (Figure 3.5). This means that each distinguished data class was learned through a separate channel of a multichannel CNN architecture.

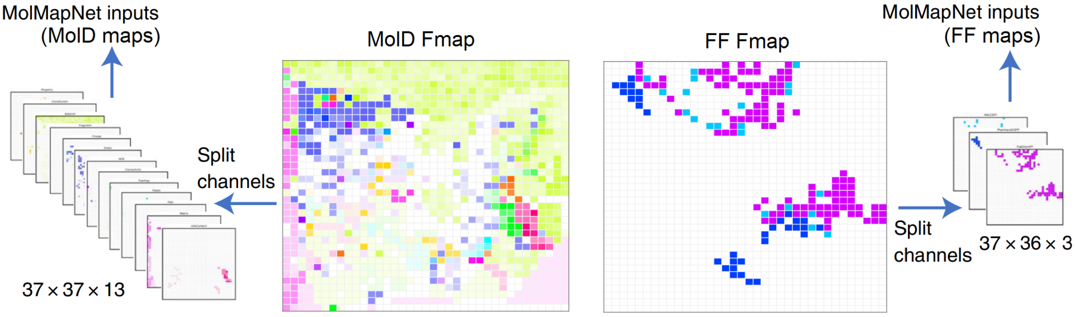


FIGURE 3.5: MolMapNet multichannel networks from Fmaps of MolDs and FFs. Source: adapted from Shen et al., 2021.

The architecture of MolMapNet consists of three components: the multichannel input Fmaps, the dual-path CNN feature learning and the nonlinear task learning [3]. The first convolution layer contains 48 kernels with size 13×13/1, followed by the max-pooling layer (3×3/2) with stride 2 that ensures lower computational cost. To achieve optimal out-of-the-box performances, MolMapNet adopts the naive inception layer which has three parallel small kernels of sizes 1×1, 3×3 and 5×5 for enhanced local perception. Subsequently, a global max-pooling layer is used for reduced parameters and, finally, two or three dense layers for improved nonlinear transformation capability. The maximum number of parameters is 830.000.

Regarding the hyperparameter tuning, a learning rate of 0.0001 and a batch size of 128 were set since they were proved to be optimal across the comprehensive benchmarks. Other regularization options such as dropout and weight decay were not used because the models were easily trained to convergence. Finally, the early-stopping strategy was used for model training, which has been extensively used in the GCNs and other deep learning models for reduced over-fitting and computing cost (Shen et al., 2021).

3.4 Other applications of MolMap

In this section we will explore other situations, besides making predictions by learning pharmaceutical properties, where MolMap components can be useful.

[3] For enhanced interpretation of the following network description, it is advisable to refer to Figure B.1 when needed.

Regions of activity in the chemical space

First, let's imagine a situation where we have computed the chemical descriptors for 1.000 inactive molecules and 1.000 active ones. By keeping the label of each molecule, we can use UMAP to see how they are placed on the plane according to the learned correlation relationships of the features (Figure 3.6). This projection can be useful for detecting regions of active and inactive molecules and later identifying which properties the grouped molecules have in common. In fact, if we perceive very different clusters between the two classes, this will also be an indication for any model to make a good classification.

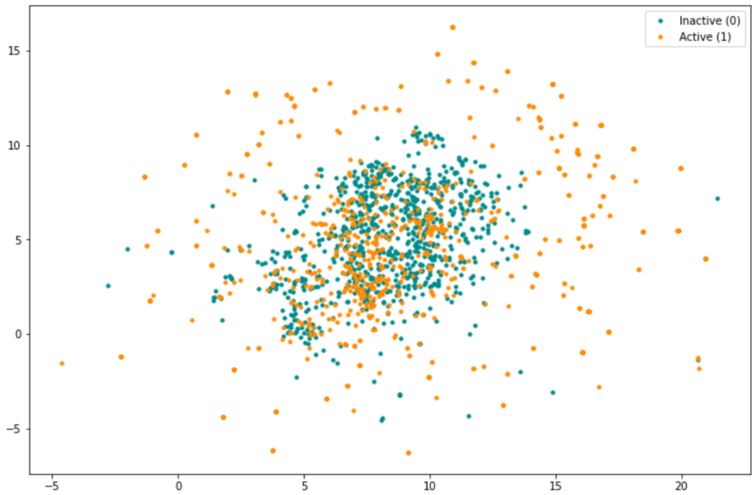


FIGURE 3.6: Labeled UMAP.

Splitting the test and train sets

Before applying MolMap to the MMV dataset, we wanted to forecast how difficult it would be to make a good prediction of our classes.

First, by using the k Nearest Neighbors (k-NN) algorithm, we selected the 10 inactive molecules closest to each of the active ones in the UMAP representation. Thus, by keeping the more similar molecules between the two classes, we ensured to make the prediction harder. Then, we made a stratified splitting into training and test sets and we trained a Logistic Regression with FFs as input. In this case, the result was surprisingly good (see first row of Table 3.1).

TABLE 3.1: Results of two splitting strategies with k-NN.

Model	Accuracy	AUC
Logistic Regression without supervised splitting	0.944	0.955
Logistic Regression with supervised splitting	0.756	0.734

To test a more challenging scenario, we did a supervised split such that the test set was composed of the molecules of one of the four quadrants of the UMAP plane (Figure 3.7). In this way we guarantee that, apart from having a high correlation between active and inactive molecules, the training and test sets were as disparate as possible.

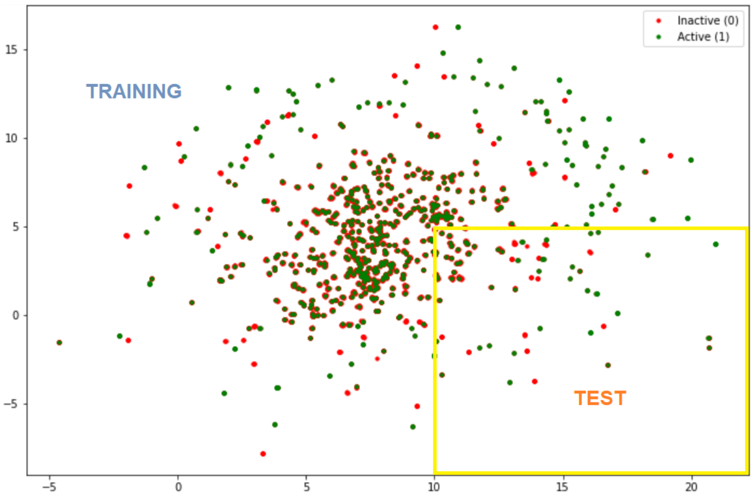


FIGURE 3.7: k-NN UMAP approach with $k = 10$ for active molecules and supervised splitting.

As expected, the results for this strategy were worse (check second row of Table 3.1). In any case, that was useful for getting an idea of extreme cases of prediction.

Molecule comparison

The last application is related to the Fmaps generated by MolMap and will be also presented with an example. It is known that the molecules Aspirin and its analogue N-acetylanthranilic acid are highly similar in structure. However, their MolD and FF Fmaps contain regions of substantially different patterns (Shen et al., 2021). These patterns (for example, the purple and light-blue dashed boxes of Figure 3.8), besides being captured by a typical CNN filter, can be also significant for drug development.

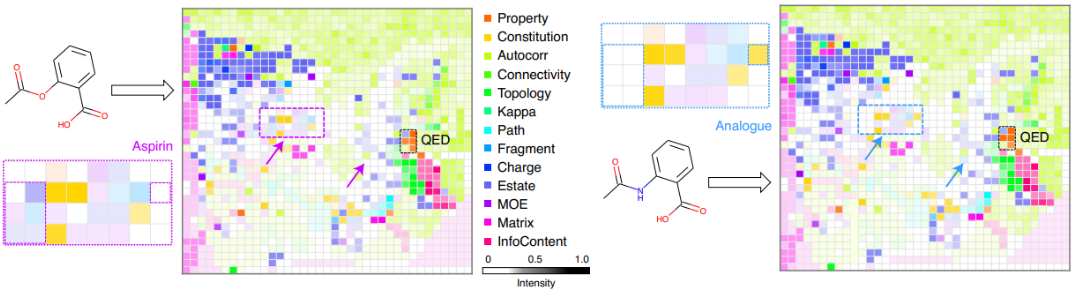


FIGURE 3.8: MolDs Fmaps for aspirin (left) and its analogue N-acetylanthranilic acid (right). Source: extracted from Shen et al., 2021.

Moreover, if we picture a situation where we are comparing one active molecule with an inactive one, by spotting the regions in which they differ, we can know which features are important for later synthesizing new active molecules. For the same reason, it may also be interesting to look at the invariant clusters between two molecule Fmaps. In summary, this reinforces the idea that MolMap Fmaps present distinguished representations and intrinsic correlations of molecular and structural features.

Chapter 4

Results

4.1 Discussion

Before applying the whole pipeline to the MMV dataset, we focused on guaranteeing and checking that MolMap fulfilled our requirements.

First, we had to ensure that this method was giving good performances in front of other baseline approaches. Secondly, we verified that the model parameters given by default fitted well in our problem. And finally, we analyzed how robust the MolMap image construction was. These three ideas are discussed in the next sections.

4.1.1 Baselines

Our first goal was to assess standard models as a baseline for model performance. For simplicity, we used a public dataset from ChemBench (Charleshen, 2020)[1], which contained approximately **10k SMILES** and its corresponding activity against a protein target of the cytochrome family: 0 for inactive and 1 for active. Although this dataset was considerably smaller than the provided one of MMV, was enough to do the set up.

For this approach, we train a classical **Logistic Regression** and a **Random Forest** classifier. Since we used both MolDs and FFs separately in both classifiers, that turned out in a total of four different baselines. Besides, the python library FLAML was used in all of them for model hyperparameter tuning. Basically, this method can handle the searching of the best hyperparameters among different models in a given period of time.

Once the models were trained, we inspected the **calibration**, which consists of readjusting, if necessary, the predicted probabilities so that they correspond to the proportion of actual cases observed. This is very useful for the evaluation of a classification model. It can help us understand how sure a model is while predicting a class label and may help us interpret how decisive a classification model is. Generally, classifiers that have a linear probability of predicting each class's labels are called calibrated. Thus, the curve of the ideal calibrated model is a linear straight line from (0, 0) moving linearly (Bora, 2021). See, for example, the calibration curve of the Logistic Regression approach with MolDs in Figure 4.1 (although the other three baselines also turned out to be well calibrated).

In fact, the performance was quite similar for the four different implementations. Table 4.1 shows the results obtained for all the baselines with accuracy and AUC as evaluation metrics.

[1] CYP PubChem BioAssay CYP 1A2, 2C9, 2C19, 2D6, 3A4 inhibition. This dataset has been downloaded from the ChemBench's GitHub Repository.

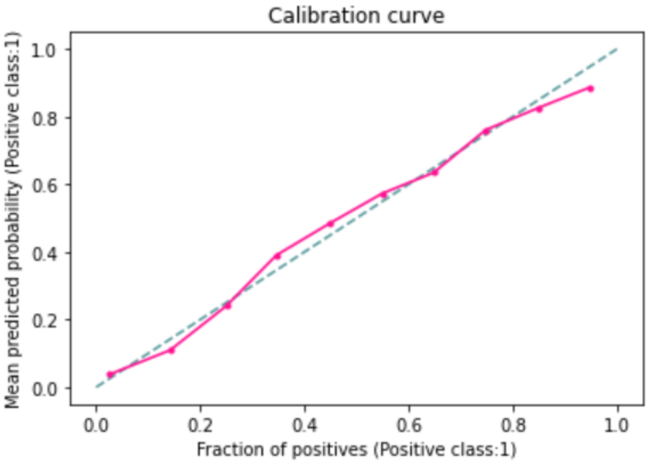


FIGURE 4.1: Calibration curve of the Logistic Regression baseline trained with MolDs from ChemBench benchmark.

TABLE 4.1: Baselines and results.

Model	**Accuracy**	**AUC**
Logistic Regression with MolDs	0.857	0.905
Logistic Regression with FFs	0.856	0.906
Random Forest with MolDs	0.852	0.911
Random Forest with FFs	0.849	0.912

Given these results, the next step was seeing how this same dataset behaved in front of MolMap in order to compare performances.

4.1.2 Parameters adjustment

In this case, the dataset from ChemBench was applied to MolMap taking benefit from the dual-path approach (that is, using both MolDs and FFs as input). Although we expected a good performance, we still wanted to see how critical were the default model parameters given by MolMap (Table 4.2).

TABLE 4.2: Default parameters of MolMapNet.

Parameter	**Value**
Learning rate	10^{-4}
Batch size	128
Dense layers	128

To do so, we applied the full MolMap pipeline several times by changing some parameters at each time, with the goal to observe if we could improve the performance with respect to the predetermined parameters. We tried giving different values to the learning rate ($10^{-2}, 10^{-5}$), the batch size (32, 64, 256) and even changing the number of the dense layers of the MolMapNet (64, 256). In addition, we try to include different subsets of FFs while keeping all the MolDs of the provided feature list.

MolMap performance did not get much worse in any of the hyperparameter combinations tested, but we could not improve it either. For this reason, we decided to maintain the model hyperparameters and the sets of MolDs and FFs given by default. With this configuration, we were already getting a similar AUC compared to the baselines (Table 4.3).

TABLE 4.3: Results of MolMap applied to ChemBench benchmark.

Model	AUC
MolMap with MolDs and FFs	0.900

Finally, note that the implementation of MolMap in the MMV dataset was also set with this same configuration.

4.1.3 Robustness of image construction

In this section the goal is to see up to what point MolMap downgrades or reconstructs the generated Fmaps. That means, if we apply MolMap several times to the same data, are we getting the same images each time? And also, can we understand why MolMap displays the chemical descriptors in this way? Does it make sense what it is doing?

This problem was addressed with a very common strategy. As it is known, the MNIST dataset is the most frequently used baseline for image classification algorithms. In our case, one of the main reasons to use it was because, unlike Fmaps, the images from this dataset can be easily understanded and it's also easy to evaluate how they have changed throughout the transformation process.

To do so, we developed a new python library called **Griddify** (see Appendix A). In essence, Griddify reproduces the different steps of the MolMap process: from building the distance matrix of the input features until generating the stacked Fmap of a given sample. Since MolMap was specifically created for inputting molecule SMILES, it was not straightforward to change the original code for learning pixels instead of chemical descriptors. Thus, that was the main reason to create this library from scratch.

The **MNIST dataset** consists of a total of 60.000 images for the training set and 10.000 images for the test set, each of them composed of 28 x 28 pixels. Since each pixel has a value between 0 and 255, we could treat them as if they were, for example, MolDs (since they take continuous values).

The first step, then, was to flatten the training images to 1-dimensional matrices of shape 784 x 1. After all images were flattened, we obtained a correlation matrix of shape 784 x 784, which captured the relations between each of the 784 pixels of each image. The metric that Griddify uses by default to compute this matrix is cosine distance. Then, by using the **UMAP** tool, we reproduced the configuration step of MolMap in order to embed all 784 pixels in a 2D map. Figure 4.2 displays the curious shape we obtained by applying UMAP, where each dot represents a pixel.

Then, for the grid assignment (Figure 4.3) we used the LAPJV library for basically mapping each pixel of the above cloud plot to a unique coordinate of the [0,1] x [0,1] grid.

Before applying Griddify, we hypothesized that MolMap would be able to correctly group the white pixels (located at the background of the image) and the black ones (defining the number at the center of the image) independently. The reason

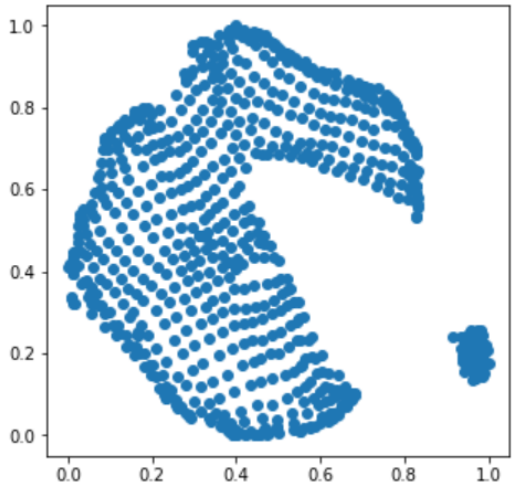


FIGURE 4.2: UMAP for MNIST.

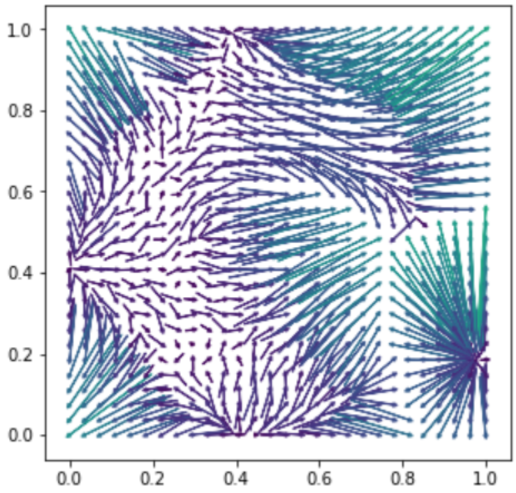


FIGURE 4.3: Grid assignment for MNIST.

behind this is that black and white pixels have constant values of 0 and 255, respectively. The real unknown was if it would be able to correctly allocate the rest of the pixels: those that are on the border of the number and have values between 1 and 254. In fact, we expected the gray pixels to be correlated in complex ways. Surprisingly, the generated Fmaps turned out to give quite outstanding and interesting results (Figure 4.4).

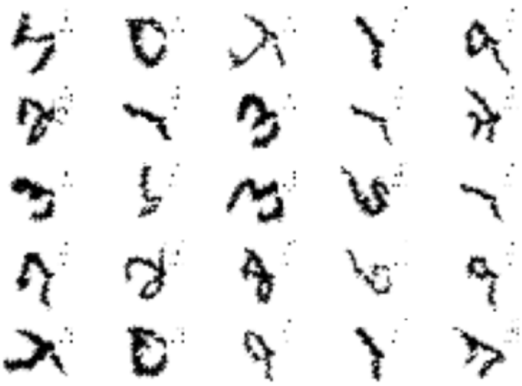


FIGURE 4.4: Fmaps for MNIST.

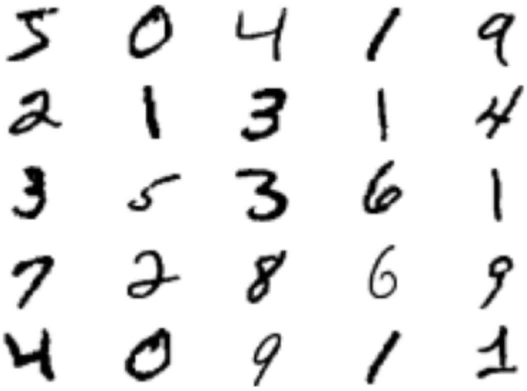


FIGURE 4.5: Original MNIST images.

Ignoring the slight rotation suffered by the transformation, the numbers that they reproduce turn out to be totally recognizable and pretty similar to the original ones (Figure 4.5).

As we did not get very fuzzy images, at that point we could expect the CNN to make good predictions. To this end, we trained a simple CNN with the processed training images. At prediction time, we applied the full Griddify pipeline to the MNIST test set and the results obtained were extraordinarily good (see Table 4.4). Therefore, we can conclude that the MolMap method is coherent and robust to image construction.

TABLE 4.4: Results of the two MNIST approaches on the same CNN.

Model	AUC
CNN on the original MNIST dataset	0.99979
CNN on the griddified MNIST dataset	0.99956

4.2 MolMap on MMV dataset

Now, we are going to focus on the relevant points of the implementation of MolMap library to the MMV dataset, previously introduced in Chapter 3.1.

Firstly, we took advantage of the completeness of the **dual-path approach** by computing chemical features of both MolD and FFs. In particular, as we already indicate in Chapter 4.1.2, we reused the MolMap computation for 13 different MolD classes and 3 different sets of FFs as well as the model parameters given by default.

The MolMap library provides a function that directly computes the molecule Fmaps from the chemical features. That means that the steps of computing the distance matrix, the feature 2D embedding (UMAP) and the grid assignment are made internally. This function returned two large matrices (one for MolDs and another for FFs) of shapes (226.921, 37, 37, 13) and (226.921, 72, 72, 3), respectively. Note that for both matrices the first dimension corresponds to the total number of molecules, followed by the image's height and the width, and finally the number of channels.

The next step was to split the data into train, validation and test sets by keeping the size proportions of 0.8, 0.1 and 0.1, respectively.

Before starting training the MolMapNet with the Fmaps as input, two points had to be taken into account.

The first one is that we were operating with a huge **imbalanced dataset** with a ratio of approximately 1 over 200.000 of the total number of active molecules (class 1) over the inactive ones (class 0). The Figure 4.6 captures this imbalance and shows a distribution based on the correlation relationships of the selected MolD and FF sets.

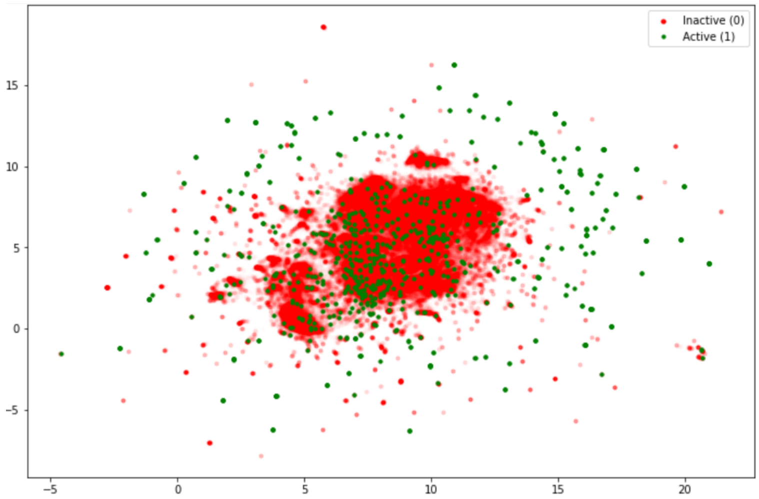


FIGURE 4.6: UMAP displaying the active/inactive imbalance of the MMV dataset.

To handle this problem, we set the parameter `sample_weight` of the fit function, which consists in giving the same 'importance' to the samples of the minority class as if they were from the majority class. For example, in our case, if we weight the inactive molecules as 1, then the active ones have to be weighted as 200.

The second problem we had to face was dealing with a **large amount of data**. Indeed, the memory of our computers couldn't bear with an input matrix X of shape ((226.921, 37, 37, 13), (226.921, 72, 72, 3)) for training the MolMapNet.

Against this setback, we first went for the most feasible solution: creating a generator. At first, this seemed a wise approach, as it is one of the most common solutions when it comes to large data training, but it was still not sufficient. At that point, we realized that the MolMap code would have required a very large refactoring to solve the 'memory leak' problem and this was beyond the scope of this project. Therefore, our final solution was doing an ensemble along with two appropriate baselines that would justify the performance of the MolMap model when applied to the complete dataset. Detailed explanations for both implementations are given below.

Generator

Initially, we created a generator for obtaining batches of our training, validation and test sets. A **generator** can be successfully used for fitting and predicting *keras* models the same way as if they were ordinary matrices. The procedure is simple: the generator iterates over the whole object by returning at each time a portion of a fixed batch size such that the memory can handle. Thereby, going over the entire object without needing to load it all at once is an achievable task. Moreover, for a proper learning of our data, the model must read the training and validation data in a random order at each epoch. This can be also attained by shuffling the order of the read indices at each epoch. At the end, the generator class can be customized according to the user needs.

In addition, we also had to slightly adapt the MolMapNet model class to deal with generators instead of standard matrices. Broadly, we took advantage of the existing model class to modify some parameters that were passed to the *Keras* fit function, as well as setting the callbacks and evaluation functions.

Concurrently, in order to handle reading and writing large files in `H5` format, we developed a tool called **Ondisk-xy** (see Appendix A). This library includes useful functions such as reading and writing an object in slices, or accessing and managing big objects from given indices. Ondisk-xy definitely played an important role for handling and solving the big-size problem.

Although the generator class worked well for a subsample of data (20.000 molecules), it did not run when applied to the whole dataset. As we were still getting memory hindrances and an unsustainable long time on execution, we decided to take an alternative route as a solution.

Baselines and Ensemble

The alternative approach consisted in performing three different models (with lower memory requirements) on the MMV activity data.

Firstly, we took advantage of the `FLAML` method to choose between a **Random Forest** and a **Logistic Regression** and automatically search for the hyperparameters that best fit each model. As we were training with all the data and the search was going to be exhaustive, we set the scanning time parameter of `FLAML` to 30min. We did two trials, one with MolDs and other with FFs, and we kept the one achieving the best performance. In all cases, the parameter `sample_weight` was passed to the fit function so that the model would not be influenced by the class imbalance.

Secondly, in order to use a baseline more similar to the MolMapNet of MolMap, we trained a **standard CNN**. Again, the feature maps for both MolDs and FFs were trained separately in the same network and we kept the model giving the most successful results.

Finally, we perform an **ensemble of 20 different MolMapNet models** such that each one was trained with a different subset of inactive molecules (class 0) and the whole set of active ones (class 1). In this way, we make sure to test the model with the entire space of inactive molecules. Indeed, this implementation was manageable as we were passing to the network a smaller set each time instead of the whole data. Once the 20 MolMapNet models were trained, we did an averaging ensemble of the performances obtained. The variance of the 20 AUC values was less than 0.001, so we can assert that the averaged result given is representative of the ensemble.

The final results for the three implementations are summarized in the following table.

TABLE 4.5: Results of the three approaches applied to MMV dataset.

Model	**AUC**
Random Forest with FFs	0.836
CNN with FFs	0.891
MolMap ensemble with MolDs and FFs	0.905

These results suggest that MolMap is a good DL library to learn from both MolDs and FFs computed for the molecules of the MMV dataset. In addition, we obtain a similar (and not worse) accuracy that the one obtained for the cytochrome dataset (Table 4.3), which points out to the possibility of obtaining even better results if more elaborated architectures are implemented.

Chapter 5

Conclusions and Future work

5.1 Conclusions

The first steps of this project were focused on understanding the formulation of compound data in cheminformatics and the role of molecular properties that can be extracted from them. The chemical descriptors obtained from molecule SMILES are used as features for training models capable of predicting pharmaceutical properties, such as the activity against a particular organism.

The MolMap model has been specifically designed to transform these features into images preserving their correlation relationships. The MolMapNet architecture learns from these intrinsic correlations and has proven to be proficient when applied to the MMV activity data.

Another quality that characterizes MolMap is that it belongs to the group of knowledge-based molecular representation models, since the interpretation and analysis of the generated feature maps for drug development requires scientific knowledge. This is an advantage over simpler models, as well as its versatility by allowing both regression and classification tasks, the proven robustness of the feature maps construction and the bimodal architecture of the MolMapNet.

Finally, dealing with the magnitude of the MMV dataset was not an easy task. Unless having sufficient computational support and the necessary infrastructure to train such amount of data, MolMap is not specifically prepared to be used directly in this case. In fact, even more alternatives could have been explored to achieve training with large data. To conclude, we will propose one of them as well as other lines of work with which the project could be extended.

5.2 Future work

At this point, there is still a wide range of further work and research that can be done to complement this project.

The first improvement relies on the personalization of the MolMap framework in some aspects. For example, instead of using the default computation of MolDs and FFs in our experiments, other descriptors could have been added so that only those desired for our purpose would have been calculated. Moreover, another visualization tool that could have been interesting to explore in place of UMAP is TMAP, having the particularity that can display very large data sets as trees.

About the experimentation we did to prove the robustness of the image reconstruction with the MNIST dataset, we could have still done some significant changes that would shed light on the question "what would have happened if...?". For example, which would be the result if tested with a coloured data set? And, what if we had mixed the pixels once they were flattened and before the distance matrix was computed?

With respect to the large data approach, still more efforts could have been dedicated to understanding why it hasn't been possible to train the full dataset even using the generator. In fact, as stated before, the problem could have been addressed in many other possible ways. For example, just by doing a PCA on the channels dimension of the feature maps, we would have already managed to considerably reduce the size of the images. Therefore, we could try to train the MolMap model with the generator on these new images and, hopefully, overcome the computational and memory issues.

Related to model predictions, we could also put the focus on the uncertainty. In the end, in order to select the most promising candidates forward the drug discovery pipeline, those molecules that have been classified the most times as active among all the trained models are the ones that should be prioritized.

Finally, in the line of Ersilia principles and mission, it could be appropriate attempt to lighten the developed assets and deploy them in the cloud, so that they can be run in resource-limited environments and reached by scientists based in Low and Middle Income Countries.

Appendix A

Source code

A.1 GitHub Repositories

This is the main repository of the project. Here can be found the Python libraries generated, scripts and Jupyter Notebooks for code replication. The dataset of MMV has not been included in this repository for privacy reasons. Otherwise, the example notebooks have been tested for simplicity with a smaller and public dataset[1].

Besides, we shall mention two other libraries that have been used along the project. Both of them have been mainly developed by Ersilia members, although I also collaborated by making some contributions.

The first one is Griddify, which has been useful to replicate and visualize the MolMap procedure step by step. And the second one is Ondisk-xy, essential for handling reading and writing large `H5` files.

[1] CYP PubChem BioAssay CYP 1A2, 2C9, 2C19, 2D6, 3A4 inhibition, from ChemBench.

Appendix B

Supplementary materials

B.1 MolMapNet architecture

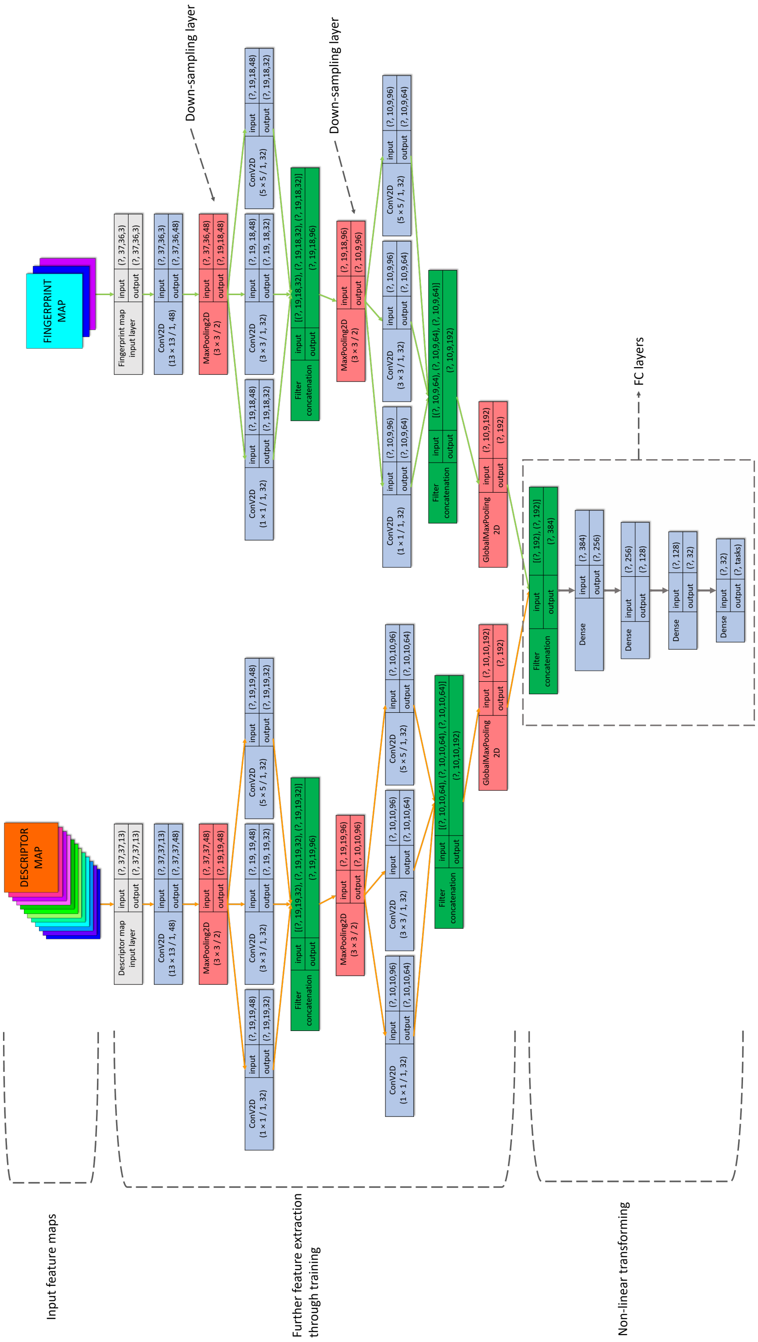

FINGERPRINT MAP
DESCRIPTOR MAP
Fingerprint map input layer
input (?, 37,36,3) output (?, 37,36,3)
Descriptor map input layer
input (?, 37,37,13) output (?, 37,37,13)
ConV2D (13×13 / 1, 48)
MaxPooling2D (3×3 / 2)
Down-sampling layer
ConV2D (1×1 / 1, 32)
ConV2D (3×3 / 1, 32)
ConV2D (5×5 / 1, 32)
Filter concatenation
input
output
GlobalMaxPooling 2D
Dense
input [(?, 192), (?, 192)] output (?, 384)
input (?, 384) output (?, 256)
input (?, 256) output (?, 128)
input (?, 128) output (?, 32)
input (?, 32) output (?, tasks)
FC layers
Input feature maps
Further feature extraction through training
Non-linear transforming

FIGURE B.1: MolMapNet architecture. Source: Shen et al., 2021.

Bibliography

Bora, Alakesh (2021). *Calibration Curves*. URL: https://www.geeksforgeeks.org/calibration-curves/ (visited on 06/06/2022).

Charleshen (Sept. 2020). *ChemBench: The molecule benchmarks and MolMapNet datasets*. Version v0. DOI: 10.5281/zenodo.4054866. URL: https://doi.org/10.5281/zenodo.4054866.

Clemons, Paul A. et al. (2011). "Quantifying structure and performance diversity for sets of small molecules comprising small-molecule screening collections". In: *Proceedings of the National Academy of Sciences of the United States of America* 108.17, pp. 6817–6822. ISSN: 00278424. URL: http://www.jstor.org/stable/41242085 (visited on 06/29/2022).

Dobilas, Saul (2021). *UMAP Dimensionality Reduction — An Incredibly Robust Machine Learning Algorithm*. URL: https://towardsdatascience.com/umap-dimensionality-reduction-an-incredibly-robust-machine-learning-algorithm-b5acb01de568 (visited on 06/20/2022).

Dong, Wei, Moses Charikar, and Kai Li (Jan. 2011). "Efficient K-nearest neighbor graph construction for generic similarity measures". In: *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, pp. 577–586. DOI: 10.1145/1963405.1963487.

Ferdinand, Nushaine (2020). *A Simple Guide to Convolutional Neural Networks*. URL: https://towardsdatascience.com/a-simple-guide-to-convolutional-neural-networks-751789e7bd88 (visited on 06/01/2022).

Goh, Garrett B. et al. (2017). *SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties*. DOI: 10.48550/ARXIV.1712.02034. URL: https://arxiv.org/abs/1712.02034.

Jonker, R. and A. Volgenant (Dec. 1987). "A shortest augmenting path algorithm for dense and sparse linear assignment problems". In: *Computing* 38.4, pp. 325–340. DOI: 10.1007/bf02278710. URL: https://doi.org/10.1007/bf02278710.

Landrum, Greg et al. (2022). *rdkit/rdkit:* $2022_03_3(Q12022)Release$. DOI: 10.5281/ZENODO.591637. URL: https://zenodo.org/record/591637.

McInnes, Leland et al. (2018). "UMAP: Uniform Manifold Approximation and Projection". In: *Journal of Open Source Software* 3.29, p. 861. DOI: 10.21105/joss.00861. URL: https://doi.org/10.21105/joss.00861.

McNaughton, Andrew D. et al. (2022). *De novo design of protein target specific scaffold-based Inhibitors via Reinforcement Learning*. DOI: 10.48550/ARXIV.2205.10473. URL: https://arxiv.org/abs/2205.10473.

Oskolkov, Nikolay (2019). *How Exactly UMAP Works*. URL: https://towardsdatascience.com/umap-dimensionality-reduction-an-incredibly-robust-machine-learning-algorithm-b5acb01de568 (visited on 06/20/2022).

Pere, Christophe (2020). *What are Loss Functions?* URL: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904 (visited on 06/01/2022).

Saha, Sumit (2018). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 06/01/2022).

Shahid, Md (2019). *Convolutional Neural Network*. URL: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529 (visited on 06/01/2022).

Shen, Wan et al. (Apr. 2021). "Out-of-the-box deep learning prediction of pharmaceutical properties by broadly learned knowledge-based molecular representations". In: *Nature Machine Intelligence* 3, pp. 1–10. DOI: 10.1038/s42256-021-00301-6.

Vamathevan, Jessica et al. (2019). "Applications of machine learning in drug discovery and development." In: *Nature reviews. Drug discovery* 18, pp. 463–477. ISSN: 1474-1784. DOI: 10.1038/s41573-019-0024-5.

Wainberg, Michael et al. (Sept. 2018). "Deep learning in biomedicine". In: *Nature Biotechnology* 36, pp. 829–838. DOI: 10.1038/nbt.4233.

Weininger, David (1988). "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *J. Chem. Inf. Comput. Sci.* 28, pp. 31–36.

World malaria report 2021 (2021). Licence: CC BY-NC-SA 3.0 IGO. URL: https://www.who.int/publications/i/item/9789240040496.

Wu, Zhenqin et al. (Mar. 2017). "MoleculeNet: A Benchmark for Molecular Machine Learning". In: *Chemical Science* 9. DOI: 10.1039/C7SC02664A.