

Universidade do Minho
Escola de Engenharia

João Filipe Silva Correia

DeepRetro: A computational framework for retrosynthesis and pathway design towards optimizing compound bioproduction

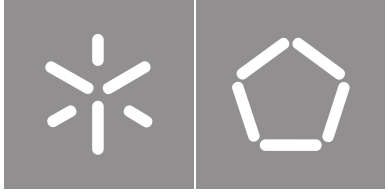
July, 2024

DeepRetro: A computational framework for retrosynthesis and pathway design towards optimizing compound bioproduction

João Correia

UMinho | 2024

Fundação para a Ciência e a Tecnologia
REPÚBLICA PORTUGUESA
CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
NORTE2020
PROGRAMA OPERACIONAL REGIONAL DO NORTE
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu



Universidade do Minho
Escola de Engenharia

João Filipe Silva Correia

DeepRetro: A computational framework for retrosynthesis and pathway design towards optimizing compound bioproduction

Doctorate Thesis
Doctorate in Informatics

Work developed under the supervision of:
Miguel Francisco de Almeida Pereira da Rocha
Vítor Manuel Sá Pereira
Rafael de Castro Carreira

July, 2024

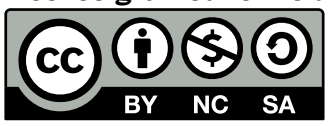
COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositoriUM of Universidade do Minho.

License granted to the users of this work



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
CC BY-NC-SA 4.0

`https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en`

Acknowledgements

Ao longo dos últimos cinco anos, a minha vida passou por grandes mudanças tanto profissionais como pessoais. O que começou como uma caminhada incerta evoluiu para um percurso de crescimento profissional e realização pessoal. No entanto, uma coisa permaneceu inalterada. O amor e apoio daqueles que sempre foram os meus pilares.

Em primeiro lugar, gostaria de expressar os meus agradecimentos à Fundação para a Ciência e Tecnologia pelo financiamento da bolsa de doutoramento (SFRH/BD/144314/2019), sem a qual este trabalho não teria sido possível. Agradeço também à instituição de acolhimento, Centro de Engenharia Biológica da Universidade do Minho.

Agradeço, também, aos meus orientadores, Professor Miguel Rocha, Vítor Pereira e Rafael Carreira. Sou muito grato pela oportunidade que me proporcionaram e pelo apoio constante ao longo deste projeto. Os vossos conselhos e feedback foram cruciais para o seu sucesso. Agradeço a vossa orientação e empenho, que foram fundamentais para o meu crescimento tanto académico como pessoal.

A todos os colegas do grupo de investigação BiSBi, com um agradecimento especial ao João Capela pela colaboração no Capítulo 3 e pela camaradagem, e à Delora Baptista e ao Tiago Sousa pela sua colaboração em diversos "side quests".

À minha família por todo o apoio e carinho incondicional. Vocês foram e sempre serão um porto de abrigo.

À Ana, a minha companheira para toda a vida. É um prazer compartilhar contigo tudo o que a vida tem de bom. Obrigado por estares sempre do meu lado nos bons e maus momentos. Foste sem dúvida o meu maior apoio nesta jornada. Sei que enquanto te tiver ao meu lado tudo será melhor.

Ao João Pedro, a minha melhor prenda desta vida. Obrigado por me mostrares que nada é impossível (mesmo com noites mal dormidas). Espero ser uma boa referência na tua vida, tudo farei para tal.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Resumo

DeepRetro: Uma plataforma computacional de retrosíntese e desenho de vias metabólicas para a optimização da bioprodução de compostos

A química computacional tem tradicionalmente recorrido a métodos clássicos como a mecânica quântica, a mecânica molecular e a termodinâmica estatística para tarefas como a previsão de propriedades moleculares, a descoberta de fármacos e a análise de vias metabólicas. Avanços recentes no poder computacional e a proliferação de dados químicos e biológicos têm melhorado significativamente a nossa capacidade de entender e prever o comportamento de pequenas moléculas. Técnicas computacionais modernas, incluindo a aprendizagem máquina, aprendizagem profunda e algoritmos evolucionários têm permitido a manipulação de vastos conjuntos de dados e sistemas moleculares complexos com alta precisão e eficiência. Estes avanços não só melhoram a descoberta de fármacos e a análise de vias metabólicas, como também promovem uma química mais sustentável ao favorecer métodos biossintéticos em deterimento da síntese química tradicional, reduzindo assim os impactos ambientais e económicos.

Este trabalho foca-se no desenvolvimento de plataformas e algoritmos de inteligência artificial na química computacional em áreas chave: previsão de propriedades/atividades moleculares, otimização do desenho molecular, planeamento de síntese e previsão de metabolismo. Para tal desenvolvemos o DeepMol para previsão automática de propriedades/atividades moleculares usando aprendizagem máquina e profunda. Utilizámos modelos de reações bioquímicas para melhorar a eficiência no planeamento de síntese. Com o ReactEA integramos algoritmos evolucionários com templates de reações para a otimização do desenho de novas moléculas. Por fim, o BioCatalyzer prevê o metabolismo de moléculas, fornecendo informação crítica sobre o seu processo metabólico.

Em geral, a nossa investigação foca-se na melhoria da previsão de propriedades moleculares, na otimização do desenho de novas moléculas e na compreensão de vias metabólicas. Ao integrar inteligência artificial com templates de reações bioquímicas, pretendemos melhorar os processos de descoberta e desenvolvimento de fármacos.

Palavras-chave: Algoritmos Evolucionários, Aprendizagem Profunda, Desenho de Vias Metabólicas, Quimioinformática, Templates de Reações

Abstract

DeepRetro: A computational framework for retrosynthesis and pathway design towards optimizing compound bioproduction

Computational chemistry has traditionally relied on methods such as quantum mechanics, molecular mechanics, and statistical thermodynamics for tasks like molecular property prediction, drug discovery, and metabolic pathway analysis. Recent advancements in computational power and the proliferation of chemical and biological data have significantly enhanced our ability to design and predict the behavior of small molecules. Modern techniques, including machine learning, deep learning, and evolutionary algorithms, enable the handling of vast datasets and complex molecular systems with high accuracy and efficiency. These advancements not only improve drug discovery and metabolic pathway analysis but also promote green chemistry by favoring sustainable biosynthetic methods over traditional chemical synthesis, thereby reducing environmental and economic impacts.

This work explores artificial intelligence frameworks and algorithms in computational chemistry across key areas: molecular property/activity prediction, molecular design optimization, synthesis planning, and metabolism prediction. We developed DeepMol for automated molecular property/activity prediction using machine and deep learning. Biochemical reaction templates were used to enhance synthesis planning efficiency. ReactEA integrated evolutionary algorithms with reaction rules for molecular design optimization. Lastly, BioCatalyzer predicts small molecule metabolism, offering insights into the metabolic fate of chemicals.

Overall, our research focuses on improving molecular property prediction, optimizing design, and understanding metabolic pathways. By integrating artificial intelligence with biochemical reaction rules, we aim to enhance drug discovery and development processes.

Keywords: Chemoinformatics, Deep Learning, Evolutionary Algorithms, Pathway Design, Reaction Rules

Contents

List of Figures **xi**

List of Tables **xiv**

Acronyms **xvi**

1 Introduction **1**

1.1 Context and Motivation 1
1.2 Research Objectives 2
1.3 Thesis Outline 3

2 Background **5**

2.1 Metabolic Engineering 5
2.1.1 Pathway Design 5
2.1.2 Metabolism Prediction 8
2.1.3 Databases for Metabolic Engineering 11
2.2 Machine Learning 13
2.2.1 Types of Learning 13
2.2.2 Supervised Learning: Model Selection, Overfitting and Underfitting 14
2.2.3 Supervised Learning Algorithms 15
2.2.4 Deep Learning 20
2.2.5 Graph Neural Networks 25
2.2.6 Transformers 28
2.2.7 Generative Models 28
2.2.8 Automated Machine Learning 31
2.3 Evolutionary Algorithms 33
2.3.1 Genetic Algorithms 34
2.3.2 Evolutionary Multi-Objective Optimization 38
2.4 Machine and Deep Learning in Chemoinformatics 40

2.4.1 General overview . . . 40
2.4.2 Compounds . . . 41
2.4.3 Reactions . . . 43

3 DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry 48
3.1 Introduction . . . 48
3.2 Implementation . . . 49
3.2.1 Data Loading . . . 50
3.2.2 Molecular Standardization . . . 50
3.2.3 Feature extraction . . . 52
3.2.4 Data Scaling . . . 53
3.2.5 Feature Selection . . . 53
3.2.6 Machine and Deep Learning models . . . 53
3.2.7 Hyperparameter Optimization . . . 54
3.2.8 Pipelines . . . 54
3.2.9 AutoML: pipeline optimization . . . 54
3.2.10 Unsupervised Learning . . . 55
3.2.11 Data splitting . . . 56
3.2.12 Imbalanced data . . . 56
3.2.13 Feature importance and model interpretability . . . 58
3.3 Results . . . 59
3.3.1 Results on the ADMET benchmarks . . . 59
3.3.2 Practical applications of DeepMol . . . 63
3.4 Discussion: comparison with other tools . . . 64
3.5 Conclusion . . . 66

4 Predicting the number of biochemical transformations needed to synthesize a compound 67
4.1 Introduction . . . 67
4.2 Materials and Methods . . . 69
4.2.1 Data . . . 69
4.2.2 Molecular Representations . . . 71
4.2.3 Deep Learning Models . . . 71
4.2.4 Performance metrics . . . 73
4.2.5 Development Environment . . . 74
4.3 Results and Discussion . . . 74
4.4 Conclusion . . . 79

5 Combining Evolutionary Algorithms with Reaction Rules Towards Focused Molecular Design **80**
5.1 Introduction . . . 80
5.2 Materials and Methods . . . 82
5.2.1 Biochemical Reaction Rules . . . 82
5.2.2 Framework Overview . . . 83
5.2.3 Initial population . . . 85
5.2.4 Population Fitness . . . 85
5.2.5 Selection Operator . . . 85
5.2.6 Mutation Operator . . . 85
5.2.7 Replacement of solutions . . . 86
5.2.8 Termination Criterion . . . 87
5.2.9 Evolutionary Algorithms . . . 87
5.2.10 Development Environment . . . 87
5.3 Results and Discussion . . . 88
5.3.1 Optimization of Drug-Likeness, Solubility, and Synthetic Accessibility . . . 88
5.3.2 Similarity to Aspartame . . . 89
5.3.3 Docking to PARP1 . . . 91
5.4 Conclusions . . . 93

6 BioCatalyzer: a rule-based tool for compound metabolism prediction and identification **94**
6.1 Introduction . . . 94
6.2 Materials and Methods . . . 96
6.2.1 Biochemical Reaction Templates Knowledgebase . . . 96
6.2.2 Gut Microbiome Drug Metabolism Data . . . 96
6.2.3 Framework Overview . . . 97
6.2.4 Development Environment . . . 101
6.3 Results and Discussion . . . 102
6.4 Conclusion and Future Work . . . 105

7 Conclusions and Future Work **107**
7.1 Main Contributions . . . 107
7.2 Publications . . . 108
7.3 Future Work . . . 110

Bibliography **112**

Appendices

A Supplementary Data for Chapter 3 **142**
A.1 Supporting Information - DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry 142
A.1.1 Extract features 142
A.1.2 Supplementary tables 143
A.1.3 Comparison with similar tools 158

B Supplementary Data for Chapter 5 **159**
B.1 Supplementary Material - Combining Evolutionary Algorithms With Reaction Rules Towards Focused Molecular Design 159
B.1.1 Supplementary Figures: 159
B.1.2 Supplementary Tables: 162

List of Figures

1 Influence of model complexity on underfitting and overfitting. 15

2 Fully Connected Artificial Neural Network with a single hidden layer. 19

3 Activation functions. (a) Sigmoid function; (b) Tanh function; (c) ReLU function. 20

4 Structure of the Artificial Neuron. 20

5 General CNN structure and illustration of convolution and average pooling operations. . . 23

6 Schematic function of a simple RNN cell (right), LSTM unit (center) and GRU unit. 26

7 Example of an undirected graph and its adjacency list. 27

8 End-to-end prediction workflow with GNNs. 27

9 Diagram of an autoencoder architecture. 29

10 General workflow of an AutoML system. 32

11 General overview of a genetic algorithm. 35

13 Two parent individuals and respective offspring after different types of crossover. (a) one-point crossover; (b) two-point crossover; (c) uniform crossover. 37

12 Individual before and after binary mutation. 37

14 The DeepMol AutoML framework offers a comprehensive approach that includes an optimization framework that samples pipeline configurations from the configuration space. This pipeline begins with an optional **(a)** standardization step (configuration space: 3 different methods), followed by **(b)** the extraction of features (configuration space: four options for sets with 34 methods in total, all with their respective parameters). These features can then be optionally **(c)** scaled and selected (configuration space: 14 methods and respective parameters), preparing them for **(d)** the training phase of the model (configuration space: 140 models, architectures and hyperparameters). Post-training, the model's performance is evaluated based on a predetermined metric. This evaluation feedbacks to the *Optuna* optimization engine. *Optuna* uses these results to inform its selection of new parameters and methods for enhancing the pipeline's efficiency using state-of-the-art optimization algorithms. This process is repeated *n* times (chosen by the user), also called trials, and, in the end, the best pipeline is selected. Each individual box represented in the configuration space is expanded and schematized in the boxes below. Finally, the result of the AutoML engine is an optimized pipeline that can be used to transform new data and perform new predictions. 51

15 Depiction of dataset splits in the chemical space for the mutagenicity dataset from [214] provided by TDC. A similarity matrix was generated using the Tanimoto similarity metric over Morgan fingerprints. A t-SNE was applied to the matrix to reduce dimensionality and visualization. **A)** represents the similarity splitter and examples of molecules belonging to small clusters. One can regulate the homogeneity of the splits by tweaking the homogeneity parameter, which considers all the compounds with a similarity lower than the homogeneous parameter to be in the same set. The higher the parameter is, the more heterogeneous the split will be. **B)** represents the scaffold splitter and scaffolds belonging to the splits on the plot on the right. This splitter can separate the data by putting compounds with the same scaffolds in different data splits (homogeneous split) or not (heterogeneous split). 57

16 Depiction SHAPE values for a ridge classifier and the most relevant MACCS keys in the molecules. The dataset used was the bioavailability dataset from [222] provided by TDC. **A)** provides the overall feature importance of all labels for all data points and depicts the two of the most relevant MACCS keys for the prediction. The highlighted atoms and bonds are the ones that correspond to the MACCS key. **B)** provides a depiction of the most important features of the drug cinacalcet, showcasing how it is also possible to assess features' importance for individual cases. 58

17 Heatmap showcasing the presence or absence of an integrated way of performing relevant tasks for constructing QSAR/QSPR systems. Green stands for presence, red for absence and yellow for incomplete integration. 65

18 Example of a reaction rule and respective original reaction. 69

19 Test and independent set accuracy for all models. 76

20 Test and independent set accuracy allowing miss-classification by one step for all models. 77

21 Test and independent set MAE for all models. 78

22 Reaction rule encoding an Aromatic Decarboxylation transformation (A) and original reactions (B), in which a carboxylic acid group (-COOH) is removed from the compound. 83
23 General outline of the ReactEA workflow. 84
24 This is an example of how the mutation operator works: three different reaction rules are applied to create three distinct products (mutants) from a single reactant (parent). It is possible that different rules apply to the same compound, and one rule can also generate multiple different products. 86
25 The molecule with the highest binding affinity and the reaction templates that were applied to transform the starting compound (Olaparib Fragment 652934) into the final molecule are presented, along with the intermediate molecules and the corresponding EC numbers for the original reactions. 92

26 Overview of Biocatalyzer metabolism prediction and identification. 102
27 Metabolism of Itraconazole by the Human Microbiota: Identification of Five Potential Drug Metabolites. 103
28 Predicted metabolism of the drug Melphalan by two different reaction rules produces two distinct compounds with the same mass. Reaction rules SMARTS were created using SMARTSview (`https://smarts.plus/`). 104

29 Best molecule according to its similarity to aspartame generated by each EA. 159
30 Evolution of solutions across generations for the different initial populations. Top left represents the average similarity to aspartame of all solutions (average 10 runs). Top right the best solution (average 10 runs). Bottom left the mean similarity of the best run only. Bottom right the best solution of the best run. 160
31 Binding affinity of the best molecule at each generation for the different EAs. The best inhibitor in the initial population, Olaparib, is shown for reference. 161
32 Best molecule and respective binding affinity of the best generated molecules by each EA. 162

List of Tables

1 A List of Pathway Design Tools for Biochemistry. 9
2 Comparison of AutoML tools. 33
3 A selection of diverse recent studies that use ML and DL for biological activity prediction. 44
4 Performance of DeepMol in the six absorption-related properties in the TDC ADMET benchmark group. 60
5 Performance of DeepMol in the three distribution-related properties in the TDC ADMET benchmark group. 61
6 Performance of DeepMol in the six metabolism-related properties in the TDC ADMET benchmark group. 62
7 Performance of DeepMol in the three excretion-related properties in the TDC ADMET benchmark group. 62
8 Performance of DeepMol in the four toxicity-related properties in the TDC ADMET benchmark group. 63
9 New compounds generated at each step. 70
10 Parameters optimized using a 5-fold RandomizedSearch for the FCNNs. 75
11 Parameters optimized using a 3-fold RandomizedSearchCV for the 1D CNNs. 75
12 Classification report of the FCNNs with Morgan fingerprints. 77
13 Confusion matrix of the FCNN with Morgan fingerprints. 77
14 Regression metrics test set. 78
15 Examples of the different gradient-free atom, fragment, and reaction-based methods. . . 82
16 Impact of different initial populations on the performance of ReactEA. The percentage of unique and novel molecules, the similarity between the molecules, the mean and best similarity to the aspartame, and the QED and SAS are showcased. 90
17 Results of the Docking to PARP1 optimization. The binding affinities of the initial population and the performance of the different EAs are shown. 93

18 Top 5 drugs with the highest percentages of matches for all possible drug-compound pairs. 103
19 Matched mass shifts associated with known transformations. 104

20 0D, 1D and 2D descriptors integrated into only one class in Descriptors 143
21 Molecular Fingerprints provided by DeepMol. 146
22 3D descriptors provided by DeepMol. 147
23 DeepChem featurizers provided by DeepMol. 148
24 Scikit-Learn scalers available in DeepMol. 149
25 Scikit-Learn feature selection methods available in DeepMol. 150
26 Data splitters in DeepMol - all of them can perform train-test, train-validation-test and k-fold splitting. 150
27 Imbalanced learning techniques provided by DeepMol. 151
28 Scikit-learn, Deepchem and Keras models provided by DeepMol by default. 151
29 Feature explainability using Shapley values (SHAP [221]) in DeepMol. 155
30 Ensembles and voting pipelines. 156

31 Results of the QED, pLogP, SAS and CLScore optimization. 162

Acronyms

AAE	adversial autoencoders
ADMET	adsorption, distribution, metabolism, elimination, and toxicity
AI	Artificial Intelligence
ANN	artificial neural network
AUPRC	area under the precision-recall curve
AUROC	area under the receiver operating characteristic
AutoML	automated machine learning
BRENDA	BRaunschweig ENzyme DAtabase
BRICS	Breaking of Retrosynthetically Interesting Chemical Substructures
CADD	computer-aided drug design
CLScore	ChEMBL-Likeness Score
CNN	convolutional neural network
DL	Deep Learning
DNN	Deep Neural Network
DT	decision tree
EA	evolutionary algorithm
EC number	Enzyme Commission number
ECFP	Extended-Connectivity Fingerprint
ES	evolution strategy
ETKDG	Experimental-Torsion basic Knowledge Distance Geometry
FCFP	Functional-Class Finger-print
FCNN	fully connected deep neural network

FN flase negative
FNN feedforward neural network
FP false positive

GA genetic algorithm
GAN generative adversial network
GAT graph attention neural network
GCNN graph convolutional neural network
GNN graph neural network
GRU gated recurrent unit

HPO hyperparameter optimization

IBEA Indicator-based Evolutionary Algorithm
InChI IUPAC Chemical Identifier

KEGG Kyoto Encyclopedia of Genes and Genomes
KNN k-nearest neighbors

LBDD ligand-based drug design
LC-MS liquid chromatography-mass spectrometry
LgR logistic regression
logP octanol–water partition coefficient
LR linear regression
LS local search
LSTM long short term memory

MACCS Molecular Access System
MAE mean absolute error
ME metabolic engineering
MINE Metabolic In silico Network Expansions
ML Machine Learning
MLP multi layer perceptron
MO multi-objective
MOEA multi-objective evolutionary algorithm
MS mass spectrometry
MSE mean squared error
MTE molecular transformer embeddings

NAS neural architecture search
NB naive bayes
NLP natural language processing
NMR nuclear magnetic resonance
NN neural network
NP natural product
NSGAII Non-dominated Sorting Genetic Algorithm II
NSGAIII Non-dominated Sorting Genetic Algorithm III

PARP Poly (ADP-ribose) polymerase
PARP1 Poly [ADP-ribose] polymerase 1
PCA principal component analysis
pLogP penalized octanol–water partition coefficient

QED quantitative estimate of drug-likeness
QSAR/QSPR quantitative structure-activity/property relationships

RF random forest
RL reinforcement learning
RNN recurrent neural network

SA simulated annealing
SAS synthetic accessibility score
SBDD structure-based drug design
SC Spearman's correlation
SHAPE SHapley Additive exPlanations
SLP single layer perceptron
SMARTS SMiles ARbitrary Target Specification
SMILES Simplified Molecular-Input Line-Entry System
SMOTE Synthetic Minority Over-sampling Technique
SO single-objective
SPEA2 Strength Pareto Evolutionary Algorithm 2
SVM support vector machine

t-SNE T-distributed Stochastic Neighbor Embedding
TDC Therapeutics Data Commons
TN true negative

TP true positive

UMAP Uniform Manifold Approximation and Projection

VAE variational autoencoder

VS virtual screening

1 Introduction

1.1 Context and Motivation

The quest for sustainable and efficient bioproduction of compounds has become a focal point in modern biotechnology and pharmaceutical sciences [1]–[3]. As traditional chemical synthesis faces increasing scrutiny due to its environmental footprint and economic limitations, there is a growing shift towards biochemical processes that leverage renewable carbon sources. Microbial metabolism, in particular, shows great promise [4]. However, optimally rewiring microbial metabolism for bioproduction remains a significant challenge [5], [6]. Designing *de novo* biosynthetic pathways and predicting feasible synthesis routes are complex and labor-intensive tasks that need advanced computational tools capable of addressing these issues effectively.

Even with the most advanced tools available today, efficiently exploring the vast and complex chemical and biological space remains extremely difficult [7]. The multitude of possible metabolic pathways, enzyme activities, and potential byproducts creates a hard challenge. Successfully navigating this vast array of possibilities requires sophisticated algorithms that can manage and analyze large datasets, simulate numerous scenarios, and predict optimal pathways with high precision.

A key area of focus in optimizing microbial metabolism is the bioproduction of compounds with enhanced properties or activities. Predicting the properties and activities of compounds based on their molecular structures requires integrating diverse computational techniques such as molecular docking, quantitative structure-activity/property relationships (QSAR/QSPR) modeling, and Machine Learning (ML) algorithms [8]–[14]. These computational tools facilitate the virtual screening of extensive chemical libraries to identify promising candidates for subsequent experimental validation. This not only saves time and resources but also enhances the likelihood of discovering compounds that exhibit desired biological activities.

However, identifying compounds with the desired activities is not enough. It is equally important to identify feasible biochemical synthesis routes for efficient and sustainable production. Retrobiosynthesis plays a pivotal role here, involving reverse engineering from a target compound to identify enzymatic transformations and potential biosynthetic pathways feasible within a biological system. A notable challenge in retrobiosynthesis lies in effectively guiding the design of synthetic biology experiments [15], [16]. By

deconstructing complex molecules into simpler precursor molecules and then mapping out enzymatic reactions to convert these precursors, researchers can methodically construct metabolic pathways. This systematic approach aids in predicting suitable enzymes or pathways for engineering. However, it often generates a large number of intermediate compounds, many of which are experimentally impractical [17], [18].

Additionally, understanding how drugs are metabolized within the organism is crucial for predicting their efficacy, safety, and potential side effects [19]. Drug metabolism typically involves enzymatic processes that modify the chemical structure of the drug, often converting it into metabolites that may exhibit different pharmacological properties compared to the parent compound. Predicting drug metabolism is essential for optimizing dosing regimens, understanding potential drug-drug interactions, and predicting adverse effects. Computational methods play a pivotal role in this predictive process by simulating enzymatic reactions and predicting the metabolic fate of drugs based on their chemical structures. By identifying and characterizing metabolites, researchers can gain insights into the pharmacokinetic profiles of drugs and design drug development strategies accordingly. This knowledge is instrumental in designing drugs with improved efficacy and safety profiles while minimizing unexpected metabolic outcomes. Therefore, integrating computational tools for predicting drug metabolism is crucial in advancing pharmaceutical research and development, ensuring the development of safer and more effective therapeutic drugs.

Given this landscape, this thesis aims to integrate state-of-the-art synthetic biology techniques with novel Artificial Intelligence (AI) methodologies to enhance the efficiency and accuracy of drug discovery and pathway design. By addressing the limitations of existing tools and leveraging the strengths of emerging technologies, we aim to contribute significantly to the optimization of compound bioproduction, thereby advancing the field of synthetic biology and computational chemistry.

1.2 Research Objectives

This work aims to develop computational tools capable of generating novel metabolic pathways for the bioproduction of target compounds with improved properties and access the effects of those on biological organisms. This will encompass the following scientific and technological objectives:

- To review the state-of-the-art in chemical/ biotechnological retrosynthesis and pathway design methods and computational tools, as well as in ML and Deep Learning (DL)-based algorithms for related tasks.

- To develop ML methods that can take as input molecular structures and can predict molecular properties and activities, testing different compound representations including simple text representations (e.g., SMILES), graph convolutions, molecular descriptors, or pre-trained embeddings. These methods will be validated using portfolios of benchmark datasets from external databases;

- To develop methods capable of guiding pathway design experiments by filtering out impractical intermediate transformations/ compounds;
- To develop AI approaches capable of designing new molecules and proposing potential synthesis paths;
- To develop computational methods for the accurate, rapid, and comprehensive prediction of small-molecule metabolism;
- To apply the developed methods/tools to diverse relevant case studies;
- To make the developed tools open-source so that they can be used and extended by the scientific community;
- To write articles for peer-reviewed international journals and conferences with the main results of the work.

1.3 Thesis Outline

This thesis is organized into seven chapters. A brief description of each one of them is given as follows:

- Chapter 1: Introduction - Provides an overview of the thesis topic, its motivation, main goals, and a summary of the thesis structure;
- Chapter 2: Background - Introduces the theoretical foundations of metabolic engineering, including topics such as pathway design, metabolism prediction, and their intersection with ML, DL, and evolutionary algorithms (EAs);
- Chapter 3: DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry - Describes the development of DeepMol, a framework designed to automate the selection and optimization of preprocessing techniques, data engineering methods, ML/DL models, and their hyperparameters for molecular property/activity prediction;
- Chapter 4: Predicting the Number of Biochemical Transformations Needed to Synthesize a Compound - Reports the use of various DL architectures and molecular representations to predict the number of biochemical transformations required to synthesize a compound. The creation of a diverse dataset using reaction templates for model training and validation is also discussed;
- Chapter 5: Combining Evolutionary Algorithms with Reaction Rules Towards Focused Molecular Design - Explores the integration of reaction rules and EAs to develop a framework for designing new compounds with optimized properties and suggesting potential synthetic routes;
- Chapter 6: BioCatalyzer: A Rule-Based Tool for Compound Metabolism Prediction and Identification - Details the development of BioCatalyzer, which utilizes reaction rules to predict drug metabolism and compares predictions with experimental mass spectrometry (MS) data;

- Chapter 7: Conclusions and Future Work - Summarizes the main findings of the thesis, discusses conclusions drawn from the research, and suggests avenues for future work and improvements.

2

Background

2.1 Metabolic Engineering

The term "Metabolic Engineering" was first introduced in 1991 by Bailey to describe the application of recombinant DNA methods for restructuring metabolic networks and enhancing the production of metabolites and proteins by modifying the enzymatic, transport, and regulatory functions of cells [20]. Today, the goals of metabolic engineering (ME) have evolved from those outlined by Bailey and later by Stephanopoulos and Vallino [21]. Current studies focus on optimizing the capacity and performance of biological systems as "biological hardware" rather than their natural biological functions. Key applications of modern ME include direct modulation of metabolic pathways for metabolite overproduction [22], [23], improving cellular properties, engineering native and heterologous pathways [24], [25] to transform biological systems into efficient cell factories. By leveraging advanced genetic tools and computational models, scientists can redesign microbial metabolism to optimize the yield of biofuels, pharmaceuticals, and other industrial chemicals, offering a sustainable alternative to traditional chemical synthesis methods.

2.1.1 Pathway Design

Metabolism is the process of synthesis and degradation of molecules in living organisms. It is formed by a complex network of biochemical reactions where metabolites are interconnected by many enzymatic reactions. To better understand and model this complex network, it is often broken down into smaller and simpler segments known as metabolic pathways. A metabolic pathway can be defined as a series of enzyme-catalyzed biochemical reactions that transform a set of source compounds into a set of target compounds.

Nowadays, computational tools play a crucial role in the design and optimization of metabolic pathways. By leveraging computational biology and bioinformatics, researchers can now predict and analyze metabolic pathways with precision and efficiency that would be challenging through traditional experimental approaches alone. These tools use algorithms that simulate biochemical reactions [26], [27], predict enzyme functions [28], [29], and assess the feasibility of pathways based on multiple criteria [30], [31].

2.1.1.1 Pathway Enumeration

Pathway enumeration is a critical step in the computational design of metabolic pathways involving the systematic generation of the metabolic pathways within an organism or a metabolic network. The enumeration process typically begins with the identification of precursor molecules and enzymes that can catalyze the necessary biochemical transformations to produce target molecules. These biochemical transformations are then systematically connected to form linear or branched pathways that link starting metabolites (precursors) to end metabolites (targets). Once potential pathways are enumerated, they are evaluated based on multiple criteria such as biochemical feasibility, thermodynamic constraints, and regulatory considerations [32]. Experimental validation through metabolomics and fluxomics further refines and validates these predicted pathways.

Given the complexity of metabolic networks, computational tools are essential for effective pathway enumeration. These tools use algorithms to predict and map all possible pathways by combining known biochemical reactions. Some tools try to optimize product yield [33], [34] while others prioritize finding the shortest pathways from source to target compounds [35], [36], reducing the number of steps and potentially increasing the efficiency of the pathway. Introducing fewer non-native (heterologous) reactions into a host organism is often desirable to minimize potential disruptions and incompatibilities [37]. Additionally, some tools assess whether the predicted pathways are thermodynamically feasible [38], ensuring that the biochemical reactions can occur under physiological conditions. The presence of necessary enzymes in the host organism is also considered to ensure that the proposed pathway can be effectively implemented [39].

A typical pathway enumeration workflow involves several key steps. First, appropriate data sources, including metabolic databases and genome-scale metabolic models, which provide comprehensive sets of reactions and metabolites for the organism of interest, are chosen. Then, a detailed representation of the metabolic network is constructed, capturing all known reactions and their interconnections. Pruning methods are applied to reduce the complexity of the network by eliminating irrelevant or redundant reactions, making the enumeration process more manageable. Searching algorithms are utilized to systematically search through the metabolic network to identify all potential pathways. Common algorithms include depth-first search, breadth-first search, and more advanced heuristics. Once potential pathways are identified, they are filtered and ranked based on multiple criteria, such as avoiding toxic intermediates, maximizing yield, and ensuring overall pathway efficiency.

2.1.1.2 Pathway Prediction

Pathway prediction involves the computational inference of potential metabolic pathways that can convert specific starting compounds into desired target compounds. This process relies on the integration of genomic, biochemical, and/or structural data to identify enzymes capable of catalyzing the biochemical transformations necessary for pathway construction.

Computational tools used for pathway prediction leverage various algorithms and databases. For instance, sequence similarity searches and protein structure prediction algorithms help identify enzymes

with known or predicted activities that can catalyze specific reactions. These tools rely on vast repositories of genomic and biochemical data to propose potential enzymes and reactions that could be integrated into a metabolic pathway. Furthermore, pathway prediction often incorporates metabolic pathway databases and bioinformatics platforms that catalog known metabolic reactions and pathways across different organisms. By comparing structural and functional similarities, computational tools can suggest analogous pathways or promiscuous enzymes from related organisms that could be used for the desired metabolic outcomes.

Reaction prediction is a critical aspect of effective pathway prediction, aimed at predicting the outcomes of chemical reactions. There are two primary methodologies for reaction prediction: rule-based approaches and ML approaches. Rule-based approaches can be quick, utilizing manually curated knowledgebases of reaction templates. While these systems can yield good results within a specific chemical domain, they have several limitations. Their effectiveness is constrained by the extent of human curation, and no exhaustive set of rules covering the chemical space currently exists. Additionally, these approaches often generate an overwhelming number of potential reactions and metabolites, most of which are not biologically feasible. As new fields of chemistry develop, rule-based systems need continuous updates and do not scale well over time.

On the other hand, ML approaches are fast and scalable, making them an attractive alternative to rule-based methods. ML approaches ideally require large datasets from which to learn, presenting a significant challenge as many chemical datasets are proprietary and not readily available for academic use. These approaches utilize embedded data of enzymatic reactions, which are described using the molecular structure of substrate-product pairs, along with enzymatic data represented as amino acid sequences or Enzyme Commission numbers (EC numbers) for metabolic pathway design.

2.1.1.3 Bioretrosynthesis

Bioretrosynthesis involves working backwards from a target molecule to identify potential precursor metabolites and the corresponding enzymatic reactions that can produce them. The bioretrosynthesis process starts by defining the structure of the target molecule and then systematically breaking it down into simpler components, identifying potential biosynthetic steps at each stage. Computational tools and algorithms play a crucial role in this process, leveraging extensive databases of metabolic reactions, enzyme functions, and genomic data. These tools can suggest possible enzymatic transformations based on known metabolic pathways, enzyme specificity, and reaction mechanisms.

One of the key advantages of bioretrosynthesis is its ability to explore a wide array of biosynthetic routes, including those not naturally occurring in the host organism. By identifying and ranking alternative pathways based on criteria such as thermodynamic feasibility, enzyme availability, and metabolic efficiency, researchers can select the most promising routes for experimental validation.

Advanced bioretrosynthesis tools also incorporate ML and AI to improve prediction accuracy and pathway optimization. These technologies can analyze large datasets of metabolic information, learn patterns of enzyme-substrate interactions, and predict novel enzymatic activities. This improves the ability

to design *de novo* pathways that are highly efficient and designed to specific production goals.

Bioretrosynthesis tools can be used for both single-step and multi-step pathway prediction. In single-step pathway prediction, different DL architectures can be used to suggest candidate precursors for the target molecule and generate novel reaction templates. For multi-step pathway prediction, different search algorithms are combined with DL models to filter and evaluate the synthetic complexity and enzyme availability for each predicted reaction step. In both of these, compounds can be represented in various formats such as line notations, molecular fingerprints, molecular graphs, and learned feature vectors derived from neural networks (NNs). Enzymes are generally represented by their amino acid sequences and EC numbers.

2.1.1.4 Computational Pathway Design Tools

Computational tools for designing pathways are crucial in synthetic biology and ME, facilitating the creation and refinement of biological pathways. Currently, numerous open-source tools are accessible, each varying in approach and functionalities. These tools follow a similar pipeline organized into five main steps:

1. Defining the Metabolic Network: Tools start by defining a metabolic network using databases of known reactions or by generalizing to predict novel reactions. This step involves gathering and integrating comprehensive data on metabolic reactions and pathways;
2. Constructing the Computational Network: The metabolic network is then represented as a computational graph, consisting of nodes (compounds) and edges (reactions);
3. Searching for Pathways: Using the constructed graph, tools search for potential pathways within the network. Search algorithms and heuristics are applied to identify feasible routes from starting compounds to target products;
4. Enumerating Pathways: The identified pathways are enumerated, providing a comprehensive list of possible routes. This step involves detailed analysis and listing of all potential pathways generated by the search algorithms;
5. Ranking Pathways: Finally, the resulting pathways are ranked based on specific criteria such as efficiency, yield, cost, or feasibility. This ensures that only the most promising candidates are selected for further experimental validation.

Table 1 lists some of the main available tools for pathway design in biochemistry. Other relevant tools for general organic chemistry include [40]–[45].

2.1.2 Metabolism Prediction

Metabolism prediction is a crucial aspect of understanding how various substances are processed within an organism. It involves predicting the chemical changes that compounds will undergo, mainly driven

Table 1: A List of Pathway Design Tools for Biochemistry.

Tool	Type	Algorithm
BNICE.ch [36]	Pathway Enumeration	Reaction Templates
MetQuest [46]	Pathway Enumeration	BFS Variation
MetaPath Online [47]	Pathway Enumeration	BFS Variation
PyMiner [48]	Pathway Enumeration	Similarity Search
NICEpath [49]	Pathway Enumeration	k-Sortest Graph Search
Enzymatic Transformer [50]	Forward Single-Step	Transformer with transfer learning
PathPred [51]	Mixed Multi-Step	Similarity Search
Biocatalysis model [52]	Mixed Single-Step	Molecular transformer with transfer learning
Baranwal *et al.* [53]	Retro Single-Step	GNN/random forest (RF)
RDEnzyme [54]	Retro Single-Step	Similarity Search
novoPathFinder [55]	Retro Multi-Step	Maximum Common Substructure Search
BioNavi-NP [56]	Retro Multi-Step	Transformer/A* Search
Levin *et al.* [57]	Retro Multi-Step	DNN/MCTS
RetroPath RL [58]	Retro Multi-Step	Reinforcement Learning
RetroBioCat [59]	Retro Multi-Step	Beam Search
RetroPath 2.0 [60]	Retro Multi-Step	EFMs

BFS: Breadth First Search; GNN: Graph Neural Network; RF: Random Forest; DNN: Deep Neural Network; MCTS: Monte Carlo Tree Search; EFMs: Elementary Flux Modes.

by enzymatic reactions. These transformations can significantly influence the pharmacokinetics, efficacy, and toxicity of the substances, impacting their overall behavior and effects on health [61]. Metabolism prediction aids in identifying potential metabolites and understanding their possible roles, thereby providing insights that are essential for drug development, toxicological assessments, and various biomedical applications.

One particular area of interest is the metabolism of xenobiotics. Xenobiotics are compounds that are not normally found in an organism. They can include drugs, environmental pollutants, and dietary components. These substances are mainly metabolized in the liver through a series of chemical enzymatic reactions, so they can be excreted from the body.

Understanding the biological and toxicological characteristics of xenobiotics is essential for developing safe and effective products. Yet, understanding the properties of the unmodified compounds is not enough. Organisms frequently metabolize these compounds, resulting in metabolites that can exhibit substantial differences in their physical, chemical, biological, and toxicological characteristics compared to the original compounds. The initial three generations of metabolites produced from a parent compound by xenobiotic-metabolizing enzymes are typically of great interest, as they can normally exist in concentrations that are pharmacologically or toxicologically meaningful [62]. Therefore, accurately predicting the metabolism of xenobiotics is vital for comprehending their pharmacokinetics, potential toxicity, and overall implications for health.

Currently, sophisticated *in vitro* and *in vivo* techniques are used to explore the metabolic pathways of small organic compounds. *In vitro* methodologies often involve purified enzymes, cell lysates, cells

(especially hepatocytes), or more sophisticated cell-based models [63], [64]. *In vivo* studies into xenobiotic metabolism are predominantly carried out in rats. However, these experimental methods remain both time and cost-intensive, making *in silico* approaches attractive alternatives for preliminary screening and hypothesis generation. These computational methods leverage simulations and modeling to predict metabolic reactions, offering a more efficient and cost-effective complement to traditional *in vitro* and *in vivo* studies.

2.1.2.1 Metabolism Prediction Tools

In metabolite structure prediction, the most established *in silico* methods utilize sets of biotransformation rules, commonly known as reaction rules, to predict the probable products of compound metabolism. These rules can be encoded using patterns such as SMIRKS [65] or SMARTS [66]. Reaction rules can be created manually by experts [67], [68], extracted through computational methods [27], [69], [70], or developed using a combination of both approaches.

Most *in silico* metabolism prediction tools are developed for specific reaction classes or particular metabolic or biotransformation processes. Some focus exclusively on secondary metabolism, while others are dedicated to primary metabolism transformation products.

Rule-based tools for metabolite structure prediction are highly valued for their interpretability. However, a significant drawback of this approach is the tendency to generate a large number of metabolites, many of which are not observed experimentally. This issue becomes more pronounced when predicting multiple generations of metabolites, often resulting in a combinatorial explosion. To mitigate this challenge, researchers have increasingly integrated rule-based approaches with ML. These hybrid methods leverage ML to predict sites or bonds of metabolism, guiding the application of biotransformation rules to specific atom or bond positions, or to estimate the likelihood of a compound serving as a substrate for particular metabolic enzymes.

Some computational tools for the prediction of the structure of compound metabolism include:

BioTransformer 3.0 [71] is a freely available web server that combines both ML and rule-based approaches. BioTransformer predicts the metabolism of small molecules across various human tissues, the human gut, and external environments such as soil and water microbiota. Specifically, BioTransformer takes a molecular structure as input and generates predicted metabolites or transformation products, along with the enzymes predicted to catalyze these reactions.

SyGMa [72], a prominent freely available tool for predicting metabolites, integrates expert knowledge formulated into reaction rules with an empirical scoring approach to prioritize specific biotransformations. The current version of SyGMa encompasses 175 biotransformation rules, encompassing both phase I (oxidation, reduction, hydrolysis) and phase II (conjugation) reactions.

MetaTrans [73] is a a rule-free, end-to-end learning-based method for predicting human metabolites of small molecules. This approach leverages a transformer architecture, pre-trained on general chemical

reactions extracted from patents and fine-tuned with human metabolic reaction data from sources such as DrugBank [74] and Recon3D [75]. MetaTrans models the metabolite prediction problem as a sequence translation problem using the SMILES [76] representation of molecules. Additionally, an ensemble model is constructed to account for the generation of multiple and diverse metabolites. This ensemble combines several fine-tuned models, each trained under different conditions. The final output is the union of the sets of predicted metabolites from each individual model.

GLORYx [77] is an enhanced version of GLORY [78] that expands its scope to encompass a wide range of both phase I and phase II metabolic enzymes, rather than focusing solely on cytochrome P450-mediated metabolism. GLORYx applies reaction rules to all relevant positions in a molecule, as determined by the matching of each reaction rule SMIRKS. It predicts the likelihood of biotransformation for all atom positions within a molecule and, based on these predictions, generates and ranks the likely metabolites.

CyProduct [79] is a ML-based, *in silico* metabolism predictor specifically designed to predict the metabolic byproducts of a given molecule when exposed to the nine most relevant human CYP450 enzymes involved in Phase I metabolism. CyProduct consists of three modules. The first module, CypReact, predicts whether a query compound will act as a reactant for a specific CYP450 enzyme. The second module, CypBoM, identifies which part of the query molecule, if any, will react with the given CYP450 enzyme. The third module, MetaboGen, predicts the metabolic byproducts based on the predictions from the previous modules.

MetaTox [80] is a rule-based metabolite prediction method that considers nine distinct reaction classes: aliphatic and aromatic hydroxylation, N- and O-glucuronidation, N-, S-, and C-oxidation, as well as N- and O-dealkylation. These reactions are catalyzed by five human cytochromes P450s (1A2, 2C19, 2C9, 2D6, 3A4) and human UDP-glucuronosyltransferase. Initially, MetaTox predicts the biotransformation reaction class applicable to the molecule. Subsequently, it identifies the specific atom within the substrate likely to undergo modification during this reaction. Lastly, MetaTox computes the overall probability of the metabolite being metabolized by computing the probabilities derived from the predicted reaction classes and reacting atoms.

2.1.3 Databases for Metabolic Engineering

ME tasks, especially pathway design, rely heavily on the quality of available data. Selecting appropriate data stands as a critical initial step for all pathway design tools. Currently, a variety of freely available web repositories offer extensive resources such as biochemical reactions, pathways, biochemical reaction templates, metabolites, and enzymes across multiple organisms. These repositories typically host diverse types of information, each specializing in specific data categories.

Currently, several databases contain known biochemical reactions and pathways, including MetaCyc [81], BRaunschweig ENzyme DAtabase (BRENDA) [82], Kyoto Encyclopedia of Genes and Genomes (KEGG) [83], Reactome [84], ModelSEED [85], among others. MetaCyc, a curated repository of metabolic

pathways and enzymes, unfortunately, transitioned to a subscription-only model starting January 1, 2024. It encompasses approximately 3,153 pathways, 19,020 reactions, and 19,372 metabolites, all validated through experimentation. Despite this change, MetaCyc remains a crucial resource across various disciplines. Access to MetaCyc is available at `https://MetaCyc.org`.

BRENDA (`www.brenda-enzymes.org`) is a comprehensive enzyme information system. It is a widely used resource in biochemistry, providing detailed data on enzyme function, including their biochemical properties, reaction mechanisms, substrate specificity, and molecular and catalytic properties. The database is curated and continually updated with information from scientific literature, ensuring its reliability and relevance. It provides access to a collection of over 84000 EC classes and almost 23000 reactions.

KEGG (`https://www.genome.jp/kegg/`) is a comprehensive resource that integrates biological information from genomes, pathways, diseases, drugs, and chemical substances. It is widely used in bioinformatics and systems biology for understanding biological systems at molecular levels. It includes over 1 million pathways from over 10 thousand organisms.

The Reactome Knowledgebase (`https://reactome.org`) is a curated biological database that provides detailed pathway information on human biological processes. It contains almost 3 thousand curated human metabolic pathways, over 15 thousand reactions, 11 thousand proteins, and 2 thousand small molecules.

Additionally, a small group of databases that contain hypothetical reactions include the Atlas of Biochemistry [86], Metabolic In silico Network Expansions (MINE) [87] and RetroRules [27] databases. The ATLAS of Biochemistry is a database of both known and hypothetically possible enzymatic reactions. This repository is organized in a web database available at `http://lcsb-databases.epfl.ch/atlas/`. It contains around 149 052 known and novel reactions generated by BNICE.ch.

RetroRules is a database of reaction rules for metabolic pathway discovery and ME. It provides more than 350 000 reaction rules extracted from several public databases augmented by a rule representation at different levels of specificity. The rules are available for download at `https://retrorules.org/dl`.

The MINE database expands upon existing metabolite databases by incorporating potential molecules that have not yet been observed but are predicted to exist based on known metabolites in biochemical databases. It utilizes the BNICE.ch algorithm and expert-curated reaction rules to propose new metabolites and reactions. In its latest version 2.0, MINE has predicted over 10 million putative compounds using approximately 500 reaction templates. MINE database is freely available via web-interface at `https://minedatabase.ci.northwestern.edu/`.

While numerous resources provide extensive data on metabolites, enzymes, and metabolic pathways, integrating and cross-referencing information from these databases can be challenging. Databases often contain discrepancies such as incorrect stoichiometries, imbalanced charges, and redundancies arising from varying nomenclatures [88]. This fragmentation generally requires further curation and integration to standardize data across different sources and ensure the coherence and connectivity of metabolic networks.

2.2 Machine Learning

AI is a fairly comprehensive computer science branch that aims to build machines capable of performing tasks that typically require human intelligence [89]. AI can be implemented in many ways, some of the major technological branches include ML, Evolutionary Computation, Robotics, Expert Systems, Computer Vision, Speech Processing, natural language processing (NLP), and Planning.

One of the most popular sub-fields of AI, ML, encompasses a set of techniques concerned with the development of algorithms that instead of being explicitly programmed, can progressively improve their performance at a given task by learning directly from data [90].

2.2.1 Types of Learning

Depending on whether feedback is given during the learning process, ML can be divided into four main categories. These categories are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (RL).

2.2.1.1 Unsupervised Learning

In unsupervised learning [91], there are no labeled data. In this case, the algorithms try to find similarities or relationships between the input data. Two of the most known unsupervised learning techniques are clustering and dimensionality reduction.

In cluster analysis, without receiving any external feedback, unlabeled data is grouped by similarity. The core objectives of these algorithms are to identify the relationships or hidden patterns shared among the data inputs to get meaningful insights, segment datasets or reduce the complexity of the data to facilitate interpretation. Two of the most known clustering approaches are K-means and hierarchical clustering.

Dimensionality reduction algorithms try to meaningfully represent overcomplex data in fewer dimensions, which has many advantages such as allowing better visualization and reducing the processing time and storage required. Some of the most known dimensionality reduction algorithms include principal component analysis (PCA) and T-distributed Stochastic Neighbor Embedding (t-SNE).

2.2.1.2 Supervised Learning

In supervised learning [92], given input-output pairs (x_i, y_i), a function to map the input x_i to the output y_i is learned so the model can predict unseen examples. Depending on whether the output y_i is numerical or categorical, supervised learning problems can be divided into two main groups, regression and classification problems, respectively.

Some well known regression and classification problems include linear regression (LR), logistic regression (LgR), RFs, support vector machines (SVMs), k-nearest neighbors (KNN), and NNs.

2.2.1.3 Semi-Supervised Learning

Semi-supervised learning [93] is very useful in case where getting labeled data is expensive or time-consuming, while unlabeled data is readily available. Initially, the model is trained on the labeled data only. Then the model is used to generate "pseudolabels" for the unlabeled data. Finally, the model is retrained using both the labeled data and the unlabeled data with the "pseudolabels". This process can be iterated many times to improve the performance and generalization power of the model.

2.2.1.4 Reinforcement Learning

RL [94] is a dynamic approach where algorithms are trained using a system of cumulative rewards and punishments. In a RL algorithm, an agent learns by interacting with an environment and receiving rewards or penalties by evaluating the impact of actions to maximize a given cumulative reward. In this type of learning, there is no need for labeled input-output pairs. In RL, the agent decides what to do to perform the given task to maximize a reward and is capable of learning from experience. In RL, the output (reward) and the next state in each step depend on the current state and the action taken, so it is all about sequentially making good decisions.

2.2.2 Supervised Learning: Model Selection, Overfitting and Underfitting

The central challenge in supervised ML is to find patterns and relations between data that best capture regularities in the underlying population from which our training dataset was drawn. However, how can we truly know that our models will perform well on new data and not simply memorize the training data? The capacity of a model to perform well on previously unseen data is called generalization.

When training a supervised ML model, we need to take into consideration two types of errors, the training error and the test error. The training error is computed based on the difference between the training dataset values and the model predictions. The test error or generalization error is calculated the same way, but using a test dataset containing data never seen by the model drawn from the same distribution as our training data.

When dealing with simple models and high amounts of data, it is expected that the training error and the test error would be very similar. However, when a large volume of data is not available or more complex models are used, it is common for the training error to be much lower than the test error. Model complexity can be measured in many ways and can vastly influence its generalization power, being normally measured by the number of tunable parameters. In a model, there are two types of configuration variables that are internal or external to the model. The internal configuration variables, the model parameters, are estimated directly from the data. Model parameters include weights in artificial neural network (ANN) and support vectors in a SVM. The external configuration variables, the model hyperparameters, cannot be estimated from the data and must be set prior to the training of the model. Model hyperparameters include the learning rate in ANN and the K in the KNN algorithm.

When training a ML model, a variety of different aspects need to be taken into account. Depending on particular metrics, the model and respective set of hyperparameters that best address the problem of interest need to be selected. The selection of the best model can be done using different approaches. For instance, in some cases, an independent validation dataset, that has never been used either for training or to calculate the generalization error, is employed to select the best hyperparameters and values for a given model. When few training examples are available, it is also common to adopt an approach called K-fold cross-validation. In this approach, the complete dataset is split into K non-overlapping subsets and the model is trained K times using, each time, one of the K different subsets for validation and the remaining $K - 1$ subsets for training. In the end, the training and validation errors are the average of the K experiments.

When training ML models one needs to be aware of two common phenomena. It is possible to look at the training and validation errors to detect both of these situations. When the training and validation errors are above some values that are expected, but are similar between them, we can be in the presence of a model that is too simple or needs more time to train. In this case, the model may be underfitting. On the other hand, when the training error is very low and significantly lower than the validation error the model may be overfitting. Both these situations are generally not wanted, and the ideal is to find a balance between these two cases. Figure 1 shows the influence of model complexity in the training process.

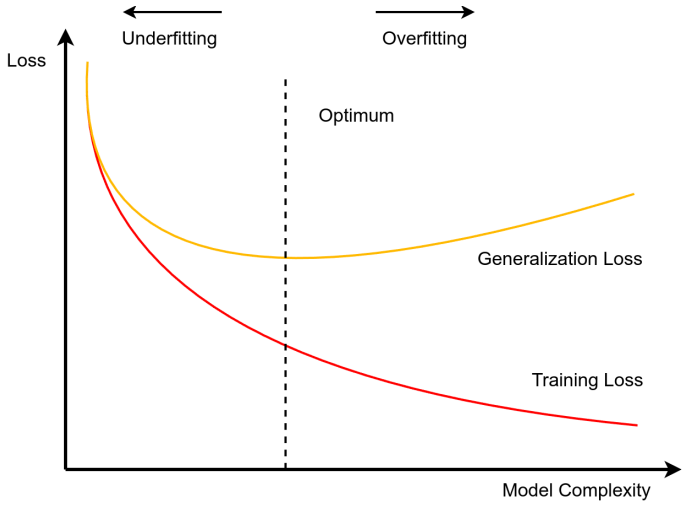


Figure 1: Influence of model complexity on underfitting and overfitting.

2.2.3 Supervised Learning Algorithms

2.2.3.1 Multiple Linear Regression

Regression analysis is a statistical technique to investigate the relationships between one continuous dependent variable and one or more independent variables. Depending on the number of independent variables, LR can be divided into simple LR and multiple LR.

In multiple LR, multiple independent variables x_i are used to predict the outcome of the dependent variable y. The multiple LR equation 2.1 can be expressed as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon \tag{2.1}$$

where β_0 represents the y-axis intercept, $\beta_1, \beta_2, ..., \beta_k$ the slope coefficients for each independent variable and ϵ the error component.

2.2.3.2 Logistic Regression

LgR is used instead of LR when the dependent variable is categorical. LR is suited to deal with regression problems and LgR to classification problems. In LgR, the probability of a certain outcome is predicted given a number of continuous input variables.

LgR can be divided in three main types, binary LgR when the dependent variable has only two possible outcomes (0 or 1), multinomial LgR when more than two non-sequential categories are possible, and ordinal LgR when more than two ordered categories are possible.

2.2.3.3 K-Nearest Neighbors

The KNN algorithm is one of the simplest supervised ML algorithms capable of performing regression and classification tasks. It is defined as a lazy learning algorithm because it does not have a specialized training phase and uses all the data every time it is required to make a prediction. It is also considered a non-parametric learning algorithm because it does not make assumptions about the data distribution.

This algorithm assumes that similar data points are close to each other. It calculates the similarity between the training samples and the new data points and classifies them by selecting the majority class of the k closest training samples (in case of a classification task) or the mean of the output variable of the k closest training examples values (in regression tasks). The KNN algorithm can use multiple metrics to calculate the distance between the data points.

2.2.3.4 Support Vector Machines

SVM is a supervised learning algorithm originally introduced in 1992 by Boser, Guyon, and Vapnik [95]. SVMs are one of the most widely used ML algorithms for nonlinear regression and classification. The main idea of SVMs is to map the data into a high-dimensional space and identify a lower-dimensional hyper-plane that best separates the data points using non-linear kernels. The optimal hyperplane is defined as the one that maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class in the training set.

Any hyper-plane can be represented in $\mathbb{R}^N$ by the following equation 2.2:

$$w \cdot x + b = 0 \tag{2.2}$$

where w is the normal vector to the hyper-plane and b is a bias scalar. Equation 2.3 represents the margin m between the hyper-plane and the closest data point that we want to maximize.

$$m = \frac{2}{||w||} \tag{2.3}$$

Maximizing the equation 2.3 is equivalent to the following equation 2.4:

$$\min_{w} \frac{||w||^2}{2} \tag{2.4}$$

Besides maximizing the margin, we also want to prevent data points from falling into the margin, so the previous equation is subjected to the following constraint function $f(x_i)$:

$$f(x_i) = \begin{cases} 1, & w \cdot x_i + b \geq 1 \\ \text{-1}, & w \cdot x_i + b \leq -1 \end{cases} \tag{2.5}$$

where x_i are called support vectors. These constraints ensure that each data point must lie on the correct side of the margin.

This constrained optimization problem can be solved in multiple ways. Linear separation and non-linear kernels can be used in SVMs to solve this problem. When the data are not linearly separable, it is common to use two different approaches. It is possible to allow the SVM to make some mistakes in order to keep the margin as wide as possible so the most number of data points can be classified correctly. This formulation is called soft margin. Using a soft margin can also help in avoiding overfitting. In this case, we aim to minimize the following objective:

$$\min_{w} \frac{||w||^2}{2} + C \tag{2.6}$$

where C is a hyperparameter of the SVM algorithm that controls the trade-off between maximizing the margin and minimizing the number of mistakes. The higher the value given to C, the higher the importance of avoiding miss classifications at the expense of keeping the margin at a lower value.

The second approach is called the kernel trick. Briefly, to separate the non-linearly separable data we need to transform our data into higher dimensional data. And for that, we theoretically need to compute the exact transformation of our data, which is actually a hard task. However, we only need the inner product of our data in that higher dimensional space, which is a lot easier to get. Polynomial, Gaussian, and other kernels can easily do that to an infinite dimension space. In this way, we can get our solution without even calculating the exact coordinates of our data points in a higher dimensional space.

2.2.3.5 Random Forests

RF, first proposed by Breiman in 2001 [96], is a supervised ML algorithm used for both classification and regression consisting of an ensemble of multiple decision trees (DTs) trained on random subsets of the training set and using a random subset of features. Each DT provides a classification for each input data point and the RF decides the final classification by choosing the most voted prediction in classification tasks and the average in regression tasks.

A concept behind the RF function is Bagging. DTs are trained by selecting random subsets of the training data with replacement. One of the main advantages of using Bagging to train individual DTs is that it decreases the model's variance. Individual trees are highly prone to overfitting and susceptible to noisy data. However, combining DTs with Bagging in RFs enables the creation of more robust models without increasing the bias.

The DT is a model representation that recursively applies binary partitioning of the feature space. In a DT, each feature in the sample can be represented as a node in the tree and each possible value of the feature as a branch. The root node of the DT should be the feature that best divides the whole data, and so on for all the internal nodes considering the subset of examples that reaches the node. The identification of the best features is made by measuring node impurity that can be calculated using mostly four different measures, the Gini Index or Entropy for classification tasks, and the standard deviation of the mean squared error (MSE) and mean absolute error (MAE) for regression tasks. To further improve performance, it is possible to apply a process called pruning that involves removing branches from the tree that make use of features with low importance.

Another key feature of RFs is the out-of-bag error, which estimates the generalization error during the training phase. Each time a random sample is used to train a decision tree, a smaller subset of data is held out and used as a test set for that tree, providing an approximation of the model's performance.

2.2.3.6 Artificial Neural Networks

ANNs are ML algorithms that were originally inspired by the way biological neural networks in the human brain process information [97]. ANNs became very popular due to the many breakthroughs achieved in many fields like text processing, speech recognition, pattern recognition, and computer vision.

An ANN is an oriented graph consisting of artificial neurons, like neurons in a brain, represented as nodes connected by arcs, the last mimicking the dendrites and synapses, that transmit information to other neurons. Every ANN is constituted by a single input layer, a single output layer, and zero or more hidden layers. The number of neurons in the input layer must be the same as the number of input variables in the data being used (eventually after pre-processing). The number of neurons in the output layer is typically one for binary classification or regression, or equal to the number of possible classes in a multiclass classification task. The number of hidden layers and their respective neurons can be variable and depend on the data being used. Figure 2 shows an ANN with an input layer with 4 neurons, one hidden layer with 3 neurons, and an output layer with a single neuron.

The basic unit of an ANN is the neuron. The neuron receives input from an external source (the input variables of the dataset), in the case of the neurons in the input layer, or from other neurons in previous layers (in the hidden and output layers), and computes an output. The neuron also receives another input called bias b, a constant that helps the model to better fit the given data ranges. Each input has an associated weight w, that reflects its relative importance to the calculation of the output.

After summing all inputs multiplied by their weights, the neuron applies a function f to the weighted

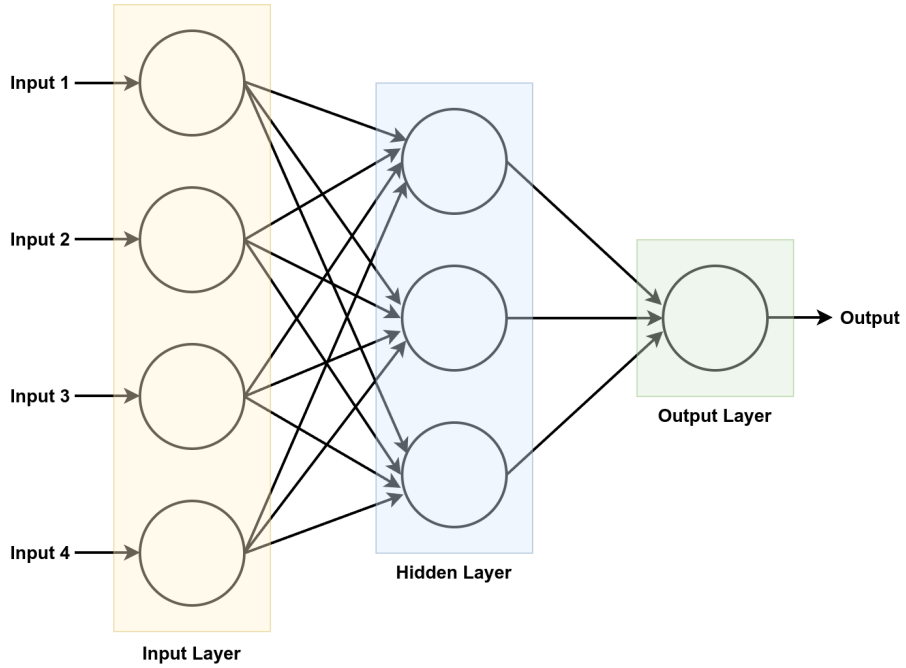


Figure 2: Fully Connected Artificial Neural Network with a single hidden layer.

sum of the received inputs. The function f is called the activation function and typically consists of a non-linear function, which is important because most real-world data are non-linear. Many activation functions can be applied, some of the most common include:

- Sigmoid (Figure 3 (a)): receives a real number and returns a value between 0 and 1

$$\sigma(x) = \frac{1}{1 + e^{(-x)}} \tag{2.7}$$

- TanH (Figure 3 (b)): receives a real number and returns a value between -1 and 1

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

- ReLU (Figure 3 (c)): receives a real number and returns the same number if it is positive or 0 if it is negative

$$f(x) = \max(0, x) \tag{2.9}$$

Figure 4 shows a scheme of an artificial neuron with respective inputs x and weights w, bias b, weighted sum function $\sum$, activation function f, and output y.

The simplest type of ANN is the feedforward neural network (FNN). In a FNN, the flow of information only moves in one direction, forward, from the input nodes, through the hidden nodes, if any, until reaching the output nodes. In this type of network, there are no cycles or loops. Two examples of FNN are the single layer perceptron (SLP) and multi layer perceptron (MLP). The SLP is the simplest FNN and only contains one input layer and one output layer with no hidden layers. The MLP has one or more hidden layers. In the next section, Deep Learning, more types of ANNs will be addressed.

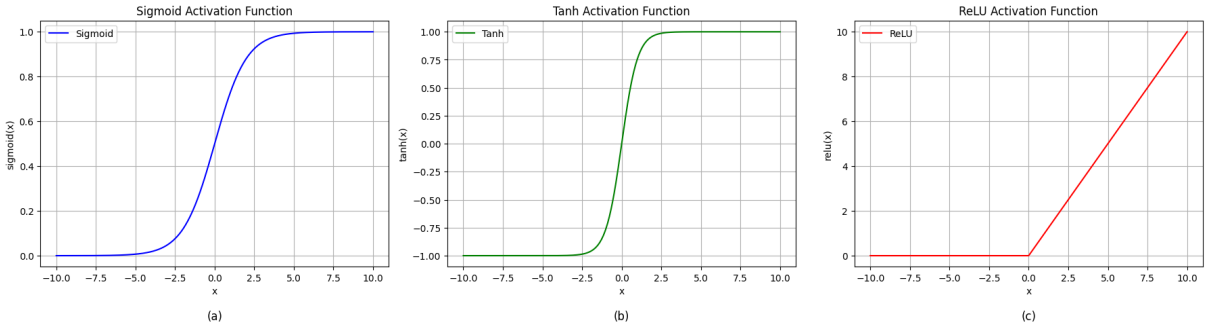


Figure 3: Activation functions. (a) Sigmoid function; (b) Tanh function; (c) ReLU function.

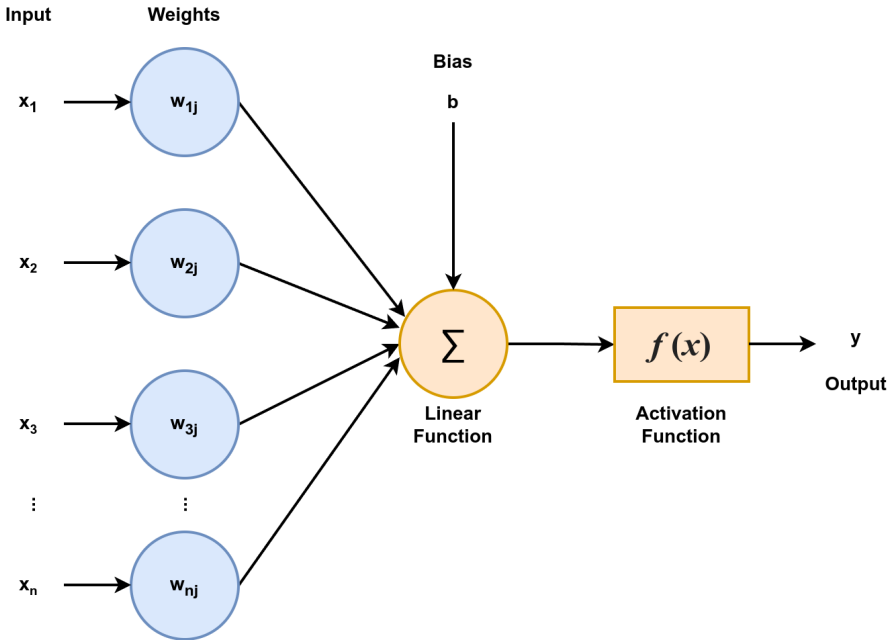


Figure 4: Structure of the Artificial Neuron.

The training of these types of ANN is done in two main steps, forward propagation and backpropagation with weights update. In the forward propagation, at first, all the weights are randomly set. Then, the performance of the model is measured by passing the inputs through the network layers and comparing the generated outputs with the real ones. The difference between the generated outputs and the real ones is calculated using a loss function. In the second step, backpropagation and weight update, using an optimization technique the internal weights of the network are modified to minimize the total loss function. A common approach is to use Gradient Descent [98] methods as the optimization method to update the network weights. This process is repeated for each training example multiple times and evaluated using the loss function that keeps track of the changes and associated costs.

2.2.4 Deep Learning

DL is a subfield of ML that uses multi-layered ANNs. It can automatically learn representations from data like text, images, and video without the need for hand-coded rules of human knowledge. DL is commonly

used in tasks such as object detection, speech recognition, language translation, and many others.

Until the end of the last century, given the scarcity of available data and computational power, traditional ML methods like SVMs and RFs provided empirically superior results than most ANN. ANNs required long periods of training and provided results without the strong theoretical statistical guarantees offered by other approaches. However, this situation changed with the availability of large volumes of data introduced by the World Wide Web and the advent of companies serving millions of users online. At the same time, the advent of more advanced and cheaper computational resources, in particular GPUs, also provided an advantageous environment for the development and implementation of more complex DL approaches. Because of that, algorithms and models that seemed computationally infeasible became relevant and started to draw the attention of the scientific community.

2.2.4.1 Deep Feedforward Networks

Deep Feedforward Networks are essentially MLPs with some hidden layers, which gives them further complexity. Also known as fully connected deep neural networks (FCNNs), these are the most common deep networks and, as explained in the previous section, can be formed by many layers of neurons that are fully connected to the ones in the previous and in the following layers. The training of such interconnected networks is not easy due to the risk of overfitting. To avoid this problem, regularization approaches can be used. The following topics will address this and other aspects related to the training and tuning of Deep Neural Networks (DNNs).

Regularization for Deep Learning DNNs can have high degrees of complexity which can make them prone to overfitting. This issue can be fought using regularization, a set of techniques that make modifications in the learning algorithm, so the model can generalize better and improve performance in unseen data. Regularization penalizes high-values in the weight matrices and when used adequately can result in simpler data fittings and, therefore, in a model with more generalization power. In DL, different regularization techniques can be used to reduce overfitting and improve generalization. The most used regularization techniques include:

- L1 and L2 regularization: when using these types of regularization, the general cost function is updated to include an extra term known as the regularization term.

$$costfunction = Loss + Regularizationterm \tag{2.10}$$

  With the addition of this regularization term, the values of the weight matrices are lower and, therefore, the complexity of the model decreases reducing overfitting. The regularization term is different when using L1 or L2 regularization.

  In L2 regularization, the cost function is:

$$\text{cost function } = \text{Loss} + \lambda * \sum ||w||^2 \tag{2.11}$$

where λ is a hyper-parameter of the L2 regularization that needs to be "manually" optimized to generate the best possible results. L2 regularization is also known as weight decay.

In L1 regularization, the cost function is:

$$\text{cost function } = \text{Loss} + \lambda * \sum ||w|| \tag{2.12}$$

Here, the main difference is that the absolute values of the weights are used. L1 regularization is often used to compress the model weights and perform variable selection.

- Dropout: this is one of the most used regularization techniques. At every iteration of the training process, a set of random nodes of a given layer are selected and are set to zero removing their influence in the calculation of the outputs. This approach resembles an ensemble method, where each iteration has a different set of active nodes, resulting in different outputs. The number of nodes that are set to zero in each training iteration depends on a hyperparameter named dropout probability that must be selected in advance.

- Data augmentation: the simplest way of avoiding overfitting is by training a model with as much data as possible. However, in practice, the amount of data that we have access to is limited. A way of dealing with this problem is to artificially create fake data and add it to the training dataset. Performing data augmentation is, however, not so straightforward for some types of data, being quite problem-specific.

- Early stopping: this is a very simple strategy that consists of stopping the training process when the validation error starts getting worse.

2.2.4.2 Convolutional Neural Networks

Convolutional neural networks (CNNs), also known as ConvNets, are a specialized type of NNs for processing data that has a grid-like topology like time-series data, which is a 1D grid, image data, which are 2D grids of pixels, and video data, which are 3D grids. CNNs are similar to normal ANNs, but use the convolution operation in at least one of their layers instead of general matrix multiplication. CNNs can be applied in many tasks and have been proven very effective in many areas such as image recognition and classification [99], NLP [100] and financial time series [101].

Convolutional Operation Convolution is a specialized type of linear operation. The objective of layers that apply convolution operations, the convolutional layers, is to detect specific features like edges and other visual elements. For that, the convolutional layers use filters, also known as kernels, that slide, or convolve, around the input and do weighted element-wise multiplication. In ML, the input is normally a multidimensional array of data and the kernel is a smaller multidimensional array of parameters. In the end, each value in the input will be a weighted sum of the values around that value. This kernel is normally applied over multiple axes at the same time. After each convolutional operation, an additional non-linear

operation called ReLU is applied to perform element-wise operations to replace all negative values in the feature map with zero.

Pooling Operation Aside from convolutional layers, CNNs also uses other layers such as the pooling layer and dense (feedforward) layer. After computing the output of convolutions in the neurons, connected to local regions in the input, a nonlinear activation function is applied. Then, a pooling function, in the pooling layer, is used to perform a downsampling operation along the input resulting in a statistical summarization of the nearby outputs. For instance, the average pooling operation reports the average output within a rectangle neighborhood. Dense layers are normally used as the last layer to compute the final output of the network. Figure 5 shows a schematic version of a simple CNN with the four main operations, convolution, ReLU, pooling, and fully connected dense layer. An illustration with a convolution and average pooling operation is also shown.

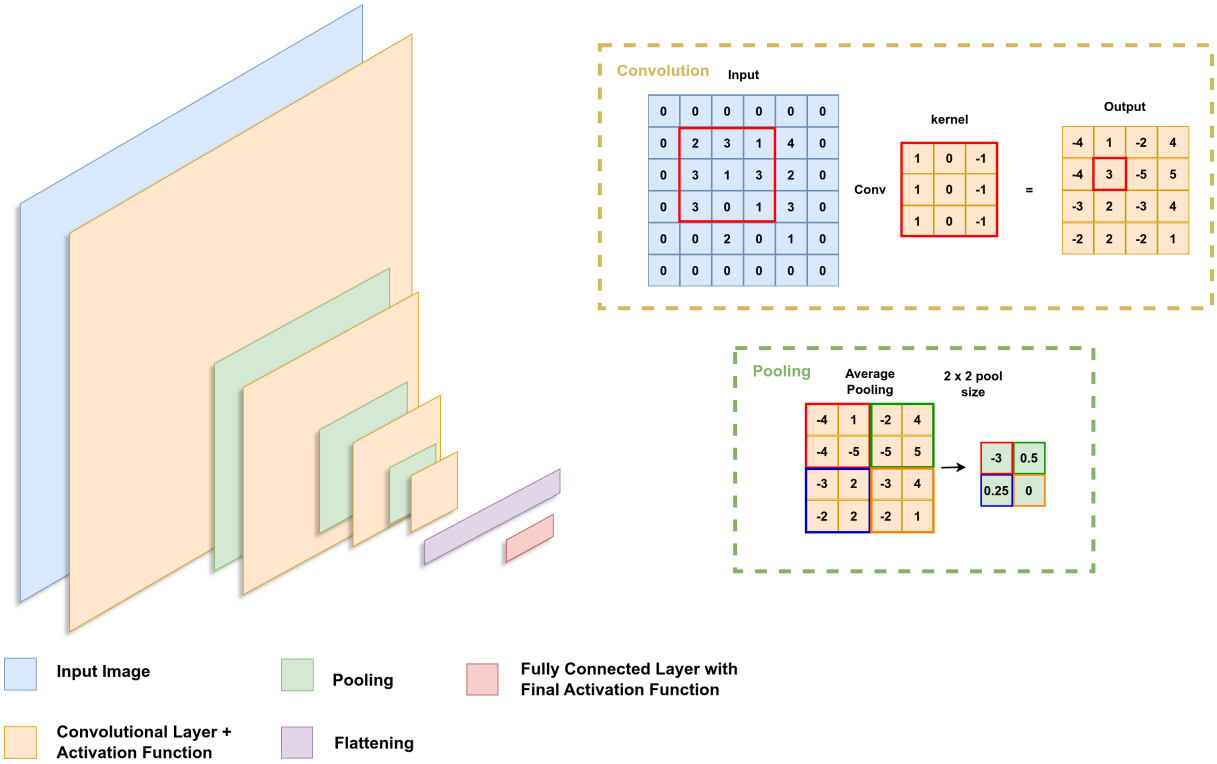


Figure 5: General CNN structure and illustration of convolution and average pooling operations.

2.2.4.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a specialized type of ANN for processing sequential data. This type of network is especially suited to deal with longer and variable size sequences of values than other ANN without sequence-based specialization. In a RNN, the input is normally a vector with values for each time step $x(t)$ that is mapped to an output vector $y(t)$. The network includes an internal hidden state $h(t)$, that is recalculated in each time step, which depends on the input for that time step and on the state for

the previous one. Thus, each output depends on the computations made in the previous time step and so on. These operations are represented as loops in the network.

$$y_t = f\left(x_t, h_{t-1}\right) \tag{2.13}$$

$$h_t = g\left(h_{t-1}, x_t\right) \tag{2.14}$$

In equations 2.13 and 2.14 the f and g functions represent the operations made to the input and previous hidden state to generate the output at time t and the new hidden state. These operations are done in a RNN cell. Multiple RNN cells can be layered with other ANNs structures to form a final model. The ability to deal with sequential data makes RNNs very suitable to handle tasks such as language modeling, text generation, automatic summarization, sentiment analysis, and question answering.

However, it is very hard for basic RNNs to remember information for long periods of time. Also, networks with many cells can lead to problems in the training process. The two major problems that often occur in basic RNNs are the exploding gradients and the vanishing gradients.

The exploding gradients occur when large gradient errors accumulate during the backpropagation process resulting in very drastic updates in the weights of the ANN model. This aspect makes the model very unstable and unable to learn from the training data. This problem can be handled by limiting the values that the gradients can have. The second problem, the vanishing gradients, is the opposite of the previous one. In this case, the gradient errors have very small values which may prevent the models' weights from significantly changing their values and prevent the model from further learning. To deal with these problems, several improvements were made to basic RNNs. Two of the major contributions include long short term memory (LSTM) and gated recurrent unit (GRU) networks.

Long Short Term Memory Networks LSTMs [102] are an adaptation of basic RNNs that solves the vanishing gradient problem by being capable of learning long-term relationships. The flow of information within a LSTM cell, the cell state, is controlled by internal mechanisms called gates. These gates include the input gate, forget gate, and output gate, which regulate which information is added to, removed from, or output from the cell state. Each gate is represented by a matrix of weights that are learned during the training process.

1. Input gate: decides which information from the input is important and can be used to modify the cell state. First, a sigmoid function is used to decide which values will be updated. Then using the tanH function, a weighted vector of new values is computed and added to the cell state. Because in this state we only want to add information to the cell state, a summation between the previous cell state and the result of applying a sigmoid function to the previous hidden state and input is done.

2. Forget gate: decides which information holds no importance and hence can be discarded. With such a purpose, the sigmoid function is used over the weighted input and previous hidden state to

select which information to omit (0) and which to keep (1) forming a binary vector. This vector is multiplied by the cell state matrix to define which information to keep and which to discard.

3. Output gate: this gate defines what the LSTM cell will output. First, the previous hidden state and the current input are passed through a sigmoid function. This new modified cell state is then passed through a tanH function. The output of the sigmoid and tanH functions and finally multiplied to decide what information the hidden state should carry forming the final output. The new cell and hidden states are then carried over to the next time step.

LSTMs can also be implemented in two different ways. Considering only information from previous time steps, the so-called Unidirectional LSTMs, or considering information from previous time steps and from future time steps, the Bi-Directonal LSTMs. This is done by training two LSTMs instead of only one: the first is trained with the normal input and the other with the reverse sequence of the input. This network will use two combined hidden states instead of only one. The network can then learn better and faster the problem at hand and provide better results.

Gated Recurrent Unit Networks As already stated, the GRU network works similarly to the LSTM but only uses two gates, the reset, and the update gate. The update gate is a simpler combination of the forget and input gate in the LSTMs network. It also combines the cell state and the hidden state present in the LSTM in a single variable hidden state.

1. Update gate: this gate in a GRU decides how much of the past information needs to be passed along to the future. It combines the functions of the forget and input gates found in an LSTM. Essentially, the update gate controls the extent to which the unit updates its content with new information, balancing between keeping the old memory and incorporating new input.

2. Reset gate: the reset gate determines how much of the past information to forget. It controls the influence of the previous hidden state on the current input, allowing the model to reset the hidden state if necessary.

LSTMs performs better when longer sequences are used because it has a better performance in remembering long-distance relationships. On the other hand, GRUs trains faster and regularly outperforms LSTMs when less training data is available. Figure 6 shows a simple RNN cell, a LSTM and GRU units.

2.2.5 Graph Neural Networks

Real-world data often exhibit relationships that can be effectively represented using graphs, where objects are nodes and connections between them are edges. To effectively utilize such structured data, NNs have been adapted specifically for graphs. These adaptations, known as graph neural network (GNN), are designed to process and learn from graph-structured data. GNNs have been applied across various domains

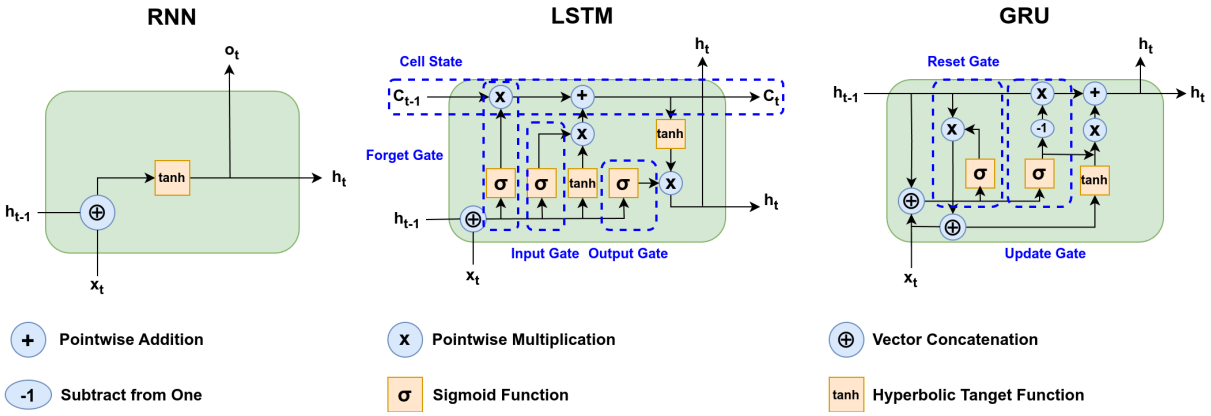


Figure 6: Schematic function of a simple RNN cell (right), LSTM unit (center) and GRU unit.

such as in chemistry for the generation of molecular fingerprints [103] and for chemical reaction prediction [104], in recommendation systems [105], [106], text classification [107], [108], image classification [109], [110].

Graphs contain up to four types of information that can be used for predictive tasks: nodes, edges, global context, and connectivity. Nodes and edges are straightforward components. Nodes can be organized into a node feature matrix N, where each node is indexed i and its corresponding features are stored.

Representing the connectivity of a graph offers more challenges. The most common approach is to use adjacency matrices. However, this has several drawbacks as graphs can contain millions of nodes with highly variable numbers of edges per node. This frequently results in adjacency matrices that are very sparse and inefficient in terms of space usage. Additionally, another issue is that multiple adjacency matrices can encode identical connectivity patterns and there is no assurance that different matrices representing the same connectivity would yield consistent results when used in a DNN.

An elegant and memory-efficient method for representing sparse matrices is through adjacency lists. Here, the connectivity of an edge e_k between nodes n_i and n_j is described as a tuple (i, j) within the $k-th$ entry of the adjacency list. This approach is advantageous because it avoids computation and storage for disconnected parts of the graph, given that the number of edges is typically much lower than the number of potential entries in an adjacency matrix (n^2 vs n nodes). Figure 7 shows an adjacency list of an undirected graph with 5 nodes and 5 edges.

The core idea of a GNN is to iteratively update the feature representation of each node by aggregating information from its neighboring nodes. This process is known as message passing. During message passing, each node collects information from its immediate neighbors, combines this information with its own features, and then updates its feature representation based on this combined information.

This process typically involves two main steps: aggregation and update. In the aggregation step, a node gathers features from its neighboring nodes. This can be done through simple operations like summation or averaging, or more complex ones like learned NN functions. In the update step, the node's own features are updated based on the aggregated information, often using a NN layer.

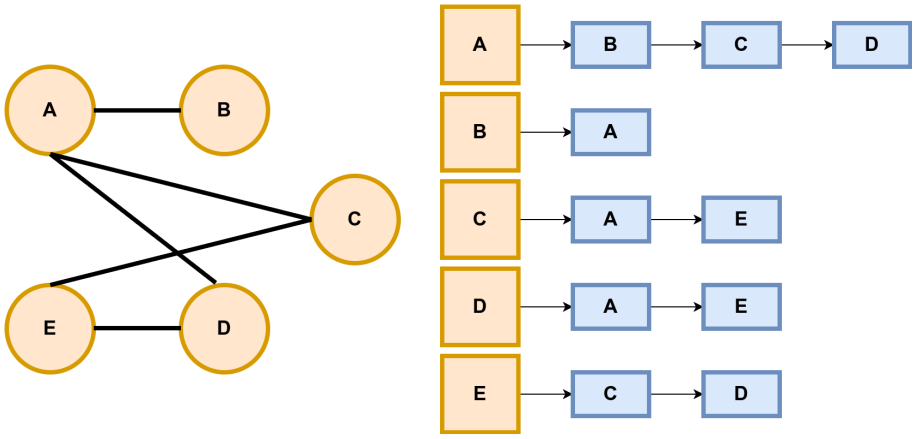


Figure 7: Example of an undirected graph and its adjacency list.

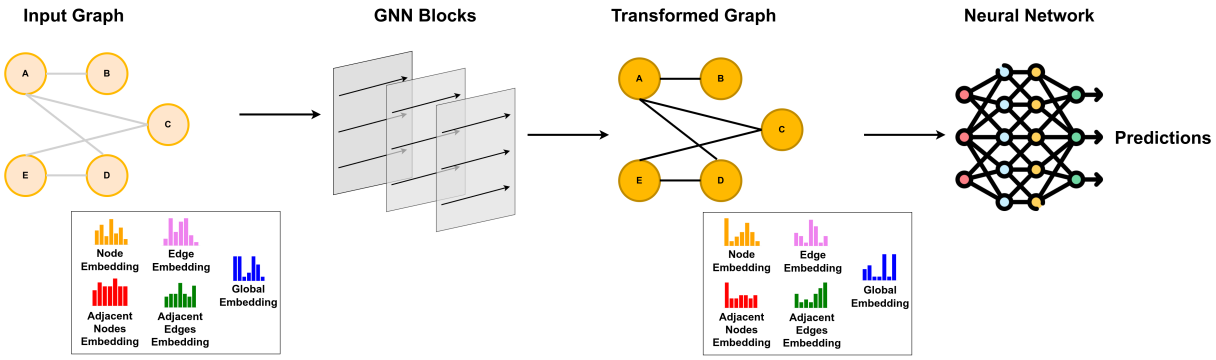


Figure 8: End-to-end prediction workflow with GNNs.

The message-passing process is repeated for several layers and iterations, allowing information to propagate through the graph. Each layer allows nodes to incorporate information from nodes further away in the graph, leading to increasingly rich feature representations.

Once the iterative message-passing process is complete, the final node features can be used for various tasks. For instance, pooling can be applied to aggregate the features of all nodes into a single feature vector that represents the entire graph. This feature vector can then be fed to regular DNNs for multiple purposes. Figure 8 shows the end-to-end workflow of a general GNN from input data to predictions.

Graph convolutional neural networks (GCNNs) are a type of GNN that extend the concept of convolution to graph-structured data. The key difference lies in the aggregation method: GCNNs use a specific convolutional approach where node features are updated through a weighted sum of their neighbors' features, structured by the graph's adjacency matrix. This convolution operation in GCNNs allows for a more structured and efficient way to capture local neighborhood information, making GCNNs particularly powerful for capturing hierarchical patterns within graph data.

Graph attention neural networks (GATs) are a specialized type of GNN that incorporate an attention mechanism to dynamically weigh the importance of neighboring nodes during feature aggregation. Unlike

traditional GNNs, GATs calculate attention coefficients for each neighbor, allowing the network to focus more on relevant neighbors and less on irrelevant ones. This is achieved by using a NN to compute these coefficients, which are then used to perform a weighted sum of the neighbors' features. This attention mechanism enables GATs to handle graphs with nodes of varying importance more effectively.

2.2.6 Transformers

The Transformer architecture was introduced in 2017 in a ground-breaking study by Vaswani *et al.* [111]. It consists of an encoder-decoder framework that relies on self-attention mechanisms rather than recurrent networks. This change from sequential processing enabled Transformers to handle long-range dependencies more effectively, making them particularly suited for tasks requiring understanding of context across large sequences.

The Transformer architecture includes several core components. It utilizes a self-attention mechanism that allows the model to assess relationships between elements by computing attention scores across all pairs of tokens. The encoder processes input data, while the decoder generates outputs. Each part is composed of multiple layers of self-attention and feed-forward neural networks.

To handle sequence order, Transformers incorporate positional encodings into input embeddings, providing positional information. This compensates for the absence of inherent sequential understanding present in other architectures. Additionally, FNNs within each layer independently processes outputs from the self-attention mechanism for each element in the sequence.

One additional key innovation introduced by Vaswani *et al.* is the multi-head attention, where each attention layer in the Transformer has multiple heads. Each head learns independently, allowing the model to specialize in different aspects of the input sequence. For example, in text-to-text problems, one head might focus on punctuation while another focuses on identifying the subject of a sentence.

The Transformer architecture offers significant advantages over traditional models like RNNs. Its parallelization capabilities enable efficient computation across entire sequences and drastically reduce training times. Transformers excel in capturing long-range dependencies within sequences through their self-attention mechanism, surpassing the common limitations in handling distant relationships between elements. This capability makes them well-suited for tasks requiring extensive contextual understanding, such as machine translation and document summarization.

Since its introduction, the Transformer architecture has been extended and adapted in various ways to improve performance and address different types of tasks. Examples of Transformer variants and extensions include, for instance, the well-known BERT [112] and the GPT architectures [113], [114].

2.2.7 Generative Models

Generative models are a class of unsupervised ML models that recently started to show promising results. Recent improvements in many stochastic optimization models and in the parameterization of generative models using DNNs, allowed the creation of improved and more complex models that can efficiently learn the probability distribution of a given training dataset and generate new realistic samples based on that

distribution. These models can deal with high-dimensional data such as images, text, and speech providing promising results in areas such as computer vision, speech recognition and NLP, data mining, and RL. There are many variations of deep generative models, the most popular being variational autoencoders (VAEs) and generative adversial networks (GANs).

2.2.7.1 Autoencoders

Autoencoders are a type of unsupervised NNs that tries to learn how to efficiently compute data into an encoded and compressed representation and then reconstruct it back to its original form as closely as possible. This new representation should only contain relevant information, therefore autoencoders must learn to ignore the noise in the data. This representation is commonly called the latent-space representation.

There are 4 main parts that should be mentioned when describing autoencoders: 1) the encoder, which is the part of the network that should learn how to encode the input data into a reduced and encoded latent-space representation; 2) the bottleneck layer, containing the latent-space representation and is the part of the network where the data has its lowest dimension; 3) the decoder, that tries to do the opposite of the encoder, since it tries to decode the latent-space representation back to its original form; 4) the reconstruction loss, that measures how close the output of the network is to the original input, therefore measuring how well the network is performing.

The training process of autoencoders is done using backpropagation to try to reduce the reconstruction loss as much as possible. Depending on the objective in mind, the architecture of the autoencoder can change from a simple FNN to a LSTM network or CNN. Figure 9 shows the general architecture of an autoencoder. Autoencoders are often used in tasks such as denoising images, compression, dimensionality reduction, and recommender systems.

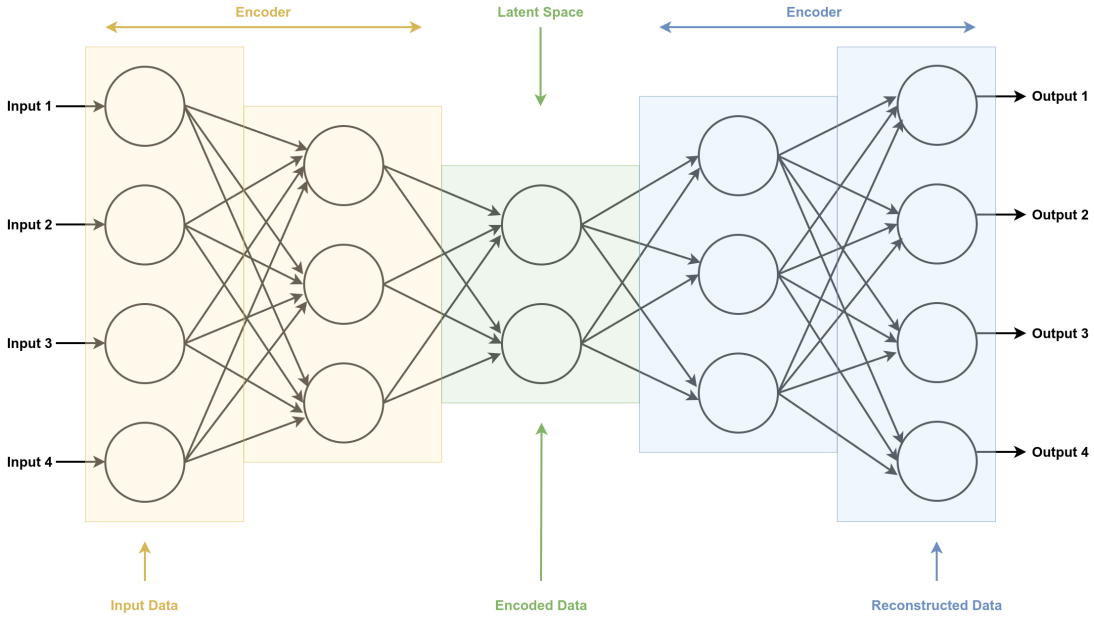


Figure 9: Diagram of an autoencoder architecture.

Variational Autoencoders VAEs [115], [116] are variants of autoencoders that are used to generate new data samples based on the probability distribution of the data in the training dataset. They also use the concepts of encoder, decoder, and latent-space representation. Random samples from the latent space are decoded using the network decoder to generate new data with characteristics similar to those of the training data. This type of model can learn the parameters of probabilistic distributions that are typically analytically intractable by using variational inference to find approximations of those parameters and therefore distributions.

The main difference between VAEs and normal autoencoders is the way that the latent space is constructed. In a VAE, the encoder outputs two vectors instead of only one. One is a vector of means μ and the second is a vector of standard deviations σ. This allows the model to easily perform random sampling and interpolation of the latent space because the initialization of a probability distribution with the parameters μ and σ creates a continuous latent space that will vary on each iteration due to sampling. This property allows the model to generate different data samples in the same distribution as the training data.

Generative Adversial Networks GANs [117] are DNNs architectures based on the game theory concept, Nash Equilibrium [118], where the objective is to find the balance between two networks denominated generator and discriminator. These networks compete against each other, thus the term adversarial, to generate new data samples that can pass as real data. The generator learns to generate successively more realistic new data samples that try their best to fool the discriminator. The discriminator does the opposite and tries to distinguish whether the data sample generated by the generator is a "real" data sample or a "fake" one. The generated data samples that do not pass the discriminator test become negative training examples for the discriminator. Every time the generator produces implausible results, the discriminator produces a signal that provides information that the generator uses to update its weights through backpropagation.

At the beginning of the training process of a GAN, the generator produces obviously fake data that is easily classified as fake by the discriminator. As the training keeps going, the generator gets better at producing new data samples, and the discriminator in parallel needs to improve its discrimination ability in telling if the data is real or fake. At the end of the process, both models should be competent in their respective tasks.

Adversarial Autoencoders Adversial autoencoderss (AAEs) have the same objective as the above-mentioned autoencoder but use a different approach. The objective is, like in VAEs, to generate a continuous latent space representation using the encoder and sample that latent space to generate new data samples using the decoder. However, it also uses a prior distribution to model the encoder output $d(\mu, \sigma)$. This prior distribution is forced into the model using a discriminator that decides if the latent space generated by the encoder follows that distribution or not influencing the model to only generate latent spaces following that prior known distribution. This type of autoencoder is a combination of the two autoencoders mentioned before, the VAE and the GAN.

2.2.8 Automated Machine Learning

Automated machine learning (AutoML), is a rapidly evolving field that aims to automate the end-to-end process of developing and applying ML techniques to data. Its primary goal is to make ML more accessible, efficient, and scalable, reducing the need for extensive domain knowledge. This democratization allows both non-experts and experienced users to leverage ML effectively in their studies and projects.

AutoML covers a range of techniques and methodologies designed to automate various stages of the ML pipeline. These stages typically include:

- Data Preprocessing: This step involves applying common preprocessing techniques to clean and prepare the data for analysis. Tasks include handling missing values and outliers, feature scaling, encoding categorical variables, and data augmentation;
- Feature Engineering: The aim here is to create and/or select the most relevant features from the data to enhance model performance;
- Model Selection: This step focuses on evaluating different ML models to identify the most suitable ones for the given data and task. Techniques such as meta-learning, which leverages knowledge from previous experiments to select promising algorithms, and ensemble learning, which combines multiple models to boost predictive performance and robustness, are commonly used;
- Hyperparameter Tuning: Optimizing the hyperparameters of the chosen models to improve their performance is crucial. Techniques like grid search, random search, and Bayesian optimization are often employed. More advanced techniques like neural architecture search (NAS) to efficiently explore and select neural network architectures can also be applied;
- Model Training and Evaluation: In this step, the selected models are trained using the optimized hyperparameters and then evaluated using methods like cross-validation with appropriate performance metrics;
- Model Deployment: The final step is deploying the trained models to production environments, enabling easy integration into applications and services.

By automating these stages, AutoML accelerates workflows and enables users to focus more on complex aspects of their projects rather than on repetitive tasks. For non-experts, AutoML serves as a valuable tool by lowering the barrier to entry into ML. It facilitates rapid prototyping and iteration, allowing users to explore a wide range of algorithms and hyperparameters efficiently. This capability often leads to good performance outcomes without requiring deep expertise in ML techniques.

Moreover, AutoML supports experts by speeding up their experimentation processes. Techniques such as meta-learning and ensemble learning help in selecting and combining models effectively, thereby boosting predictive performance and robustness. This automation also enhances productivity by streamlining the deployment of trained models into production environments. Figure 10 illustrates a general end-to-end workflow of AutoML systems.

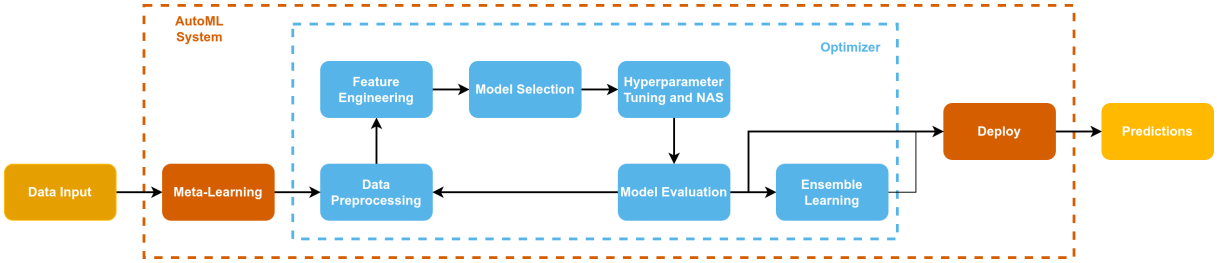


Figure 10: General workflow of an AutoML system.

Despite the significant advancements enabled by AutoML, several challenges persist. While a few solutions offer fully automated end-to-end ML pipelines that cover all necessary steps, these are often specialized for specific domains. The majority of AutoML solutions only support a limited subset of these steps. Additionally, issues such as interpretability and scalability remain significant limitations. AutoML models can be complex and difficult to interpret, making it difficult to understand their decision-making processes. Efficiently managing large datasets and handling complex tasks also poses technical challenges.

Moreover, achieving a balance between automation and the need for customized solutions tailored to specific problems is challenging. Customization often requires integrating automated processes with domain expertise, which can be difficult and time-consuming.

2.2.8.1 Open-Source AutoML Tools

Throughout the last decade, AutoML has evolved from experimental concepts to practical tools that democratize ML by automating complex tasks traditionally handled by data scientists and ML experts. The field continues to grow, driven by advancements in AI research, improvements in computational power, and increasing demand for accessible ML solutions across industries.

Describing and comparing all available AutoML tools comprehensively is a daunting task due to the rapid proliferation and diversity within the field. The number of available tools has grown exponentially in recent years, encompassing a wide range of commercial and open-source frameworks. Each tool varies in its focus, from automating model selection and hyperparameter optimization (HPO) to offering advanced capabilities like automated feature engineering and NAS. Given this vast array, it is nearly impossible to cover every tool in a brief overview. Therefore, here, we focus on a selection of prominent and widely adopted open-source AutoML frameworks (Table 2).

In the rapidly evolving field of AutoML, selecting the appropriate tool can be challenging. Most tools automate preprocessing, thereby easing the initial data preparation burden, though have limited preprocessing capabilities. Some tools focus solely on ML models without NAS capabilities, while others support both ML and DL, incorporating advanced NAS features. Various HPO strategies are employed, such as Bayesian Optimization, Genetic Programming, Hyperband, Random Search, and other flexible algorithms. Meta-learning is integrated into some tools, leveraging prior knowledge for improved performance. Tools vary in their framework compatibility, from traditional ML frameworks to those supporting DL. Several tools integrate well with cloud services, facilitating scalability, while others do not. Scalability varies, with

Table 2: Comparison of AutoML tools.

Tool	Optimization Strategy	Model Ensembling	NAS	Meta-Learning	Supported Frameworks	Cloud Integration
auto-sklearn [119]	Bayesian Optimization	Bag, Boost, Stack, Vote	No	Yes	scikit-learn	No
tpot [120]	Genetic Programming	Stack	No	No	scikit-learn	No
autogluon [121]	Fixed Defaults	Stack, Bag	Yes	No	MXNet, PyTorch	Yes
autokeras [122]	Hill-climbing	No	Yes	No	Tensorflow	Yes
h2o automl [123]	Random Search	Stack	No	No	-	Yes
nni [124]	Various	No	Yes	No	Agnostic	Yes

Bag: Bagging; Boost: Boosting; Stack: Stacking; Vote: Voting

some tools suited for large datasets and complex models. Additionally, all listed tools are open-source, making them accessible without licensing costs. Each AutoML tool offers unique strengths, making some ideal for traditional ML tasks, others for environments requiring DL and NAS, and some balancing ease of use with powerful ML capabilities. Selecting the right tool depends on specific project needs, such as the type of models required, the necessity for NAS, and integration with existing workflows and cloud services.

2.3 Evolutionary Algorithms

EAs are a class of optimization algorithms inspired by the natural selection process and biological evolution. They are used to solve complex optimization and search problems by iteratively improving a population of candidate solutions. These candidate solutions evolve through techniques inspired by Darwin's theory of evolution, including crossover, mutation, natural selection, and survival of the fittest.

During the 1960s and 1970s, many attempts to computationally model the process of evolution were undertaken, leading to the emergence of three major paradigms:

- evolution strategies (ESs) [125], which originated from an experimental optimization method developed for solving fluid dynamics problems;
- evolutionary programming [126], aimed at evolving the structure of finite-state automata through iterative selections and mutations, envisioned as an alternative to early artificial intelligence approaches;
- genetic algorithms (GAs) [127], designed to explore the mechanisms underlying systems capable of self-adaptation to their environment.

To complete the four main EAs paradigms, genetic programming [127] emerged as another significant approach. It focused on evolving computer programs rather than fixed structures, allowing the creation of complex solutions through iterative improvements and genetic operators.

Given that this thesis primarily utilizes GAs, our focus will be on providing a detailed description of them.

2.3.1 Genetic Algorithms

GAs are heuristic iterative algorithms that make the individuals of a population evolve over a number of generations. A generic GA begins with an initial population of candidate solutions, represented as chromosomes. These chromosomes are typically generated randomly or using a heuristic method that reflects prior knowledge about the problem domain.

The next step in the GA is fitness evaluation, where each chromosome's fitness is assessed based on its ability to solve the problem at hand. The fitness function quantitatively measures how well each solution performs. Higher fitness indicates a better solution to the problem, guiding the algorithm towards potential optimal solutions.

Following fitness evaluation, the selection phase determines which chromosomes will act as parents for the next generation. Selection is often biased towards chromosomes with higher fitness, though stochastic methods ensure diversity within the population. This diversity is crucial for maintaining genetic variability, which allows the algorithm to explore a broader solution space.

To produce offspring, genetic operators such as crossover and mutation are used, enhancing the genetic algorithm's ability to explore and exploit solutions. Crossover involves exchanging genetic material between selected parent chromosomes to create offspring. This process combines beneficial characteristics from different parents, potentially producing offspring with improved fitness.

Mutation introduces random changes into offspring chromosomes, ensuring the exploration of new solutions and preventing premature convergence to suboptimal solutions. It adds diversity by introducing novel genetic material not present in the current population, thereby enhancing the algorithm's ability to better explore the solution space.

After generating offspring, the next step in the genetic algorithm is replacement. Replacement involves selecting individuals from both the current population and the offspring population to form the new population for the next generation. Typically, individuals with higher fitness are favored during selection to ensure that the population evolves towards better solutions over successive generations.

Throughout successive iterations (generations), the genetic algorithm evolves the population by iteratively applying selection, crossover, mutation, and replacement steps. The algorithm continues this iterative process until a termination condition is met, such as finding a satisfactory solution or reaching a predefined number of generations. Figure 11 shows a general workflow of a generic genetic algorithm.

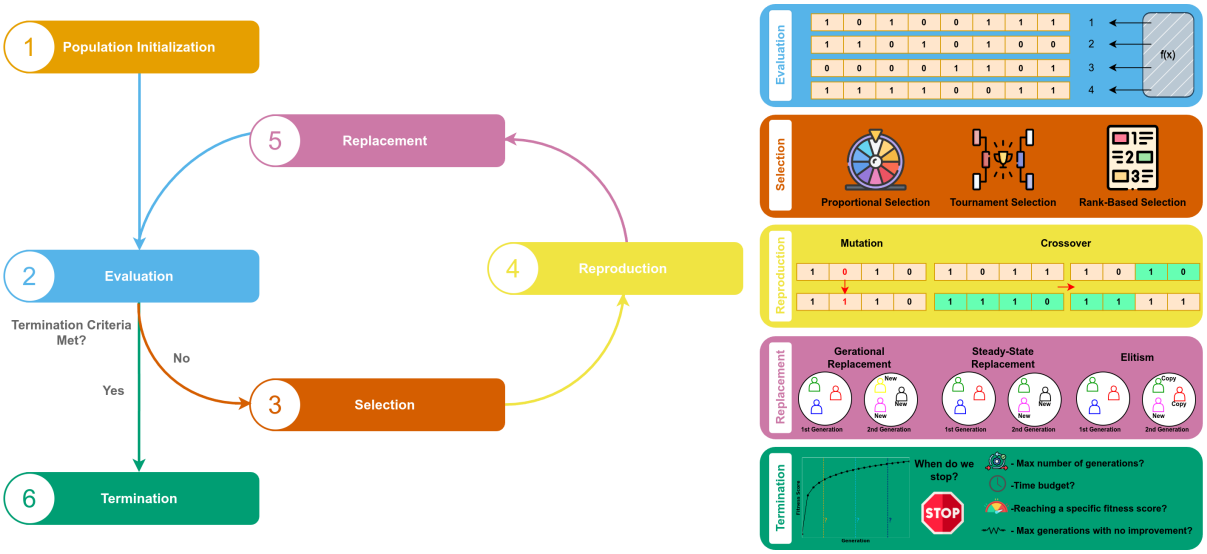


Figure 11: General overview of a genetic algorithm.

2.3.1.1 Selection Operators

Selection operators are mechanisms in evolutionary algorithms that determine which individuals from a population are chosen for reproduction. Their main goal is to favor individuals with higher fitness, which helps in the propagation of good solutions to future generations. Sometimes, less fit individuals are intentionally included as well to maintain genetic diversity and encourage the exploration of different solutions within the population. These methods aim to balance the exploration of new areas of the solution space with the exploitation of known good solutions [128], thereby enhancing the algorithm's ability to find optimal or nearly optimal solutions for complex problems.

Selection pressure plays a crucial role in these processes. It refers to the intensity with which selection operators favor individuals with higher fitness over those with lower fitness [129]. High selection pressure accelerates the convergence of the algorithm towards fitter solutions but may reduce genetic diversity if not carefully managed. Conversely, low selection pressure preserves diversity but may slow down convergence. Different selection methods vary in their inherent selection pressures, contributing to the overall performance and effectiveness of evolutionary algorithms.

Here are the main types of selection operators:

Proportional Selection Also known as Roulette Wheel Selection, Proportional Selection selects individuals for reproduction based on their fitness. In this approach, each individual in the population is assigned a slice of a circular "roulette wheel" directly proportional to their fitness score relative to the total fitness of the population. This method ensures that individuals with higher fitness have a higher probability of being selected. While simple to implement, it can be sensitive to scaling issues when fitness values vary widely, potentially leading to premature convergence or reduced exploration of the solution space if not properly adjusted.

Rank-Based Selection In rank-based selection, individuals are ranked based on their fitness, and selection probabilities are assigned according to their rank rather than their absolute fitness values. This approach aims to mitigate issues related to fitness scaling and fluctuation by focusing on relative fitness rankings. This effectively reduces the influence of outliers or extreme fitness values. However, it may result in slower convergence compared to fitness proportionate methods in certain scenarios.

Tournament Selection This approach involves randomly selecting a subset of individuals from the population and conducting a "tournament" where the fittest individual is chosen as a parent. This process is repeated to select the desired number of parents. Tournament Selection allows for adjustable selection pressure by varying the tournament size; larger tournaments increase selection pressure by favoring individuals with higher fitness, whereas smaller tournaments promote diversity by giving lower-fitness individuals a chance to be selected.

2.3.1.2 Genetic Operators

Genetic or variation operators in evolutionary algorithms are stochastic mechanisms that transform and combine individuals in a population to create offspring. These offspring may inherit beneficial traits from their parents and have the potential to be fitter.

There are two main types of variation operators: mutation operators, which alter individuals independently, and crossover operators, which generate offspring by combining traits from multiple parents. The way these operators modify individuals depends on how solutions are represented in the problem domain. For example, in optimization problems involving continuous spaces like $\mathbb{R}^n$, solutions are often represented as vectors in that space.

The effectiveness of variation operators lies in their ability to explore different parts of the solution space to find promising areas that likely contain optimal solutions, while also exploiting these areas to refine current solutions. Finding a balance between exploration and exploitation is crucial. Operators that explore widely may struggle to converge to optimal solutions efficiently, while operators that exploit too narrowly risk getting stuck in local optima. Achieving this balance is a key challenge in designing effective variation operators for evolutionary algorithms.

Mutation The mutation operator is a stochastic process that introduces small, random changes to individual solutions within a population. The idea is to modify individuals in a way that keeps the offspring close to the original, while still allowing for exploration of different parts of the search space. One important parameter of the mutation operator is the mutation rate which determines the proportion of individuals in the offspring population that undergo mutation. Early algorithms relied solely on mutation with a 100% rate due to the absence of crossover. A sufficiently high mutation rate aids in preserving population diversity, crucial for exploring the search space effectively. In binary representations, where solutions are represented as binary vectors (0s and 1s), mutation involves flipping individual bits (Figure 12).

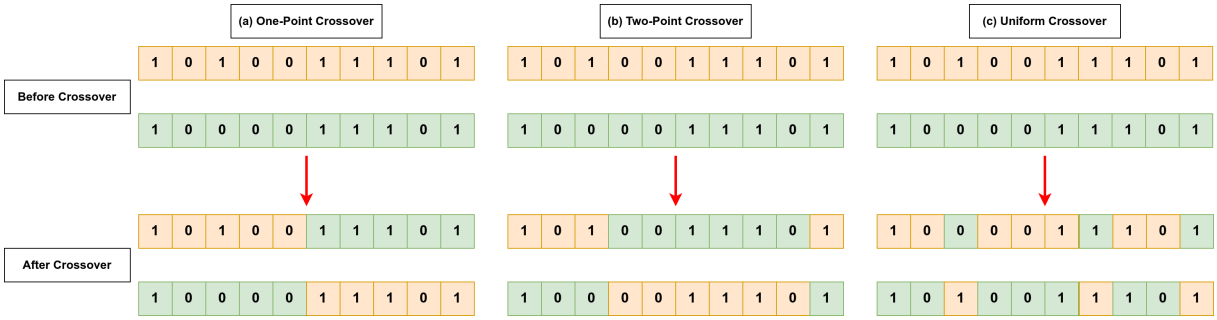


Figure 13: Two parent individuals and respective offspring after different types of crossover. (a) one-point crossover; (b) two-point crossover; (c) uniform crossover.

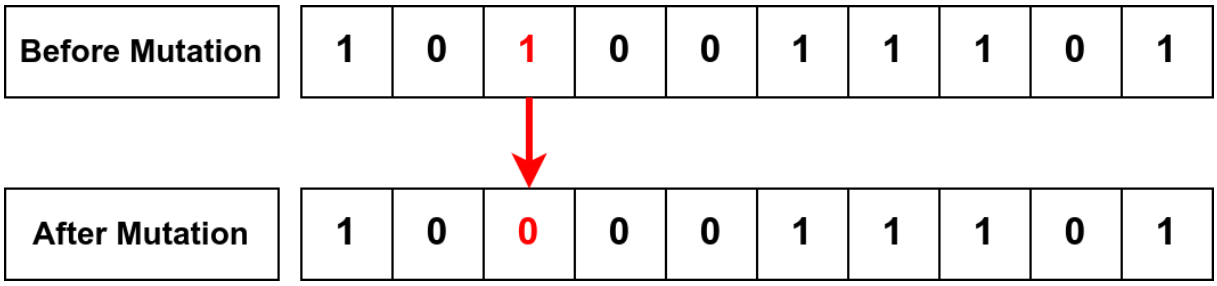


Figure 12: Individual before and after binary mutation.

Crossover The crossover operator is a stochastic process that combines genetic material from at least two parents to generate offspring. The crossover rate, a parameter of the algorithm, determines the proportion of individuals that undergo this process. Generally, crossover ensures that offspring from identical parents are identical to the parents themselves, and parents close to each other in the search space tend to produce offspring that are also close.

In simpler evolutionary algorithms, parents selected randomly might be located on different peaks of the fitness function. This scenario can lead to "flat crossover," where offspring have suboptimal fitness values. This case of crossover is known as lethal as it generates, from high-fitness parents, offspring that is too low fitness to survive. To mitigate this, it is common to restrict crossover to similar parents and implement variations of crossover that increase the likelihood of offspring being closer to their parents.

In binary representations, three main variants of the crossover operator are used: one-point, two-point, and uniform crossover. One-point crossover involves splitting the parents at a single randomly chosen point and swapping the tails to create offspring (Figure 13 (a)). Two-point crossover is similar to one-point but involves two points of exchange. Segments between these points are swapped between parents to create offspring (Figure 13 (b)). In uniform crossover, each bit position in the offspring is chosen randomly from one of the parents allowing a more uniform exchange of genetic information between parents (Figure 13 (c)).

2.3.1.3 Replacement Strategies

Replacement strategies play a crucial role in determining how individuals within a population evolve over time. These strategies dictate which individuals are selected to survive and produce offspring in subsequent generations. By determining which individuals are kept or replaced, these strategies affect how diverse the population is, how quickly the process reaches a solution, and how well the evolutionary method works overall. Here are some widely used replacement strategies:

Generational Replacement This is a straightforward replacement strategy where the entire population of individuals is replaced by a new generation of offspring at each iteration. While it offers the potential for rapid convergence and clear generational transitions, it must be carefully managed to maintain genetic diversity and avoid premature convergence.

Steady-State Replacement Here, only a small portion of the population is replaced with new offspring at each iteration, rather than replacing the entire population as in generational replacement. This approach allows for continuous and gradual integration of new individuals into the population, maintaining a steady population size throughout the evolutionary process. This helps maintain genetic diversity, ensuring robust exploration of the solution space while steadily driving the population towards optimal solutions. It is particularly suited for complex and long-term optimization problems where maintaining a diverse population is essential.

Elitism Elitism is a replacement strategy used in evolutionary algorithms to ensure that the individuals with the highest fitness are preserved from one generation to the next. By ensuring that top-performing individuals are preserved, elitism contributes to the efficiency and effectiveness of the optimization process. However, it is essential to manage the balance between preserving elite solutions and maintaining genetic diversity to avoid premature convergence and ensure robust exploration of the solution space.

2.3.2 Evolutionary Multi-Objective Optimization

Real-world problems are often complex, involving multiple criteria that need careful consideration. These problems may have multiple (potentially infinite) incomparable alternative solutions. Unlike traditional optimization approaches that focus on a single objective, multi-objective optimization seeks to find a set of solutions that represent trade-offs between conflicting goals, known as Pareto-optimal solutions. For instance, in drug design experiments, researchers may need to optimize a molecule's effectiveness (e.g., binding affinity to a target receptor) while minimizing toxicity and maximizing bioavailability. These objectives are often conflicting, as improving one may degrade another.

Evolutionary algorithms, such as genetic algorithms, are well-suited for multi-objective optimization because they can explore a diverse set of solutions simultaneously by maintaining a population of candidate solutions that evolve over successive generations. This population-based approach allows them to search through the solution space and discover Pareto-optimal solutions that balance these competing objectives.

In molecular design, genetic algorithms can generate a variety of molecular structures that offer different compromises between binding affinity, toxicity, and other pharmacokinetic properties. Researchers can then analyze these solutions to identify promising candidates that meet desired criteria across multiple dimensions, enabling informed decisions in drug discovery and other complex domains.

2.3.2.1 Pareto Optimal Solutions

Pareto optimal solutions represent the best trade-offs achievable between conflicting objectives, where improving one objective comes at the expense of another. Named after Vilfredo Pareto, who first introduced the concept, Pareto optimality ensures that no other feasible solution can simultaneously improve all objectives without worsening at least one.

Pareto dominance is a fundamental concept in determining Pareto optimal solutions. A solution A is said to dominate another solution B if A is at least as good as B in all objectives and strictly better in at least one objective. In other words, solution A dominates B if it offers superior or equal performance across all objectives without any trade-off worsening.

Identifying Pareto optimal solutions typically involves using specialized algorithms such as evolutionary approaches (e.g., genetic algorithms) that systematically explore the solution space to discover and evaluate these trade-offs. Once Pareto optimal solutions are identified, researchers can analyze and prioritize them based on specific needs and constraints, ensuring informed choices in complex scenarios.

2.3.2.2 Multi-Objective Evolutionary Algorithms

Multi-objective evolutionary algorithms (MOEAs) are designed to find Pareto optimal solutions in complex scenarios with conflicting objectives. Unlike traditional methods focusing on single objectives, they aim to generate diverse solutions that balance these competing goals.

Different MOEAs may differ in how they handle Pareto optimality, maintain diversity among solutions, achieve convergence and spread along the Pareto front, and manage computational efficiency. Their effectiveness can vary based on problem characteristics, thus, understanding these distinctions is essential for selecting the most suitable MOEA for a given multi-objective optimization problem.

In addition to achieving specific objectives, MOEAs are evaluated using diverse quality indicators that provide a detailed evaluation of algorithm performance. These indicators, such as hypervolume, spacing metrics, and dispersion metrics, among others, serve distinct purposes in assessing MOEAs performance. They measure aspects ranging from how well the Pareto front is covered to the distribution and overall quality of solutions obtained. Selecting appropriate indicators is crucial for a comprehensive and accurate evaluation of MOEAs, facilitating informed decisions in choosing the most suitable algorithm for a given multi-objective optimization problem.

NSGA-II The Non-dominated Sorting Genetic Algorithm II [130] is one of the most well-known MOEAs. It uses non-dominated sorting and crowding distance to maintain diversity among the solutions in the population. It emphasizes both convergence (getting closer to the true Pareto front) and diversity (maintaining

a spread of solutions across the Pareto front).

NSGA-III NSGA-III [131] extends the capabilities of NSGA-II to handle optimization problems that involve a large number of objectives. It is known that NSGA-II has good performance on problems of up to three objectives while NSGA-III excels in many objective problems [132]. It enhances diversity preservation mechanisms using a reference point-based approach and improves the scalability of the algorithm in high-dimensional objective spaces. NSGA-III adapts its selection mechanisms to efficiently guide the search towards diverse and well-distributed solutions on the Pareto front.

SPEA2 The Strength Pareto Evolutionary Algorithm 2 [133] focuses on maintaining a diverse set of non-dominated solutions. It uses an external archive to store non-dominated solutions. It also combines dominance and density for fitness assignment, making it suitable for a moderate number of objectives but potentially less effective for very high-dimensional objective spaces when compared to NSGA-III.

IBEA The Indicator-Based Evolutionary Algorithm [134] is another MOEA that uses quality indicators to guide the search process by maintaining diversity through the indicators. These indicators include the Hypervolume Indicator which measures the volume of the objective space dominated by a set of solutions and the Epsilon Indicator which measures the minimum factor by which a solution set needs to be translated in the objective space to dominate another set of solutions. This algorithm is suitable for a high number of objectives but can be computationally intensive depending on the chosen indicator.

2.4 Machine and Deep Learning in Chemoinformatics

2.4.1 General overview

Over human history, the search for compounds able to address relevant challenges and improve quality of life has been a constant endeavor. In many cases, this implies assessing their interactions with biological systems, seeking for possible drugs to fight distinct diseases or simply new nutrients and food ingredients. Still today, there is an ever-growing need for sustainable compounds with improved activity in both the pharmaceutical and food industries.

Due to its clear advantages like high accuracy and cost-effectiveness, AI is currently widely used in a variety of distinct areas, which include chemoinformatics. By taking advantage of the large available datasets on compound biological activity, novel approaches using ML and DL are continuously being developed. During the last decades, chemoinformatics has evolved at a fast pace shaping the way how researchers approach their studies. As datasets increase in size and algorithms augment their scope, complexity and improve their performance and accuracy, traditional methods become increasingly unmanageable giving place to new complex and innovative tools.

The whole process of discovering, testing and launching a new compound in the market is difficult, time-consuming, costly and risky. In this scenario, virtual screening through the use of computational

approaches toward the identification and design of new compounds with desired chemical and pharmacological properties has an important role and is providing great contributions. These approaches have the potential to identify higher quality candidate compounds to further be tested in the laboratory, reducing in this way the possible issues associated with this process.

Over the last decades, the number of publications involving the development of methods for the identification and design of new compounds has been growing and is witnessing an increasing interest. More recently, DL approaches are being widely used and are showing promising results in many areas and in special in *de novo* compound design. Several approaches, including traditional ML and more recent DL methods, have been used to predict many physical, chemical and biological properties among other information from structural data. Despite early attempts to use computer-assisted synthesis design, new approaches are still needed. The identification and design of new compounds with improved capabilities, with a lower impact on human health and on the environment, remain of pivotal importance. Therefore, chemicals should be introduced to the consumer only if they have been proven to be safe. For instance, models to unveil the relationships between chemical structure and toxicity are of great relevance.

Chemoinformatics can assist chemists both in academic research to increase our knowledge, as well as in industrial applications aiming at producing new compounds with desired properties. In industrial biotechnology, the synthesis of new compounds can be modeled by using a biological organism's metabolic network and performing alterations in that network to achieve the overproduction of a desired compound. So, in conjunction with chemoinformatics, the understanding of the events occurring in living organisms can be used to design novel strategies for many challenging tasks.

In this section, we will focus on machine and DL approaches for the prediction of biological compound activity. as well as in reactions' type and outcome prediction. We provide a brief introduction to known approaches and architectures relevant to these problems along with a selection of the most relevant studies published over the last years and a brief discussion of their critical differences.

2.4.2 Compounds

2.4.2.1 Compound Representation

Compounds are generally represented in manners that are not suited to ML or DL tasks. There has been a lot of research on how to efficiently represent compounds so that ML and DL methods can better learn to generalize the properties and substructures that are shared among sets of compounds. The set of features used for these tasks are primarily generated from the chemical structure of the compounds.

Generally, Simplified Molecular-Input Line-Entry System (SMILES) strings [76] are the molecular representation that is used by these methods to encode the chemical structure of the compounds into more complex and specialized molecular representations. In many cases, this extra layer of complexity is needed to overcome the lack of data or even to better guide the exploration of the complex and vast chemical space. Some of the most used and successful cases of compound featurization approaches include line notations, one-hot encoding, molecular fingerprints, graph convolutions, weave module, and NLP-inspired embeddings:

- Line Notations: line notations represent chemical structures as a linear string of characters. They are widely used in chemoinformatics due to the ability of computers to process linear strings of data. They express the 2D structure and properties of the molecules including atoms, bonds, aromaticity, chiral, and isotopic information. Examples of line notations include SMILES and IUPAC Chemical Identifier (InChI) [135].

- One-Hot Encoding: one-hot encoding is a process by which categorical variables are converted into binary vectors. Initially, the compounds represented as SMILES strings are tokenized character by character, representing each atom, bond, or special character. Then, an array of binary vectors indicating which character is present in each position of the SMILES string is computed.

- Molecular Fingerprints: molecular fingerprints are a way of encoding the structure of a compound. Most molecular fingerprints are series of bits that represent the presence or absence of a particular property or substructure in the compound. In some cases, molecular fingerprints can also be composed of non-binary values indicating other properties like molecular weight, melting point, and conductivity value. Molecular fingerprints differ from each other by the properties or substructures they encode. Extended-Connectivity Fingerprint (ECFP), Functional-Class Finger-print (FCFP), and the 166-bit Molecular Access System (MACCS) are the most common molecular fingerprints in chemoinformatics and biological activity prediction.

- Graph Convolutions: in graph convolutions, compounds are represented as graphs with atoms as nodes and bonds as edges. The connectivity between atoms and the local chemical environment of each atom is encoded by this representation. Different neighbor levels of the compound are used as input to a CNN to generate fixed-length numerical vectors representations of the compound. The convolutional layer applies a pooling-like operation using the softmax function and a weighted summed representation to output the final real vector representation of the molecule [136].

- Weave Module: weave modules are a type of graph CNNs similar to graph convolutions. Weaves are focused on encoding the connectivity between atoms in a molecule and each atom's local chemical environment but uses more detailed features focusing on atom pairs, allowing to consider information of non-adjacent atoms. To that end, weave modules combine and transform atom-level and pair-level features by applying convolution operators [137].

- NLP-inspired Embeddings: inspired by NLP techniques, several techniques were proposed to encode compounds as vector representations of molecular substructures. Two of the most known and used molecular embeddings methods are the Seq2Seq fingerprint [138] and the Mol2vec method [139]. The Seq2seq fingerprint translates molecules represented as SMILES strings into continuous embeddings using a model based on the sequence to sequence model [140]. Mol2vec is a method inspired by the Word2vec algorithm [141] that also learns continuous embeddings of compound substructures.

Among the described molecular featurization techniques, molecular fingerprints are the most widely used in the context of biological activity prediction. They are normally used in ML and DL approaches to identify compounds with desired properties. Theoretically, graph convolutions were proposed as an improvement over molecular fingerprints, however, when compared in several studies these do not consistently outperform fingerprint-based methods [103].

The use of NLP-based approaches has gained a lot of popularity over the last years due to recent outstanding results in other fields [111], [112]. For instance, Xu *et al.* [138] compared the use of Seq2Seq and several molecular fingerprints, and concluded that in many cases molecular embeddings could outperform regular molecular fingerprints. In *de novo* compound generation, the majority of published studies use a one-hot encoding approach to encode the SMILES represented compounds. An exception is presented in the work of Li *et al.* [142] that used graph representations for molecule generation. This study has shown that graph-based methods can present good results, especially regarding the rate of validly generated molecules.

2.4.2.2 Machine and Deep Learning for Biological Activity Prediction

The application of ML and DL in biological activity prediction has demonstrated its potential for the design and identification of molecules possessing relevant properties. Every day, expert chemists face the challenge of working on the design, synthesis and test of new molecules. The use of ML-based approaches can radically reduce the costs and resources needed for these tasks. The main objective is to select a subset of compounds from all the compounds under consideration that have a higher probability of being active when compared to random samples.

One critical aspect for the success of ML and in particular DL is the access to large datasets. Multiple public databases are available and store large amounts of chemical data suited for specific tasks. Examples of resources storing this information include PubChem [143], MoleculeNet [144], DrugBank [74], ChEBI [145], ChemSpider [146] and multiple others are valuable assets to the future of this field.

In the last years, researchers have been publishing several studies regarding the use of different ML and DL-based techniques. RFs, SVMs and DNNs are widely used to identify compounds with desired biological properties. Table 3 lists selected studies, showing examples of ML and DL in tasks related to activity prediction. From the table, we can realize that DNNs seem to provide better results than classic ML methods [150], [151], where RF-based models are the most prevalent and the ones performing best, even better than DNNs in some cases [159].

2.4.3 Reactions

Identifying a series of reaction steps to transform a set of available reactants into a target compound or to decompose a target compound to a set of available reactants is not an easy task. These processes have been performed by expert chemists over the last decades. Assisting chemists in these tasks using computational approaches was introduced over 40 years ago and still remains a relevant effort.

Table 3: A selection of diverse recent studies that use ML and DL for biological activity prediction.

Study	**ML Methods**	**Study Description**
Stokes *et al.* [147]	directed-message passing DNN	DLs approach to predict antibacterial molecules. Halicin and other antibacterial compounds were predicted to be effective against various pathogens.
Ashraf *et al.* [148]	26 shallow ML models and one FCNN	DL approach for predicting the potency of drug compounds against SARS-CoV-2 3CLpro. The model was validated on a large dataset.
Sakai *et al.* [14]	GCNN	Training DL models using 2D structural information of compounds to predict activity against diverse targets. A new serotonin transporter inhibitor was successfully identified.
Galushka *et al.* [149]	VAE coupled with a FCNN	DL approach using a VAE to generate chemical compound fingerprints. The study showed good results for lipophilicity and binding prediction in selected assays from the ChEMBL database.
Mayr *et al.* [150]	SVMs, KNN, RFs, naive bayes (NB), DNNs	Comparing distinct DL methods with different ML and target prediction methods on a drug discovery dataset. DL methods, in particular feed-forward NNs, outperformed the other methods.
Merget *et al.* [151]	RF, KNN, NB, DNN	Ligand-based activity prediction for over 280 kinases using multiple ML methods. RFs were chosen over other ML methods.
Lenselink *et al.* [152]	NB, SVMs, LgR, RFs, DNN	ML and DL approaches compared using a dataset from ChEMBL [153]. DNN outperformed traditional ML methods especially when using multitask DNNs.
Korotcov *et al.* [154]	NB, LgR, DTs, RFs, SVMs, DNNs.	Comparison between DNNs and other ML algorithms for diverse endpoints (bioactivity, solubility, and ADME properties) relevant to pharmaceutical research. In general, DNNs outperform other methods.
Koutsoukas *et al.* [155]	DNNs, SVMs, RFs, NB, KNN	Shows that DNNs are capable of outperforming shallow methods across diverse activity classes when optimized. The impact of different hyper-parameter configurations was also explored.
Mayr *et al.* [156]	DNNs, SVMs and RFs	Results of the DeepTox pipeline, the winner of the Tox21 Data Challenge. DNNs show good predictivity on 12 different toxic endpoints outperforming traditional ML methods.
Kearnes *et al.* [157]	Single and multitask DNNs	Multitask learning provided benefits over single task models. Smaller datasets tend to benefit more from multitask learning than larger datasets.
Ramsundar *et al.* [158]	Multitask DNNs	DeepChem implementation of multitask DNNs. The models were tested over multiple pharmaceutical datasets. Multitask DNNs can frequently provide improvements over RFs.
Ma *et al.* [159]	DNNs, RFs	Comparing performance of DNNs and RFs for QSAR/QSPRs using different datasets from Merck's drug discovery effort. DNNs achieved better performance than RFs in most of the datasets used.
Xu *et al.* [160]	DNNs	Use of the UGRNN architecture [161] to predict drug-induced liver injury. Showed improvements over previous drug-induced liver injury prediction models.

The majority of computer-aided synthesis design methods involve the use of reaction templates containing information on sub-molecular patterns that encode changes in atom connectivity of compounds. The successive application of these reaction templates over a target compound and resulting products produces a candidate synthesis tree that needs to be evaluated. However, the translation of a retrosynthetic route to a forward synthesis is not always a straightforward task and commonly leads to functional group conflicts that might lead to a lack of specificity or selectivity. It is of pivotal importance that, during pathway prediction, the most promising reaction templates, given a set of reactants, are applied. Once a pathway is proposed, it is also important to evaluate each step in the forward direction to identify potential problems. However, in the end, the true measure of success is whether the predicted reaction or pathway matches what is observed experimentally.

In computational reaction prediction tasks, there are essentially two main approaches, rule-based expert systems and ML-based systems, that can generally be combined. The lack of scalability of rule-based approaches due to limited knowledge of human expert-defined rules can be overcome by combining fast and scalable ML approaches. Rule-based expert approaches can be used to provide training examples for ML-based approaches that are data-dependent and, in this way, address specific cases or express strong priors about conditions that can be redundant or problematic.

Databases of hypothetical reactions can be generated using many different reaction operators that encode chemical transformation mechanisms. Bond-electron matrices encode molecules in a matrix representation, where elements in the off-diagonal of the matrix represent non-bonded valence electrons and the diagonal elements represent the connectivity between atoms and bond orders. A graph-based matrix is also used to represent enzyme reaction rules and biochemical compounds. This formulation is used in the BNICE.ch tool [36]. Molecular signatures can also be used to generate reaction rules based on the substructure of adjacent atoms as in XTMS [162]. Biochemical structure transformation patterns, called RDM patterns, can also be used for reaction rule generation as shown in the studies by Moriya *et al.* [51] and Law *et al.* [163]. In UM-PPS [164] and GEM-Path [165] string-based methods, SMIRKS and SMARTS, to encode chemical reactions were used.

2.4.3.1 Reaction Representations

Reaction SMILES Just as compounds can be individually represented as SMILES strings, reaction SMILES [166] can also be used to represent the molecules in a chemical reaction. This way of modeling reactions is defined by specifying the reactants, agents, and product molecules separated by the > symbol. One reaction SMILES must have exactly two > symbols, reactant > agent > product or reactant >> product. Multiple molecules in the same role are separated by a dot. Examples of valid reaction SMILES are $C=CCBr >> C=CCI$ that have no agent molecules specified and $C=CCBr.[Na+].[I-] > CC(=O)C > C=CCI.[Na+].[Br-]$ that includes an agent.

Reaction SMARTS SMiles ARbitrary Target Specification (SMARTS) [66] is a straightforward extension of the SMILES notation. It allows to describe molecular patterns to match both molecules and reactions.

Beyond using atomic and bond symbols to specify a graph, SMARTS are extended to include logical operators and special atomic and bond symbols. For instance, the SMARTS atomic symbol $[C, N]$ defines an atom that can be either carbon or nitrogen. Another example is the use of the wildcard symbol $*$ which can be used to match any atom and the symbol to match any bond. For example, the SMARTSs $C(=O)O.OCC >> C(=O)OCC.O$ matches esterifications, $(C(=O)O).(OCC) >> C(=O)OCC.O$ matches intermolecular esterifications and $(C(=O)O.OCC) >> C(=O)OCC.O$ matches intramolecular esterifications (lactonizations).

Reaction SMIRKS The SMIRKS line notation is defined for generic reactions to represent a group of reactions that undergo the same set of atom and bond changes. SMIRKS is a hybrid notation of both SMILES and SMARTS and it is generally used to create new reactions, manipulate molecules and to generate new molecules. Virtually, any chemical transformation can be represented using SMIRKS allowing an easier computational manipulation and processing of the known and potential chemical space. For example, the SMIRKS $[*:1][N:2](=[O:3]) = [O:4] >> [*:1][N+:2](=[O:3])[O-:4]$ encodes the transformation to interconvert nitro-group representations.

2.4.3.2 Reaction Type Prediction and Reaction Outcome Prediction

Over the last years, many studies involving reaction type and reaction outcome prediction have been published. These two tasks are key aspects in computer-aided synthesis design enabling the prediction of more accurate pathway suggestions that best match what actually can be experimentally feasible.

In an earlier study, Kayala and collaborators [167] proposed a ML approach called ReactionPredictor for reaction prediction that models elementary mechanistic reactions as interactions between molecular orbitals. Molecules were represented as graphs and approximate molecular orbital was assigned to each molecule so that interaction between donor and acceptor orbital can be considered. The ML model was trained using a set of reactions known to occur derived from an existing rule-based system. Reaction types were then predicted based on the individual molecular orbital of the respective molecules. Promising results were obtained, however, the need for manually encoded reaction rules to generate data to train the model may limit it in terms of scalability.

Wei *et al.* [168] suggested a new reaction fingerprint method to predict reaction types using NNs. They combined SMARTS transformations with reactant fingerprints to predict the most likely products and applicable template, out of 16 types of reactions, given a set of reactants and reagents. Despite the good results, the method was tested on problems using artificially generated examples with limited chemical functionality instead of experimental data. Another limiting factor was the fact that their study was limited to only 16 types of reactions covering only a portion of the possible reactions.

In two studies by Segler and Waller [41], [169], two approaches based on DNNs to substitute rule-based expert systems for the prediction of reaction types and outcomes, and invent new reactions are investigated. In the first study, a model to mimic chemical transformations and formalize reaction prediction using a knowledge graph containing all the chemical reactions ever published in the scientific

literature was proposed. Here, possible reaction products are generated by two given reactants and by looking at the known reactions in which those reactants participate. This data-driven approach is capable of predicting reaction types and effectively rediscovering novel transformations. In the second study, DNNs were used to learn to resolve reactivity conflicts and to prioritize the most suitable transformation rules given reactant fingerprints.

In a recent study, Gupta [170] used RNNs to automate the prediction of the reaction type given two reactants and to generate the reaction products given the reactants and reaction type. For reaction type classification, a RNN classifier consisting of two LSTM layers was trained achieving a test accuracy of around 95%. For reaction outcome prediction, a sequence to sequence architecture with attention was implemented achieving an average 83% validity of the products generated and an average Tanimoto similarity of around 88% between the generated products and the actual ones.

Coley and collaborators [171] proposed a framework to predict reaction outcomes by combining traditional reaction templates and NNs. The model was trained to select the major product of a reaction by ranking a self-generated list of potential product candidates generated by applying reaction templates to a pool of reactants using NNs. The major product was identified 72% of the time and ranked in the top 5 91% of the times.

In the last years, there has been a shift towards data-driven approaches to predict reaction types and reaction outcomes. However, the lack of "negative" cases, i.e. reactions that are not chemically feasible, and the bias towards reactions with high yields, with reactions with negligible or zero yield not being reported, still poses a considerable barrier. The release of this type of data from both academic and proprietary studies can make the field evolve faster improving supervised learning approaches for better synthetic-related predictions.

3

DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry[1]

3.1 Introduction

In recent years, computational chemistry has undergone a remarkable transformation, driven by advances in ML and DL techniques and by the immense growth of available chemical data [173]–[176]. These have facilitated exploring and understanding the complex relationships between chemical structures and their properties [177]. As a result, using these computational methods within the chemical discovery pipeline has emerged as a powerful and promising way to expedite the discovery of new chemicals with improved properties [178].

QSAR/QSPR models have always been a focal topic in computational chemistry [179]–[182]. Initially, simple statistical models were applied to small datasets of molecules characterized by a restricted array of descriptors [183], offering a straightforward way to correlate molecular structure with biological activity [184]. Nowadays, QSAR/QSPR modelling consists of large sets of molecules with an extensive set of molecular descriptors.

As the amount of available data and the complexity of QSAR/QSPR models continue to grow, the demand for advanced techniques has led to the emergence of DL as a viable alternative to traditional ML models [185]. With the current plethora of ML/DL models and diverse chemical descriptors available, researchers face the challenge of selecting the most suitable alternative for their data regarding data representation and processing and ML/DL models [184]. Choosing the optimal combination of features/descriptors and models requires exhaustive testing to comprehensively understand their performance on a given dataset [186], [187]. Recognizing this challenge, the importance of AutoML frameworks becomes unequivocal. In the fields of QSAR/QSPR modelling, where pre-processing steps, models, and their combination can significantly impact results, the need for a comprehensive and easily customizable AutoML

[1]This chapter is an unmodified version of an article [172] submitted in a peer review scientific journal, written in co-authorship with João Capela and Miguel Rocha. This thesis author and João Capela contributed equally to this work.

framework is even more evident.

However, despite this need, only a few resources offer an easily customizable framework capable of providing a wide range of possibilities. ZairaChem [188] provides the first AutoML framework described in the literature that can automatically optimize all the pre-processing steps and model hyperparameters for specific QSAR/QSPR problems. However, users cannot easily customize this system to suit their needs. For example, ZairaChem does not provide ways of creating custom objective functions, i.e. how to generate the final metric to give feedback to the AutoML system, e.g. whether in a cross- or hold-out validation or other scenarios. Another example is the impossibility of defining the type of models and descriptors to test. Furthermore, ZairaChem is restricted to binary classification tasks, which can be very limiting.

Herein, we present *DeepMol*, an AutoML python-based open-source framework for the prediction of activities and properties of chemical molecules. Although fully automated, *DeepMol* is built modularly, allowing for the customization of every step of the ML pipeline, starting from the data loading and processing to the model prediction and explainability. Nonetheless, what sets DeepMol apart is its robust AutoML functionality, allowing the automatic optimization of different scenarios involving pre-processing methods, data engineering techniques, and ML/DL models and respective hyperparameters. Even for users with minimal coding experience, with just a few lines of code, *DeepMol* allows testing thousands of configurations to determine the most effective ones for their specific datasets. Notably, *DeepMol* provides documented ways of defining objective functions and customizing AutoML experiments. Moreover, it is suitable for several ML tasks such as binary, multi-class, multi-label/multi-task classification and regression. The framework leverages other well-known and established packages, like *RDKit* [189] for molecular operations, *Scikit-Learn* [190], *Tensorflow* [191] and *DeepChem* [192] for model building and *Optuna* [193] for end-to-end ML pipeline optimization.

A rigorous experimental framework was established to ensure an impartial assessment of the AutoML engine's capabilities, enabling evaluation of its performance across 22 benchmark datasets for predicting adsorption, distribution, metabolism, elimination, and toxicity (ADMET) derived from the Therapeutics Data Commons (TDC) repository [194].

3.2 Implementation

DeepMol was primarily developed as a comprehensive end-to-end AutoML framework for computational chemistry. Yet, its modular design permits the independent utilization of its components. The framework provides a wide range of techniques for each step of a general ML pipeline, from data pre-processing and feature extraction to model training, evaluation, and explainability. Furthermore, the package has been extended with additional functionalities of significant relevance, including unsupervised learning models, data-splitting strategies specifically designed for molecular datasets, and approaches to address data imbalance.

The *DeepMol* AutoML engine, as illustrated in Figure 14, comprehensively explores a vast array of potential methodological combinations. In this context, state-of-the-art computational chemistry methods

are employed automatically and sequentially in a pipeline, following a specific configuration, ensuring it is tailored to address specific chemical tasks. The configuration space encompasses several data standardization methods (Figure 14a: three methods), feature extraction methods (Figure 14b: four options for sets with 34 methods in total, all with their respective parameters), scaling and selection methods (Figure 14c: 14 methods and respective parameters), various ML models and ensembles (Figure 14d: 140 models), and their respective hyperparameters.

Upon starting the AutoML experiment, the engine first processes the training data following a predefined sequence of steps, known as the pipeline configuration, and then uses this data to train an ML/DL model. Post-training, a separate set of data, the validation set, is processed and assessed to evaluate the model's performance. The outcomes of this evaluation are then fed back into the optimization framework, guiding it in choosing new parameters and methods for improvement. This cycle of training and evaluating is repeated a user-specified number of times, referred to as *trials*. After completing all the trials, the system analyzes the results to identify and select the most effective pipeline. The optimization framework is efficiently powered by the *Optuna* search engine, which provides nine different optimization algorithms compatible with *DeepMol*. Once the optimal pipeline has been identified, it can be applied to analyze new data, enabling predictions on untested data. Additionally, this pipeline facilitates the virtual screening of extensive databases, efficiently identifying potential molecules of interest.

This section provides an overview of the key components and capabilities of *DeepMol*, highlighting its significant contributions and advantages in chemical compound analysis.

3.2.1 Data Loading

The representation of molecular structures can be achieved using various formats. One such format is the SMILES, which is readable and concise, as short as single-line ASCII strings [195]. The other format is a connection table, which can be stored as a Structure Data File (SDF) and used to represent three-dimensional structures of molecules [196], [197]. These are the *de facto* standard starting points for representing molecules in ML tasks [195], [198] and *DeepMol* provides loaders for both formats. Once loaded, the information is converted into a structured data format that allows easy access to relevant details, such as the SMILES representation of all molecules, their identifiers, and known outputs associated with the input data (labels). Alternatively, if one prefers to load molecules from an already-loaded data frame, a *SmilesDataset* class can be created. This dataset accepts SMILES or *RDKit Mol* objects as input, along with their corresponding identifiers and labels.

3.2.2 Molecular Standardization

The availability of vast amounts of chemical data found in molecular databases containing hundreds of millions of compounds can be a double-edged sword. While it pushes the field forward, it makes the human curation process infeasible, resulting in the frequent occurrence of incorrect and inconsistent molecular structures [199]. Even minor structural errors and inconsistencies within a dataset can result

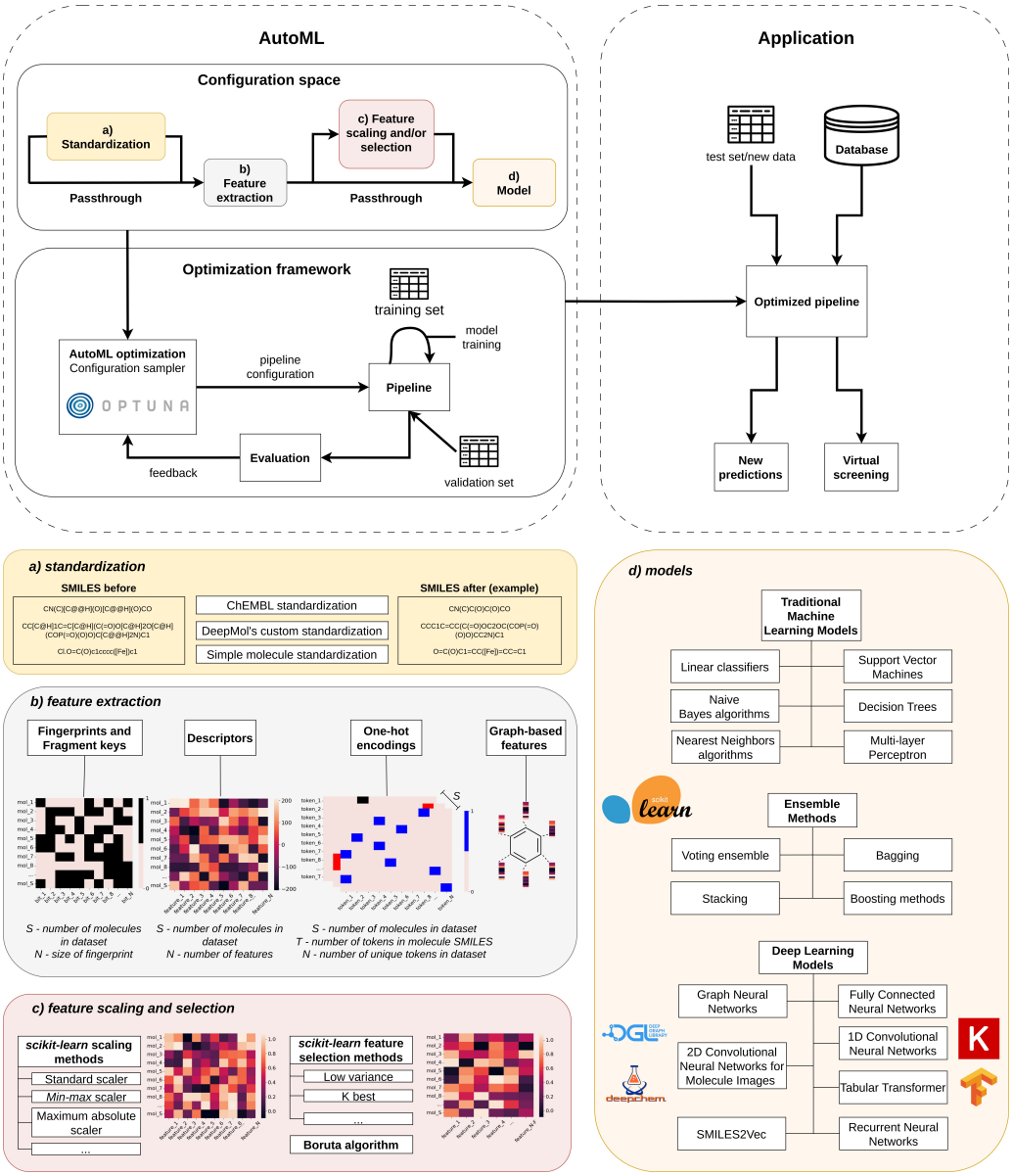


Figure 14: The DeepMol AutoML framework offers a comprehensive approach that includes an optimization framework that samples pipeline configurations from the configuration space. This pipeline begins with an optional **(a)** standardization step (configuration space: 3 different methods), followed by **(b)** the extraction of features (configuration space: four options for sets with 34 methods in total, all with their respective parameters). These features can then be optionally **(c)** scaled and selected (configuration space: 14 methods and respective parameters), preparing them for **(d)** the training phase of the model (configuration space: 140 models, architectures and hyperparameters). Post-training, the model's performance is evaluated based on a predetermined metric. This evaluation feedbacks to the *Optuna* optimization engine. *Optuna* uses these results to inform its selection of new parameters and methods for enhancing the pipeline's efficiency using state-of-the-art optimization algorithms. This process is repeated *n* times (chosen by the user), also called trials, and, in the end, the best pipeline is selected. Each individual box represented in the configuration space is expanded and schematized in the boxes below. Finally, the result of the AutoML engine is an optimized pipeline that can be used to transform new data and perform new predictions.

in significant losses in the predictive ability of QSAR/QSPR models [200]. Consequently, the development of strategies that can tackle this problem has received increased attention in recent years [201]–[203].

With *DeepMol*, it is possible to standardize molecules using three different options:

- *BasicStandardizer* performs basic sanitization using *RDKit* to ensure that a molecular structure is represented consistently and validly according to a set of predefined rules. These include kekulization, check of valencies, set aromaticity, conjugation, and hybridization.
- *CustomStandardizer* performs a set of steps defined by the user, using *RDKit*. Some steps include molecular sanitization (same as in the BasicStandardizer), removal of isotope and/or stereochemistry information, neutralizing charges, fragment removal, and kekulization.
- *ChEMBLStandardizer* performs a set of rules as used by the ChEMBL database [201]. It consists of three components: a Checker that tests the validity of the chemical structures, a Standardizer that formats compounds based on the U.S. Food and Drug Administration and International Union of Pure and Applied Chemistry (IUPAC) guidelines; and a *GetParent* component that removes any salts and solvents from the compound.

3.2.3 Feature extraction

In ML, feature extraction from molecules is a common task. Different types of molecular features are categorized from 0-dimensional (D) to 4D. 0D features provide overall information about the molecule (e.g. atom and bond counts), while 1D features describe substructures within the molecule (e.g. fingerprints) [204], [205]. 2D features capture molecular topology, and 3D features capture geometric information of the molecule's three-dimensional structure [206]. 4D features are descriptors with an additional dimension representing interactions with receptors' active sites or multiple conformational states.

DeepMol provides a set of 0D, a few 1D and 2D descriptors in only one class. The descriptors contained in this class are enumerated and described in Supplementary Table 20. *DeepMol* offers a wide range of 1D features provided by *RDKit*, including circular, atom pair, layered and *RDKit* fingerprints. *DeepMol* also includes MACCS keys implemented by *RDKit*, which encode the presence or absence of specific molecular fragments or substructures in a molecule. These are further enumerated and described in Supplementary Table 21.

3D molecular conformations must be generated or loaded before generating 3D descriptors. For this matter, *DeepMol* provides conformer generation methods like Experimental-Torsion basic Knowledge Distance Geometry (ETKDG) and methods like Merck Molecular Force Field (MMFF) and Universal Force Field (UFF) for optimizing the generated conformers. Once generated, *DeepMol* offers methods to extract features from these conformers. These methods include AutoCorr3D, Radial Distribution Function (RDF), the plane of best fit, MORSE, WHIM descriptors, Radius of Gyration, shape descriptors, and principal moments of inertia. More details on these methods can be found in Supplementary Table 22.

DeepMol also provides one-hot encoding schemes. Herein, *DeepMol* provides an implementation of an atom-level tokenizer and *k*-mer tokenizer from Li et al. [207]. The former method permits treating each

character in the SMILES string linked to an atom (e.g., [C@] or [N@+]) as an identical token, whereas the latter provides groups of *k* characters of the SMILES string. Both create a vocabulary dynamically given a dataset. If no tokenizer is passed as input, the one-hot encoding uses the atom-level tokenizer.

DeepMol extends *DeepChem* to provide inputs for GNNs, including molecular graphs (2D descriptors), representing the molecule by a list of neighbors and a set of initial feature vectors for each atom. The feature vectors represent the atom's local chemical environment, including atom types, charge, and hybridization, among others [103]. Other implementations for using GNNs were also included, such as methods for implementing Duvenaud graph convolutions [136], Weave convolutions [103], and MolGAN [208]. Coulomb matrices and their eigenvalues [209] are also provided as methods of extracting 3D features, providing information on electrostatic interaction between atoms.

Finally, *DeepMol* also extends *DeepChem* to convert molecules into images and encodings to be passed to an embedding layer. All DeepChem-derived feature extractors are further detailed in Supplementary Table 23.

3.2.4 Data Scaling

Many feature extraction methods encode molecules as bit vectors, while others use vectors of real numbers. In the latter, a challenge arises when the features have different numerical ranges. This discrepancy can cause certain features to dominate the learning process due to their larger scales. Data scaling becomes crucial to address this issue. Bringing all features into a similar numerical range can enhance the performance and convergence of many ML algorithms. *DeepMol* offers a wide range of data scalers, as provided in the *Scikit-Learn* package (Supplementary Table 24).

3.2.5 Feature Selection

Feature selection is a common task in ML problems. By choosing the most relevant features, the performance of an ML model can be significantly enhanced. Eliminating irrelevant or redundant features helps prevent overfitting and improve the generalization power of the model, while reducing the computational burden. *DeepMol* supports many types of feature selection as provided by *Scikit-Learn*, including removing features with low variability, supervised filters based on statistical tests, as well as wrappers such as Recursive Feature Elimination, or selecting features based on importance weights (Supplementary Table 25). In addition, *DeepMol* also provides feature selection based on the Boruta algorithm [210].

3.2.6 Machine and Deep Learning models

DeepMol offers compatibility with three popular ML/DL frameworks: *Tensorflow*, *Scikit-Learn*, and *DeepChem*. This allows seamless integration of any model built upon these frameworks into *DeepMol*. A wide range of pre-built models from these frameworks is readily available, supporting single and multi-task problems, binary and multi-class classification and regression, making it convenient for users to utilize these models for specific tasks. Available models are described in Supplementary Table 28.

Through *Scikit-Learn*, *DeepMol* offers an extensive selection of popular ML models, including, among others, linear and logistic regression, support vector machines, decision trees, random forests, and gradient boosting. In conjunction with *DeepChem*, it also provides several DL models specifically tailored for chemical data, including GNNs, RNNs, and transformer models. Moreover, the flexibility offered by *Tensorflow* enables integration of any DL architecture into *DeepMol*'s pipeline. By consolidating these capabilities, *DeepMol* serves as a comprehensive and versatile framework, facilitating implementation and comparison of various ML and DL models in one unified platform [211], [212].

3.2.7 Hyperparameter Optimization

In *DeepMol*, users can perform hyperparameter optimization for all the above-mentioned models. This can be done either by using a validation dataset or through cross-validation. Fine-tuning the model's hyperparameters is crucial to obtaining optimal results, mitigating overfitting, and improving overall efficiency. Currently, *DeepMol* utilizes *randomized* or *grid search* methods to explore and find the best hyperparameters.

3.2.8 Pipelines

In *DeepMol*, we have introduced a pipeline class that simplifies the creation of an ML pipeline. This class empowers users to construct an ML pipeline tailored to their specific needs. It offers the flexibility to incorporate a diverse range of steps, enabling the inclusion of any combination of the following methods:

- Data standardization;
- Feature extraction or transformation (data scaling and selection);
- Model training and hyperparameter tuning.

Pipelines in *DeepMol* work the same as *Scikit-Learn* pipelines: use the *fit_transform* method to fit and transform the train data and the *transform* method for solely transforming the validation and test sets. The pipeline class also allows users to save and load the pipeline again. Moreover, it allows to evaluate the pipeline and make predictions on new data using the *evaluate* and *predict* methods, respectively. The model herein declared can also be optimized.

Users can design and customize their ML pipeline, adapting it to their unique requirements and desired outcomes. This approach significantly reduces the complexity of constructing an ML pipeline, promoting efficient experimentation and development.

3.2.9 AutoML: pipeline optimization

Pipelines provide a convenient and efficient way to transform raw data into trained and validated models, but determining the appropriate sequence of steps often requires expertise. Moreover, if one wishes to

explore multiple combinations of pre-processing techniques (data *Transformers*) and models (*Predictors*), manually constructing a *Pipeline* for each combination can be repetitive and time-consuming.

Automating the process of building and optimizing each step and respective parameters of a *Pipeline* can greatly assist researchers with limited ML expertise in building and deploying effective QSAR/QSPR systems, accelerating the development process. In *DeepMol*, we offer the *PipelineOptimization* module (AutoML), which enables the search for the best configurations from a large number of possibilities. This module leverages the powerful *Optuna* library [193] and its state-of-the-art optimization algorithms.

As with any optimization problem, pipeline optimization needs an objective function. By default, *DeepMol* provides the objective of maximizing or minimizing a metric (e.g. accuracy) for a given validation set. However, other custom objective functions can be added.

Furthermore, a function can be created with the configuration space that will be optimized by *Optuna*. This function will receive a *Trial* object from *Optuna* and will return a set of steps (i.e. *Transformers* and *Predictors*) from the configuration space to define the trial pipeline. Alternatively, *DeepMol* provides a set of predefined configuration spaces (pre-sets) for convenience. The available pre-sets include:

- 'sklearn': optimizes between all available Standardizers, Featurizers, Scalers, Feature Selectors in *DeepMol*, and all available models in *Scikit-Learn*.
- 'deepchem': optimizes between all available Standardizers in *DeepMol* and all available models in *DeepChem* (depending on the model, different Featurizers, Scalers and Feature Selectors may also be optimized).
- 'keras': optimizes between Standardizers, Featurizers, Scalers, Feature Selectors in *DeepMol*, and some core Keras models (FCNNs, 1D CNNs, RNNs, Bidirectional RNNs, and transformer models).
- 'all': optimizes between all the above.

Finally, a *Voting Pipeline* composed of the best five pipelines is created after running the AutoML, where the voting process for classification occurs through soft and hard voting mechanisms. In soft voting, the predicted probabilities from each classifier are averaged to determine the final prediction. On the other hand, hard voting involves a simple majority vote, where the class predicted by the majority of the individual classifiers is selected. Additionally, for regression tasks, the final prediction is obtained by averaging the predictions from each regressor, providing a robust ensemble prediction from the diverse models in the voting pipeline.

3.2.10 Unsupervised Learning

Unsupervised learning has become increasingly important in exploring the vast amounts of available and sometimes unlabeled chemical data. Through techniques like clustering and dimensionality reduction, unsupervised learning allows the identification of patterns and relationships in large datasets without prior knowledge.

DeepMol provides a simple interface to various techniques, such as PCA, t-SNE, K-means clustering, and Uniform Manifold Approximation and Projection (UMAP). These techniques can help scientists to efficiently analyze and visualize complex molecular data, providing valuable insights to guide subsequent analyses.

3.2.11 Data splitting

Splitting the data into training, validation and test data sets or in a cross-validation setting is a crucial step in ML pipelines. It is common for practitioners to require data separation based on molecular structures or similarities. Practitioners may opt for a homogeneous data split to maintain a balanced distribution of molecular structures or similarities across the sets. This approach aims to capture the diversity of the dataset, while providing reliable performance estimates for the ML model. Alternatively, practitioners may emphasize the separation of highly similar molecules from less similar ones (heterogeneous splits). By isolating similar molecules, practitioners can better assess the model's ability to capture nuanced differences and generalize to unseen data.

DeepMol provides molecular splitters that split the inputted dataset based on similarity, scaffolds or Butina clusters [213], while maintaining the stratification of classes. Figure 15 illustrates how the similarity and scaffold splits are distributed in the overall chemical space and how the homogeneity parameters control the split.

Moreover, *DeepMol* provides random and stratified splitters for single and multitask classification problems. More information about each splitter is provided in Supplementary Table 26.

3.2.12 Imbalanced data

Imbalanced data is a common problem in many real-world applications that heavily affect the quality and reliability of ML and DL approaches [215]. Likewise, imbalanced data is an issue in QSAR/QSPR studies, where the difference between the number of active and inactive molecules can be extreme [216]. This difference generally leads to biased and sub-optimal models, as traditional ML algorithms may not be able to learn the minority class effectively.

Data balancing techniques have shown potential in mitigating the effect of imbalanced data on the model performance [216]. *DeepMol* provides various methods for handling imbalanced data, including over-sampling, under-sampling, and combination methods, as in the *imbalanced-learn* package [217]. For over-sampling, it provides a random over-sampler and Synthetic Minority Over-sampling Technique (SMOTE) [218], while Random under-sampler and *ClusterCentroids* can be used for under-sampling. Finally, for simultaneous over and under-sampling, it provides SMOTE with Edited Nearest Neighbours (SMOTEENN) [219] and SMOTE using Tomek links (SMOTETomek) [220]. More information about each imbalanced learning technique is provided in Supplementary Table 27.

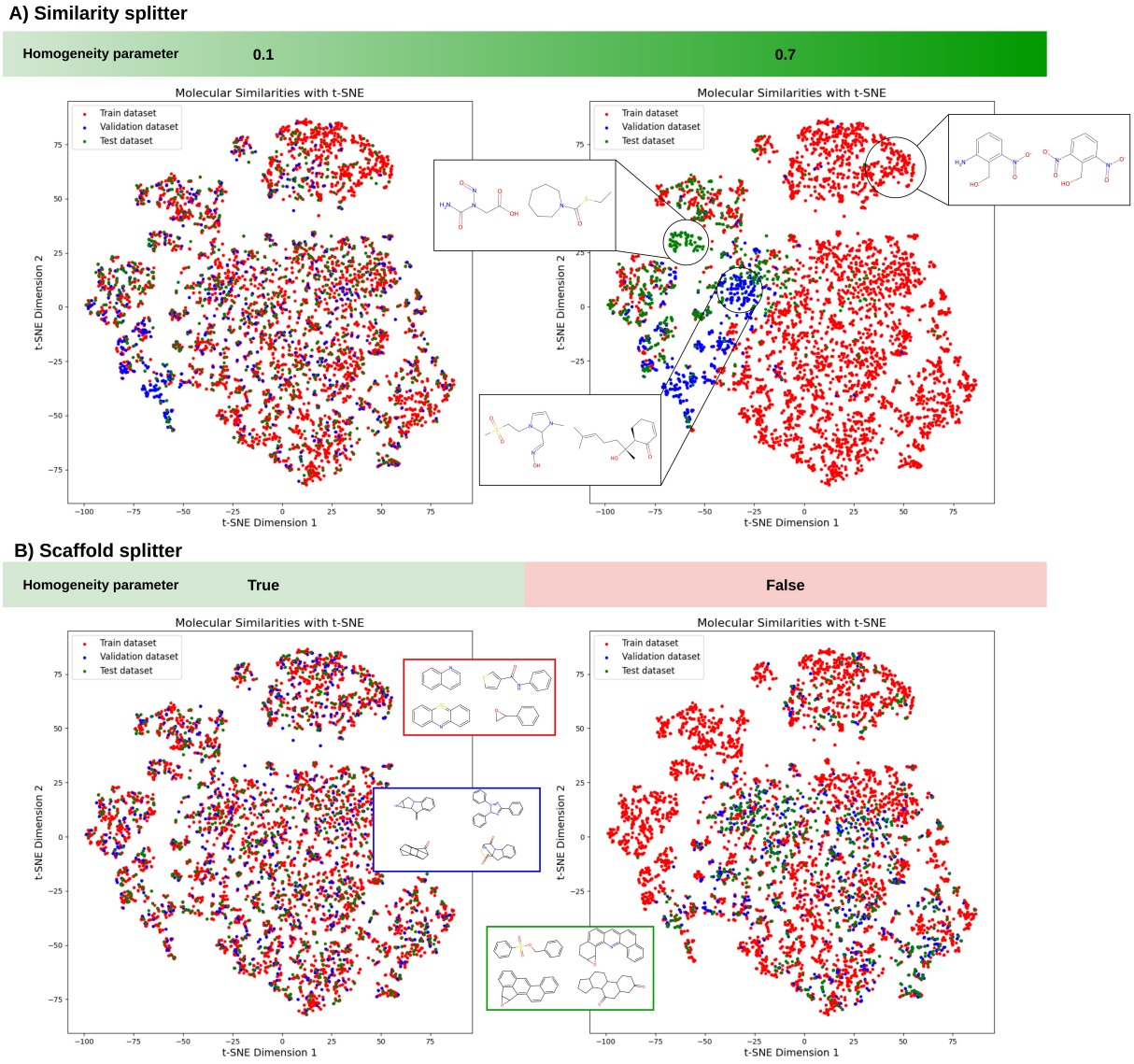


Figure 15: Depiction of dataset splits in the chemical space for the mutagenicity dataset from [214] provided by TDC. A similarity matrix was generated using the Tanimoto similarity metric over Morgan fingerprints. A t-SNE was applied to the matrix to reduce dimensionality and visualization. **A)** represents the similarity splitter and examples of molecules belonging to small clusters. One can regulate the homogeneity of the splits by tweaking the homogeneity parameter, which considers all the compounds with a similarity lower than the homogeneous parameter to be in the same set. The higher the parameter is, the more heterogeneous the split will be. **B)** represents the scaffold splitter and scaffolds belonging to the splits on the plot on the right. This splitter can separate the data by putting compounds with the same scaffolds in different data splits (homogeneous split) or not (heterogeneous split).

3.2.13 Feature importance and model interpretability

Model interpretability, an essential aspect in the field of ML, refers to the capacity to comprehend and elucidate the internal mechanisms of an ML model. This ability provides insights into the rationale behind the model's predictions or decisions, fostering a sense of trust in the model's outputs.

SHapley Additive exPlanations (SHAPE) [221] stands out as a widely utilized technique for achieving model explainability. It leverages the Shapley value, a fundamental concept derived from cooperative game theory, to assign significance to each input feature within a model. By comparing the predictions of various feature subsets, both with and without a specific feature, the Shapley value quantifies the marginal contribution of that feature to a prediction.

Visualizing SHAPE values can take various forms, including bar charts, scatter plots, and summary plots. These visual representations facilitate the interpretation of how individual features influence the model's predictions, aiding in identifying the most crucial features for a given prediction scenario.

Along with these methods (Supplementary Table 29), *DeepMol* allows one to visualize the structures associated with bits in Morgan and *RDKit* fingerprints and MACCS keys. Therefore, after computing the SHAPE values, one can directly cross information from the most relevant features to the molecule and draw biological and chemical conclusions, as depicted in Figure 16.

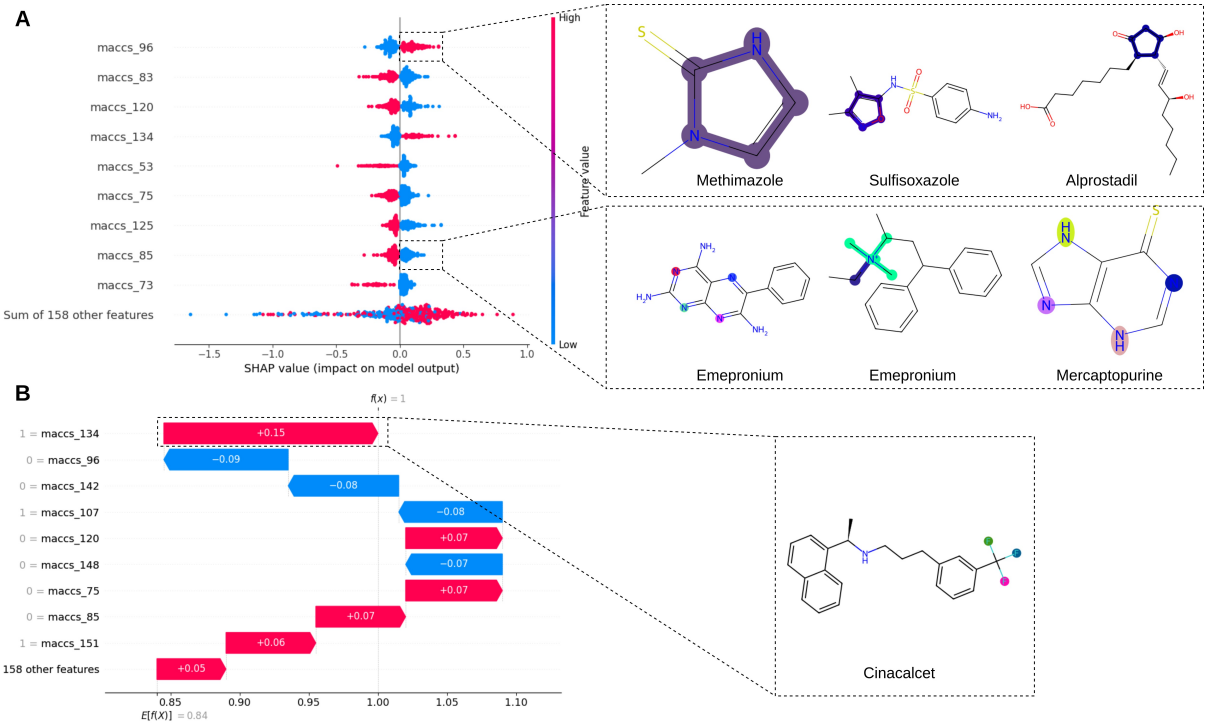


Figure 16: Depiction SHAPE values for a ridge classifier and the most relevant MACCS keys in the molecules. The dataset used was the bioavailability dataset from [222] provided by TDC. **A)** provides the overall feature importance of all labels for all data points and depicts the two of the most relevant MACCS keys for the prediction. The highlighted atoms and bonds are the ones that correspond to the MACCS key. **B)** provides a depiction of the most important features of the drug cinacalcet, showcasing how it is also possible to assess features' importance for individual cases.

3.3 Results

3.3.1 Results on the ADMET benchmarks

In this section, we discuss the performance of DeepMol AutoML on the 22 TDC ADMET benchmark group datasets. Thousands of pre-processing and feature extraction steps, models, and respective hyperparameters were tested and optimized using a fully automated process.

Each benchmark dataset within TDC includes its own training, validation, and test sets, alongside an evaluation metric. Out of the 22 datasets used, a combination of binary classification and regression tasks is covered. The dataset splits are computed through a scaffold split method, ensuring that the training and test sets feature distinct molecular structures. For regression tasks, the primary evaluation metric was the MAE, with some exceptions where Spearman's correlation (SC) is employed. Binary classification datasets were evaluated using the area under the receiver operating characteristic (AUROC) metric when the balance between positive and negative examples was even. Alternatively, when negative examples substantially outnumber positive ones, the area under the precision-recall curve (AUPRC) was used.

Benchmarking involves assessing an ML model's performance using specific ADMET datasets. This is done by first dividing these datasets into training, validation, and test sets using five distinct data seeds to ensure variability. The model is then trained using the training and validation sets. Its effectiveness is evaluated based on how it performs on the test set. Finally, the process wraps up by calculating the average and standard deviation of the model's predictions across all five seeds, providing a comprehensive measure of its reliability and accuracy.

Our experimental setup involved using the DeepMol AutoML engine to train models with the training sets and evaluate them on the validation sets. The optimization criterion for the *Optuna* search engine was to optimize the mean of the selected metric values across all validation sets. The Tree-structured Parzen Estimator algorithm was utilized across 100 trials for hyperparameter and method selection.

Upon the conclusion of each experiment, the optimal pipeline was identified following the objective function criteria. Subsequently, the model underwent retraining by integrating both the validation and training datasets and its performance was assessed on the test set in alignment with the guidelines specified for submission. *Voting Pipelines* are also included in this analysis.

We compared our results with the TDC ADMET benchmark group leaderboard submissions. Overall, our AutoML framework demonstrated good performance, securing top-1 placement in one benchmark and achieving top-3 and top-5 rankings in four and eleven benchmarks, respectively. While our performance may not have been outstanding across all benchmarks, it is crucial to understand the remarkable nature of these results. This is particularly noteworthy, considering that the majority of benchmark submissions typically rely on handcrafted pre-processing steps and models selected for their performance. Our AutoML framework, operating without explicit instructions and requiring no prior domain knowledge from the users, efficiently navigated an extensive array of possibilities. That resulted in fully reproducible end-to-end pipelines that demonstrated competitive results. More details on each of the voting pipelines and ensembles are given in Supplementary Table 30.

3.3.1.1 Absorption benchmarks

The *absorption benchmarks* measure how a drug travels from the administration site to its action site. These benchmarks consist of six datasets, three of which involve classification, and three involve regression. The AutoML framework of DeepMol achieved the top rank on the leaderboard for the human intestinal absorption (HIA) dataset and the second position for the aqueous solubility (AqSol) dataset. Interestingly, the HIA dataset, comprising 578 molecules, is the smallest in the benchmark, while the AqSol dataset, consisting of 9982 molecules, is the largest. This shows the effectiveness of our framework in finding robust models across varying data sizes. Additionally, it is noteworthy that graph-based models outperformed others in three out of the six benchmarks. The results for the six benchmarks and respective pre-processing steps and models can be seen in Table 4.

Although DeepMol did not provide so much competitive performance for the drug oral bioavailability (Bioav) and lipophilicity (Lipo) datasets, the AutoML results provided invaluable hints regarding the best-performing methods. *Optuna* provides a dashboard to explore the results across all the trials. We analysed the results and focused our pipeline optimization on the best-performing methods and ML models. This allowed us to identify Histogram-based Gradient-boosting classification trees as top-performing models for the Bioav dataset and stacking ensembles for the Lipo dataset, leading us to set up smaller experiments and reduce the configuration space of the method search. We improved the AUROC on the Bioav test set from 0.617 to 0.753 and secured 1^{st} place in the leaderboard. Not so impressively, we secured the 6^{th} spot in the Lipo dataset leaderboard.

Table 4: Performance of DeepMol in the six absorption-related properties in the TDC ADMET benchmark group.

Dataset	**Standardization**	**Featurization**	**Feature Selection**	**Model**	**Metric**	**Result**	**Rank***
Caco2	-	Molecular Graph	-	AttentiveFP	MAE ↓	0.297 ± 0.008	6/19
HIA	Custom	Molecular Graph	-	AttentiveFP	AUROC ↑	0.990 ± 0.002	1/18
Pgp	Custom	Morgan FP	-	Logistic Regression	AUROC ↑	0.922 ± 0.000	7/18
Bioav	-	-	-	Voting Pipeline	AUROC ↑	0.617 ± 0.009	11/18
Lipo	Custom	Morgan FP	Percentil FS	Stacking Ensemble	MAE ↓	0.656 ± 0.012	14/17
AqSol	-	Molecular Graph	-	Attentive FP	MAE ↓	0.775 ± 0.006	2/16

**Assessed April 16th, 2024.*

3.3.1.2 Distribution benchmarks

The *distribution benchmarks* measure how a drug moves to and from the various tissues of the body and the amount of drug concentration in those tissues. These benchmarks consist of three datasets, two involving regression and one classification. The results of this benchmark were less satisfactory, particularly in the case of the classification dataset that measures the drug's ability to penetrate the blood-brain barrier, ranking in the lower half of the entire benchmark. Notably, this particular benchmark appears to be among the most challenging, with only ensemble-based strategies showing the best performance. The results for the three benchmarks and respective pre-processing steps and models can be seen in Table 5.

Table 5: Performance of DeepMol in the three distribution-related properties in the TDC ADMET benchmark group.

Dataset	**Standardization**	**Featurization**	**Feature Selection**	**Model**	**Metric**	**Result**	**Rank***
BBB	-	-	-	Voting Pipeline	AUROC ↑	0.868 ± 0.007	15/23
PPBR	Custom	Descriptors	Percentil FS (62%)	Stacking Ensemble	MAE ↓	7.99 ± 0.104	5/17
VDss	Custom	MACCS keys	Low Variance FS	Random Forest	SC ↑	0.497 ± 0.011	8/17

**Assessed April 16th, 2024.*

3.3.1.3 Metabolism benchmarks

The *metabolism benchmarks* evaluate the breakdown of drugs by specialized enzymatic systems, determining the duration and intensity of their effects. These benchmarks consist of six classification datasets. We achieved top-2 performance in two datasets and top-5 in three. Notably, unlike the absorption benchmarks, which thrived on graph-based approaches, simpler methods such as molecular descriptors and fingerprints yielded superior results in the metabolism benchmark. Similarly, ensemble learning techniques outperformed individual models in the *distribution benchmark*. The results for the six benchmarks and respective pre-processing steps and models can be seen in Table 6.

3.3.1.4 Excretion benchmarks

The excretion benchmarks measure the removal of drugs from the body using various routes. These benchmarks consist of three regression datasets.

The exploration of predictive models across these datasets reveals that GNNs, alongside the ChEMBL standardizer, emerge as standout methodologies. For the Half-Life dataset, a DeepMol custom standardization and scaled descriptors combined with a 1D CNN achieved an SC score of 0.485, ranking it fifth. On the other hand, a voting pipeline that standardized molecules with the ChEMBL standardizer comprises three Directed Message Passing Neural Networks (D-MPNN) and two GCNNs and reaches fourth place for

Table 6: Performance of DeepMol in the six metabolism-related properties in the TDC ADMET benchmark group.

Dataset	**Standardization**	**Featurization**	**Feature Selection**	**Model**	**Metric**	**Result**	**Rank***
CYP2C9 Inhibition	-	-	-	Voting Pipeline	AUPRC ↑	0.758 ± 0.002	8/18
CYP2C9 Substrate	-	MACCS keys + RDK FP	Low Variance FS	Bagging	AUPRC ↑	0.417 ± 0.005	5/19
CYP2D6 Inhibition	Custom	RDK FP	Boruta	Stacking	AUPRC ↑	0.685 ± 0.000	5/17
CYP2D6 Substrate	-	Descriptors (binarized)	Percentil FS	Gradient Boosting	AUPRC ↑	0.731 ± 0.037	2/17
CYP3A4 Inhibition	-	-	-	Voting Pipeline	AUPRC ↑	0.867 ± 0.002	7/17
CYP3A4 Substrate	Custom	Atom Pair FP	Low Variance FS	Bagging	AUROC ↑	0.655 ± 0.003	2/18

**Assessed April 16th, 2024.*

the CL-Hepa dataset. For the CL-Micro dataset, optimal performance was achieved using a voting pipeline that combined five pipelines employing the ChEMBL standardizer with five distinct TextCNNs. The pipeline achieved a modest 11th place, lagging behind the first place by a small SC margin of 0.05.

The results for the three benchmarks and respective pre-processing steps and models can be seen in Table 7.

Table 7: Performance of DeepMol in the three excretion-related properties in the TDC ADMET benchmark group.

Dataset	**Standardization**	**Featurization**	**Feature Selection**	**Model**	**Metric**	**Result**	**Rank***
Half Life	Custom	Descriptors (scaled)	-	1D CNN	SC ↑	0.485 ± 0.039	5/18
CL-Hepa	-	-	-	Voting Pipeline	SC ↑	0.44 ± 0.011	4/16
CL-Micro	-	-	-	Voting Pipeline	SC ↑	0.553 ± 0.013	11/18

**Assessed April 16th, 2024.*

3.3.1.5 Toxicity benchmarks

The toxicity benchmarks measure how much damage a drug can cause to organisms. These benchmarks consist of four datasets, one involving regression and three for classification.

DeepMol achieved good results for the Ames and LD50 datasets, which ranked fifth and fourth, respectively. For the former dataset, a voting pipeline with five GNNs standardizing molecules with the ChEMBL standardized, while for the latter, a pipeline using a custom standardizer, layered fingerprints, a

feature selection method that selected features based on the percentile 79 of the scores of a univariate linear regression test (for more information refer to [190]), and a voting regressor composed of 5 different models. On the other hand, the results for the rest of the datasets were far from impressive, with a GCNN ending up in the 9th place of the hERG dataset leaderboard and a Voting pipeline securing a place in the half of the leaderboard for the DILI dataset.

The results for the four benchmarks and respective pre-processing steps and models can be seen in Table 8.

Table 8: Performance of DeepMol in the four toxicity-related properties in the TDC ADMET benchmark group.

Dataset	**Standardization**	**Featurization**	**Feature Selection**	**Model**	**Metric**	**Result**	**Rank***
hERG	-	Molecular Graph	-	GCNN	AUROC ↑	0.763 ± 0.015	9/18
Ames	-	-	-	Voting Pipeline	AUROC ↑	0.847 ± 0.007	5/17
DILI	-	-	-	Voting Pipeline	AUROC ↑	0.885 ± 0.014	9/18
LD50	Custom	Layered FP	Percentil FS	Voting Regressor	MAE ↓	0.614 ± 0.004	4/19

**Assessed April 16th, 2024.*

3.3.2 Practical applications of DeepMol

Several publications have already showcased the versatility of *DeepMol* in various domains, such as drug discovery, food science and natural product discovery, across different tasks. Numerous experiments have exploited the diverse array of methods offered by DeepMol, highlighting its user-friendly nature and contributing to significant research advancements.

In their studies [211], [223], Baptista and colleagues used *DeepMol* to investigate which compound representation methods are most suitable for drug sensitivity prediction in cancer cell lines. They benchmarked twelve compound representations, including molecular fingerprints and DL-based representation learning methods, using two classifications and three regression datasets from human cancer single-cell line drug screenings. The authors found that most compound representations performed similarly. Still, some end-to-end DL models performed on par with or even outperformed traditional fingerprint-based models, even when dealing with smaller datasets. Furthermore, the authors utilized *DeepMol*'s feature importance methods to enhance the interpretability of fingerprint-based DL models. They demonstrated that consistently highlighted features were known to be associated with drug response.

Capela *et al.* [212] conducted a study in which they trained and evaluated 66 different model configurations to predict the relationships between chemical structures and sweetness. Throughout the study pipeline, *DeepMol* was utilized for several tasks, such as molecular standardization, feature generation, feature selection, model construction, hyperparameter tuning, and model explainability. Furthermore, a

subset of the trained models was employed to screen 60 million molecules from PubChem [224] in search of potential sweeteners. The authors successfully identified numerous derivatives of potent and artificial sweeteners, some of which were patented as sweetening agents and were not included in the original training data. This demonstrated the remarkable capability of *DeepMol* in helping design new sweeteners and repurposing existing compounds.

In a recent study [225], DeepMol AutoML was used to address the challenge of predicting precursors of specialized plant metabolites, which play critical roles in plant defense and have significant economic implications. Despite these compounds' complexity, vast diversity, and the current gaps in understanding their biosynthesis, DeepMol's methodology stood out for its efficacy. It was useful to find regularized linear classifiers that surpass existing state-of-the-art approaches in performance and also offer chemical explanations for their predictions. This approach marks a significant advancement in expediting the discovery of biosynthetic pathways, highlighting the potential of DeepMol's AutoML in finding high-performing models.

3.4 Discussion: comparison with other tools

Several chemoinformatics toolkits have been developed in recent years [182], [226]–[229], including *Deep-Chem* (`https://deepchem.io/`), OpenChem [228], AMPL [227], and ZairaChem [188]. These open-source projects, developed in Python, are used to construct pipelines for QSAR/QSPR modeling. A comparison between DeepMol and the other tools is showcased in Figure 17.

Regarding dataset loading, *DeepMol* and *DeepChem* support CSV and SDF formats, while OpenChem, AMPL, and ZairaChem only support CSV files. Hence, a limitation of the latter two tools is their incapability to import pre-computed 3D structures. Molecule standardization is available in *DeepMol* through various methods. OpenChem allows sanitizing molecules, padding sequences and canonicalizing SMILES. In contrast, AMPL allows the striping of salt groups, neutralizing and kekulizing the molecules and replacing any rare isotopes with the most common ones for each element. *DeepChem* only allows sanitizing molecules when loading datasets. ZairaChem applies the MELLODDY-Tuner protocol, which includes disconnecting metal atoms from the rest of the molecule, reionizing the molecule, adjusting charges and protonation states, assigning stereochemistry to the molecule, ensuring consistent representation of chiral centers, and additional standardization procedures. All the tools can perform data splitting, except OpenChem, and only *DeepMol* can perform a stratified split specific to molecular data and multitask classification. Additionally, all the tools are capable of generating numerical features from molecules. Hyperparameter and architecture optimization are possible in all the toolkits except for OpenChem.

Model construction in these toolkits primarily focuses on DL models, but all the tools provide shallow learning models except for OpenChem. Hyperparameter and architecture optimization are possible in all the tools except OpenChem. Moreover, while all the methods offer pipelines, only *DeepMol* and ZairaChem provide a comprehensive and efficient approach to automating pipeline optimization (AutoML).

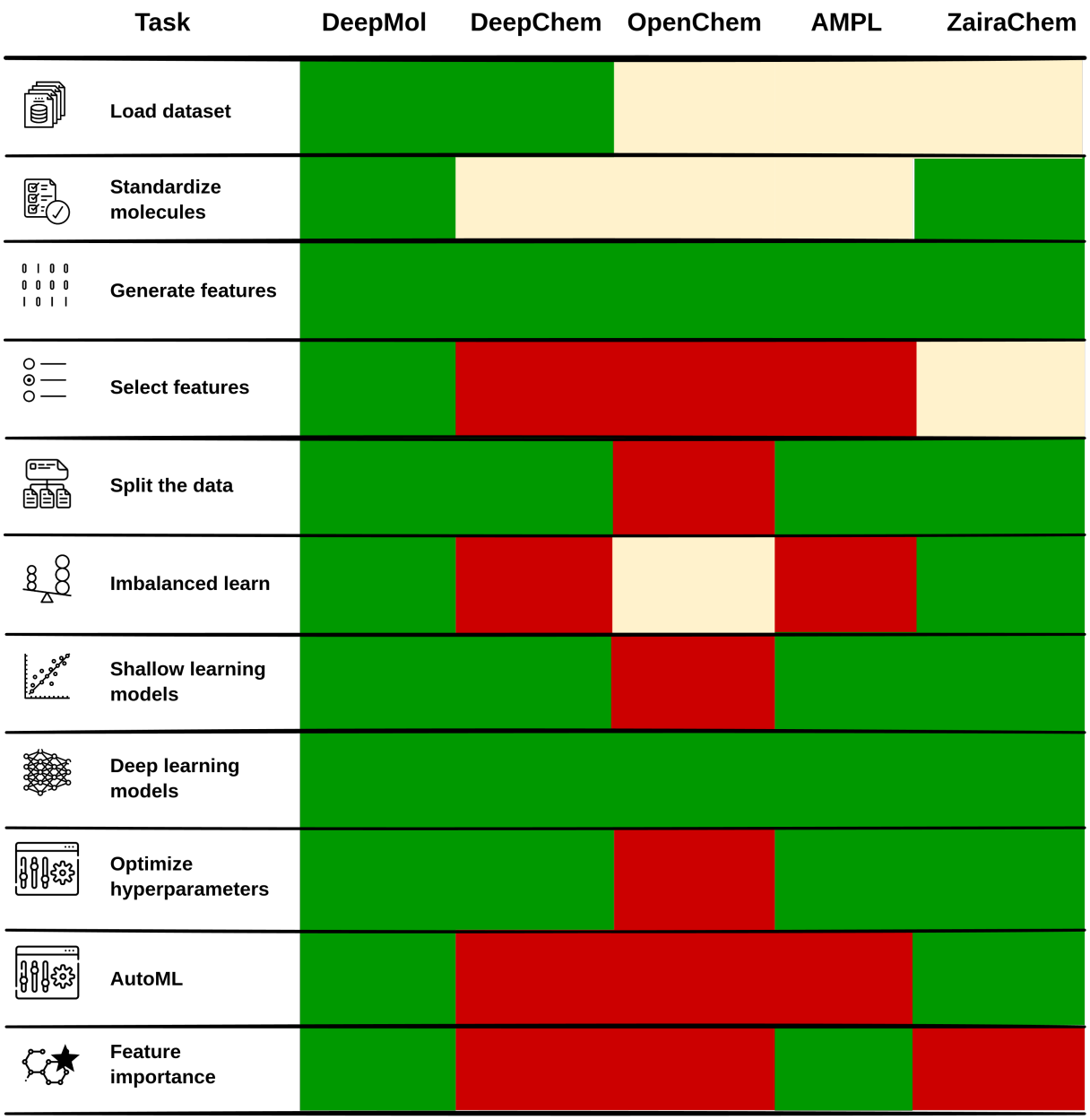


Figure 17: Heatmap showcasing the presence or absence of an integrated way of performing relevant tasks for constructing QSAR/QSPR systems. Green stands for presence, red for absence and yellow for incomplete integration.

DeepMol excels compared to other tools in providing feature importance analysis and effectively connecting features to molecular structures. Both ZairaChem and *DeepMol* also offer methods to address unbalanced datasets and perform integrated and automated feature selection, distinguishing them from the other assessed tools. ZairaChem applies feature selection through Autogluon, which offers only one feature selection method.

Even though *DeepMol* and ZairaChem appear to perform similarly for these tasks, it is important to note that ZairaChem is limited to binary classification problems. Consequently, the features provided by ZairaChem, as indicated in Figure 17, are specifically designed for binary classification tasks.

3.5 Conclusion

In conclusion, *DeepMol* emerges as a powerful alternative to similar tools, offering a Python-based open-source framework for predicting activities and properties of chemical molecules. Its modular design allows researchers to customize every aspect of the ML pipeline, from data processing to model prediction and explainability, catering to users with varying levels of computational expertise. *DeepMol*'s AutoML modules further simplify the process by automatically optimizing pre-processing, data engineering techniques, and ML/DL models and their hyperparameters, streamlining the selection of the most suitable combinations for a given dataset. By providing a user-friendly and easily customizable platform, *DeepMol* empowers researchers to efficiently explore thousands of configurations, making it an invaluable resource for accelerating chemical discovery. With the support of well-established packages like *RDKit*, *Scikit-Learn*, *Tensorflow*, and *DeepChem*, *DeepMol* represents a promising avenue for optimizing and deploying pipelines, offering a vital contribution to the advancement of computational chemistry. The framework's availability as an open-source resource encourages collaboration and innovation in the field, facilitating progress and empowering researchers in their quest to find new chemicals with improved properties.

4

Predicting the number of biochemical transformations needed to synthesize a compound[1]

4.1 Introduction

During the last decades, microorganisms have been extensively used as a platform for the production of added-value compounds with a wide set of applications in the pharmaceutical, chemical, food, and agriculture industries. It was in 1928 when Sir Alexander Fleming discovered Penicillin from *Penicillium notatum* [231], that microorganisms were seriously regarded as a source of natural products triggering a big wave of interest in the subject. Since then, the production of drugs, chemicals, and biofuels using microorganisms has grown exponentially.

Microbial biosynthesis offers many advantages over traditional chemical synthesis. While traditional chemical synthesis demands high energetic resources and often produces toxic intermediates, microbial biosynthesis is safer and eco-friendly [232]. Additionally, taking into account how developed the fields of ME, protein engineering, and synthetic biology are nowadays, the high accessibility of engineered strains to produce specific compounds makes the process of redesigning microbial cellular networks and fine-tuning physiological capabilities much easier [233]. ME also offers established pathway optimization methods for improvements in the yield and productivity of target compounds [234].

While multiple organisms can be used and are optimized to produce specific compounds, none is as well characterized as *Escherichia coli*. *E. coli* has been studied over several decades and has shown its potential in many functional genomics and ME efforts [235], [236]. Moreover, its ability to quickly grow on minimal media, while maintaining its full metabolic function offers an important advantage over other popular organisms, such as the yeast [237].

In the last years, the application of ML and DL in bioinformatics has seen a growing interest as biological data becomes more accessible. In particular, the availability of comprehensive datasets of

[1] This chapter is an unmodified version of an article [230] published and presented in the *2022 International Joint Conference on Neural Networks (IJCNN)*, written in co-authorship with Rafael Carreira, Vítor Pereira and Miguel Rocha.

omics data and metabolic pathways propelled the use of DL within the field of ME. DL is a subfield of ML that uses multi-layerNNs to learn hierarchical abstract features using a set of non-linear modules that transform, at each step, the original input data into more abstract representations [238]. DL has achieved remarkable results in many fields including computer vision [239], speech recognition [240] and bioinformatics [241]. In particular, the use of DL in ME is evolving at a fast pace. Some of the most prolific applications include product maximization by, for example, predicting optimal reactor conditions, *de novo* pathway design, phenotypic profiling, and robust system modeling [242].

One recent challenging subject in ME involving the application of DL, in particular RL, is retrobiosynthesis. Retrobiosynthesis consists of efficiently breaking a target compound by finding a series of producing reactions, until readily available starting materials are obtained. Retrobiosynthesis is commonly carried out using template-based approaches by identifying a set of reaction rules, representing enzymatic reactions that have been observed in biological systems, that can transform native metabolites of a host into the target molecule [60], [243]. Other approaches do not rely on reaction templates and use DL models to directly predict the outcome of reactions in the reverse order [40], [244]. However, the debate on whether template-based approaches are long-term feasible, due to the sheer number of reaction templates available for matching, or if the role of human experts is indispensable due to the complexity of the reaction mechanism is still ongoing [245].

The tree of possibilities generated by retrosynthesis approaches results in a combinatorial explosion of possible pathways. Thus, exhaustive exploration of these pathways is not computationally feasible. To effectively explore the most promising routes, a set of heuristics to prune the tree needs to be defined. Many retrosynthesis studies approach this problem in different manners. For example, in Simpheny [246], to constrain the network size a maximum molecule size is defined. In RetroPath [247], the network complexity is limited by coding substrates, products, and reactions into molecular signatures. A reaction signature is given by the difference between the signatures of products and substrates. This signature can be controlled using predefined distances, defining graph distances of atoms, which leads to different levels of specificity, thus the number of *de novo* pathways can also be controlled. In RetroPath 2.0 [60], they also use signatures, but introduce an additional enzyme score reflecting the ability to retrieve enzyme sequences catalyzing defined transformations. SimZyme/SimIndex [248] and PathPred [51] use similarities between the molecules and the typical substrate of an enzyme.

Despite many studies suggesting different strategies to define which compounds to prioritize when searching for the best pathways to synthesize a compound, there is no single method that is regarded as the best. Another important factor to consider when computing these routes is how easy they are to reproduce in a host microorganism. For this, different factors such as atom conservation, thermodynamics, presence or absence of toxic intermediates, product yield and pathway length are taken into consideration when ranking the obtained pathways.

In this study, we propose the use of DL to predict the number of biochemical transformations needed to synthesize a compound. We tested different DL architectures, in specific, FCNNs and 1D CNNs. We also tested different compound representations, such as molecular fingerprints and transformer-based molecular embeddings. To our knowledge, this is the first attempt to leverage DL to model the number

of steps needed to synthesize a compound. To train the DL models, we generated data using reaction rules from public databases with compounds from the *E. coli* metabolism as starting materials. The performance of our models indicates that DL can promote the study of compound synthetisability in host microorganisms and it is expected that it can represent a useful tool to effectively narrow down a large number of retrobiosynthesis-derived compound candidates to more promising routes.

4.2 Materials and Methods

4.2.1 Data

In this work, multiple DL architectures and compound representations were tested. The data used to train the models was generated using a set of 13055 reaction rules and a set of 673 compounds from *E. coli* metabolism as available starting precursor compounds. The reaction rules were collected from two public resources, RetroRules [27] and MINE databases [249].

Reaction rules. Reaction rules are generic descriptions of reactions that encode the way reactants are converted into products. Reaction rules are very important for a variety of synthetic biology approaches, mainly in *de novo* pathway discovery. A reaction rule can be applied to a compound if the compound contains a particular substructure that is encoded by the reaction rule. Then, new product compounds can be generated by applying the transformation encoded by the reaction rule. These rules can encode single and multi reactant reactions and can generate multiple products if they match multiple parts of the reactant molecules. In Fig. 18, an example of a known reaction is shown (id MNXR94682 in the MetaNetX database [250]), which was used to generate a reaction rule.

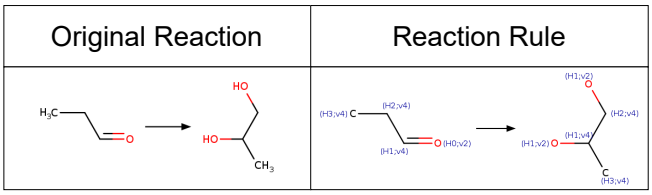


Figure 18: Example of a reaction rule and respective original reaction.

The reaction rules from RetroRules were downloaded directly from the website `https://retrorules.org/dl`. Only the reaction rules that apply in the forward direction and with a diameter of 2 were selected resulting in a set of 5732 rules. RetroRules contains reaction rules that were generated using different diameters, in specific, 2 to 16. This diameter consists of the distance considered around the reaction center of the original metabolic reaction when generating the reaction rule. So one reaction can be used to generate multiple reaction rules at different diameter levels. The higher the diameter, the more specific the rules will be. This diameter directly links with the known promiscuity of enzymes. We selected the reaction rules with diameter 2 for these reasons, they are less specific and promote enzyme promiscuity.

The 7323 MINE database reaction rules were downloaded from the GitHub page `https://github.com/tyo-nu/MINE-Database/`and were used as they are. The final dataset consisted of 13055 reaction rules identified using the SMARTS notation [66]. The validity of each reaction rule SMARTS was determined using RDKit [189], an open-source toolkit for cheminformatics.

Starting compounds. The list of starting precursors that we assume to be available are the ones existing in the metabolism from *E. coli*. We selected this microorganism because it is widely used in a great set of biochemical processes including in the synthesis of added-value compounds. These *E. coli* sink compounds were downloaded from the RetroPathRL [58] GitHub `https://github.com/brsynth/RetroPathRL`. The authors extracted these compounds from the *E. coli* iJO1366 genome-scale metabolic model [251]. They selected the compounds that lie in the cytosol compartment except for the ones that cannot be produced by any reactions in a steady-state metabolic model. This was obtained by performing a Flux Variability Analysis [252] using the COBRApy package [253]. After downloading these data, we selected only the compounds with available and valid identifiers, in this case, the InChI [135], resulting in a set of 673 compounds. The validity of each compound InChI was determined using RDKit. For further usability, the compounds were converted into the SMILES notation [76] also using RDKit.

Generated Dataset. The dataset used to train and evaluate our DL models was generated by successively trying to apply randomly selected reaction rules to randomly selected compounds from the previous step starting from the *E. coli* iJO1366 sink compounds (step 0). In the first step, because the number of compounds was low, we applied all possible reaction rules to all *E. coli* iJO1366 sink compounds, resulting in a set of 146157 compounds. After this step, we tried to apply randomly selected reaction rules to randomly selected compounds over 5 million iterations. This second step resulted in a set of 464994 compounds. We repeated this process up to five steps, resulting in the dataset described in Table 9. When a compound that was already generated in a previous step was generated again, only the first one was kept. We defined a maximum of five biochemical transformations based on the fact that, in general, microbial networks have small-world properties and thus small average path length [254], [255]. This means that, in theory, with a small number of biochemical transformations we can go from any compound to any compound in the network.

Table 9: New compounds generated at each step.

Step	Generated Dataset	Independent Dataset
1	146157	16439
2	464994	27151
3	600280	44681
4	698529	97249
5	773586	70342
Total	2683546	255862

After the dataset was generated it was divided into train, validation, and test sets using ScikitLearn [190]. The splits were made in a stratified fashion based on the step value with the proportions of

approximately 60%, 20%, and 20% for train, validation, and test sets respectively.

Independent Dataset. Since the compounds present in later steps were generated using the compounds from the previous step, there is a dependency between the compounds generated at each step. To validate if the previously generated dataset was representative enough we generated an independent set using the same approach. The newly generated independent test set comprises 255862 unique compounds. The number of compounds for each step is shown in Table 9.

4.2.2 Molecular Representations

There has been a lot of research on how to better represent compounds in a suitable form so that ML algorithms can learn and generalize the information shared among sets of compounds. Several approaches to encode the properties and structural characteristics of compounds have been reported in the literature. From single descriptors to complex multi-dimensional graph-based formulations, compound representation remains a hot topic. The most traditional molecular features include molecular descriptors and fingerprints, such as the ECFPs [256]. Other successful approaches that make use of representations like other types of fingerprints, weave, graph convolutions, and NLP-inspired embeddings are actively being used [14], [257], [258]. In this study, we focus on two distinct molecular representations, the well-known Morgan fingerprints [256] and the NLP-based molecular transformer embeddingss (MTEs) [259].

Morgan Fingerprints. Also known as circular fingerprints, Morgan fingerprints are built by applying the Morgan algorithm to a defined set of atom invariants up to a defined radius around each atom of the molecule. We computed Morgan fingerprints of radius 2 hashed to 1024 bits using the RDKit package.

Molecular Transformer Embeddings. With the emergence of the Transformer architecture [111], the field of NLP saw some considerable improvements. The use of such architecture has been explored in the field of chemoinformatics, in particular by translating between two distinct text-based molecular representations in well-studied subsets of the chemical space. In the study by Morris *et al.* [259], the authors trained and repurposed, through transfer learning, a transformer network to predict binding affinity. The intermediate set of features representing abstract features that describe general molecular structures that are generated by this architecture can be used as embeddings. These embeddings can then be used as features to train other models for diverse purposes. We computed these MTE for our datasets. We defined a maximum length of our compound SMILES of 300 characters and an embedding size of 512.

4.2.3 Deep Learning Models

In this study, we leverage the use of DL models to predict the number of biochemical transformations needed to synthesize a compound. We defined this problem as both a multiclass classification problem with 5 labels, the 5 steps, and as a regression problem, where the number of biochemical steps is modulated as a continuous variable. For this, we use FCNNs and 1D CNNs for both classification and regression tasks.

Fully Connected Neural Networks. A FCNN consists in a set of fully connected layers. In a fully connected layer, all possible neuron connections between layers are present, meaning that every input

dimension influences each output dimension. Fully connected layers are the most common layers in ANNs. One of the major advantages of FCNNs is that no special assumptions about the input data need to be made, making them very broadly applicable. However, these networks are very computationally intense, prone to overfitting, and tend to have weaker performance than specific networks tuned for the problem in question.

In this study, we used FCNNs for both classification and regression tasks using Morgan fingerprints and MTE as inputs. All our FCNN models follow a similar architecture. The models consist of an initial input Dense layer with 1024 units in the case of using Morgan fingerprints and 512 for the MTE. Then a variable number of Dense, with L1 and L2 regularizers, BatchNormalization and Dropout layers are added to the model. A final output Dense layer concludes the model. The initial and intermediate Dense layers use the Relu activation function. In the case of classification, the final dense layer has 5 units and uses softmax as the activation function. In this case, we also use the categorical cross-entropy as the loss function and accuracy as the metric to follow during training. In the case of regression, the last Dense layer has 1 unit and uses the linear function as activation function. The loss function and metric to follow during training was the MAE. We used the Adam as our optimization algorithm and a ReduceLROnPlateau callback with a minimum learning rate of 0.000001, a factor of 0.25, and patience of 10 epochs that reduces the learning rate by 25% if there is no improvement in the validation loss for 10 epochs. We defined a maximum of 250 epochs and a batch size of 256. We also use the EarlyStopping callback with patience of 15 epochs that stops training when there is no improvements in the validation loss for 15 epochs.

1D Convolutional Neural Networks. CNNs are a class of feed-forward ANNs that gained popularity during the last decades and became the dominant method for computer vision applications. CNNs employ mathematical operations called convolutions in at least one of its layers. Other building blocks of CNNs are pooling and fully connected layers. Both convolution and pooling layers are designed to identify spatial hierarchies of features that are then mapped to the final output by the fully connected layers. Despite being designed for two-dimensional inputs, CNNs can also deal with other dimensional inputs such as 1D and 3D. As opposed to 2D convolutions that act in two dimensions, 1D convolutions operate in only one dimension applying 1D convolutions (scalar multiplications and additions). This allows the use of these types of models in data such as 1D numerical vectors, including molecular descriptors, fingerprints, and embeddings.

In this study, we used 1D CNNs for both classification and regression tasks using Morgan fingerprints and MTE. All our 1D CNN models follow a similar architecture. The models consist of an initial GaussianNoise layer that adds noise to the data helping to mitigate overfitting, followed by 2 Conv1D layers using the Relu activation function. Then one Flatten and one Dropout layer are added to the model. Before the output layer, a set of 1 Dense, 1 BatchNormalization, and 1 Dropout layers are added to the model. The model is then concluded with an output Dense layer. We used Adam as our optimization algorithm and a ReduceLROnPlateau callback with a minimum learning rate of 0.000001 with a factor of 0.25 with patience of 10 epochs. We defined a maximum of 150 epochs and a batch size of 512. We also used the EarlyStopping callback with patience of 15 epochs.

4.2.4 Performance metrics

The use of performance metrics is a key step in any ML pipeline to ensure that the model is performing as expected. There are dozens of metrics for both classification and regression, we will discuss the ones used in this study.

Classification metrics: Some of the most popular metrics in classification tasks include classification accuracy, precision, recall, F1-score, and the confusion matrix, which is essential to compute some metrics.

- Confusion Matrix: The confusion matrix is a two-dimensional matrix that summarizes the classification performance of a classifier regarding the model label predictions versus ground-truth labels [260]. Each row in the confusion matrix represents the predicted classes by the classifier and each column represents the actual class. In a confusion matrix for a binary classification there are four important terms: true positives (TPs), true negatives (TNs), false positives (FPs), and flase negatives (FNs). TPs represent the number of positive class samples that were predicted correctly. TNs represent the number of negative class samples that were predicted correctly. FPs represent the number of negative class samples that were predicted incorrectly. FNs represent the number of positive class samples that were predicted incorrectly. In multiclass problems, there are no negative or positive classes, so the TPs, TNs, FPs, and FNs are calculated for each class.

- Accuracy: This classification metric represents the ratio of correct predictions in the total number of input samples [260].

- Precision: Focused on Type-I errors, precision measures the fraction of positive class predictions that were actually positive [260]. Precision for each class can be calculated using the following formula: $\frac{TPs}{TPs+FPs}$. Values of precision near 1 mean that the model is performing well on classifying the positive cases as positive, whereas low precision scores mean that the model is classifying a high portion of negative cases as positive, i.e. produces a high number of FPs.

- Recall: Focused on Type-II errors, recall, also known as sensitivity, measures the fraction of all TP samples that were actually predicted as positive by the model [260]. Recall for each class can be calculated using the following formula: $\frac{TPs}{TPs+FNs}$. Recall near 1 means that the model is not missing many TPs, i.e. can correctly classify positive samples as positive. Low recall values mean that a high number of FNs are being predicted by the model, i.e. the model is classifying positive samples as negative.

- F1-score: This metric combines both precision and recall, actually it is the harmonic mean of these two metrics [261]. The f1-score can be calculated using the following formula: $\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = \frac{2TPs}{2TPs+FPs+FNs}$. In a perfect model, the respective f1-score would be equal to 1, which means that both precision and recall would also be equal to 1 which also means that the model would have a 100% accuracy. The f1-score is particularly important on imbalanced classification problems as it

tells you how precise your model is, i.e. how many cases it classifies correctly, and how robust it is, i.e. if it misses a significant number of samples.

Regression metrics: The output of regression models are continuous values. So, we need metrics that can calculate the numeric distance between the predicted value and the ground truth. Some of the most popular metrics in regression tasks include MAE, MSE and, R^2.

- MAE: This metric computes the average difference between the predicted values and the ground truth values [260]. Mathematically, it can be represented as: $\frac{1}{N}\sum_{j=1}^{N}\left|y_j - \check{y}_j\right|$, where y_j is the ground truth value, $\check{y}_j$ is the predicted value and N is the number of instances. MAE gives a measure of how far the predictions are from the actual value. However, it does not give the direction of the error, i.e. if we are under-prediction or over-predicting the data.

- MSE: Perhaps the most popular metric in regression problems, MSE computes the average of the squared difference between the predicted values and the ground truth values [260]. Mathematically, it is represented by the following formula: $\frac{1}{N}\sum_{j=1}^{N}\left(y_j - \check{y}_j\right)^2$, where y_j is the ground truth value, $\check{y}_j$ is the predicted value and N the number of instances. When compared with MAE it penalizes outliers harder by squaring them. However, this sometimes can lead to an overestimation of how bad the model is.

- R^2: Also known as the coefficient of determination, the R^2 calculates the proportion of variance which is explained by the predictor variables in the sample [262]. When the R^2 is close to 1, it means that the model was able to capture a high proportion of the variance in the target variable. If it is close to 0 it means that the model wasn't able to capture any variance in the target variable.

4.2.5 Development Environment

This work was developed using Python version 3.6.12. Molecular operations like compound SMILES validity, standardization, reaction SMARTS validity, and Morgan fingerprints generation were done using RDKit version 2019.09.3. Molecular Transformer Embeddings were generated using the release from September 16, 2020. All the models were implemented using Tensorflow version 2.2.0. Hyperparameter optimization and model training and evaluation were done using scikit-learn version 0.23.2. Source code and small data examples are available at `https://github.com/jcorreia11/WCCI2022_scripts`.

4.3 Results and Discussion

Before training any DL model, we divided our data into three sets, training set, validation set, and test set. We made sure that there was no duplicated data in and between all our datasets. Only the train and validation sets were used in the parameter optimization and training phases. The models only saw the test set and the independent set in the evaluation phase to produce the metrics shown here.

We performed 5-fold hyperparameter optimization using RandomizedSearchCV for 15 iterations for all our FCNN models. We optimized the number of hidden dense layers, the number of units in the hidden layers, the first dropout, the dropouts that followed hidden layers, and the l1 and l2 regularizers. In Table 10, the optimized parameters and the ones that produced the best results for each case are shown. For

Table 10: Parameters optimized using a 5-fold RandomizedSearch for the FCNNs.

Parameter	**Values**	**Morgan Classification**	**MTE Classification**	**Morgan Regression**	**MTE Regression**
# of hidden layers	2, 4, 6	2	2	6	2
Hidden layers units	1024, 512, 256	512	1024	256	512
First dropout	0, 0.2, 0.5	0.2	0	0.2	0
Dropout hidden layers	0, 0.3, 0.4	0	0.4	0	0.3
l1	0, 0.001, 0.01	0	0	0	0
l2	0, 0.001, 0.01	0	0.01	0	0

our 1D CNN models, we performed a 3-fold hyperparameter optimization using RandomizedSearchCV for 10 iterations. We optimized the standard deviation of an initial GaussianNoise layer, the number of output filters and kernel size of the Conv1D layers, the number of units in the Dense layers, and the dropout ratio introduced by the Dropout layers. In Table 11, the optimized parameters and the ones that produced the best results for each case are shown.

Table 11: Parameters optimized using a 3-fold RandomizedSearchCV for the 1D CNNs.

Parameter	**Values**	**Morgan Classification**	**MTE Classification**	**Morgan Regression**	**MTE Regression**
Gaussian noise stddev	0.01, 0.05	0.05	0.01	0.05	0.05
Size of output filters	4, 8, 16	16	8	16	8
Kernel size	32, 64, 128	32	32	64	64
Dense layers units	512, 256, 128	512	512	256	128
Dropout	0, 0.3, 0.5	0.5	0.3	0.5	0

In Fig. 19, the accuracy for the four classification models using the test and independent sets are

shown. The use of Morgan fingerprints to encode the compounds was considerably superior when compared with the embeddings generated with the MTE. The use of FCNNs versus 1D CNNs produced better results, but the differences were smaller than the differences obtained when comparing the molecular representation methods. The FCNNs with Morgan fingerprints obtained a test accuracy of 63% which was 4% better than our second best performing model, the 1D CNN also with Morgan fingerprints.

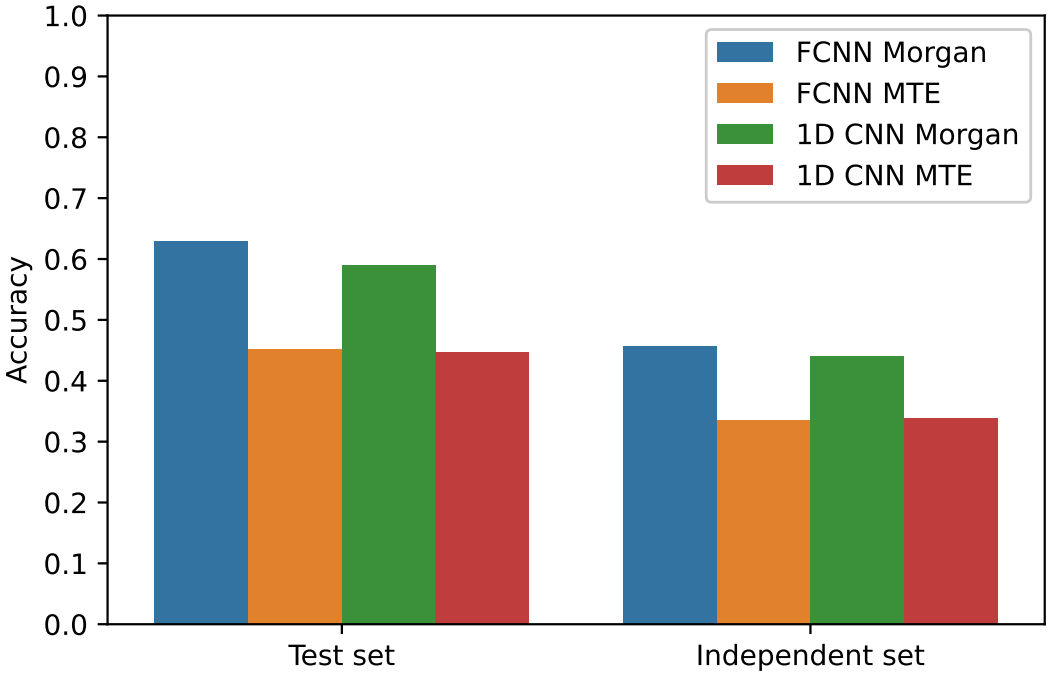


Figure 19: Test and independent set accuracy for all models.

The best classification model, the FCNNs with Morgan fingerprints, was trained using the best parameters from the hyperparameter optimization search for 27 epochs (early stopping with a maximum of 250 epochs) and obtained a training accuracy of 0.735, a validation accuracy of 0.646, and a test accuracy of 0.630.

By analyzing the results shown in Table 12, we can see that the model performs the best in classifying compounds that require 1, 2, and 5 biochemical transformations to be synthesized with higher precision, recall, and f1-score metrics. The compounds that require 3 and especially 4 biochemical transformations have worse performance metrics. Additionally, if we take a closer look at the confusion matrix in Table 13 we can see that the majority of the mispredictions, around 78%, only fail by one step, which may be a reasonable estimate in practical applications since we are looking for an estimation of how near we are from our available starting precursors. This can also mean that this problem can better be modeled as a regression task. If we consider that our model predictions are correct if the output step is no more than 1 step, up and down, far from the labeled value, our accuracies increase by a considerable margin achieving a 92% test and 91% independent test accuracies using the FCNN with Morgan fingerprints. The

Table 12: Classification report of the FCNNs with Morgan fingerprints.

Step	Precision	Recall	F1-Score
1	0.77	0.81	0.79
2	0.61	0.79	0.69
3	0.52	0.55	0.54
4	0.49	0.37	0.42
5	0.66	0.65	0.66

Table 13: Confusion matrix of the FCNN with Morgan fingerprints.

Step	1	2	3	4	5
1	25141	3316	228	101	82
2	4073	77539	9361	1632	753
3	1229	21039	76363	19234	2115
4	841	10837	30449	75675	22091
5	594	7145	17113	46150	83609

test and independent test accuracies, when allowing a 1 step margin error, for all our models are shown in Fig. 20.

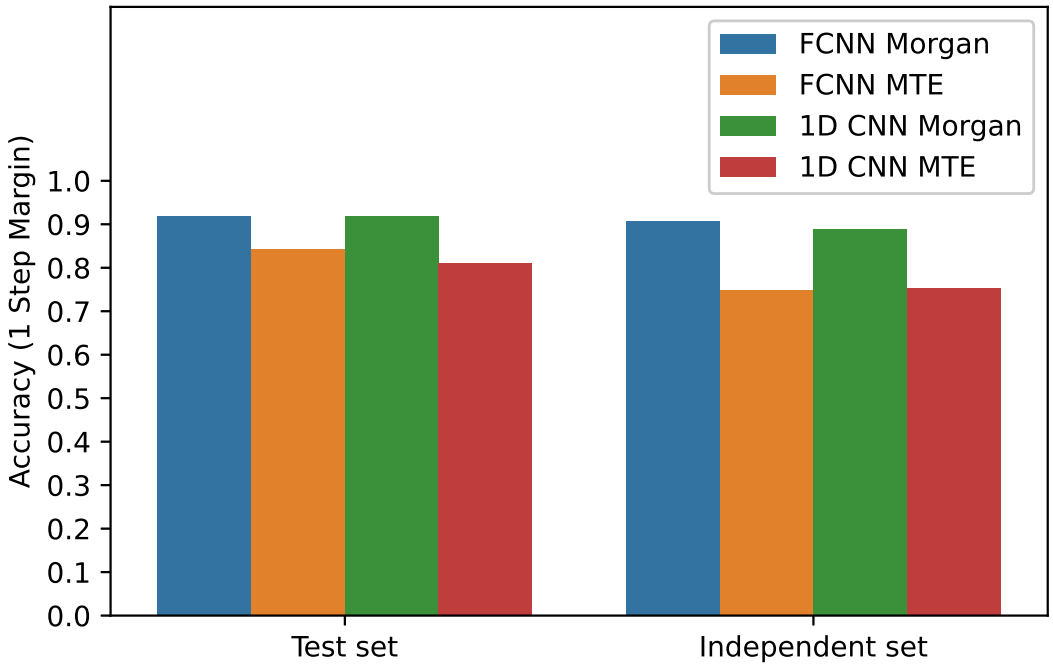


Figure 20: Test and independent set accuracy allowing miss-classification by one step for all models.

Fig. 21 shows the main results for the regression models. Looking at these results, we can argue if we obtained better results using FCNNs or 1D CNNs. Regarding the molecular representation, again, the use of Morgan fingerprints showed considerably better results when compared with the MTE. The FCNN with Morgan fingerprints performed better than the 1D CNN also with Morgan fingerprints with a MAE of 0.465. However, as we can see in Table 14, despite the lower MAE obtained by the FCNN with Morgan fingerprints, the 1D CNN with Morgan fingerprints obtained a slightly lower MSE and slightly higher R^2. Additionally, as shown in Fig. 21, if we check the performance of these two models in the independent set, both MAEs are similar.

Table 14: Regression metrics test set.

Model	**Features**	**MAE**	**MSE**	**R^2**
FCNN	Morgan	0.465	0.623	0.583
FCNN	MTE	0.691	1.165	0.220
1D CNN	Morgan	0.595	0.615	0.588
1D CNN	MTE	0.737	0.888	0.405

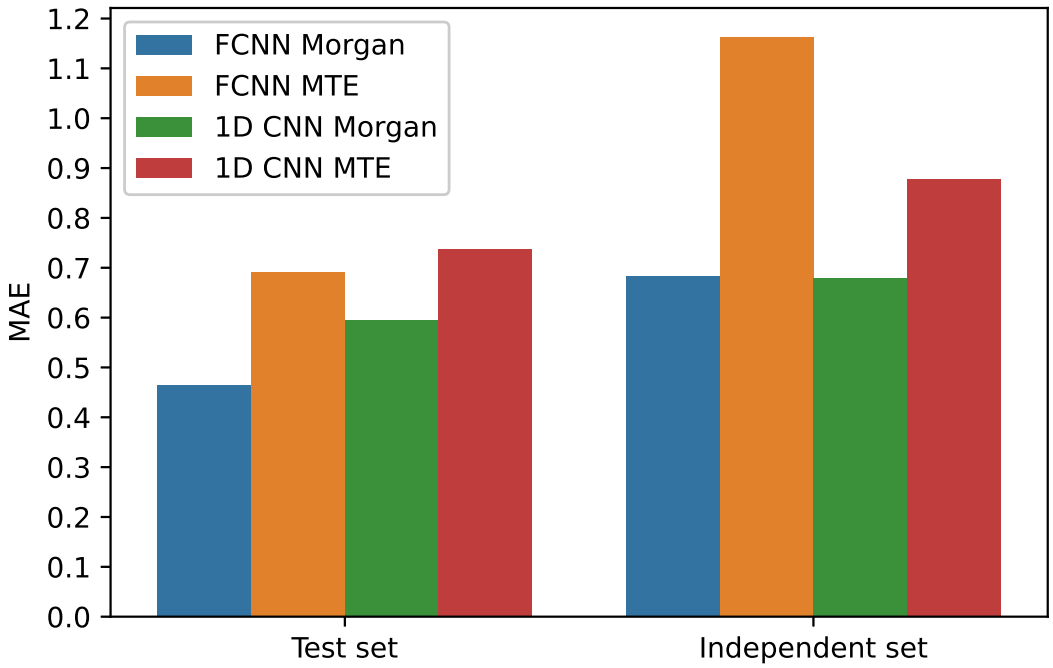


Figure 21: Test and independent set MAE for all models.

The FCNN with Morgan fingerprints was trained, using the best parameters from the hyperparameter optimization search, for 196 epochs (early stopping with a maximum of 250 epochs) and obtained a test MAE of 0.465. This means that on average the predictions are 0.465 steps away from the real step value.

As far as we know, there is no similar study in the literature that followed a similar approach to the one shown in this study. However, we are confident to say that the prediction of the number of biochemical transformations needed to synthesize a compound can be modeled as a DL problem. Despite the lack of other studies to compare our results with, we can say that the results obtained by our best models, a 63% accuracy, 92% if we give a one step margin, in a 5-label classification and 0.465 MAE in the regression, are promising. We also shown that, in some cases, the use of more complex molecular representations like molecular embeddings does not lead to better results. The results shown in this study corroborates that the use of traditional methods is still relevant in many cases.

4.4 Conclusion

In this study, we propose the use of different DL architectures and molecular representations to predict the approximate number of biochemical transformations needed to synthesize a compound having the *E. coli* metabolites as available starting materials. The models were trained using newly generated compounds obtained by successively applying reaction rules to compounds generated in the previous step, starting with the compounds in the *E. coli* sink, up to five steps. Both Morgan fingerprints and MTE were computed and tested using FCNNs and 1D CNNs. As a result, these models showed good performance both in classification and regression tasks especially when using Morgan fingerprints and FCNNs.

As far as we know, this is the first time that the prediction of the number of biochemical transformations needed to synthesize a compound using DL is described in the literature which makes the task of comparing results hard. However, we think that the results obtained in this study indicate that approaches like this one can benefit the field of ME and especially be useful in retrobiosynthesis tools to narrow the number of generated compounds allowing the exploration of the most promising pathways for the synthesis of target compounds.

In the future, it would be interesting to test other compound representations and models like RNNs and the Transformer architecture. Further exploration of the data can also be conducted to understand if the generated data are representative of what happens in microbial networks and which types of biochemical reactions are being used and left out when generating new data. Additionally, model interpretability could also be used to understand why the models make certain predictions and which properties of the molecules are more impactful for those predictions.

5

Combining Evolutionary Algorithms with Reaction Rules Towards Focused Molecular Design[1]

5.1 Introduction

Throughout history, humans have been searching and cataloging compounds, studying their effects on biological systems in an effort to find products that can improve quality of life. Natural products (NPs) which are produced by living organisms, such as bacteria, fungi, and plants have long been a rich source of drug candidates, with many of the most essential drugs in use today, including antibiotics, anticancer agents, and anti-inflammatory drugs being NPs or based on NPs. The unique and vast chemical diversity of NPs, which have been optimized through the natural evolution process to serve specific biological functions, makes them an ideal source for drug discovery, as they are enriched with bioactive molecules that span a broader range of the chemical space when compared to synthetic small molecules [264].

Discovering a drug is a long and expensive process, typically taking over 8 years and costing between 314 million to 2.8 billion US dollars [265]. For decades, drugs have been discovered by searching through libraries of biologically active natural and synthetic chemical molecules. However, the recent adoption of computational methods, such as computer-aided drug design (CADD) techniques, into the drug discovery process has resulted in a reduction of both time and cost compared to traditional trial-and-error experimentation. CADD techniques are mainly used for the rapid assessment of chemical libraries to guide and accelerate the early stages of developing new active compounds. These techniques include virtual screening, designing virtual libraries, optimizing leads, and designing new compounds from scratch.

CADD techniques can generally be grouped into two main types: ligand-based drug design (LBDD) and structure-based drug design (SBDD) [266]. LBDD involves the identification of small molecules (ligands) that can bind and modulate the activity of a particular target protein, such as a receptor or enzyme. LBDD relies on the physicochemical properties of known ligands and does not consider the three-dimensional structure of the target protein. However, LBDD is limited by the availability of known ligands for a given

[1] This chapter is an unmodified version of an article [263] published and presented in the *2023 Genetic and Evolutionary Computation Conference (GECCO)*, written in co-authorship with Vítor Pereira and Miguel Rocha.

target protein, which may not always exist. In contrast, SBDD involves using the three-dimensional structure of a protein or other biological target to design and develop small molecule drugs that can bind to and inhibit or activate the target.

In many CADD projects, virtual objective functions are used to predict properties of molecules, such as their biological activity and ADME (Absorption, Distribution, Metabolism, and Excretion) properties. In virtual screening (VS), these objective functions are evaluated for all molecules in a virtual library to identify the most promising ones. Typically, the molecules being screened are either commercially available or have well-defined synthetic routes allowing for a quick transition from *in silico* to *in vitro* studies. However, these libraries only represent a very small and biased portion of the drug-like chemical space leading to a lack of chemical novelty.

De novo drug design partially solves the chemical novelty problem by designing novel structures that provide a desired biological response, while maintaining certain pharmacokinetic properties. However, navigating and sampling the vast possible chemical space of drug-like molecules in an efficient way is not a trivial task. CADD techniques face the challenge of finding optimal solutions, finding a trade-off between exploring global solutions and exploiting local optimum, as there may be many good regions of the chemical space.

Over the last decades, gradient-based and gradient-free molecular optimization methods have received attention for *de novo* drug design. In gradient-free approaches, the molecular generation is guided towards optimal molecules by the use of population-based stochastic optimization algorithms, such as EAs and swarm intelligence [267]–[272]. On the other hand, gradient-based molecular optimization approaches are based on DL architectures, such as VAEs, GANs, RNNs, and RL [273]–[276]. These models, once trained on large datasets of chemical structures, are capable of sampling the learned chemical space.

Regarding the level of specificity of molecular structure generation, gradient-free tools can be divided into three main groups: atom-based, fragment-based, and reaction-based. Atom-based approaches act by applying simple atom/bond level operations such as adding, removing, and replacing individual atoms or bonds from the molecules. In theory, these types of operations are able to generate every possible structure resulting in high novelty and diversity of the generated molecules. However, these approaches can produce invalid or less accessible structures, so it is essential to control these properties during the generation process.

Fragment-based approaches use fragmentation schemes and retrosynthetic disconnections to generate new molecules by adding, replacing, or removing groups of atoms. The novelty and diversity of the resulting molecules depend on the initial set of fragments, and their size and number can control the level of space exploration. However, like atom-based approaches, the validity and synthetic accessibility of the molecules can still be problematic.

Reaction-based approaches use libraries of *in silico* chemical reaction rules to generate new molecules by applying them to a set of reactants. This approach can produce highly novel and diverse molecules in fewer steps by making significant structural changes. However, the effectiveness of this method can be limited by the small number of available reaction templates, which may hinder its ability to explore unknown parts of chemical space and reduce the diversity of the generated molecules. Both atom, fragment, and

Table 15: Examples of the different gradient-free atom, fragment, and reaction-based methods.

Method	Molecule construction method	Evolutionary technique
Kawai *et al.* [277]	Atom-based	Genetic Algorithm
iSyn [278]	Reaction-based	Genetic Algorithm
GB-GA [279]	Atom-based	Genetic Algorithm
MolFinder [269]	Atom-based	Conformational Space Annealing
AutoGrow4 [270]	Reaction-based	Genetic Algorithm
EvoMol [280]	Atom-based	Genetic Algorithm
LEADD [271]	Fragment-based	Genetic Algorithm
ChemGE [281]	Fragment-based	$(\mu + \lambda)$ Evolutionary Strategy
MSO [268]	Atom-based	Particle Swarm Optimization
MOARF [282]	Fragment-based	Multi-objective Evolutionary Algorithm
CReM [283]	Fragment-based	Stochastic exploration

reaction-based gradient-free approaches have been demonstrated to be effective in many studies (Table 15).

In this study, we introduce ReactEA, a modular and problem-agnostic evolutionary CADD approach that utilizes biochemical reaction rules to manipulate molecules. To optimize user-specified objective functions, a suite of EAs from the jMetalPy framework [284] is employed. The initial population of seed molecules is used to generate a new population, and the fitness of these new molecules is calculated using the objective functions. The top-scoring molecules then proceed to the next generation, and this process is repeated until a stopping criterion is met. ReactEA was used to optimize existing inhibitors and design novel molecules with various objectives, such as drug-likeliness, synthetic accessibility, similarity to a target molecule, and docking affinity against the Poly [ADP-ribose] polymerase 1 (PARP1) enzyme.

5.2 Materials and Methods

5.2.1 Biochemical Reaction Rules

In computational chemistry, Reaction Rules are templates that encode the conversion of reactants into products. They identify substructures in the reactants based on a given chemical reaction scheme, enabling researchers to evaluate potential reactions and generate new product molecules. This process can easily be automated using several chemoinformatics tools like RDKit [189]. Figure 22 shows an example of an aromatic decarboxylation reaction rule (Fig. 22 A) and two reactions used to create the rule (Fig. 22 B).

Although the ReactEA framework is compatible with any set of reaction rules, in this work we focus specifically on exploring rules encoding biochemical transformations to enable the exploitation of sustainable cell factories for the production of natural drugs. The used reaction rules were sourced from two primary databases: the RetroRules database [27] and the MINE Databases [285]. RetroRules includes 350,224 reaction rules, covering over 15,000 biochemical transformations. These rules vary in enzyme

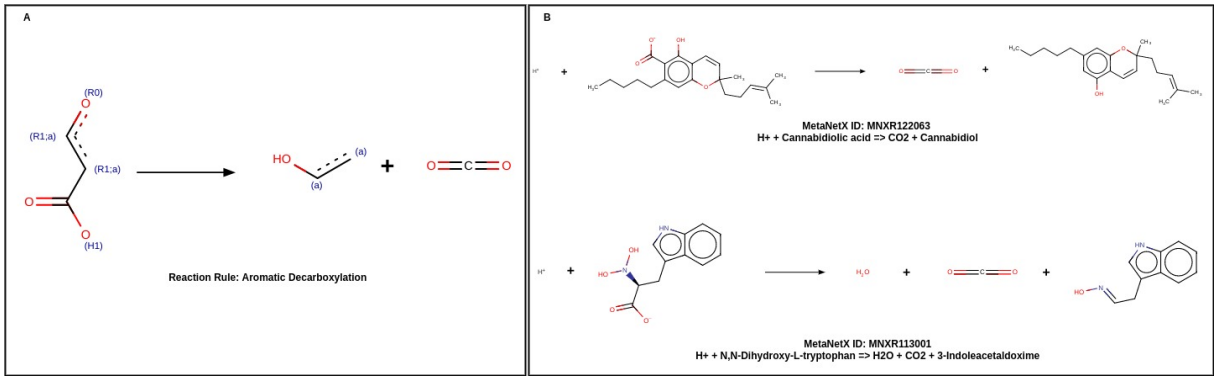


Figure 22: Reaction rule encoding an Aromatic Decarboxylation transformation (A) and original reactions (B), in which a carboxylic acid group (-COOH) is removed from the compound.

specificity and consider the atomic environment around the reaction center at different diameters (2 to 16, increments of 2). By predicting *de novo* reactions of promiscuous enzymes, this approach expands natural chemical diversity. In this study, only 13,035 reaction rules that can be expressed in the forward direction and with a diameter of 2 (the most permissive) were used.

The MINE databases include known and predicted metabolites generated with a comprehensive set of 9,914 hand-curated reaction rules. The reaction rules from MINE were combined with those from RetroRules to create a comprehensive set of 22,949 biochemical reaction rules. These rules are represented in the SMARTS notation, which can be used by open-source chemoinformatics tools. The validity of each rule was confirmed with the RDKit chemoinformatics toolkit.

5.2.2 Framework Overview

The general workflow starts with the definition of the problem, which includes defining the objective functions that are being minimized/ maximized, the constraints that the solutions must satisfy (if any), and the representation of the solutions. This determines the search space that the algorithm will explore and the criteria by which the solutions will be evaluated.

ReactEA is designed to optimize molecular properties by exploring the chemical space of NPs using biochemical reaction rules. The framework starts by receiving the molecular properties to optimize and an initial population of molecules that will serve as the seed for the next generations. First, the initial population of molecules is initialized and evaluated. At every generation, a selection operator retrieves the mating pool from the solution list (the population) for reproduction. A mutation operator is then applied to yield a new list of solutions (the offspring). The solutions of this offspring population are evaluated using the specified objective functions, and a replacement strategy is applied to update the population for the next generation. This process is repeated until a certain stopping criterion is met (e.g. maximum number of generations or mean fitness of all individuals). A general overview of the ReactEA framework is presented in Figure 23.

ReactEA is highly modular and extensible, making it easy for users to customize and extend its functionalities. It provides an interface for molecular property optimization using single-objective (SO) and

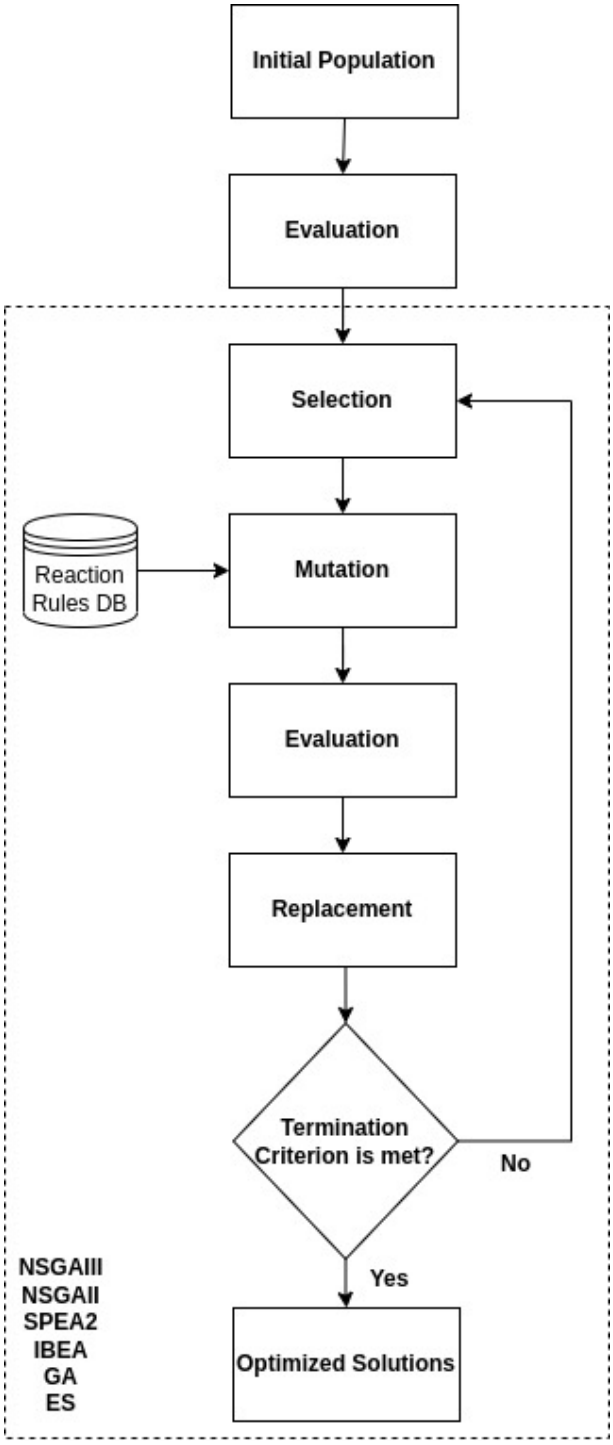


Figure 23: General outline of the ReactEA workflow.

multi-objective (MO) EAs. This allows users to quickly and easily experiment with different configurations without having to write complex code. Additionally, ReactEA includes a wide range of pre-defined reaction rules that serve as mutation operators to generate diverse molecules with optimized properties.

5.2.3 Initial population

The initial population serves as the starting point for the search for optimal solutions. Its definition plays an important role in determining the diversity and novelty of the compounds generated. A diverse initial population allows for a wider exploration of potential solutions, increasing the chances of finding multiple optimal solutions. In ReactEA, the initial population is defined by the user. This can be, for instance, a set of available precursors in a particular organism, a diverse set of molecular fragments for a *de novo* design experiment, or known ligands for lead optimization. The framework accepts molecules represented as SMILES strings [76], which are then transformed into RDKit Mol objects. All molecular operations, including the application of the Mutation operator, are performed using RDKit.

5.2.4 Population Fitness

In an EA, the fitness of a solution is a measure of its ability to solve the problem at hand, being determined by evaluating its performance on pre-defined objective functions.

ReactEA can be used in both SO and MO problems. For that, a set of predefined objective functions, such as octanol–water partition coefficient (logP), drug-likeliness, molecular weight range, number of large rings, and stereoisomer count are provided. However, its modular design allows for the optimization of virtually any objective function, allowing those with computational background to easily implement their own objective functions of interest.

To implement a custom objective function, the user needs to implement a function that calculates some property of interest, specify if it is a maximization or minimization problem, and provide the worst possible fitness for invalid molecules. Additionally, MO functions can be combined into a single one through the use of a weighted aggregated sum. In this case, a list of weights for each objective function must also be provided.

5.2.5 Selection Operator

The selection operator is used to choose which solutions will be used to create the next generation of solutions. There are different methods that can be used for selecting the mating population in EAs. ReactEA supports a wide range of selection operators as provided by the jMetalPy framework including a variety of Random, Rank, Roulette, and Tournament selection strategies. The best selection method to use depends on the specific problem being solved, while the different algorithms also work better with different selection schemes.

5.2.6 Mutation Operator

The Mutation Operator in ReactEA introduces variations into the population by creating novel compounds from the existing ones, by utilizing *in silico* biochemical reactions, executed using RDKit. Given a parent molecule, the process starts by randomly selecting a reaction rule from a predefined set of rules and attempting to apply it to the parent. This step is repeated until a successful reaction occurs or until a

maximum number of attempts is reached. The resulting product molecules (offspring) are evaluated, and one of the most similar ones, within a specified range, is selected. This strategy avoids selecting byproducts or undesirable reaction outcomes, such as water or carbon dioxide molecules.

An illustration of the mutation operation is presented in Figure 24, where three different reaction rules are applied to a glycerol molecule, resulting in different product molecules. This approach ensures the validity and higher synthetic feasibility of the generated molecules, while also providing the ability to track the lineage of any mutant molecule to identify its origin and propose potential synthetic routes.

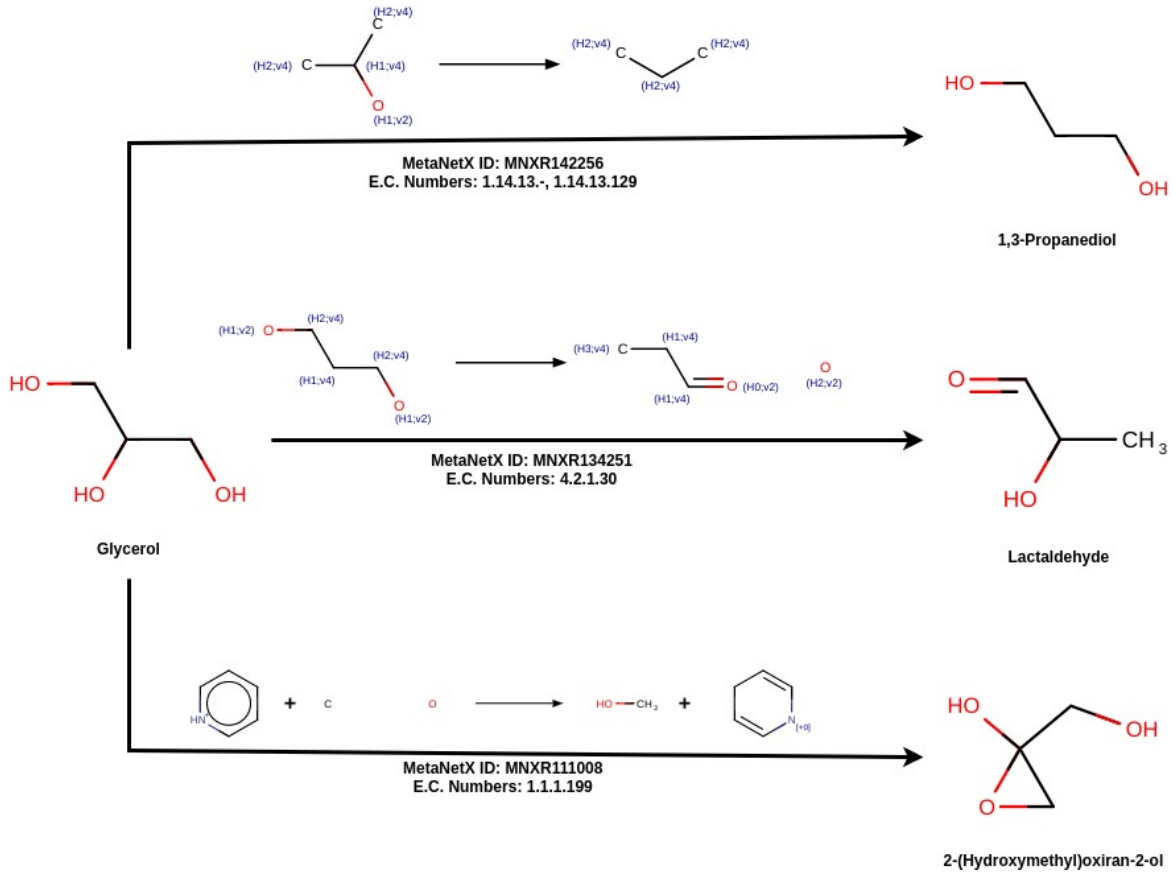


Figure 24: This is an example of how the mutation operator works: three different reaction rules are applied to create three distinct products (mutants) from a single reactant (parent). It is possible that different rules apply to the same compound, and one rule can also generate multiple different products.

5.2.7 Replacement of solutions

In an EA, the replacement of solutions refers to the process of updating the current population with new solutions. The replacement strategy used can vary based on the specific problem being solved and the desired outcome of the algorithm. ReactEA offers a range of replacement strategies for its various EAs. Some algorithms replace the entire population with a new set of solutions generated from the current population, which helps to maintain diversity and avoid being trapped in a local optimum. On the other hand, some strategies only replace a portion of the population, striking a balance between exploring new

solutions and exploiting the best solutions found so far. More complex algorithms may involve complex variations of these strategies. In addition to these strategies, some EAs utilize the concept of elitism, preserving the best-performing solutions from one generation to the next. This helps to ensure that the best solutions are not lost.

5.2.8 Termination Criterion

The termination criterion determines when the algorithm comes to a halt. The criterion ensures that the EA produces results within a reasonable time frame. The specific criterion used will vary depending on the optimization problem and the requirements of the application. When choosing a termination criterion, it is crucial to balance exploration of the search space and finding good solutions within a reasonable time. The criterion should be well-defined and easily evaluated during the course of the algorithm. ReactEA implements several criteria, including a maximum number of generations, a maximum run time, reaching a predetermined quality indicator, and a mean value of fitness of all objectives. The termination module can be easily extended to implement other termination criteria as needed.

5.2.9 Evolutionary Algorithms

Choosing the right EA to tackle a specific problem is not straightforward and requires careful consideration. The algorithm's effectiveness depends on various factors such as the problem's complexity, the size of the search space, and the design of the objective functions. Whether the problem requires SO or MO optimization or if the objective functions are continuous or discrete, different algorithms may perform better. Therefore, it is advisable to try out multiple algorithms and determine the most suitable one for a particular case.

ReactEA offers a comprehensive collection of EAs and some stochastic optimization algorithms for molecular property optimization: simulated annealing (SA) [286], GA [287], ES [133], and local search (LS) [288] for SO optimization, and Non-dominated Sorting Genetic Algorithm III (NSGAIII)) [131], Non-dominated Sorting Genetic Algorithm II (NSGAII) [130], Indicator-based Evolutionary Algorithm (IBEA) [134], and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [289] for MO optimization. Although designed for many objectives, MO EAs can also benefit SO optimization by improving convergence and exploration of the search space [290]. This approach is especially advantageous for complex problems that require a diverse set of solutions to cover different regions of the search space.

5.2.10 Development Environment

ReactEA was developed using Python version 3.8. Molecular operations like SMILES validity, standardization, reaction SMARTS validity, and reaction product generation were done using RDKit version 2022.03.1. EAs were implemented with jMetalPy version 1.5.5. Source code, small data files, and usage examples are available at `https://github.com/BioSystemsUM/ReactEA`. For reproducibility, all data and

code used to generate the presented results are also available at `https://zenodo.org/record/7630352`.

5.3 Results and Discussion

5.3.1 Optimization of Drug-Likeness, Solubility, and Synthetic Accessibility

Evaluating the performance of *de novo* design methods often involves testing them on simplified tasks, such as maximizing quantitative estimate of drug-likeness (QED) or logP. These objectives are straightforward to calculate and demonstrate the ability to generate molecules that meet specific goals. While they do not reflect the complexity of real-world drug discovery experiments, they can still serve as good indicators to identify compounds that are more likely to be successful drugs.

In this study, we compared the performance of ReactEA in optimizing simple tasks such as the QED, the penalized octanol–water partition coefficient (pLogP), the synthetic accessibility score (SAS), and the ChEMBL-Likeness Score (CLScore) with some state-of-the-art methods. The QED score, calculated using RDKit and ranging between 0 and 1, indicates how closely a compound resembles known drugs in terms of physical and chemical properties. A higher score suggests better drug-like properties, including oral bioavailability, low toxicity, and the ability to pass through the blood-brain barrier.

The logP value measures the solubility of a molecule in water versus lipids, with positive values indicating lipophilicity and negative values indicating hydrophilicity. In drug discovery, logP is crucial for determining the absorption, transportation, and distribution of drugs. Lipinski's Rule of 5 states that for optimal oral and intestinal absorption, drugs must have a logP value below 5, with a preferable range of 1.35 to 1.8. In drug design studies, pLogP is often optimized instead, which is a penalized version of logP that accounts for large rings and synthetic accessibility (Equation 5.1). The pLogP score was calculated using RDKit and normalized across the ZINC dataset.

$$pLogP(m) = logP(m) - SA(m) - RingPenalty(m) \tag{5.1}$$

Synthetic complexity is an estimate of how difficult it would be to synthesize a molecule. We used the SAS metric proposed by Ertl and Schuffenhauer [291] to predict synthetic accessibility. SAS takes into account the molecule's similarity to reference drug-like compounds and the presence of synthetically complex features such as unusual rings and many stereo centers. SAS ranges from 1 to 10, with higher values indicating synthetic complexity. We also calculated a normalized SAS score, ranging from 0 to 1. RDKit was used to calculate the SAS scores.

The CLScore [292] is another metric that predicts synthetic accessibility. It compares a molecule to a subset of biologically active compounds in ChEMBL [293] using circular substructures called molecular shingles. Each shingle is assigned a weight based on its frequency of occurrence in the ChEMBL subset, and the sum of these weights is divided by the total number of shingles in the molecule to calculate a

score. Higher scores indicate greater similarity to ChEMBL molecules. RDKit was used to calculate the CLScore scores.

The initial population consisted of a set of precursors from the *Escherichia coli* iJO1366 metabolic model [251], retrieved from the RetroPathRL GitHub repository [58]. Only molecules with valid InChI [135] were considered, resulting in a set of 648 precursors. The validity and conversion to SMILES notation were performed using RDKit.

We run GA, ES, NSGAIII, NSGAII, SPEA2, and IBEA with default parameters for 100 generations or until no improvements were observed for 5 generations, using the Ecoli Sink Set as the initial population. The results showed similar performance to state-of-the-art methods under comparable conditions. Direct comparisons are difficult due to differences in the conditions used by different methods (e.g. number of generations, restrictions on molecular size), thus results were compared to a limited set of studies reported in [280]. Fair comparisons were made by limiting the maximum number of heavy atoms to 38 in the pLogP optimization.

In the QED optimization, the ReactEA achieved a score of 0.948, comparable to other best-performing methods [268], [280], [294], [295], except for the method proposed by Zhang *et al.* [274] with a score of 0.954. The regular and normalized pLogP optimization resulted in maximum scores of 13.88 and 11.19, respectively, which were outperformed only by EvolMol [280]. The SAS optimization resulted in the best possible score of 1.0, outperforming scores of 0.95 reported in [280] and [208]. In the CLScore optimization, a top score of 6.78 was achieved, outperforming the score of 6.552 reported in [280]. The complete set of results is available in Supplementary Table 31.

It is important to note that optimizing these metrics does not reflect the complexity of real-world drug discovery experiments. For instance, the best molecule obtained in the pLogP optimization was a non-drug-like compound consisting of 38 carbon atoms. While the best-scoring molecules in the QED optimization display desirable drug-like properties, optimizing this metric alone may result in unrealistic molecules as it does not consider synthetic accessibility, pharmacokinetics, and target activity. The generated molecules from the SAS and CLScore optimizations are quite simple, mostly consisting of small alkanes and cycloalkanes. In the case of CLScore, it may be more meaningful to optimize values between 3 and 5, as they correspond to the peak in the ChEMBL distribution. These limitations suggest that objective functions must be carefully designed and that MO optimization should be integrated into drug discovery studies, along with additional targeted objectives.

5.3.2 Similarity to Aspartame

Designing novel molecules based on existing bioactive ones, or incorporating specific core structures and scaffolds is a common strategy in molecular design. This approach allows for the preservation of desirable properties while exploring new areas of chemical space, simplifies synthesis by using the core structure or scaffold as a template, and increases the likelihood of obtaining bioactive molecules due to their pre-determined structural features.

Table 16: Impact of different initial populations on the performance of ReactEA. The percentage of unique and novel molecules, the similarity between the molecules, the mean and best similarity to the aspartame, and the QED and SAS are showcased.

Set	Unique/ Novel	Internal Sim.	Mean Sim. to Aspartame	Best Sim. to Aspartame	QED	SAS
ChEMBL Representative	100%	0.457	0.598	0.803	0.436	3.030
ChEMBL Similar	100%	0.500	0.642	1.000	0.452	2.953
ChEMBL Top100	100%	0.510	0.653	1.000	0.447	2.971
ChEMBL Worst100	100%	0.506	0.639	1.000	0.447	2.971
Scaffolds Representative	100%	0.487	0.623	0.828	0.445	3.004
Scaffolds Similar	100%	0.508	0.645	1.000	0.428	2.973
Scaffolds Top100	100%	0.476	0.630	0.869	0.450	2.991
Scaffolds Worst100	100%	0.519	0.649	1.000	0.447	3.015

In order to assess the ability of ReactEA in finding paths to target molecules, we evaluated its performance in optimizing for the similarity to the Aspartame molecule. The similarity was defined as the Tanimoto Similarity [256] between Morgan fingerprints with radius 2 and 1024 bits. Additionally, to understand the impact of the initial population on the performance of ReactEA, we conducted experiments using the GA. We tested 8 different initial populations, each consisting of 100 molecules, including NP-based approved drugs from ChEMBL and scaffolds from known NPs.

The molecules from ChEMBL were selected by choosing NP-based approved small molecule drugs with molecular weights between 100 and 399, resulting in 184 valid molecules. We created 4 different initial populations from this set. For the first set, called "Representative ChEMBL", we selected the 100 most representative molecules by computing the Morgan fingerprints of each molecule, then reducing the dimensionality to two components using t-SNE. Next, using KMeans clustering with 100 clusters, we selected the molecule closest to the centroid of each cluster. For the second set, called "Similar ChEMBL", we selected 100 molecules from the same cluster by following a similar approach as for the Representative ChEMBL set, except that only 2 clusters were computed. The remaining two sets, "ChEMBL Top100" and "ChEMBL Worst100", consist of the 100 molecules with the best and worst similarity to Aspartame, respectively.

The set of 368 NP scaffolds was retrieved from the study by Lai et al. [296]. We also created 4 different initial populations of 100 molecules each using the same approach as for the ChEMBL data, and these sets are referred to as "Scaffolds Representative", "Scaffolds Similar", "Scaffolds Top100", and "Scaffolds Worst100".

For each initial population, the algorithm was run over 100 generations and repeated 10 times. The selection operator was Binary Tournament Selection, and the replacement of solutions was done by selecting the 100 fittest unique solutions among the previous solutions and offspring.

It is usually advantageous to start with a diverse initial population as this can increase the chances of exploring a wider search space. Despite this, ReactEA has demonstrated the ability to quickly improve

solutions even with a non-diverse initial population. In our experiments, we found no significant differences in the average score of generated molecules when using different initial populations (Table 16, Supplementary Figure 30). This can be associated with the fact that applying reaction rules to molecules can drastically change their score. The similarity to Aspartame scores ranged between 0.598 and 0.653. Interestingly, the initial population with the lowest average score was the Representative ChEMBL, which suggests that the diversity of the initial population does not significantly impact the final solutions. With the exception of the Representative ChEMBL, Representative Scaffolds, and Scaffolds Top100 sets, ReactEA was able to reach the Aspartame molecule or one of its stereoisomers (Supplementary Figure 29).

In addition to evaluating the similarity of the generated molecules to Aspartame, we also assessed their novelty, diversity, QED, and SAS. We found that all generated molecules were unique and distinct from the starting population. As expected, the internal similarity was high, with the generated molecules sharing many substructures. The average QED and SAS values of the generated molecules were comparable to those of Aspartame.

5.3.3 Docking to PARP1

The PARP1 enzyme plays a crucial role in DNA repair, cell signaling, and gene regulation. It was the first member of the Poly (ADP-ribose) polymerase (PARP) family of enzymes, which add ADP-ribose groups to target proteins, to be identified. PARP1 primarily mediates multiple DNA damage repair pathways, which repair DNA damage caused by various internal and external factors. If left unrepaired, this damage could lead to the development of several diseases, including cancer. Therefore, the inhibition of PARP1 is being explored as a treatment option for various cancers, such as ovarian, breast, and prostate cancer.

To show how ReactEA can be used for lead optimization, we used a library of 94 seed molecules, including 11 known PARP inhibitors (retrieved from `http://www.clinicaltrials.gov`) and 83 molecular fragments derived from them using Breaking of Retrosynthetically Interesting Chemical Substructures (BRICS) decomposition [297] by the authors of AutoGrow4.

Our objective function uses the DOCKSTRING package [298] for optimizing the binding affinity against the PARP1 enzyme. DOCKSTRING is a user-friendly Python wrapper of the AutoDock Vina package [299]. AutoDock Vina produces high-quality docking poses and reasonable binding free energy predictions while having a relatively low computational cost.

To ensure the generated molecules had desirable physical and chemical properties, we applied a set of molecular filters before docking. If a compound did not pass all the filters, it was assigned the worst fitness. These filters ensured that the molecules had a logP value between -0.4 and 5.6, a molecular weight between 160 and 500, a molar refractivity between 40 and 130, and between 20 and 70 heavy atoms.

The best-scoring molecule, along with the reaction rules used to generate it, is displayed in Figure 25. It has a predicted binding affinity of -14.5 kcal/mol, considerably better than the best-known inhibitor in the initial population (Olaparib with -12.3 kcal/mol). The molecule was generated with the GA. Other EAs also had comparable performance with the best binding affinities ranging from -12.5 kcal/mol to -14.5

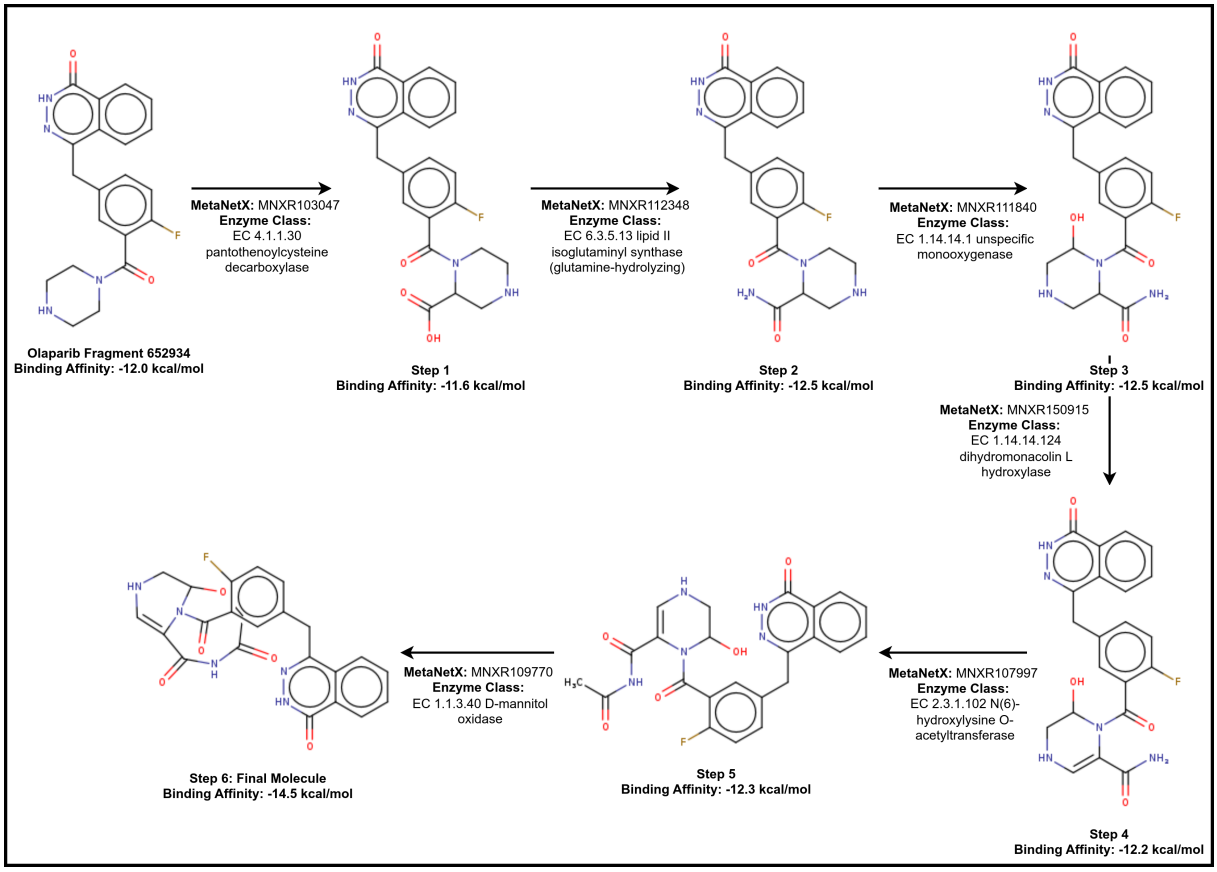


Figure 25: The molecule with the highest binding affinity and the reaction templates that were applied to transform the starting compound (Olaparib Fragment 652934) into the final molecule are presented, along with the intermediate molecules and the corresponding EC numbers for the original reactions.

kcal/mol (Table 17). The binding affinity of the best molecule of each EA at each generation and the best molecules generated by each EA can be consulted in Supplementary Figures 31 and 32.

The predicted binding affinities could easily be improved by simply raising the number of generations. However, doing so would come with a number of drawbacks. For one, AutoDock Vina and other docking tools tend to favor larger molecules [300]. Additionally, longer runs may lead to the emergence of unwanted functional groups that have the potential to be mutagenic or have unfavorable pharmacokinetic properties. Similarly, the synthesizability of compounds tends to decline in later generations as the accumulation of mutations causes the population to significantly deviate from the original molecules. Therefore, in general, a lower number of generations is preferred when optimizing molecular docking. However, these challenges can be addressed by using a MO optimization approach, such as the one offered by ReactEA, that can consider multiple objectives, such as binding affinity to the target, molecular weight, synthetic accessibility, and other relevant factors.

Table 17: Results of the Docking to PARP1 optimization. The binding affinities of the initial population and the performance of the different EAs are shown.

EA	Worst	Best	Mean	Std. Dev.
Init. Pop.	50.0000	-12.1000	35.277660	25.8735
NSGAIII	-11.1000	-14.4000	-11.769149	0.5998
NSGAII	-10.6000	-12.5000	-11.198936	0.4759
SPEA2	-10.7000	-12.7000	-11.427660	0.5284
IBEA	-10.9000	-13.5000	-11.547872	0.5303
GA	-10.7000	-14.5000	-11.378723	0.6341
ES	-11.5000	-13.1000	-11.954255	0.3935

5.4 Conclusions

We have introduced ReactEA, a new open-source reaction-based EA framework for molecule generation. ReactEA can be used in SO and MO problems, optimizing any (set of) measurable objective functions with its large set of biochemical reaction rules. This allows for the generation of valid structures and assures chemical feasibility, with potential synthesis paths provided for the generated molecules.

Compared to previous reaction-based EAs, such as Autogrow4 and iSyn, which use a limited set of click chemistry rules, ReactEA utilizes a more comprehensive set of reaction rules to enumerate possible biosynthetic routes connecting target molecules to precursors. This represents a step forward towards a more sustainable, economically viable, and environmentally responsible approach to chemical production. Our results demonstrate ReactEA's high configurability and versatility, achieving excellent results in optimizing simple objectives like QED or SAS, as well as more complex tasks like similarity to a target molecule and docking to proteins.

One of the limitations of ReactEA and other reaction-based approaches is their dependence on the available reaction rules and their coverage of the chemical space. While ReactEA uses a comprehensive set of biochemical reaction rules, their number still limits the chemical space that can be explored, potentially reducing the novelty and diversity of the generated molecules. Additionally, the reaction rules are limited to human knowledge, which may limit the exploration of new areas of the chemical space. Despite its relatively low computational demands, evaluating the results provided by ReactEA requires chemical expertise as the quality of generated molecules is currently either subjective or measured imperfectly.

6

BioCatalyzer: a rule-based tool for compound metabolism prediction and identification[1]

6.1 Introduction

Metabolism is a fundamental biological process essential for the survival and thriving of living beings. It encompasses countless chemical reactions that convert molecules within an organism. These reactions facilitate the breakdown of nutrients to generate energy, the synthesis of essential compounds, and the elimination of waste products. Without these processes, organisms would be unable to grow, develop, or maintain their internal balance.

An important part of metabolism involves transforming xenobiotics, substances like environmental toxins, pollutants, and drugs that are not naturally found in the body. These substances constitute the largest portion of the human chemical exposome, with the vast majority remaining either unknown or largely uncharacterized [301]. These metabolites often result from promiscuous or nonspecific enzymatic reactions [302], microbial or gut metabolism [303], and phase I and II metabolism [304].

Understanding how xenobiotics are metabolized, distributed, and eliminated is critical in determining the impact of those chemicals and their metabolites on the human organism. This is particularly vital for pharmaceutical drugs, as how they are metabolized can greatly affect how well they work and how safe they are. Drug metabolism is mainly carried out by specialized enzymes, with the cytochrome P450 family playing a central role. These enzymes perform a variety of reactions like oxidation, reduction, and hydrolysis, which change drug molecules to make them easier to dissolve and eliminate. However, drug metabolism is not consistent, it is affected by factors such as genetic differences, diet, and interactions with other substances. Recent studies have also shown that the gut microbiome can significantly impact how drugs are processed [305], [306]. These microbes can modify drug availability and effectiveness, making it more challenging to predict how drugs will be metabolized.

Predicting drug metabolism is thus a challenging yet essential task in pharmacology. Accurate predictions are crucial for understanding drug behavior and minimizing adverse effects. However, characterizing

[1] At the time of writing, this chapter is being prepared for submission to a peer-reviewed scientific journal.

or identifying drug metabolites presents significant challenges. Traditional analytical methods can be time-consuming and may not capture the full spectrum of metabolites produced *in vivo*. These methods often rely on techniques such as MS, liquid chromatography-mass spectrometry (LC-MS) or nuclear magnetic resonance (NMR) spectroscopy, which, while highly sensitive, are limited by their ability to detect only known metabolites or those that match existing reference spectra [307].

Furthermore, the integration of computational predictions with experimental data, such as MS results, remains a significant hurdle. One of the key challenges lies in accurately matching predicted masses with experimental MS data. MS provides precise mass-to-charge (m/z) ratios of metabolites, but translating these readings into specific molecular structures requires robust computational algorithms and comprehensive reference databases.

As a result, over the past decades, *in silico* strategies have been increasingly used to predict the metabolism of xenobiotics, particularly drugs. Recent computational tools, such as BioTransformer [71], SMARTCyp [308], PreMetabo [309], and others [72], [310], [311], have been developed to predict biotransformation products. Most of these tools employ pattern recognition techniques, hand-made reaction templates, or ML algorithms and are often specialized for specific reaction types, such as phase I or phase II metabolism. Some tools are comprehensive, covering a wide range of biotransformations, while others focus on specific subsets. The majority are designed for mammalian metabolism, with only a few targeting microbial degradation. Additionally, most tools are commercial, with only a limited number freely available. Unfortunately, despite the increasing number of *in silico* metabolism prediction tools, significant limitations remain, particularly in terms of their performance, scope, and accessibility.

In response to these challenges, we introduce BioCatalyzer, a novel rule-based tool designed to predict compound metabolism and identify metabolites. BioCatalyzer employs a comprehensive set of biochemical rules to simulate metabolic transformations. These rules are derived from public databases, and they account for mammalian, plant, and microbial metabolism ensuring broad coverage of possible metabolic reactions. Additionally, BioCatalyzer can match predicted metabolites with experimental MS data, enabling the validation and refinement of its predictions.

This paper details the development and application of BioCatalyzer. We explore its key features, including the integration of gut microbiome interactions and the matching of predictions with MS data. Our contributions include the development of a comprehensive reaction template knowledgebase, the implementation of a versatile prediction tool, the validation of our predictions against experimental data, and the demonstration of BioCatalyzer's utility in identifying previously unknown metabolites. Additionally, we highlight the potential of BioCatalyzer to help the study of metabolic pathways, providing valuable insights for drug development and personalized medicine.

6.2 Materials and Methods

6.2.1 Biochemical Reaction Templates Knowledgebase

Our reaction template knowledgebase consists of 29,297 reaction rules. Each reaction rule acts as a template that describes how reactants convert into products, identifying specific substructures within the reactants based on a defined chemical reaction scheme. This enables researchers to evaluate potential reactions and their corresponding product molecules. These reaction templates have been widely utilized in computational tools to enumerate possible promiscuous reactions, aiding in the discovery of novel reactions.

The reaction template knowledgebase used in this study was retrieved from two major public databases. Around 18,698 reaction rules were sourced from the RetroRules database [27]. This database comprises 350 224 reaction rules, encompassing over 15,000 biochemical transformations. These rules vary in enzyme specificity and take into account the atomic environment surrounding the reaction center at different diameters, ranging from 2 to 16 in increments of 2. We only used valid reaction rules expressed in the forward direction with the most promiscuous diameter of 2. The remaining 10,599 reaction rules were retrieved from MINE [249] database.

Approximately 63% of the reaction rules include annotated EC numbers, covering around 8,421 unique organisms. Additionally, the dataset includes 117 spontaneous reactions. Most reaction rules from RetroRules follow a one-to-one format (one substrate to one product), whereas reaction rules from MINE are many-to-many.

These rules are encoded in SMARTS notation, ensuring compatibility with open-source chemoinformatics tools. The validity of each rule was verified using the RDKit chemoinformatics toolkit. Any duplicated reaction rules, those with missing or incorrect structural information, or those with unknown cofactors were removed. Cofactors, essential for many enzymatic reactions, act as key molecules for various enzyme classes and are involved in a wide range of reactions. Common cofactors like carbon dioxide and water were explicitly identified in the reaction rules. Additionally, common reactant-product pairs with donor-acceptor relationships were annotated as generalized cofactor pairs, such as ATP/ADP and GTP/GDP as phosphate donors and acceptors. Reactions solely involving cofactor transformations were excluded, as their reaction rules would not apply to other metabolites.

6.2.2 Gut Microbiome Drug Metabolism Data

Despite being a general-purpose metabolism prediction and identification tool, BioCatalyzer was tested and validated using data from the work of Zimmermann *et al.* [305] that focus on the metabolism of drugs in the gut microbiome.

In their study, Zimmermann and collaborators investigated how the gut microbiome contributes to the variability in individual responses to medicinal drugs. By examining 76 human gut bacteria from various groups and their ability to metabolize 271 orally administered drugs, they found that many drugs, about two-thirds, are chemically modified by at least one of these microorganisms. They were then able to

identify and validate 30 microbiome-encoded enzymes that collectively convert 20 drugs to 59 candidate metabolites. The data described below, which was used to test and validate our framework, was retrieved from their publication.

Drugs The 271 drugs used in their study were chosen to cover a wide range of chemical characteristics and clinical uses, encompassing diverse physicochemical properties and predicted concentrations in the intestines, except for antibiotics. These drugs belong to various classes such as antivirals (e.g., Zidovudine), beta-blockers (e.g., Atenolol), antidepressants (e.g., Bupropion), antipsychotics (e.g., Olanzapine), and statins (e.g., Atorvastatin). They are used to treat a wide range of conditions including infections (antivirals), cardiovascular diseases (beta-blockers), psychiatric disorders (antidepressants and antipsychotics), and lipid disorders (statins). They can cause side effects ranging from mild issues like dizziness or nausea to more severe reactions.

Gut Microbes The 76 bacterial species and/or strains used in their study were chosen to represent the major phyla of the human gut microbiome. These organisms play vital roles in digestion, metabolism, and maintaining gut health. Examples include probiotic *Bifidobacterium* species like *B. adolescentis* and *B. breve*, known for fermenting carbohydrates and producing beneficial metabolites. *Bacteroides* and *Clostridium* species that are abundant in the colon, contribute to the breakdown of dietary fibers and production of short-chain fatty acids. Besides their metabolic functions, these organisms can interact with drugs within the human gut. For instance, in the treatment of various cancers, the commonly used drug Irinotecan can induce severe diarrhea and other adverse reactions due to gut microbiota (e.g., E. coli, Staphylococcus, and Clostridium) converting the less toxic SN-38-glucuronide metabolite back into the highly toxic 7-ethyl-10-hydroxycamptothecin, which can severely damage the intestines [312].

Unknown Products of Bacterial Drug Metabolism To identify products of drug metabolism in the human gut, the authors conducted untargeted metabolomics analysis experiments, resulting in the discovery of 6,572 different drug–compound pairs for all 20,596 tested bacteria–drug interactions. Additionally, to systematically identify the structural targets of microbial metabolism, they performed functional group analysis on sets of drugs that shared a specific mass difference with respect to their metabolite. This approach revealed some drug products undergoing conserved metabolic transformations such as deacetylation and hydrogenation. However, most metabolic transformations remained unidentified, highlighting the complexity and diversity of microbial drug metabolism in the human gut. The dataset of 6,572 drug-compound pairs and their respective mass differences offers valuable information for further analysis aimed at uncovering additional drug metabolism products.

6.2.3 Framework Overview

BioCatalyzer comprises two primary modules: one for metabolism prediction and another for metabolite identification. These modules are designed as command-line tools that can be utilized independently or together in a sequence. Both tools were implemented using the Python programming language. The

command line tools are compatible with the latest versions of Linux, Windows, and MacOS. To perform molecular operation it uses the open-source chemoinformatics toolkit RDKit. To parallelize Pandas operations on all available CPUs it uses the pandarallel tool.

BioCatalyzer is freely available and can be conveniently installed through PyPI by running "*pip install biocatalyzer*". You can find it on GitHub at `https://github.com/BioSystemsUM/BioCatalyzer/`.

6.2.3.1 Metabolism Prediction Tool

The workflow of the BioCatalyzer metabolism prediction tool is very simple. The tool takes as input a set of compounds represented by their SMILES strings and will make sure the provided SMILES strings correspond to valid molecular structures. These compounds will serve as the reactants, i.e., the compounds that we want to predict the metabolism for. For each compound, we try to apply each one of the reaction rules from our knowledgebase. Finally, the predicted compounds are validated and the results are saved in a file for posterior analysis. This information is then stored in a tabular file. The generated file contains information about the parent/original compound, the applied reaction rule and associated EC numbers, and the new product compounds.

The tool can be run from the terminal as follows:

```
bioreactor_cli <PATH_TO_COMPOUNDS> <OUTPUT_DIRECTORY> [--neutralize=<BOOL>]
    [--reaction_rules=<FILE_PATH>] [--organisms=<FILE_PATH>] [--
   patterns_to_remove=<FILE_PATH>] [--molecules_to_remove=<FILE_PATH>] [--
   min_atom_count=<INT>] [--n_jobs=<INT>]
```

Only the <PATH_TO_COMPOUNDS> and <OUTPUT_DIRECTORY> arguments are required, all the others are optional.

- *<PATH_TO_COMPOUNDS>* should point to a tab-separated file. This file must include a column named "smiles" containing the SMILES string representation of the compounds to be used, and another column named "compound_id" containing identifiers for each compound. Alternatively, compounds can be provided as a ";"-separated string of SMILES representations. In this scenario, random identifiers will be automatically generated for each compound;
- *<OUTPUT_DIRECTORY>* should be a path to a directory where the results will be stored. A file named "new_compounds.tsv" will be created within this directory, containing the predicted metabolites. This file contains the following columns:
  - "OriginalCompoudID": Contains the IDs of the parent/original compound;
  - "OriginalCompoudSmiles": Contains the SMILES representation of the parent/original compound;
  - "OriginalReactionRuleID": Contains the IDs of the applied reaction rule;
  - "NewCompoundCompoudID": Contains the IDs of the predicted compound products;

  - "NewCompoudSmiles": Contains the SMILES representation of the predicted compound products;
  - "NewReactionSmiles": Contains the Reaction SMILES of the applied transformation;
  - "EC_Numbers": Contains the annotated EC numbers for the applied reaction rule.

- *neutralize* defaults to False. When set to True, it neutralizes the compounds and predicted metabolites by adjusting their charge through the addition and/or removal of hydrogen atoms;

- *reaction_rules* should point to a tab-separated file structured as follows:

  - "InternalID": Contains the IDs of each reaction rule;
  - "Reactants": Lists the reactants involved in the reaction rule. Coreactants should be referenced by their ID as defined in the Coreactants file. The compound to match within the reactants should be denoted by the string 'Any'. The format must be: "coreactant1_id;Any;coreactant_id". The order in which the reactants and the compound to match are defined is relevant and specific to the reaction rule. If the reaction rules are mono-component (i.e., they do not involve additional coreactants), the format should be: "Any";
  - "SMARTS": Contains the SMARTS representation of the reaction rules;
  - "EC_Numbers": Contains the annotated EC numbers numbers associated with the reaction rules (if any);
  - "Organisms": Lists the organisms associated with the EC numbers numbers, represented by their KEGG identifiers. By default, our knowledge base of reaction rules is utilized unless explicitly specified otherwise;

- *organisms* should point to a tab-separated file that includes a column named "org_id" that lists the organisms' identifiers (KEGG identifiers). Alternatively, organisms can be provided as a ";"-separated string of their identifiers. By default, all organisms are kept;

- *patterns_to_remove* should point to a tab-separated file containing a column named "smarts" with the SMARTS representation of the patterns to remove. By default, our system utilizes a predefined set of SMARTS patterns that match common reaction byproducts, such as Coenzyme A, NAD, ATP, ADP, GTP, Adenosine-3'-5'-diphosphate, as well as molecules containing flavin compounds and heme groups;

- *molecules_to_remove* should point to a tab-separated file containing a column named "smiles" with the SMILES representation of the compounds to be removed from the analysis. These are specific compounds deemed irrelevant to the analysis. By default, our system includes a predefined set of 9 molecules such as pyruvate, acetate ion, diphosphate, and others;

- *min_atom_count* should be an integer that specifies the minimum number of heavy atoms a product must contain to be considered. By default, this value is set to 5;

- *n_jobs* should be an integer specifying the number of parallel jobs to run (-1 to use all available cores). By default, the system runs with only one core.

The BioCatalyzer metabolism prediction tool offers a straightforward yet powerful workflow for predicting metabolite transformations from input compounds specified by their SMILES strings. By leveraging our comprehensive knowledgebase of reaction rules, the tool systematically applies these rules to each compound to predict potential metabolites. Users can customize the tool's behavior through optional parameters such as neutralization of compounds, specific reaction rules, organism preferences, and exclusion criteria for patterns and molecules. Results are conveniently saved in a designated output directory for further analysis.

This tool facilitates the exploration of metabolic pathways, aiding researchers in understanding biochemical transformations and potentially discovering new metabolites relevant to various biological processes and applications.

6.2.3.2 Metabolite Identification Tool

The BioCatalyzer metabolite identification tool aligns predicted metabolic product masses with experimental MS data. This aids in identifying potential compounds formed from compounds metabolized within an organism, thereby providing insights into the possible mechanisms driving specific transformations. The tool takes as input a file containing the MS data along with the predicted compounds to match. Users can also set a tolerance level for mass matching, enabling more flexible alignments.

The tool can be run from the terminal as follows:

```
matcher_cli <MS_DATA> <COMPOUNDS_TO_MATCH> <OUTPUT_DIRECTORY> [--tolerance
    =<FLOAT>] [--n_jobs=<INT>]
```

The <MS_DATA>, <COMPOUNDS_TO_MATCH>, and <OUTPUT_DIRECTORY> arguments are required, while *tolerance* and *n_jobs* are optional.

- *<MS_DATA>* should point to a tab-separated file containing the following columns:
  - "ParentCompound": the parent/original compound identifier;
  - "ParentCompounSmiles": the SMILES representation of the compounds (optional);
  - "Mass": the mass of the compound.
- *<COMPOUNDS_TO_MATCH>* should point to a tab-separated file containing the predicted compounds to match with the MS data. The file should contain the same columns as those generated by the metabolism prediction tool;
- *<OUTPUT_DIRECTORY>* should be a path to a directory where the results will be stored. A file named "matches.tsv" will be created within this directory, containing the matched compounds. This file contains the following columns:
  - "OriginalCompoundID": Contains the IDs of the parent/original compound;

  - "OriginalCompoundSmiles": Contains the SMILES representation of the parent/original compound;
  - "ParentCompoundExactMass": Contains the exact mass of the parent/original compound;
  - "NewCompoundID": Contains the ID of the predicted compound product;
  - "NewCompoundSmiles": Contains the SMILES representation of the predicted compound product;
  - "NewCompoundExactMass": Contains the exact mass of the predicted compound product;
  - "MassDiff": Contains the mass difference between the parent/original compound and the predicted compound product;
  - "EC_Numbers": Contains the annotated EC numbers for the applied transformation.

- *tolerance* should be a value representing the mass tolerance to use when matching masses. Masses with a difference of mass ± tolerance will be counted as matches. Defaults to 0.02;
- *n_jobs* should be an integer specifying the number of parallel jobs to run (-1 to use all available cores). By default, the system runs with only one core.

6.2.3.3 Running both Tools

Although the metabolism prediction tool and metabolite identification tool are independent, they are designed to be used sequentially. The output from the metabolism prediction tool can be directly used by the metabolite identification tool without modification. However, if using the identification tool with other data, users may need to adjust their data to meet specific requirements. To run both tools in sequence—predicting compound metabolism and then matching the results with MS data—users need to run the following command:

```
biocatalyzer_cli <PATH_TO_COMPOUNDS> <OUTPUT_DIRECTORY> [--neutralize=<BOOL
    >] [--reaction_rules=<FILE_PATH>] [--organisms=<FILE_PATH>] [--
    patterns_to_remove=<FILE_PATH>] [--molecules_to_remove=<FILE_PATH>] [--
    min_atom_count=<INT>] [--match_ms_data=<BOOL>] [--ms_data_path=<
    FILE_PATH>] [--tolerance=<FLOAT>] [--n_jobs=<INT>]
```

The metabolism prediction tool can also operate independently of the metabolite identification tool by using the "biocatalyzer_cli" command-line interface and setting the *match_ms_data* parameter to False. If *match_ms_data* is set to True, both tools will be used, and the user must provide the *ms_data_path* argument. Figure 26 shows a general overview of BioCatalyzer.

6.2.4 Development Environment

BioCatalyzer was developed using Python version 3.8. Molecular operations like SMILES validity, standardization, reaction SMARTS validity, and reaction product generation were done using RDKit version 2022.3.5. Source code, small data files, and usage examples are available at `https://github.com/B`

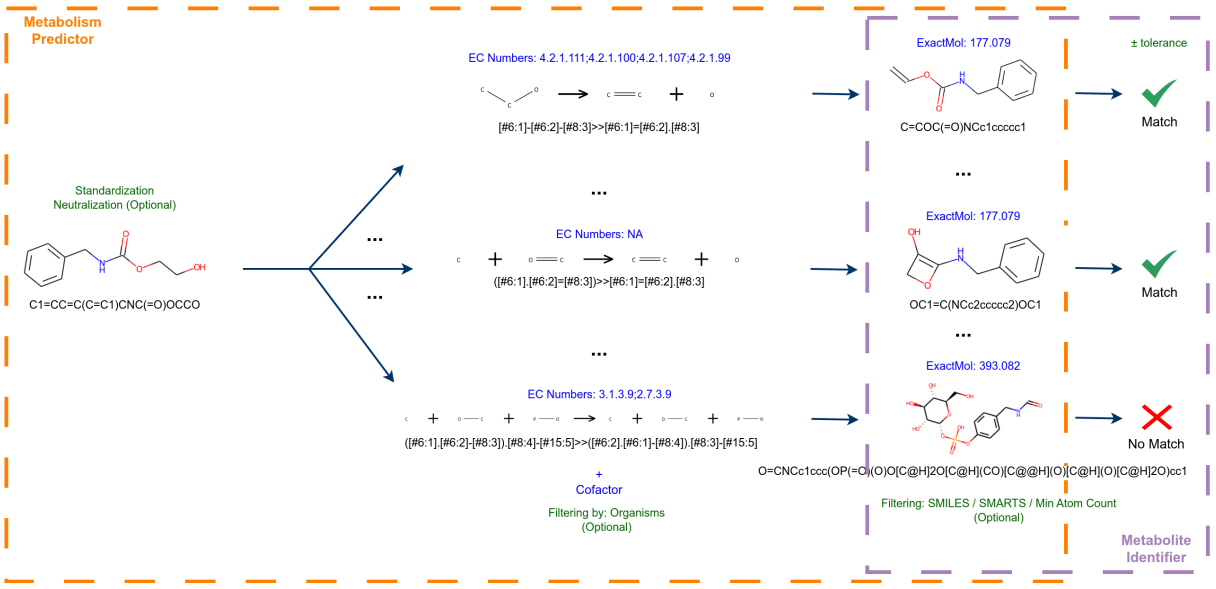


Figure 26: Overview of Biocatalyzer metabolism prediction and identification.

`ioSystemsUM/BioCatalyzer`. For reproducibility, all data and code used to generate the presented results are also available at `https://github.com/BioSystemsUM/BioCatalyzer-CaseStudy`. Results were generated in a High-Performance Computing Cluster on one node with 128 cores with a memory allocation of 32 GB.

6.3 Results and Discussion

To assess the performance of BioCatalyzer we tried to predict and identify single-step metabolism of 271 well-known drugs in the human gut microbiome (Zimmermann *et al.* data [305]).

Using the metabolism prediction tool we applied our knowledgebase of 29,297 reaction rules to the 271 drugs. This resulted in 580,820 potential new drug metabolism products from which 318,820 had annotated EC numbers. After that, we used our metabolite identification tool to match our predictions with the 6,572 different drug–compound pairs from the MS data from the study of Zimmermann *et al.* [305]. From the 580,820 predicted products, we got 39,848 matches, from which 25,588 had annotated EC numbers.

Out of the 271 drugs analyzed, we identified possible metabolism products for 214, leaving 57 drugs with no matches. Among the 6,572 different drug-compound pairs, only a small portion, 548 pairs representing approximately 8.3% of all possible matches, were matched. This means that the predicted 39,848 correspond to only 578 unique drug-compound pairs. This means that on average we have around 69 possibilities for each of the identified matches.

Table 18 lists the top 5 drugs with the highest match percentages. For example, Itraconazole, an antifungal medication used to treat fungal infections, had 5 suggested potential drug metabolites (Figure 27) out of the 9 identified drug metabolism products in the MS data. Of these 5 potential metabolites, 2 are known products of Itraconazole metabolism. Match 1, known as Keto Itraconazole, is a documented

metabolite of Itraconazole [313]. Keto Itraconazole inhibits P-glycoprotein, organic anion-transporting polypeptides 1B1 and 1B3, and the multidrug and toxin extrusion protein 1 [314]. Additionally, it is a potent *in vitro* inhibitor of CYP3A4 [315] and may contribute to the overall CYP3A4 inhibition observed *in vivo* if it circulates at sufficiently high concentrations. Match 4, with the IUPAC name 2-butan-2-yl-4-[4-[4-(4-hydroxyphenyl)piperazin-1-yl]phenyl]-1,2,4-triazol-3-one, is also a known compound. However, its activity has not been well studied.

Table 18: Top 5 drugs with the highest percentages of matches for all possible drug-compound pairs.

ParentCompound	ReportedMatches	UniqueMatches	Percentage
ITRACONAZOLE	9	5	0.56
MELPHALAN	6	3	0.50
METOCLOPRAMIDE	9	4	0.44
ANAGRELIDE	7	3	0.43
DICYCLOMINE	8	3	0.38

Match 3, a potential but previously unknown metabolism product of Itraconazole, was predicted by applying a reaction rule derived from the KEGG reaction R09527 and the Metacyc reaction RXN-11497. This rule is associated with the EC numbers 2.3.1.29, 2.3.1.194, and 2.3.1.180. Notably, the enzyme 2.3.1.180 is involved in the biosynthesis of straight-chain fatty acids through the type II fatty-acid biosynthesis system in plants and bacteria. This enzyme is found in human gut bacteria such as *Enterococcus faecalis* and *Escherichia coli* suggesting a possible interaction between Itraconazole and the gut microbiota.

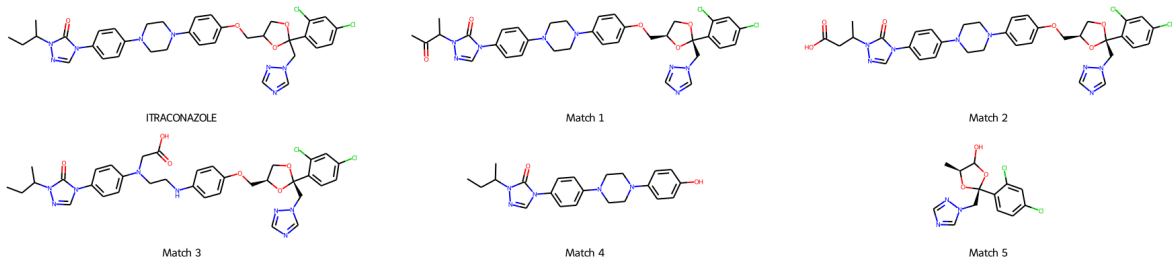


Figure 27: Metabolism of Itraconazole by the Human Microbiota: Identification of Five Potential Drug Metabolites.

As was previously stated, our tool can generate many matches for the same entry, i.e., many potentially different products for the same drug-compound pair in the untargeted MS data. Figure 28 shows one example of the metabolism of the same drug (Melphalan) by two different reactions resulting in two different compounds with the same monoisotopic mass. This redundancy can complicate data analysis by introducing ambiguity in identifying relevant metabolites. In addition, different reaction rules may be associated with distinct EC numbers and specific organisms, further adding to the complexity. To overcome these challenges, filtering reaction rules by organisms or EC numbers can provide a targeted approach to metabolite identification. This way, the tool can focus on relevant metabolic pathways and reduce the number of potential matches, thereby enhancing specificity in data analysis. However, this approach

may also limit the scope to known enzymatic activities and potentially overlook enzymatic promiscuity or homologous reactions that could produce relevant metabolites.

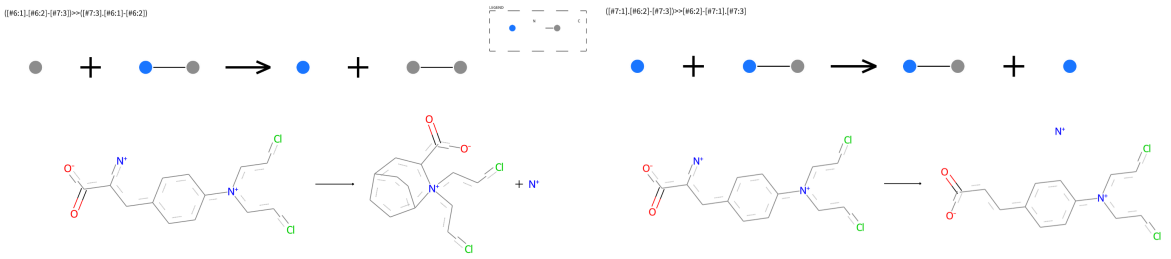


Figure 28: Predicted metabolism of the drug Melphalan by two different reaction rules produces two distinct compounds with the same mass. Reaction rules SMARTS were created using SMARTSview (`https://smarts.plus/`).

In general, we think that our tool can help in the identification of potential but previously unknown drug metabolites and associated transformations. This can help researchers enhance the understanding of drug metabolism pathways and their complexities within the human body. By uncovering new metabolites and their transformations, researchers can refine drug development strategies, improve therapeutic efficacy, and anticipate potential side effects earlier in the drug discovery process. The discovery of potential yet unknown drug metabolites and their transformations can help researchers enhance the understanding of drug metabolism pathways and their complexities within the human body. This can help optimize drug development strategies, enhance therapeutic effectiveness, and predict potential side effects earlier in the drug discovery phase.

We also examined the mass differences between each identified drug-specific metabolite and its original drug to further investigate the microbial drug metabolism chemistry. Notably, these variances were not random, rather, they revealed consistent, discrete mass differences, suggesting shared metabolic transformations across different drugs. Specific increases and decreases in mass further suggest conserved chemical modifications, with detailed examples provided in Table 19.

Table 19: Matched mass shifts associated with known transformations.

MassDiff	Transformation	Matches	Total	Percentage
-2.016	oxidation (−H2)	12	23	0.52
-15.995	reduction (−O)	7	12	0.58
-42.011	deacetylation (−C2H2O)	8	13	0.62
+2.016	hydrogenation (+H2)	11	16	0.69
+15.995	oxidation (+O)	22	39	0.56
+18.010	hydroxylation (+H2O)	7	12	0.58
+42.011	acetylation (+C2H2O)	5	7	0.71
+56.026	propionylation (+C3H4O)	1	7	0.14

Our tool has shown promising capabilities in proposing plausible chemical structures that could explain specific mass shifts, especially those related to known transformations. However, a significant

limitation arises in cases where no matches were found for other mass shifts. This discrepancy highlights a critical aspect of our current methodology: its dependency on annotated reaction rules that are derived from known biochemical transformations. This could be related to the concept of "dark matter" in metabolomics which refers to the vast portion of chemical space that remains unexplored or poorly understood, encompassing metabolic transformations, pathways, and metabolites that are yet to be characterized or even detected experimentally. This "dark matter" is not covered by our reaction rules, as they are based on known biochemical reactions hindering the ability of the tool to explore those parts of the chemical space.

Another current limitation in our methodology involves the challenge of extending beyond single-step predictions. In our study, we initially generated 580,820 potential drug metabolism products from a pool of 7,939,487 possible outcomes (comprising 271 drugs and 29,297 reaction rules). Moving to the subsequent step, where we predict the metabolism of products derived from the initial predictions, the total number of possibilities escalates significantly to 17,016,283,540. This exponential growth highlights the escalating complexity and computational demands in advancing from single-step to multi-step predictions. The need for robust computational resources and efficient algorithms capable of effectively managing such extensive combinatorial spaces becomes critical. Alternative approaches may involve leveraging a more condensed yet comprehensive knowledgebase of reaction rules, such as the one introduced by Ni *et al.* [316], or filtering out highly similar reaction rules that encode identical transformations. In this regard, the methodology proposed by Chakraborty *et al.* [317] can convert our reaction rules into reaction template fingerprints suitable for similarity searches.

6.4 Conclusion and Future Work

In this work, we introduced BioCatalyzer, a new open-source command line tool for the prediction and identification of compound metabolism. BioCatalyzer can be used to predict the metabolism of compounds using our comprehensive knowledgebase of reaction rules or a subset of those filtered by organisms. The results of the metabolism prediction can then be matched with MS data by comparing the monoisotopic mass of the predicted metabolism products with the ones detected by MS.

We tested and evaluated our tool using gut microbiome drug metabolism data. The generated results showed the capabilities of the tool to accurately predict potential drug metabolism candidates and identify specific chemical transformations. For instance, we identified known Itraconazole metabolism products like Keto Itraconazole and suggested potential unknown ones that matched specific mass shifts in the MS data. Moreover, we identified conserved mass differences between metabolites and their parent drugs, suggesting common metabolic pathways and transformations across different compounds.

Despite the promising results, we acknowledge the limitations of our current methodology. One major challenge is the tool's dependency on annotated reaction rules derived from known biochemical transformations. This limitation restricts our ability to explore unknown parts of the chemical space that include

uncharacterized metabolic transformations and pathways. Additionally, extending beyond single-step predictions presents significant computational challenges, as the number of potential outcomes grows exponentially with each additional step.

To address these limitations, future work will focus on expanding our knowledgebase to include more diverse and comprehensive reaction rules, potentially leveraging methodologies for filtering out redundant rules and condensing similar transformations. By enhancing the tool's capacity to explore multi-step metabolic pathways, we aim to provide a more robust platform for predicting and identifying drug metabolites.

In conclusion, BioCatalyzer represents a valuable resource in the field of metabolic prediction and identification. By uncovering new metabolites and their transformations, researchers can refine drug development strategies, improve therapeutic efficacy, and anticipate potential side effects earlier in the drug discovery process. The insights gained from using BioCatalyzer can enhance our understanding of drug metabolism pathways and their complexities within the human body, ultimately contributing to better healthcare outcomes.

7

Conclusions and Future Work

The overall aim of this thesis was the development of AI-based approaches to enhance the efficiency and accuracy of bio-based drug discovery and pathway design computational experiments. A variety of open-source tools and methodologies to improve drug and pathway design using biochemical strategies were implemented. While our work stands as a step forward in the field it also opens up numerous opportunities for new future research directions.

The following sections will summarize the main contributions of this thesis and discuss suggestions for further improvement.

7.1 Main Contributions

One of the main contributions of this work was the development of innovative computational tools and frameworks for advancing molecular design, synthesis planning, and metabolism prediction. A variety of methodologies were tested and integrated, aiming to enhance predictive accuracy and practical applicability.

In Chapter 3, the thesis introduces DeepMol, an automated framework designed for predicting molecular properties and activities. DeepMol uses AutoML capabilities to simplify the integration of various molecular-specific and advanced ML techniques. This automation streamlines the selection of optimal pipeline steps and the tuning of their hyperparameters, reducing manual effort and accelerating model development. Additionally, extensive validation on ADMET benchmarks demonstrates DeepMol's robustness in adapting to diverse datasets and tasks, consistently achieving good performance.

The methodologies explored in Chapter 4 focused on the use of DL techniques for estimating the number of biochemical transformations necessary for synthesizing compounds. This work explored the generation of extensive biochemical data through reaction rules and its application in training various DL architectures. Both regression and classification approaches were tested, each demonstrating strong performance. The study also investigated different molecular representations to enhance prediction performance. Approaches like the one described in Chapter 4, could potentially be used in compound generation processes, enabling focused exploration of the most promising pathways for synthesizing target compounds. By efficiently filtering intermediate results based on accurate predictions, it can help identify

optimal synthetic pathways. This capability not only improves efficiency and reduces complexity but also enhances the feasibility and selectivity of synthetic routes.

Chapter 5 presents ReactEA, a novel framework that combines evolutionary algorithms with biochemical reaction rules to optimize molecular design. By using evolutionary multi-objective optimization, ReactEA can efficiently design new molecules with optimized properties while maintaining others such as drug-likeness and synthetic accessibility. Practical applications of this framework demonstrated its potential in optimizing molecules for specific targets, enhancing the drug design process. One of ReactEA's key benefits is its effectiveness even with limited data, making it very useful when data availability is limited. Additionally, ReactEA not only suggests molecular structures but also synthesis routes, which is crucial for understanding the decision-making process in drug design.

Chapter 6 introduces BioCatalyzer, a command-line tool designed to predict and identify compound metabolism using a knowledgebase of biochemical reaction templates. The tool was validated with gut microbiome drug metabolism data, showcasing its capability to predict known metabolites and identify unknown ones. By understanding how drugs are metabolized, BioCatalyzer enhances our ability to predict the fate of drugs in the body, which is crucial for assessing drug safety, efficacy, and interactions, as well as for optimizing drug design and development processes.

This thesis also addresses the issue of accessibility in scientific research, making the developed code freely available to the scientific community. The frameworks presented in Chapters 3, 5, and 6 were made freely available as Python packages. Scripts developed in Chapter 4 were also made publicly available. This work is open for use or further enhancement by researchers, fostering collaboration and innovation without restrictions.

Finally, this thesis includes a comprehensive review of recent studies applying AI methods for molecular design, synthesis planning, and metabolism prediction, highlighting limitations and suggesting improvements. It also provides descriptions of ML architectures and publicly available data resources, contributing to the field's advancement.

7.2 Publications

Several chapters from this thesis have either already been published, submitted, or are currently being prepared for submission to peer-reviewed scientific journals (equal contribution is indicated by †):

- At the time of writing, Chapter 3 was submitted to a peer-reviewed scientific journal. Despite that, the paper was made available as a preprint in bioRxiv: Correia, J.†, Capela, J.†, & Rocha, M. (2024). DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry. Cold Spring Harbor Laboratory. `https://doi.org/10.1101/2024.05.27.595849`

- Chapter 4 was published and presented at the 2022 International Joint Conference on Neural Networks (IJCNN): Correia, J., Carreira, R., Pereira, V., & Rocha, M. (2022). Predicting the number of biochemical transformations needed to synthesize a compound. In 2022 International Joint

Conference on Neural Networks (IJCNN). 2022 International Joint Conference on Neural Networks (IJCNN). IEEE. https://doi.org/10.1109/ijcnn55064.2022.9892124

- Chapter 5 was published and presented at the Proceedings of the 2023 Genetic and Evolutionary Computation Conference: Correia, J., Pereira, V., & Rocha, M. (2023). Combining Evolutionary Algorithms with Reaction Rules Towards Focused Molecular Design. In Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '23: Genetic and Evolutionary Computation Conference. ACM. https://doi.org/10.1145/3583131.3590413

- At the time of writing, Chapter 6 is being prepared for submission to a peer-reviewed scientific journal.

Additionally, the author of this thesis contributed to the following peer-reviewed scientific publications during his PhD:

- Sousa, T., Correia, J., Pereira, V., & Rocha, M. (2021). Combining Multi-objective Evolutionary Algorithms with Deep Generative Models Towards Focused Molecular Design. In Applications of Evolutionary Computation (pp. 81–96). Springer International Publishing. https://doi.org/10.1007/978-3-030-72699-7_6

- Sousa, T., Correia, J., Pereira, V., & Rocha, M. (2021). Generative Deep Learning for Targeted Compound Design. In Journal of Chemical Information and Modeling (Vol. 61, Issue 11, pp. 5343–5361). American Chemical Society (ACS). https://doi.org/10.1021/acs.jcim.0c01496

- Baptista, D., Correia, J., Pereira, B., & Rocha, M. (2021). A Comparison of Different Compound Representations for Drug Sensitivity Prediction. In Practical Applications of Computational Biology & Bioinformatics, 15th International Conference (PACBB 2021) (pp. 145–154). Springer International Publishing. https://doi.org/10.1007/978-3-030-86258-9_15

- Capela, J., Correia, J., Pereira, V., & Rocha, M. (2022). Development of Deep Learning approaches to predict relationships between chemical structures and sweetness. In 2022 International Joint Conference on Neural Networks (IJCNN). 2022 International Joint Conference on Neural Networks (IJCNN). IEEE. https://doi.org/10.1109/ijcnn55064.2022.9891992

- Baptista, D., Correia, J., Pereira, B., & Rocha, M. (2022). Evaluating molecular representations in machine learning models for drug response prediction and interpretability. In Journal of Integrative Bioinformatics (Vol. 19, Issue 3). Walter de Gruyter GmbH. https://doi.org/10.1515/jib-2022-0006

7.3 Future Work

The work carried out during this PhD already stands as a significant contribution to the field, providing robust tools and methodologies that have the potential to drive future research and applications forward. Nevertheless, we suggest some future work and improvements that can be addressed to further enhance the impact and applicability of our research.

Perhaps the major limitation of our work relies on the usage of reaction templates in the work described in Chapters 4, 5, and 6. While these templates provide a structured and efficient way to predict reactions, they inherently restrict the scope of chemical space that can be explored, as they are based on pre-defined rules and patterns derived from existing data. This can limit the discovery of novel reactions and mechanisms that fall outside the used templates. To address this, future work could focus on developing more flexible strategies that leverage ML and data-driven approaches to explore unknown parts of the biochemical space.

Additionally, while each tool and methodology proposed in this thesis is ready to be used and integrated with other studies, providing a solid foundation for advancing research in the field, we propose several future work and improvements for each chapter to further enhance their effectiveness and applicability.

For Chapter 3, and the DeepMol framework we propose the following improvements:

- We propose to incorporate meta-learning in the DeepMol framework to enhance its ability to generalize across different molecular datasets and tasks. By leveraging meta-learning, DeepMol can learn to adapt quickly to new problems with minimal data and computational resources;
- Transfer learning strategies can also be employed to leverage pre-trained models or knowledge from related tasks, thereby accelerating learning and improving performance on new datasets or tasks;
- While it is possible to perform data balancing and explore interpretability methods with DeepMol, these are not yet included in the AutoML component. Integrating these functionalities could be a short-term objective.

For Chapter 4:

- While the models have been validated with an independent test set, it would be beneficial to further validate them with experimental pathway data to estimate their reliability;
- Exploring alternative DL architectures and molecular representations can provide deeper insights and potentially improve model performance;
- Integration with bioretrosynthesis tools should be evaluated and validated to enhance the models' applicability in biological synthesis planning.
- Incorporating interpretability would also help in understanding model predictions and supporting informed decision-making in synthetic route planning.

In Chapter 5, regarding the ReactEA framework, we identify the following as pertinent future work:

- Develop and integrate a crossover operator into ReactEA that applies multi-reactant reaction rules to compounds. This operator would combine two different parent molecules to generate offspring;
- Alternatively, exploring ML-based methods to represent molecular structures as numeric vectors could be interesting. However, maintaining interpretability, a critical feature of ReactEA, becomes more challenging with these approaches.

For Chapter 6, BioCatalyzer, we propose the following improvements:

- For this tool, efficiency is a major point. With the computational explosion of possibilities comes the necessity of efficiently navigating or filtering some of the proposed metabolites. For that, methodologies need to be formulated so we can easily predict multi-step metabolism pathways;
- Organizing reaction rules into categories based on types of metabolic transformations could also be interesting. These could include phase I and phase II metabolism reactions, human gut microbial reactions, environmental microbial transformations, and various combinations of those;
- Including ML algorithms to predict compound metabolism could also be interesting as they could help improve the accuracy and speed of predicting metabolite transformations.

There are still many ways we can improve our current work, showing the dynamic nature of research in this field. These improvements also mean we could still make big advances and come up with new ideas. Our findings open up numerous avenues for new and exciting research directions, presenting a positive outlook on the possibilities for future scientific exploration and discovery.

Bibliography

[1] S. Maiti, G. Gallastegui, S. Kaur Brar, *et al.*, "Quest for sustainable bio-production and recovery of butanol as a promising solution to fossil fuel: Bio-butanol production and recovery," *International Journal of Energy Research*, vol. 40, no. 4, pp. 411–438, Dec. 2015, issn: 0363-907X. doi: 10.1002/er.3458.

[2] K. Boodhoo, M. Flickinger, J. Woodley, and E. Emanuelsson, "Bioprocess intensification: A route to efficient and sustainable biocatalytic transformations for the future," *Chemical Engineering and Processing - Process Intensification*, vol. 172, p. 108 793, Feb. 2022, issn: 0255-2701. doi: 10.1016/j.cep.2022.108793.

[3] W. Dessie, X. Luo, G. J. Duns, M. Wang, and Z. Qin, "Towards the development of efficient, economic and environmentally friendly downstream processing for bio-based succinic acid," *Environmental Technology & Innovation*, vol. 32, p. 103 243, Nov. 2023, issn: 2352-1864. doi: 10.1016/j.eti.2023.103243.

[4] A. Akinsemolu, H. Onyeaka, O. Fagunwa, and A. H. Adenuga, "Toward a resilient future: The promise of microbial bioeconomy," *Sustainability*, vol. 15, no. 9, p. 7251, Apr. 2023, issn: 2071-1050. doi: 10.3390/su15097251.

[5] Y. Liu, M. G. Benitez, J. Chen, E. Harrison, A. N. Khusnutdinova, and R. Mahadevan, "Opportunities and challenges for microbial synthesis of fatty acid-derived chemicals (facs)," *Frontiers in Bioengineering and Biotechnology*, vol. 9, Jan. 2021, issn: 2296-4185. doi: 10.3389/fbioe.2021.613322.

[6] J. Zhu, S. Wang, C. Wang, *et al.*, "Microbial synthesis of bacitracin: Recent progress, challenges, and prospects," *Synthetic and Systems Biotechnology*, vol. 8, no. 2, pp. 314–322, Jun. 2023, issn: 2405-805X. doi: 10.1016/j.synbio.2023.03.009.

[7] M. Awale, R. Visini, D. Probst, J. Arús-Pous, and J.-L. Reymond, "Chemical space: Big data challenge for molecular diversity," *CHIMIA*, vol. 71, no. 10, p. 661, Oct. 2017, issn: 0009-4293. doi: 10.2533/chimia.2017.661.

[8] S. Vilar and S. Costanzi, "Predicting the biological activities through qsar analysis and docking-based scoring," in *Membrane Protein Structure and Dynamics*. Humana Press, 2012, pp. 271–284, isbn: 9781627030236. doi: 10.1007/978-1-62703-023-6_16.

[9] T. Tanebe and T. Ishida, "End-to-end learning for compound activity prediction based on binding pocket information," *BMC Bioinformatics*, vol. 22, no. S3, May 2021, issn: 1471-2105. doi: 10.1186/s12859-021-04440-w.

[10] M. Goel, A. Sharma, A. S. Chilwal, S. Kumari, A. Kumar, and G. Bagler, "Machine learning models to predict sweetness of molecules," *Computers in Biology and Medicine*, vol. 152, p. 106 441, Jan. 2023, issn: 0010-4825. doi: 10.1016/j.compbiomed.2022.106441.

[11] A. Karampuri and S. Perugu, "A breast cancer-specific combinational qsar model development using machine learning and deep learning approaches," *Frontiers in Bioinformatics*, vol. 3, Jan. 2024, issn: 2673-7647. doi: 10.3389/fbinf.2023.1328262.

[12] Y. Shi, "Support vector regression-based qsar models for prediction of antioxidant activity of phenolic compounds," *Scientific Reports*, vol. 11, no. 1, Apr. 2021, issn: 2045-2322. doi: 10.1038/s41598-021-88341-1.

[13] T. Tran and C. Ekenna, "Molecular descriptors property prediction using transformer-based approach," *International Journal of Molecular Sciences*, vol. 24, no. 15, p. 11 948, Jul. 2023, issn: 1422-0067. doi: 10.3390/ijms241511948.

[14] M. Sakai, K. Nagayasu, N. Shibui, *et al.*, "Prediction of pharmacological activities from chemical structures with graph convolutional neural networks," *Scientific Reports*, vol. 11, no. 1, Jan. 2021, issn: 2045-2322. doi: 10.1038/s41598-020-80113-7.

[15] S. Moon, A. Saboe, and M. J. Smanski, "Using design of experiments to guide genetic optimization of engineered metabolic pathways," *Journal of Industrial Microbiology and Biotechnology*, vol. 51, 2024, issn: 1476-5535. doi: 10.1093/jimb/kuae010.

[16] S. Hong, H. H. Zhuo, K. Jin, G. Shao, and Z. Zhou, "Retrosynthetic planning with experience-guided monte carlo tree search," *Communications Chemistry*, vol. 6, no. 1, Jun. 2023, issn: 2399-3669. doi: 10.1038/s42004-023-00911-8.

[17] G.-M. Lin, R. Warden-Rothman, and C. A. Voigt, "Retrosynthetic design of metabolic pathways to chemicals not found in nature," *Current Opinion in Systems Biology*, vol. 14, pp. 82–107, Apr. 2019, issn: 2452-3100. doi: 10.1016/j.coisb.2019.04.004.

[18] G. Zahoránszky-K□halmi, N. Lysov, I. Vorontcov, *et al.*, "Algorithm for the pruning of synthesis graphs," *Journal of Chemical Information and Modeling*, vol. 62, no. 9, pp. 2226–2238, Apr. 2022, issn: 1549-960X. doi: 10.1021/acs.jcim.1c01202.

[19] Z. Zhang and W. Tang, "Drug metabolism in drug discovery and development," *Acta Pharmaceutica Sinica B*, vol. 8, no. 5, pp. 721–732, Sep. 2018, issn: 2211-3835. doi: 10.1016/j.apsb.2018.04.003.

[20] J. Bailey, "Toward a science of metabolic engineering," *Science*, vol. 252, no. 5013, pp. 1668–1675, Jun. 1991. doi: 10.1126/science.2047876.

[21] G. Stephanopoulos and J. Vallino, "Network rigidity and metabolic engineering in metabolite overproduction," *Science*, vol. 252, no. 5013, pp. 1675–1681, Jun. 1991. doi: 10.1126/science.1904627.

[22] M. Basiony, L. Ouyang, D. Wang, *et al.*, "Optimization of microbial cell factories for astaxanthin production: Biosynthesis and regulations, engineering strategies and fermentation optimization strategies," *Synthetic and Systems Biotechnology*, vol. 7, no. 2, pp. 689–704, Jun. 2022, issn: 2405-805X. doi: 10.1016/j.synbio.2022.01.002.

[23] F. Muneer, I. Rasul, M. Qasim, A. Sajid, and H. Nadeem, "Optimization, production and characterization of polyhydroxyalkanoate (pha) from indigenously isolated novel bacteria," *Journal of Polymers and the Environment*, vol. 30, no. 8, pp. 3523–3533, Apr. 2022, issn: 1572-8919. doi: 10.1007/s10924-022-02444-y.

[24] S. Ferreira, R. Pereira, S. A. Wahl, and I. Rocha, "Metabolic engineering strategies for butanol production in escherichia coli," *Biotechnology and Bioengineering*, vol. 117, no. 8, pp. 2571–2587, Jun. 2020, issn: 1097-0290. doi: 10.1002/bit.27377.

[25] M. P. Mezzina, M. T. Manoli, M. A. Prieto, and P. I. Nikel, "Engineering native and synthetic pathways in pseudomonas putida for the production of tailored polyhydroxyalkanoates," *Biotechnology Journal*, vol. 16, no. 3, Nov. 2020, issn: 1860-7314. doi: 10.1002/biot.202000165.

[26] T. Sekiguchi, H. Hamada, and M. Okamoto, "Winbest-kit: Biochemical reaction simulator for analyzing multi-layered metabolic pathways," *Bioengineering*, vol. 8, no. 8, p. 114, Aug. 2021, issn: 2306-5354. doi: 10.3390/bioengineering8080114.

[27] T. Duigou, M. du Lac, P. Carbonell, and J.-L. Faulon, "Retrorules: A database of reaction rules for engineering biology," *Nucleic Acids Research*, vol. 47, no. D1, pp. D1229–D1235, Oct. 2018, issn: 1362-4962. doi: 10.1093/nar/gky940.

[28] W. Qian, X. Wang, Y. Kang, P. Pan, T. Hou, and C.-Y. Hsieh, "A general model for predicting enzyme functions based on enzymatic reactions," *Journal of Cheminformatics*, vol. 16, no. 1, Mar. 2024, issn: 1758-2946. doi: 10.1186/s13321-024-00827-y.

[29] N. Buton, F. Coste, and Y. Le Cunff, "Predicting enzymatic function of protein sequences with attention," *Bioinformatics*, vol. 39, no. 10, A. Valencia, Ed., Oct. 2023, issn: 1367-4811. doi: 10.1093/bioinformatics/btad620.

[30] T. Greinert, K. Vogel, T. Maskow, and C. Held, "New thermodynamic activity-based approach allows predicting the feasibility of glycolysis," *Scientific Reports*, vol. 11, no. 1, Mar. 2021, issn: 2045-2322. doi: 10.1038/s41598-021-85594-8.

[31] T. Fuji, S. Nakazawa, and K. Ito, "Feasible-metabolic-pathway-exploration technique using chemical latent space," *Bioinformatics*, vol. 36, no. Supplement$_2$, pp. i770–i778, Dec. 2020, issn: 1367-4811. doi: 10.1093/bioinformatics/btaa809.

[32] F. Liu, P. Vilaça, I. Rocha, and M. Rocha, "Development and application of efficient pathway enumeration algorithms for metabolic engineering applications," *Computer Methods and Programs in Biomedicine*, vol. 118, no. 2, pp. 134–146, Feb. 2015, issn: 0169-2607. doi: 10.1016/j.cmpb.2014.11.010.

[33] P. Carbonell, D. Fichera, S. B. Pandit, and J.-L. Faulon, "Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms," *BMC Systems Biology*, vol. 6, no. 1, p. 10, 2012. doi: 10.1186/1752-0509-6-10.

[34] M. Yousofshahi, K. Lee, and S. Hassoun, "Probabilistic pathway construction," *Metabolic Engineering*, vol. 13, no. 4, pp. 435–444, Jul. 2011, issn: 1096-7176. doi: 10.1016/j.ymben.2011.01.006.

[35] L. F. de Figueiredo, A. Podhorski, A. Rubio, *et al.*, "Computing the shortest elementary flux modes in genome-scale metabolic networks," *Bioinformatics*, vol. 25, no. 23, pp. 3158–3165, Sep. 2009, issn: 1367-4803. doi: 10.1093/bioinformatics/btp564.

[36] V. Hatzimanikatis, C. Li, J. A. Ionita, C. S. Henry, M. D. Jankowski, and L. J. Broadbelt, "Exploring the diversity of complex metabolic networks," *Bioinformatics*, vol. 21, no. 8, pp. 1603–1609, Dec. 2004. doi: 10.1093/bioinformatics/bti213.

[37] A. Kumar, L. Wang, C. Y. Ng, and C. D. Maranas, "Pathway design using de novo steps through uncharted biochemical spaces," *Nature Communications*, vol. 9, no. 1, Jan. 2018, issn: 2041-1723. doi: 10.1038/s41467-017-02362-x.

[38] J. Asplund-Samuelsson, M. Janasch, and E. P. Hudson, "Thermodynamic analysis of computed pathways integrated into the metabolic networks of e. coli and synechocystis reveals contrasting expansion potential," *Metabolic Engineering*, vol. 45, pp. 223–236, Jan. 2018, issn: 1096-7176. doi: 10.1016/j.ymben.2017.12.011.

[39] P. Pharkya, "OptStrain: A computational framework for redesign of microbial production systems," *Genome Research*, vol. 14, no. 11, pp. 2367–2376, Nov. 2004. doi: 10.1101/gr.2872004.

[40] P. Schwaller, T. Laino, T. Gaudin, *et al.*, "Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction," *ACS Central Science*, vol. 5, no. 9, pp. 1572–1583, Aug. 2019. doi: 10.1021/acscentsci.9b00576.

[41] M. H. S. Segler and M. P. Waller, "Neural-symbolic machine learning for retrosynthesis and reaction prediction," *Chemistry - A European Journal*, vol. 23, no. 25, pp. 5966–5971, Feb. 2017. doi: 10.1002/chem.201605499.

[42] J. Lu and Y. Zhang, "Unified deep learning model for multitask reaction predictions with explanation," *Journal of Chemical Information and Modeling*, vol. 62, no. 6, pp. 1376–1387, Mar. 2022, issn: 1549-960X. doi: 10.1021/acs.jcim.1c01467.

[43] K. Lin, Y. Xu, J. Pei, and L. Lai, "Automatic retrosynthetic route planning using template-free models," *Chemical Science*, vol. 11, no. 12, pp. 3355–3364, 2020, issn: 2041-6539. doi: 10.1039/c9sc03666k.

[44] K. Molga, S. Szymkuć, and B. A. Grzybowski, "Chemist ex machina: Advanced synthesis planning by computers," *Accounts of Chemical Research*, vol. 54, no. 5, pp. 1094–1106, Jan. 2021, issn: 1520-4898. doi: 10.1021/acs.accounts.0c00714.

[45] S. Genheden, A. Thakkar, V. Chadimová, J.-L. Reymond, O. Engkvist, and E. Bjerrum, "Aizynthfinder: A fast, robust and flexible open-source software for retrosynthetic planning," *Journal of Cheminformatics*, vol. 12, no. 1, Nov. 2020, issn: 1758-2946. doi: 10.1186/s13321-020-00472-1.

[46] A. Ravikrishnan, M. Nasre, and K. Raman, "Enumerating all possible biosynthetic pathways in metabolic networks," *Scientific Reports*, vol. 8, no. 1, Jul. 2018, issn: 2045-2322. doi: 10.1038/s41598-018-28007-7.

[47] T. Handorf and O. Ebenhoh, "Metapath online: A web server implementation of the network expansion algorithm," *Nucleic Acids Research*, vol. 35, no. Web Server, W613–W618, May 2007, issn: 1362-4962. doi: 10.1093/nar/gkm287.

[48] X. Song, M. Dong, and M. Liu, "Pyminer: A method for metabolic pathway design based on the uniform similarity of substrate-product pairs and conditional search," *PLOS ONE*, vol. 17, no. 4, D. Zhang, Ed., e0266783, Apr. 2022, issn: 1932-6203. doi: 10.1371/journal.pone.0266783.

[49] J. Hafner and V. Hatzimanikatis, "Nicepath: Finding metabolic pathways in large networks through atom-conserving substrate–product pairs," *Bioinformatics*, vol. 37, no. 20, A. Valencia, Ed., pp. 3560–3568, May 2021, issn: 1367-4811. doi: 10.1093/bioinformatics/btab368.

[50] D. Kreutter, P. Schwaller, and J.-L. Reymond, "Predicting enzymatic reactions with a molecular transformer," *Chemical Science*, vol. 12, no. 25, pp. 8648–8659, 2021, issn: 2041-6539. doi: 10.1039/d1sc02362d.

[51] Y. Moriya, D. Shigemizu, M. Hattori, *et al.*, "PathPred: An enzyme-catalyzed metabolic pathway prediction server," *Nucleic Acids Research*, vol. 38, no. Web Server, W138–W143, Apr. 2010. doi: 10.1093/nar/gkq318.

[52] D. Probst, M. Manica, Y. G. Nana Teukam, A. Castrogiovanni, F. Paratore, and T. Laino, "Biocatalysed synthesis planning using data-driven learning," *Nature Communications*, vol. 13, no. 1, Feb. 2022, issn: 2041-1723. doi: 10.1038/s41467-022-28536-w.

[53] M. Baranwal, A. Magner, P. Elvati, J. Saldinger, A. Violi, and A. O. Hero, "A deep learning architecture for metabolic pathway prediction," *Bioinformatics*, vol. 36, no. 8, J. Wren, Ed., pp. 2547–2553, Dec. 2019, issn: 1367-4811. doi: 10.1093/bioinformatics/btz954.

[54] K. Sankaranarayanan, E. Heid, C. W. Coley, D. Verma, W. H. Green, and K. F. Jensen, "Similarity based enzymatic retrosynthesis," *Chemical Science*, vol. 13, no. 20, pp. 6039–6053, 2022, issn: 2041-6539. doi: 10.1039/d2sc01588a.

[55] S. Ding, Y. Tian, P. Cai, *et al.*, "Novopathfinder: A webserver of designing novel-pathway with integrating gem-model," *Nucleic Acids Research*, vol. 48, no. W1, W477–W487, Apr. 2020, issn: 1362-4962. doi: 10.1093/nar/gkaa230.

[56] S. Zheng, T. Zeng, C. Li, *et al.*, "Deep learning driven biosynthetic pathways navigation for natural products with bionavi-np," *Nature Communications*, vol. 13, no. 1, Jun. 2022, issn: 2041-1723. doi: 10.1038/s41467-022-30970-9.

[57] I. Levin, M. Liu, C. A. Voigt, and C. W. Coley, "Merging enzymatic and synthetic chemistry with computational synthesis planning," *Nature Communications*, vol. 13, no. 1, Dec. 2022, issn: 2041-1723. doi: 10.1038/s41467-022-35422-y.

[58] M. Koch, T. Duigou, and J.-L. Faulon, "Reinforcement learning for bioretrosynthesis," *ACS Synthetic Biology*, vol. 9, no. 1, pp. 157–168, Dec. 2019. doi: 10.1021/acssynbio.9b00447.

[59] W. Finnigan, L. J. Hepworth, S. L. Flitsch, and N. J. Turner, "Retrobiocat as a computer-aided synthesis planning tool for biocatalytic reactions and cascades," *Nature Catalysis*, vol. 4, no. 2, pp. 98–104, Jan. 2021, issn: 2520-1158. doi: 10.1038/s41929-020-00556-z.

[60] B. Delépine, T. Duigou, P. Carbonell, and J.-L. Faulon, "RetroPath2.0: A retrosynthesis workflow for metabolic engineers," *Metabolic Engineering*, vol. 45, pp. 158–170, Jan. 2018. doi: 10.1016/j.ymben.2017.12.002.

[61] Y. Zhu, A. Boye, M. Body-Malapel, and J. Herkovits, "The toxic effects of xenobiotics on the health of humans and animals," *BioMed Research International*, vol. 2017, pp. 1–2, 2017, issn: 2314-6141. doi: 10.1155/2017/4627872.

[62] V.-A. Scholz, C. Stork, M. Frericks, and J. Kirchmair, "Computational prediction of the metabolites of agrochemicals formed in rats," *Science of The Total Environment*, vol. 895, p. 165 039, Oct. 2023, issn: 0048-9697. doi: 10.1016/j.scitotenv.2023.165039.

[63] A. S. Serras, J. S. Rodrigues, M. Cipriano, A. V. Rodrigues, N. G. Oliveira, and J. P. Miranda, "A critical perspective on 3d liver models for drug metabolism and toxicology studies," *Frontiers in Cell and Developmental Biology*, vol. 9, Feb. 2021, issn: 2296-634X. doi: 10.3389/fcell.2021.626805.

[64] D. Dalvie, R. S. Obach, P. Kang, *et al.*, "Assessment of three human in vitro systems in the generation of major human excretory and circulating metabolites," *Chemical Research in Toxicology*, vol. 22, no. 2, pp. 357–368, Jan. 2009, issn: 1520-5010. doi: 10.1021/tx8004357.

[65] *Daylight Theory: SMIRKS - A Reaction Transform Language — daylight.com*, https://www.daylight.com/dayhtml/doc/theory/theory.smirks.html, [Accessed 11-07-2024].

[66] DayLight, *Daylight theory manual*, https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html, Accessed 15/06/2024.

[67] H. Satoh and K. Funatsu, "Sophia, a knowledge base-guided reaction prediction system - utilization of a knowledge base derived from a reaction database," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 1, pp. 34–44, Jan. 1995, issn: 1520-5142. doi: 10.1021/ci00023a005.

[68] W. Wang, Q. Liu, L. Zhang, Y. Dong, and J. Du, "Retrosynx: A retrosynthetic analysis framework using hybrid reaction templates and group contribution-based thermodynamic models," *Chemical Engineering Science*, vol. 248, p. 117 208, Feb. 2022, issn: 0009-2509. doi: 10.1016/j.ces.2021.117208.

[69] A. Thakkar and J.-L. Reymond, "Automatic extraction of reaction templates for synthesis prediction," *CHIMIA*, vol. 76, no. 4, p. 294, Apr. 2022, issn: 0009-4293. doi: 10.2533/chimia.2022.294.

[70] L.-Y. Chen and Y.-P. Li, "Autotemplate: Enhancing chemical reaction datasets for machine learning applications in organic chemistry," *ChemRxiv*, Mar. 2024. doi: 10.26434/chemrxiv-2024-tq22r.

[71] D. S. Wishart, S. Tian, D. Allen, *et al.*, "Biotransformer 3.0—a web server for accurately predicting metabolic transformation products," *Nucleic Acids Research*, vol. 50, no. W1, W115–W123, May 2022, issn: 1362-4962. doi: 10.1093/nar/gkac313.

[72] L. Ridder and M. Wagener, "Sygma: Combining expert knowledge and empirical scoring in the prediction of metabolites," *ChemMedChem*, vol. 3, no. 5, pp. 821–832, May 2008, issn: 1860-7187. doi: 10.1002/cmdc.200700312.

[73] E. E. Litsa, P. Das, and L. E. Kavraki, "Prediction of drug metabolites using neural machine translation," *Chemical Science*, vol. 11, no. 47, pp. 12 777–12 788, 2020, issn: 2041-6539. doi: 10.1039/d0sc02639e.

[74] D. S. Wishart, Y. D. Feunang, A. C. Guo, *et al.*, "DrugBank 5.0: A major update to the DrugBank database for 2018," *Nucleic Acids Research*, vol. 46, no. D1, pp. D1074–D1082, Nov. 2017. doi: 10.1093/nar/gkx1037.

[75] E. Brunk, S. Sahoo, D. C. Zielinski, *et al.*, "Recon3d enables a three-dimensional view of gene variation in human metabolism," *Nature Biotechnology*, vol. 36, no. 3, pp. 272–281, Feb. 2018, issn: 1546-1696. doi: 10.1038/nbt.4072.

[76] D. Weininger, "SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Modeling*, vol. 28, no. 1, pp. 31–36, Feb. 1988. doi: 10.1021/ci00057a005.

[77] C. de Bruyn Kops, M. Šícho, A. Mazzolari, and J. Kirchmair, "Gloryx: Prediction of the metabolites resulting from phase 1 and phase 2 biotransformations of xenobiotics," *Chemical Research in Toxicology*, vol. 34, no. 2, pp. 286–299, Aug. 2020, issn: 1520-5010. doi: 10.1021/acs.chemrestox.0c00224.

[78] C. de Bruyn Kops, C. Stork, M. Šícho, *et al.*, "Glory: Generator of the structures of likely cytochrome p450 metabolites based on predicted sites of metabolism," *Frontiers in Chemistry*, vol. 7, Jun. 2019, issn: 2296-2646. doi: 10.3389/fchem.2019.00402.

[79] S. Tian, X. Cao, R. Greiner, C. Li, A. Guo, and D. S. Wishart, "Cyproduct: A software tool for accurately predicting the byproducts of human cytochrome p450 metabolism," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 3128–3140, May 2021, issn: 1549-960X. doi: 10.1021/acs.jcim.1c00144.

[80] A. V. Rudik, V. M. Bezhentsev, A. V. Dmitriev, *et al.*, "Metatox: Web application for predicting structure and toxicity of xenobiotics' metabolites," *Journal of Chemical Information and Modeling*, vol. 57, no. 4, pp. 638–642, Apr. 2017, issn: 1549-960X. doi: 10.1021/acs.jcim.6b00662.

[81] R. Caspi, R. Billington, I. M. Keseler, *et al.*, "The metacyc database of metabolic pathways and enzymes - a 2019 update," *Nucleic Acids Research*, vol. 48, no. D1, pp. D445–D453, Oct. 2019, issn: 1362-4962. doi: 10.1093/nar/gkz862.

[82] L. Jeske, S. Placzek, I. Schomburg, A. Chang, and D. Schomburg, "BRENDA in 2019: A european ELIXIR core data resource," *Nucleic Acids Research*, vol. 47, no. D1, pp. D542–D549, Nov. 2018. doi: 10.1093/nar/gky1048.

[83] M. Kanehisa, "KEGG: Kyoto encyclopedia of genes and genomes," *Nucleic Acids Research*, vol. 28, no. 1, pp. 27–30, Jan. 2000. doi: 10.1093/nar/28.1.27.

[84] B. Jassal, L. Matthews, G. Viteri, *et al.*, "The reactome pathway knowledgebase," *Nucleic Acids Research*, Nov. 2019. doi: 10.1093/nar/gkz1031.

[85] S. M. D. Seaver, F. Liu, Q. Zhang, *et al.*, "The modelseed biochemistry database for the integration of metabolic annotations and the reconstruction, comparison and analysis of metabolic models for plants, fungi and microbes," *Nucleic Acids Research*, vol. 49, no. D1, pp. D575–D588, Sep. 2020, issn: 1362-4962. doi: 10.1093/nar/gkaa746.

[86] N. Hadadi, J. Hafner, A. Shajkofci, A. Zisaki, and V. Hatzimanikatis, "ATLAS of biochemistry: A repository of all possible biochemical reactions for synthetic biology and metabolic engineering studies," *ACS Synthetic Biology*, vol. 5, no. 10, pp. 1155–1166, Jul. 2016. doi: 10.1021/acssynbio.6b00054.

[87] J. Strutz, K. M. Shebek, L. J. Broadbelt, and K. E. J. Tyo, "Mine 2.0: Enhanced biochemical coverage for peak identification in untargeted metabolomics," *Bioinformatics*, vol. 38, no. 13, Z. Lu, Ed., pp. 3484–3487, May 2022, issn: 1367-4811. doi: 10.1093/bioinformatics/btac331.

[88] A. A. Labena, Y.-Z. Gao, C. Dong, H.-l. Hua, and F.-B. Guo, "Metabolic pathway databases and model repositories," *Quantitative Biology*, vol. 6, no. 1, pp. 30–39, Mar. 2018, issn: 2095-4697. doi: 10.1007/s40484-017-0108-3.

[89] Z. Nagy, *Artificial intelligence and machine learning fundamentals*. Birmingham, UK: Packt Publishing, 2018, isbn: 9781789801651.

[90] E. Alpaydin, *Introduction to machine learning*. Cambridge, Massachusetts: The MIT Press, 2014, isbn: 9780262028189.

[91] G. Hinton and T. J. Sejnowski, Eds., *Unsupervised Learning*. The MIT Press, 1999. doi: 10.7551/mitpress/7011.001.0001.

[92] T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer, Apr. 2019, isbn: 0387848576.

[93] M. F. A. Hady and F. Schwenker, "Semi-supervised learning," in *Handbook on Neural Information Processing*. Springer Berlin Heidelberg, 2013, pp. 215–239, isbn: 9783642366574. doi: 10.1007/978-3-642-36657-4_7.

[94] R. S. Sutton, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series)*. A Bradford Book, Nov. 2018, isbn: 0262039249.

[95] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, ACM Press, 1992. doi: 10.1145/130385.130401.

[96] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/a:1010933404324.

[97] M. Hassoun, *Fundamentals of artificial neural networks*. Cambridge, Mass: MIT Press, 1995, isbn: 9780262082396.

[98] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. eprint: arXiv:1609.04747.

[99] Y. Wei, W. Xia, M. Lin, *et al.*, "HCP: A flexible CNN framework for multi-label image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1901–1907, Sep. 2016. doi: 10.1109/tpami.2015.2491929.

[100] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, *Language modeling with gated convolutional networks*, 2016. eprint: arXiv:1612.08083.

[101] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, IEEE, Jul. 2017. doi: 10.1109/cbi.2017.23.

[102] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. doi: 10.1162/neco.1997.9.8.1735.

[103] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, Aug. 2016. doi: 10.1007/s10822-016-9938-8.

[104] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, ACM, Jul. 2019. doi: 10.1145/3292500.3330958.

[105] R. v. d. Berg, T. N. Kipf, and M. Welling, *Graph convolutional matrix completion*, 2017. doi: 10.48550/ARXIV.1706.02263.

[106] Q. Wu, H. Zhang, X. Gao, *et al.*, "Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems," in *The World Wide Web Conference*, ser. WWW '19, ACM, May 2019. doi: 10.1145/3308558.3313442.

[107] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds., Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1556–1566. doi: 10.3115/v1/P15-1150.

[108] H. Peng, J. Li, Y. He, *et al.*, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, ser. WWW '18, ACM Press, 2018. doi: 10.1145/3178876.3186005.

[109] V. Garcia and J. Bruna, *Few-shot learning with graph neural networks*, 2017. doi: 10.48550/ARXIV.1711.04043.

[110] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. doi: 10.1109/cvpr.2017.10.

[111] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. doi: 10.48550/ARXIV.1706.03762.

[112] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. doi: 10.48550/ARXIV.1810.04805.

[113] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. doi: 10.48550/ARXIV.2005.14165.

[114] OpenAI, J. Achiam, S. Adler, *et al.*, *Gpt-4 technical report*, 2023. doi: 10.48550/ARXIV.2303.08774.

[115] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2013. eprint: arXiv:1312.6114.

[116] D. J. Rezende, S. Mohamed, and D. Wierstra, *Stochastic backpropagation and approximate inference in deep generative models*, 2014. eprint: arXiv:1401.4082.

[117] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. eprint: arXiv:1406.2661.

[118] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, Jan. 2009. doi: 10.1137/070699652.

[119] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: Hands-free automl via meta-learning," 2020. doi: 10.48550/ARXIV.2007.04074.

[120] T. T. Le, W. Fu, and J. H. Moore, "Scaling tree-based automated machine learning to biomedical big data with a feature set selector," *Bioinformatics*, vol. 36, no. 1, J. Kelso, Ed., pp. 250–256, Jun. 2019, issn: 1367-4811. doi: 10.1093/bioinformatics/btz470.

[121] N. Erickson, J. Mueller, A. Shirkov, *et al.*, *Autogluon-tabular: Robust and accurate automl for structured data*, 2020. doi: 10.48550/ARXIV.2003.06505.

[122] H. Jin, F. Chollet, Q. Song, and X. Hu, "Autokeras: An automl library for deep learning," *Journal of Machine Learning Research*, vol. 24, no. 6, pp. 1–6, 2023.

[123] H2O.ai, *Python interface for h2o*, Python module version 3.46.0.3, Jul. 2024.

[124] Microsoft, *Neural Network Intelligence*, version 2.0, Jan. 2021.

[125] I. Rechenberg, "Evolution strategy: Nature's way of optimization," in *Optimization: Methods and Applications, Possibilities and Limitations*. Springer Berlin Heidelberg, 1989, pp. 106–126, isbn: 9783642838149. doi: 10.1007/978-3-642-83814-9_6.

[126] D. B. Fogel, "An overview of evolutionary programming," in *The IMA Volumes in Mathematics and its Applications*. Springer New York, 1999, pp. 89–109, isbn: 9781461215424. doi: 10.1007/978-1-4612-1542-4_5.

[127] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Berlin Heidelberg, 1996, isbn: 9783662033159. doi: 10.1007/978-3-662-03315-9.

[128] A. Hussain and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 1–14, Apr. 2019, issn: 2198-6053. doi: 10.1007/s40747-019-0102-7.

[129] T. Back, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, ser. ICEC-94, IEEE, 1994. doi: 10.1109/icec.1994.350042.

[130] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, Springer Berlin Heidelberg, 2000, pp. 849–858. doi: 10.1007/3-540-45356-3_83.

[131] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug. 2014. doi: 10.1109/tevc.2013.2281535.

[132] P. Chaudhari, A. K. Thakur, R. Kumar, N. Banerjee, and A. Kumar, "Comparison of nsga-iii with nsga-ii for multi objective optimization of adiabatic styrene reactor," *Materials Today: Proceedings*, vol. 57, pp. 1509–1514, 2022, issn: 2214-7853. doi: 10.1016/j.matpr.2021.12.047.

[133] T. Bäck, "Evolution strategies: An alternative evolutionary algorithm," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 1996, pp. 1–20. doi: 10.1007/3-540-61108-8_27.

[134] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2004, pp. 832–842. doi: 10.1007/978-3-540-30217-9_84.

[135] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, "InChI, the IUPAC international chemical identifier," *Journal of Cheminformatics*, vol. 7, no. 1, May 2015. doi: 10.1186/s13321-015-0068-4.

[136] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[137] M. Ohue, R. Ii, K. Yanagisawa, and Y. Akiyama, *Molecular activity prediction using graph convolutional deep neural network considering distance on a molecular graph*, 2019. arXiv: 1907.01103 [q-bio.BM].

[138] Z. Xu, S. Wang, F. Zhu, and J. Huang, "Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics - ACM-BCB '17*, New York, New York, USA: ACM Press, 2017, pp. 285–294, isbn: 9781450347228. doi: 10.1145/3107411.3107424.

[139] S. Jaeger, S. Fulle, and S. Turk, "Mol2vec: Unsupervised machine learning approach with chemical intuition," *Journal of Chemical Information and Modeling*, vol. 58, no. 1, pp. 27–35, Jan. 2018. doi: 10.1021/acs.jcim.7b00616.

[140] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014. doi: 10.48550/ARXIV.1409.3215.

[141] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. doi: 10.48550/ARXIV.1301.3781.

[142] Y. Li, L. Zhang, and Z. Liu, "Multi-objective de novo drug design with conditional graph generative model," *Journal of Cheminformatics*, vol. 10, no. 1, Jul. 2018. doi: 10.1186/s13321-018-0287-6.

[143] S. Kim, J. Chen, T. Cheng, *et al.*, "PubChem 2019 update: Improved access to chemical data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D1102–D1109, Oct. 2018. doi: 10.1093/nar/gky1033.

[144] Z. Wu, B. Ramsundar, E. N. Feinberg, *et al.*, "MoleculeNet: A benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018. doi: 10.1039/c7sc02664a.

[145] J. Hastings, G. Owen, A. Dekker, *et al.*, "ChEBI in 2016: Improved services and an expanding collection of metabolites," *Nucleic Acids Research*, vol. 44, no. D1, pp. D1214–D1219, Oct. 2015. doi: 10.1093/nar/gkv1031.

[146] H. E. Pence and A. Williams, "ChemSpider: An online chemical information resource," *Journal of Chemical Education*, vol. 87, no. 11, pp. 1123–1124, Nov. 2010. doi: 10.1021/ed100697w.

[147] J. M. Stokes, K. Yang, K. Swanson, *et al.*, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, 688–702.e13, Feb. 2020, issn: 0092-8674. doi: 10.1016/j.cell.2020.01.021.

[148] F. B. Ashraf, S. Akter, S. H. Mumu, M. U. Islam, and J. Uddin, "Bio-activity prediction of drug candidate compounds targeting sars-cov-2 using machine learning approaches," *PLOS ONE*, vol. 18, no. 9, A. A. Al-Karmalawy, Ed., e0288053, Sep. 2023, issn: 1932-6203. doi: 10.1371/journal.pone.0288053.

[149] M. Galushka, C. Swain, F. Browne, M. D. Mulvenna, R. Bond, and D. Gray, "Prediction of chemical compounds properties using a deep learning model," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13 345–13 366, Jun. 2021, issn: 1433-3058. doi: 10.1007/s00521-021-05961-4.

[150] A. Mayr, G. Klambauer, T. Unterthiner, *et al.*, "Large-scale comparison of machine learning methods for drug target prediction on ChEMBL," *Chemical Science*, vol. 9, no. 24, pp. 5441–5451, 2018. doi: 10.1039/c8sc00148k.

[151] B. Merget, S. Turk, S. Eid, F. Rippmann, and S. Fulle, "Profiling prediction of kinase inhibitors: Toward the virtual assay," *Journal of Medicinal Chemistry*, vol. 60, no. 1, pp. 474–485, Dec. 2016. doi: 10.1021/acs.jmedchem.6b01611.

[152] E. B. Lenselink, N. ten Dijke, B. Bongers, *et al.*, "Beyond the hype: Deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set," *Journal of Cheminformatics*, vol. 9, no. 1, Aug. 2017. doi: 10.1186/s13321-017-0232-0.

[153] A. Gaulton, A. Hersey, M. Nowotka, *et al.*, "The ChEMBL database in 2017," *Nucleic Acids Research*, vol. 45, no. D1, pp. D945–D954, Nov. 2016. doi: 10.1093/nar/gkw1074.

[154] A. Korotcov, V. Tkachenko, D. P. Russo, and S. Ekins, "Comparison of deep learning with multiple machine learning methods and metrics using diverse drug discovery data sets," *Molecular Pharmaceutics*, vol. 14, no. 12, pp. 4462–4475, Nov. 2017. doi: 10.1021/acs.molpharmaceut.7b00578.

[155] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *Journal of Cheminformatics*, vol. 9, no. 1, Jun. 2017. doi: 10.1186/s13321-017-0226-y.

[156] A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter, "DeepTox: Toxicity prediction using deep learning," *Frontiers in Environmental Science*, vol. 3, Feb. 2016. doi: 10.3389/fenvs.2015.00080.

[157] S. Kearnes, B. Goldman, and V. Pande, *Modeling industrial admet data with multitask networks*, 2016. doi: 10.48550/ARXIV.1606.08793.

[158] B. Ramsundar, B. Liu, Z. Wu, *et al.*, "Is multitask deep learning practical for pharma?" *Journal of Chemical Information and Modeling*, vol. 57, no. 8, pp. 2068–2076, Aug. 2017. doi: 10.1021/acs.jcim.7b00146.

[159] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of Chemical Information and Modeling*, vol. 55, no. 2, pp. 263–274, Feb. 2015. doi: 10.1021/ci500747n.

[160] Y. Xu, Z. Dai, F. Chen, S. Gao, J. Pei, and L. Lai, "Deep learning for drug-induced liver injury," *Journal of Chemical Information and Modeling*, vol. 55, no. 10, pp. 2085–2093, Oct. 2015. doi: 10.1021/acs.jcim.5b00238.

[161] A. Lusci, G. Pollastri, and P. Baldi, "Deep architectures and deep learning in chemoinformatics: The prediction of aqueous solubility for drug-like molecules," *Journal of Chemical Information and Modeling*, vol. 53, no. 7, pp. 1563–1575, Jul. 2013. doi: 10.1021/ci400187y.

[162] P. Carbonell, P. Parutto, J. Herisson, S. B. Pandit, and J.-L. Faulon, "XTMS: Pathway design in an eXTended metabolic space," *Nucleic Acids Research*, vol. 42, no. W1, W389–W394, May 2014. doi: 10.1093/nar/gku362.

[163] J. Law, Z. Zsoldos, A. Simon, *et al.*, "Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation," *Journal of Chemical Information and Modeling*, vol. 49, no. 3, pp. 593–602, Feb. 2009. doi: 10.1021/ci800228y.

[164] J. Gao, L. B. M. Ellis, and L. P. Wackett, "The university of minnesota pathway prediction system: Multi-level prediction and visualization," *Nucleic Acids Research*, vol. 39, no. suppl, W406–W411, Apr. 2011. doi: 10.1093/nar/gkr200.

[165] M. A. Campodonico, B. A. Andrews, J. A. Asenjo, B. O. Palsson, and A. M. Feist, "Generation of an atlas for commodity chemical production in escherichia coli and a novel pathway prediction algorithm, GEM-path," *Metabolic Engineering*, vol. 25, pp. 140–158, Sep. 2014. doi: 10.1016/j.ymben.2014.07.009.

[166] DayLight, *Daylight theory manual*, https://www.daylight.com/meetings/summerschool01/course/basics/smirks.html, Accessed 15/06/2024.

[167] M. A. Kayala and P. Baldi, "ReactionPredictor: Prediction of complex chemical reactions at the mechanistic level using machine learning," *Journal of Chemical Information and Modeling*, vol. 52, no. 10, pp. 2526–2540, Oct. 2012. doi: 10.1021/ci3003039.

[168] J. N. Wei, D. Duvenaud, and A. Aspuru-Guzik, "Neural networks for the prediction of organic chemistry reactions," *ACS Central Science*, vol. 2, no. 10, pp. 725–732, Oct. 2016. doi: 10.1021/acscentsci.6b00219.

[169] M. H. S. Segler and M. P. Waller, "Modelling chemical reasoning to predict and invent reactions," *Chemistry - A European Journal*, vol. 23, no. 25, pp. 6118–6128, Jan. 2017. doi: 10.1002/chem.201604556.

[170] A. Gupta, "Predicting chemical reaction type and reaction products with recurrent neural networks," 2017.

[171] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen, "Prediction of organic reaction outcomes using machine learning," *ACS Central Science*, vol. 3, no. 5, pp. 434–443, Apr. 2017. doi: 10.1021/acscentsci.7b00064.

[172] J. Correia, J. Capela, and M. Rocha, "Deepmol: An automated machine and deep learning framework for computational chemistry," Jun. 2024. doi: 10.1101/2024.05.27.595849.

[173] G. Hessler and K.-H. Baringhaus, "Artificial intelligence in drug design," *Molecules*, vol. 23, no. 10, p. 2520, Oct. 2018. doi: 10.3390/molecules23102520.

[174] J. Shen and C. A. Nicolaou, "Molecular property prediction: Recent trends in the era of artificial intelligence," *Drug Discovery Today: Technologies*, vol. 32-33, pp. 29–36, Dec. 2019. doi: 10.1016/j.ddtec.2020.05.001.

[175] J. Gasteiger, "Chemistry in times of artificial intelligence," *ChemPhysChem*, vol. 21, no. 20, pp. 2233–2242, Sep. 2020. doi: 10.1002/cphc.202000518.

[176] W. P. Walters and R. Barzilay, "Applications of deep learning in molecule generation and molecular property prediction," *Accounts of Chemical Research*, vol. 54, no. 2, pp. 263–270, Dec. 2020. doi: 10.1021/acs.accounts.0c00699.

[177] G. Montavon, M. Rupp, V. Gobre, *et al.*, "Machine learning of molecular electronic properties in chemical compound space," *New Journal of Physics*, vol. 15, p. 095 003, 9 Sep. 2013, issn: 1367-2630. doi: 10.1088/1367-2630/15/9/095003.

[178] A. Tkatchenko, "Machine learning for chemical discovery," *Nature Communications*, vol. 11, p. 4125, 1 Aug. 2020, issn: 2041-1723. doi: 10.1038/s41467-020-17844-8.

[179] M. Karelson, V. S. Lobanov, and A. R. Katritzky, "Quantum-chemical descriptors in qsar/qspr studies," *Chemical Reviews*, vol. 96, pp. 1027–1044, 3 Jan. 1996, issn: 0009-2665. doi: 10.1021/cr950202r.

[180] W. M. Berhanu, G. G. Pillai, A. A. Oliferenko, and A. R. Katritzky, "Quantitative structure-activity/property relationships: The ubiquitous links between cause and effect," *ChemPlusChem*, vol. 77, pp. 507–517, 7 Jul. 2012, issn: 21926506. doi: 10.1002/cplu.201200038.

[181] P. C. S. Costa, J. S. Evangelista, I. Leal, and P. C. M. L. Miranda, "Chemical graph theory for property modeling in qsar and qspr—charming qsar & qspr," *Mathematics*, vol. 9, no. 1, p. 60, Dec. 2020, issn: 2227-7390. doi: 10.3390/math9010060.

[182] I. L. Guidotti, A. Neis, D. P. Martinez, F. K. Seixas, K. Machado, and F. S. Kremer, "Bambu and its applications in the discovery of active molecules against melanoma," *Journal of Molecular Graphics and Modelling*, vol. 124, p. 108 564, Nov. 2023, issn: 1093-3263. doi: 10.1016/j.jmgm.2023.108564.

[183] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, "Computational methods in drug discovery," *Pharmacological Reviews*, vol. 66, no. 1, E. L. Barker, Ed., pp. 334–395, Dec. 2013, issn: 1521-0081. doi: 10.1124/pr.112.007336.

[184] Z. Wu, M. Zhu, Y. Kang, *et al.*, "Do we need different machine learning algorithms for qsar modeling? a comprehensive assessment of 16 machine learning algorithms on 14 qsar data sets," *Briefings in Bioinformatics*, vol. 22, 4 Jul. 2021, issn: 1467-5463. doi: 10.1093/bib/bbaa321.

[185] Y. Xu, "Deep neural networks for qsar," in *Methods in Molecular Biology*. Springer US, Nov. 2021, pp. 233–260, isbn: 9781071617878. doi: 10.1007/978-1-0716-1787-8_10.

[186] Z. Li, M. Jiang, S. Wang, and S. Zhang, "Deep learning methods for molecular representation and property prediction," *Drug Discovery Today*, vol. 27, no. 12, p. 103 373, Dec. 2022. doi: 10.1016/j.drudis.2022.103373.

[187] Á. Orosz, K. Héberger, and A. Rácz, "Comparison of descriptor- and fingerprint sets in machine learning models for ADME-tox targets," *Frontiers in Chemistry*, vol. 10, Jun. 2022. doi: 10.3389/fchem.2022.852893.

[188] G. Turon, J. Hlozek, J. Woodland, and et al., "First fully-automated ai/ml virtual screening cascade implemented at a drug discovery centre in africa," *Nat Commun*, vol. 14, p. 5736, 2023. doi: 10.1038/s41467-023-41512-2.

[189] G. Landrum, P. Tosco, B. Kelley, *et al.*, *Rdkit/rdkit: 2022_09_3 (q3 2022) release*, 2022. doi: 10.5281/ZENODO.7415128.

[190] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[191] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[192] B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu, *Deep Learning for the Life Sciences*. O'Reilly Media, 2019, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[193] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, ACM, Jul. 2019. doi: 10.1145/3292500.3330701.

[194] K. Huang, T. Fu, W. Gao, *et al.*, "Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung, Eds., vol. 1, Curran, 2021.

[195] D. Probst and J.-L. Reymond, "Smilesdrawer: Parsing and drawing smiles-encoded molecular structures using client-side javascript," *Journal of Chemical Information and Modeling*, vol. 58, no. 1, pp. 1–7, Jan. 2018, issn: 1549-960X. doi: 10.1021/acs.jcim.7b00425.

[196] J. Gasteiger, J. Sadowski, J. Schuur, P. Selzer, L. Steinhauer, and V. Steinhauer, *Chemical information in 3d space*, Jan. 1996. doi: 10.1021/ci960343+.

[197] J. Polanski and J. Gasteiger, "Computer representation of chemical compounds," in *Handbook of Computational Chemistry*. Springer International Publishing, 2017, pp. 1997–2039, isbn: 9783319272825. doi: 10.1007/978-3-319-27282-5_50.

[198] D. S. Wigh, J. M. Goodman, and A. A. Lapkin, "A review of molecular representation in the age of machine learning," *WIREs Computational Molecular Science*, vol. 12, no. 5, Feb. 2022, issn: 1759-0884. doi: 10.1002/wcms.1603.

[199] K. Mansouri, C. M. Grulke, A. M. Richard, R. S. Judson, and A. J. Williams, "An automated curation procedure for addressing chemical errors and inconsistencies in public datasets used in qsar modelling," *SAR and QSAR in Environmental Research*, vol. 27, no. 11, pp. 911–937, Nov. 2016, issn: 1029-046X. doi: 10.1080/1062936x.2016.1253611.

[200] D. Fourches, E. Muratov, and A. Tropsha, "Trust, but verify: On the importance of chemical structure curation in cheminformatics and QSAR modeling research," *Journal of Chemical Information and Modeling*, vol. 50, no. 7, pp. 1189–1204, Jun. 2010. doi: 10.1021/ci100176x.

[201] A. P. Bento, A. Hersey, E. Félix, *et al.*, "An open source chemical structure curation pipeline using RDKit," *Journal of Cheminformatics*, vol. 12, no. 1, Sep. 2020. doi: 10.1186/s13321-020-00456-1.

[202] V. D. Hähnke, S. Kim, and E. E. Bolton, "PubChem chemical structure standardization," *Journal of Cheminformatics*, vol. 10, no. 1, Aug. 2018. doi: 10.1186/s13321-018-0293-8.

[203] K. Karapetyan, C. Batchelor, D. Sharpe, V. Tkachenko, and A. J. Williams, "The chemical validation and standardization platform (CVSP): Large-scale automated validation of chemical structure datasets," *Journal of Cheminformatics*, vol. 7, no. 1, Jun. 2015. doi: 10.1186/s13321-015-0072-8.

[204] R. Todeschini and V. Consonni, *Molecular Descriptors for Chemoinformatics*. Wiley, Jul. 2009, isbn: 9783527628766. doi: 10.1002/9783527628766.

[205] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors*. Wiley, Sep. 2000, isbn: 9783527613106. doi: 10.1002/9783527613106.

[206] H. Moriwaki, Y.-S. Tian, N. Kawashita, and T. Takagi, "Mordred: A molecular descriptor calculator," *Journal of Cheminformatics*, vol. 10, no. 1, Feb. 2018, issn: 1758-2946. doi: 10.1186/s13321-018-0258-y.

[207] X. Li and D. Fourches, "Smiles pair encoding: A data-driven substructure tokenization algorithm for deep learning," *Journal of Chemical Information and Modeling*, vol. 61, pp. 1560–1569, 4 Apr. 2021, issn: 1549-9596. doi: 10.1021/acs.jcim.0c01127.

[208] N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," 2018. doi: 10.48550/ARXIV.1805.11973.

[209] G. Montavon, K. Hansen, S. Fazli, *et al.*, "Learning invariant representations of molecules for atomization energy prediction," *Advances in neural information processing systems*, vol. 25, 2012.

[210] M. B. Kursa and W. R. Rudnicki, "Feature selection with thebboruta/bpackage," *Journal of Statistical Software*, vol. 36, no. 11, 2010. doi: 10.18637/jss.v036.i11.

[211] D. Baptista, J. Correia, B. Pereira, and M. Rocha, "Evaluating molecular representations in machine learning models for drug response prediction and interpretability," *Journal of Integrative Bioinformatics*, vol. 19, no. 3, Aug. 2022. doi: 10.1515/jib-2022-0006.

[212] J. Capela, J. Correia, V. Pereira, and M. Rocha, "Development of deep learning approaches to predict relationships between chemical structures and sweetness," in *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2022. doi: 10.1109/ijcnn55064.2022.9891992.

[213] D. Butina, "Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets," *Journal of Chemical Information and Computer Sciences*, vol. 39, no. 4, pp. 747–750, Jun. 1999, issn: 0095-2338. doi: 10.1021/ci9803381.

[214] C. Xu, F. Cheng, L. Chen, *et al.*, "In silico prediction of chemical ames mutagenicity," *Journal of Chemical Information and Modeling*, vol. 52, no. 11, pp. 2840–2847, Oct. 2012, issn: 1549-960X. doi: 10.1021/ci300400a.

[215] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, Mar. 2019. doi: 10.1186/s40537-019-0192-5.

[216] S. Korkmaz, "Deep learning-based imbalanced data classification for drug discovery," *Journal of Chemical Information and Modeling*, vol. 60, no. 9, pp. 4180–4190, Jun. 2020. doi: 10.1021/acs.jcim.9b01162.

[217] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[218] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, issn: 1076-9757. doi: 10.1613/jair.953.

[219] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study," in *WOB*, 2003.

[220] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004. doi: 10.1145/1007730.1007735.

[221] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[222] C.-Y. Ma, S.-Y. Yang, H. Zhang, M.-L. Xiang, Q. Huang, and Y.-Q. Wei, "Prediction models of human plasma protein binding rate and oral bioavailability derived by using ga–cg–svm method," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 47, no. 4–5, pp. 677–682, Aug. 2008, issn: 0731-7085. doi: 10.1016/j.jpba.2008.03.023.

[223] D. Baptista, J. Correia, B. Pereira, and M. Rocha, "A comparison of different compound representations for drug sensitivity prediction," in *Practical Applications of Computational Biology &amp Bioinformatics, 15th International Conference (PACBB 2021)*, Springer International Publishing, Aug. 2021, pp. 145–154. doi: 10.1007/978-3-030-86258-9_15.

[224] S. Kim, J. Chen, T. Cheng, *et al.*, "PubChem 2023 update," *Nucleic Acids Research*, vol. 51, no. D1, pp. D1373–D1380, Oct. 2022. doi: 10.1093/nar/gkac956.

[225] J. Capela, J. Cheixo, D. de Ridder, M. Rocha, and O. Dias, "Automated machine learning to predict the precursors of plant specialized metabolites," *Manuscript submitted*, 2024.

[226] S. R. Tangadpalliwar, S. Vishwakarma, R. Nimbalkar, and P. Garg, "ChemSuite: A package for chemoinformatics calculations and machine learning," *Chemical Biology & Drug Design*, vol. 93, no. 5, pp. 960–964, May 2019, issn: 1747-0277. doi: 10.1111/cbdd.13479.

[227] A. J. Minnich, K. McLoughlin, M. Tse, *et al.*, "AMPL: A Data-Driven Modeling Pipeline for Drug Discovery," *Journal of Chemical Information and Modeling*, vol. 60, no. 4, pp. 1955–1968, Apr. 2020, issn: 1549-9596. doi: 10.1021/acs.jcim.9b01053.

[228] M. Korshunova, B. Ginsburg, A. Tropsha, and O. Isayev, "OpenChem: A Deep Learning Toolkit for Computational Chemistry and Drug Design," *Journal of Chemical Information and Modeling*, vol. 61, no. 1, pp. 7–13, Jan. 2021, issn: 1549-9596. doi: 10.1021/acs.jcim.0c00971.

[229] B. P. Brown, O. Vu, A. R. Geanes, *et al.*, "Introduction to the BioChemical Library (BCL): An Application-Based Open-Source Toolkit for Integrated Cheminformatics and Machine Learning in

Computer-Aided Drug Discovery," *Frontiers in Pharmacology*, vol. 13, Feb. 2022, issn: 1663-9812. doi: 10.3389/fphar.2022.833099.

[230] J. Correia, R. Carreira, V. Pereira, and M. Rocha, "Predicting the number of biochemical transformations needed to synthesize a compound," in *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2022. doi: 10.1109/ijcnn55064.2022.9892124.

[231] A. FLEMING, "The discovery of penicillin," *British Medical Bulletin*, vol. 2, no. 1, pp. 4–5, 1944, issn: 0007-1420. doi: 10.1093/oxfordjournals.bmb.a071032.

[232] M. M. El-Sheekh and H. Y. El-Kassas, "Algal production of nano-silver and gold: Their antimicrobial and cytotoxic activities: A review," *Journal of Genetic Engineering and Biotechnology*, vol. 14, no. 2, pp. 299–310, Dec. 2016, issn: 1687-157X. doi: 10.1016/j.jgeb.2016.09.008.

[233] J. Du, Z. Shao, and H. Zhao, "Engineering microbial factories for synthesis of value-added products," *Journal of Industrial Microbiology & Biotechnology*, vol. 38, no. 8, pp. 873–890, Apr. 2011, issn: 1476-5535. doi: 10.1007/s10295-011-0970-3.

[234] S. Comba, A. Arabolaza, and H. Gramajo, "Emerging engineering principles for yield improvement in microbial cell design," *Computational and Structural Biotechnology Journal*, vol. 3, no. 4, e201210016, Oct. 2012, issn: 2001-0370. doi: 10.5936/csbj.201210016.

[235] H. Sun, H. Zhang, E. L. Ang, and H. Zhao, "Biocatalysis for the synthesis of pharmaceuticals and pharmaceutical intermediates," *Bioorganic & Medicinal Chemistry*, vol. 26, no. 7, pp. 1275–1284, Apr. 2018, issn: 0968-0896. doi: 10.1016/j.bmc.2017.06.043.

[236] S. Y. Oh, S. Y. Youn, M. S. Park, N. I. Baek, and G. E. Ji, "Synthesis of stachyobifiose using bifidobacterial -galactosidase purified from recombinant escherichia coli," *Journal of Agricultural and Food Chemistry*, vol. 66, no. 5, pp. 1184–1190, Jan. 2018, issn: 1520-5118. doi: 10.1021/acs.jafc.7b04703.

[237] J. Zaldivar, J. Nielsen, and L. Olsson, "Fuel ethanol production from lignocellulose: A challenge for metabolic engineering and process integration," *Applied Microbiology and Biotechnology*, vol. 56, no. 1–2, pp. 17–34, Jul. 2001, issn: 1432-0614. doi: 10.1007/s002530100624.

[238] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[239] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018, issn: 1687-5273. doi: 10.1155/2018/7068349.

[240] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019, issn: 2169-3536. doi: 10.1109/access.2019.2896880.

[241] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in Bioinformatics*, bbw068, Jul. 2016, issn: 1477-4054. doi: 10.1093/bib/bbw068.

[242] K. V. Presnell and H. S. Alper, "Systems metabolic engineering meets machine learning: A new era for data□driven metabolic engineering," *Biotechnology Journal*, vol. 14, no. 9, May 2019, issn: 1860-7314. doi: 10.1002/biot.201800416.

[243] N. Hadadi and V. Hatzimanikatis, "Design of computational retrobiosynthesis tools for the design of de novo synthetic pathways," *Current Opinion in Chemical Biology*, vol. 28, pp. 99–104, Oct. 2015. doi: 10.1016/j.cbpa.2015.06.025.

[244] B. Liu, B. Ramsundar, P. Kawthekar, *et al.*, "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS Central Science*, vol. 3, no. 10, pp. 1103–1113, Sep. 2017. doi: 10.1021/acscentsci.7b00303.

[245] S. Szymkuć, E. P. Gajewska, T. Klucznik, *et al.*, "Computer-assisted synthetic planning: The end of the beginning," *Angewandte Chemie International Edition*, vol. 55, no. 20, pp. 5904–5937, Apr. 2016. doi: 10.1002/anie.201506101.

[246] H. Yim, R. Haselbeck, W. Niu, *et al.*, "Metabolic engineering of escherichia coli for direct production of 1, 4-butanediol," *Nature Chemical Biology*, vol. 7, no. 7, pp. 445–452, May 2011. doi: 10.1038/nchembio.580.

[247] P. Carbonell, P. Parutto, C. Baudier, C. Junot, and J.-L. Faulon, "Retropath: Automated pipeline for embedded metabolic circuits," *ACS Synthetic Biology*, vol. 3, no. 8, pp. 565–577, Oct. 2013, issn: 2161-5063. doi: 10.1021/sb4001273.

[248] D. A. Pertusi, A. E. Stine, L. J. Broadbelt, and K. E. Tyo, "Efficient searching and annotation of metabolic networks using chemical similarity," *Bioinformatics*, vol. 31, no. 7, pp. 1016–1024, Dec. 2014. doi: 10.1093/bioinformatics/btu760.

[249] J. G. Jeffryes, R. L. Colastani, M. Elbadawi-Sidhu, *et al.*, "MINEs: Open access databases of computationally predicted enzyme promiscuity products for untargeted metabolomics," *Journal of Cheminformatics*, vol. 7, no. 1, Aug. 2015. doi: 10.1186/s13321-015-0087-1.

[250] S. Moretti, O. Martin, T. Van Du Tran, A. Bridge, A. Morgat, and M. Pagni, "MetaNetX/MNXref – reconciliation of metabolites and biochemical reactions to bring together genome-scale metabolic networks," *Nucleic Acids Research*, vol. 44, no. D1, pp. D523–D526, Nov. 2015. doi: 10.1093/nar/gkv1117.

[251] J. D. Orth, T. M. Conrad, J. Na, *et al.*, "A comprehensive genome-scale reconstruction of escherichia coli metabolism–2011," *Molecular Systems Biology*, vol. 7, no. 1, p. 535, Jan. 2011. doi: 10.1038/msb.2011.65.

[252] S. Gudmundsson and I. Thiele, "Computationally efficient flux variability analysis," *BMC Bioinformatics*, vol. 11, no. 1, Sep. 2010. doi: 10.1186/1471-2105-11-489.

[253] A. Ebrahim, J. A. Lerman, B. O. Palsson, and D. R. Hyduke, "COBRApy: COnstraints-based reconstruction and analysis for python," *BMC Systems Biology*, vol. 7, no. 1, Aug. 2013. doi: 10.1186/1752-0509-7-74.

[254] Z. Pan, Y. Chen, M. Zhou, T. A. McAllister, and L. L. Guan, "Microbial interaction-driven community differences as revealed by network analysis," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 6000–6008, 2021. doi: 10.1016/j.csbj.2021.10.035.

[255] A. C. Jones, K. D. Hambright, and D. A. Caron, "Ecological patterns among bacteria and microbial eukaryotes derived from network analyses in a low-salinity lake," *Microbial Ecology*, vol. 75, no. 4, pp. 917–929, Nov. 2017. doi: 10.1007/s00248-017-1087-7.

[256] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, Apr. 2010, issn: 1549-960X. doi: 10.1021/ci100050t.

[257] X. Li and D. Fourches, "Inductive transfer learning for molecular activity prediction: Next-gen qsar models with molpmofit," *Journal of Cheminformatics*, vol. 12, no. 1, Apr. 2020, issn: 1758-2946. doi: 10.1186/s13321-020-00430-x.

[258] M. Seo, H. K. Shin, Y. Myung, S. Hwang, and K. T. No, "Development of natural compound molecular fingerprint (nc-mfp) with the dictionary of natural products (dnp) for natural product-based drug development," *Journal of Cheminformatics*, vol. 12, no. 1, Jan. 2020, issn: 1758-2946. doi: 10.1186/s13321-020-0410-3.

[259] P. Morris, R. St. Clair, W. E. Hahn, and E. Barenholtz, "Predicting binding from screening assays with transformer network embeddings," *Journal of Chemical Information and Modeling*, vol. 60, no. 9, pp. 4191–4199, Jun. 2020, issn: 1549-960X. doi: 10.1021/acs.jcim.9b01212.

[260] K. M. Ting, "Confusion matrix," in *Encyclopedia of Machine Learning*. Springer US, 2011, pp. 209–209, isbn: 9780387301648. doi: 10.1007/978-0-387-30164-8_157.

[261] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Advances in Information Retrieval*. Springer Berlin Heidelberg, 2005, pp. 345–359, isbn: 9783540318651. doi: 10.1007/978-3-540-31865-1_25.

[262] J. Miles, *R squared, adjusted r squared*, Sep. 2014. doi: 10.1002/9781118445112.stat06627.

[263] J. Correia, V. Pereira, and M. Rocha, "Combining evolutionary algorithms with reaction rules towards focused molecular design," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '23, ACM, Jul. 2023. doi: 10.1145/3583131.3590413.

[264] H. Lachance, S. Wetzel, K. Kumar, and H. Waldmann, "Charting, navigating, and populating natural product chemical space for drug discovery," *Journal of Medicinal Chemistry*, vol. 55, no. 13, pp. 5989–6001, May 2012. doi: 10.1021/jm300288g.

[265] O. J. Wouters, M. McKee, and J. Luyten, "Estimated research and development investment needed to bring a new medicine to market, 2009-2018," *JAMA*, vol. 323, no. 9, p. 844, Mar. 2020. doi: 10.1001/jama.2020.1166.

[266] W. Yu and A. D. MacKerell, "Computer-aided drug design methods," in *Methods in Molecular Biology*, Springer New York, Nov. 2016, pp. 85–106. doi: 10.1007/978-1-4939-6634-9_5.

[267] E. van der Horst, P. Marqués-Gallego, T. Mulder-Krieger, *et al.*, "Multi-objective evolutionary design of adenosine receptor ligands," *Journal of Chemical Information and Modeling*, vol. 52, no. 7, pp. 1713–1721, Jun. 2012. doi: 10.1021/ci2005115.

[268] R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noé, and D.-A. Clevert, "Efficient multi-objective molecular optimization in a continuous latent space," *Chemical Science*, vol. 10, no. 34, pp. 8016–8024, 2019. doi: 10.1039/c9sc01928f.

[269] Y. Kwon and J. Lee, "MolFinder: An evolutionary algorithm for the global optimization of molecular properties and the extensive exploration of chemical space using SMILES," *Journal of Cheminformatics*, vol. 13, no. 1, Mar. 2021. doi: 10.1186/s13321-021-00501-7.

[270] J. O. Spiegel and J. D. Durrant, "AutoGrow4: An open-source genetic algorithm for de novo drug design and lead optimization," *Journal of Cheminformatics*, vol. 12, no. 1, Apr. 2020. doi: 10.1186/s13321-020-00429-4.

[271] A. Kerstjens and H. D. Winter, "LEADD: Lamarckian evolutionary algorithm for de novo drug design," *Journal of Cheminformatics*, vol. 14, no. 1, Jan. 2022. doi: 10.1186/s13321-022-00582-y.

[272] L. Elend, L. Jacobsen, T. Cofala, *et al.*, "Design of SARS-CoV-2 main protease inhibitors using artificial intelligence and molecular dynamic simulations," *Molecules*, vol. 27, no. 13, p. 4020, Jun. 2022. doi: 10.3390/molecules27134020.

[273] Y. Kwon, J. Yoo, Y.-S. Choi, W.-J. Son, D. Lee, and S. Kang, "Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation," *Journal of Cheminformatics*, vol. 11, no. 1, Nov. 2019. doi: 10.1186/s13321-019-0396-x.

[274] C. Zhang, X. Lyu, Y. Huang, Z. Tang, and Z. Liu, "Molecular graph generation with deep reinforced multitask network and adversarial imitation learning," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, Nov. 2019. doi: 10.1109/bibm47256.2019.8983277.

[275] W. Jin, R. Barzilay, and T. Jaakkola, *Junction tree variational autoencoder for molecular graph generation*, 2018. doi: 10.48550/ARXIV.1802.04364.

[276] X. Liu, K. Ye, H. W. T. van Vlijmen, M. T. M. Emmerich, A. P. IJzerman, and G. J. P. van Westen, "DrugEx v2: De novo design of drug molecules by pareto-based multi-objective reinforcement learning in polypharmacology," *Journal of Cheminformatics*, vol. 13, no. 1, Nov. 2021. doi: 10.1186/s13321-021-00561-9.

[277] K. Kawai, N. Nagata, and Y. Takahashi, "De novo design of drug-like molecules by a fragment-based molecular evolutionary approach," *Journal of Chemical Information and Modeling*, vol. 54, no. 1, pp. 49–56, Jan. 2014. doi: 10.1021/ci400418c.

[278] H. Li, K.-S. Leung, C. H. Chan, H. L. Cheung, and M.-H. Wong, "iSyn: WebGL-based interactive de novo drug design," in *2014 18th International Conference on Information Visualisation*, IEEE, Jul. 2014. doi: 10.1109/iv.2014.10.

[279] J. H. Jensen, "A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space," *Chemical Science*, vol. 10, no. 12, pp. 3567–3572, 2019. doi: 10.1039/c8sc05372c.

[280] J. Leguy, T. Cauchy, M. Glavatskikh, B. Duval, and B. D. Mota, "EvoMol: A flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation," *Journal of Cheminformatics*, vol. 12, no. 1, Sep. 2020. doi: 10.1186/s13321-020-00458-z.

[281] N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono, and K. Tsuda, "Population-based de novo molecule generation, using grammatical evolution," *Chemistry Letters*, vol. 47, no. 11, pp. 1431–1434, Nov. 2018. doi: 10.1246/cl.180665.

[282] N. C. Firth, B. Atrash, N. Brown, and J. Blagg, "MOARF, an integrated workflow for multiobjective optimization: Implementation, synthesis, and biological evaluation," *Journal of Chemical Information and Modeling*, vol. 55, no. 6, pp. 1169–1180, Jun. 2015. doi: 10.1021/acs.jcim.5b00073.

[283] P. Polishchuk, "CReM: Chemically reasonable mutations framework for structure generation," *Journal of Cheminformatics*, vol. 12, no. 1, Apr. 2020. doi: 10.1186/s13321-020-00431-w.

[284] A. Benítez-Hidalgo, A. J. Nebro, J. García-Nieto, I. Oregi, and J. D. Ser, "jMetalPy: A python framework for multi-objective optimization with metaheuristics," *Swarm and Evolutionary Computation*, vol. 51, p. 100 598, Dec. 2019. doi: 10.1016/j.swevo.2019.100598.

[285] J. Strutz, K. M. Shebek, L. J. Broadbelt, and K. E. J. Tyo, "MINE 2.0: Enhanced biochemical coverage for peak identification in untargeted metabolomics," *Bioinformatics*, vol. 38, no. 13, Z. Lu, Ed., pp. 3484–3487, May 2022. doi: 10.1093/bioinformatics/btac331.

[286] E. Aarts and J. Korst, *Simulated annealing and Boltzmann machines : a stochastic approach to combinatorial optimization and neural computing* (Wiley-Interscience series in discrete mathematics and optimization), English. Wiley, 1989, isbn: 0-471-92146-7.

[287] K. Sastry, D. Goldberg, and G. Kendall, "Genetic algorithms," in *Search Methodologies*, Springer US, pp. 97–125. doi: 10.1007/0-387-28356-0_4.

[288] D. Sudholt, "Local search in evolutionary algorithms: The impact of the local search frequency," in *Algorithms and Computation*, Springer Berlin Heidelberg, 2006, pp. 359–368. doi: 10.1007/11940128_37.

[289] V. J. Amuso and J. Enslin, "The strength pareto evolutionary algorithm 2 (SPEA2) applied to simultaneous multi- mission waveform design," in *2007 International Waveform Diversity and Design Conference*, IEEE, Jun. 2007. doi: 10.1109/wddc.2007.4339452.

[290] P. Murugan, S. Kannan, and S. Baskar, "Application of NSGA-II algorithm to single-objective transmission constrained generation expansion planning," *IEEE Transactions on Power Systems*, vol. 24, no. 4, pp. 1790–1797, Nov. 2009. doi: 10.1109/tpwrs.2009.2030428.

[291] P. Ertl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, Jun. 2009. doi: 10.1186/1758-2946-1-8.

[292] S. Bühlmann and J.-L. Reymond, "ChEMBL-likeness score and database GDBChEMBL," *Frontiers in Chemistry*, vol. 8, Feb. 2020. doi: 10.3389/fchem.2020.00046.

[293] D. Mendez, A. Gaulton, A. P. Bento, *et al.*, "ChEMBL: Towards direct deposition of bioassay data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D930–D940, Nov. 2018. doi: 10.1093/nar/gky1075.

[294] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, *Graph convolutional policy network for goal-directed molecular graph generation*, 2018. doi: 10.48550/ARXIV.1806.02473.

[295] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley, "Optimization of molecules via deep reinforcement learning," *Scientific Reports*, vol. 9, no. 1, Jul. 2019. doi: 10.1038/s41598-019-47148-x.

[296] J. Lai, J. Hu, Y. Wang, *et al.*, "Privileged scaffold analysis of natural products with deep learning-based indication prediction model," *Molecular Informatics*, vol. 39, no. 11, p. 2 000 057, May 2020. doi: 10.1002/minf.202000057.

[297] J. Degen, C. Wegscheid-Gerlach, A. Zaliani, and M. Rarey, "On the art of compiling and using 'drug-like' chemical fragment spaces," *ChemMedChem*, vol. 3, no. 10, pp. 1503–1507, Oct. 2008. doi: 10.1002/cmdc.200800178.

[298] M. García-Ortegón, G. N. C. Simm, A. J. Tripp, J. M. Hernández-Lobato, A. Bender, and S. Bacallado, "Dockstring: Easy molecular docking yields better benchmarks for ligand design," *Journal of Chemical Information and Modeling*, vol. 62, no. 15, pp. 3486–3502, 2022. doi: 10.1021/acs.jcim.1c01334.

[299] O. Trott and A. J. Olson, "AutoDock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of Computational Chemistry*, NA–NA, 2009. doi: 10.1002/jcc.21334.

[300] M. W. Chang, C. Ayeni, S. Breuer, and B. E. Torbett, "Virtual screening for HIV protease inhibitors: A comparison of AutoDock 4 and vina," *PLoS ONE*, vol. 5, no. 8, H. W. van Veen, Ed., e11955, Aug. 2010. doi: 10.1371/journal.pone.0011955.

[301] K. Uppal, D. I. Walker, K. Liu, S. Li, Y.-M. Go, and D. P. Jones, "Computational metabolomics: A framework for the million metabolome," *Chemical Research in Toxicology*, vol. 29, no. 12, pp. 1956–1975, Oct. 2016, issn: 1520-5010. doi: 10.1021/acs.chemrestox.6b00179.

[302] G. Piedrafita, M. Keller, and M. Ralser, "The impact of non-enzymatic reactions and enzyme promiscuity on cellular metabolism during (oxidative) stress conditions," *Biomolecules*, vol. 5, no. 3, pp. 2101–2122, Sep. 2015, issn: 2218-273X. doi: 10.3390/biom5032101.

[303] M. Afzaal, F. Saeed, Y. A. Shah, *et al.*, "Human gut microbiota in health and disease: Unveiling the relationship," *Frontiers in Microbiology*, vol. 13, Sep. 2022, issn: 1664-302X. doi: 10.3389/fmicb.2022.999001.

[304] B. M. A. van Vugt-Lussenburg, L. Capinha, J. Reinen, *et al.*, ""commandeuring" xenobiotic metabolism: Advances in understanding xenobiotic metabolism," *Chemical Research in Toxicology*, vol. 35, no. 7, pp. 1184–1201, Jun. 2022, issn: 1520-5010. doi: 10.1021/acs.chemrestox.2c00067.

[305] M. Zimmermann, M. Zimmermann-Kogadeeva, R. Wegmann, and A. L. Goodman, "Mapping human microbiome drug metabolism by gut bacteria and their genes," *Nature*, vol. 570, no. 7762, pp. 462–467, Jun. 2019, issn: 1476-4687. doi: 10.1038/s41586-019-1291-3.

[306] I. J. Dikeocha, A. M. Al□Kabsi, M. Miftahussurur, and M. A. Alshawsh, "Pharmacomicrobiomics: Influence of gut microbiota on drug and xenobiotic metabolism," *The FASEB Journal*, vol. 36, no. 6, May 2022, issn: 1530-6860. doi: 10.1096/fj.202101986r.

[307] A.-H. M. Emwas, "The strengths and weaknesses of nmr spectroscopy and mass spectrometry with particular focus on metabolomics research," in *Metabonomics*. Springer New York, 2015, pp. 161–193, isbn: 9781493923779. doi: 10.1007/978-1-4939-2377-9_13.

[308] L. Olsen, M. Montefiori, K. P. Tran, and F. S. Jørgensen, "Smartcyp 3.0: Enhanced cytochrome p450 site-of-metabolism prediction server," *Bioinformatics*, vol. 35, no. 17, A. Valencia, Ed., pp. 3174–3175, Jan. 2019, issn: 1367-4811. doi: 10.1093/bioinformatics/btz037.

[309] S. Hwang, H. K. Shin, S. E. Shin, *et al.*, "Premetabo: An in silico phase i and ii drug metabolism prediction platform," *Drug Metabolism and Pharmacokinetics*, vol. 35, no. 4, pp. 361–367, Aug. 2020, issn: 1347-4367. doi: 10.1016/j.dmpk.2020.05.007.

[310] P. Banerjee, M. Dunkel, E. Kemmler, and R. Preissner, "Supercypspred—a web server for the prediction of cytochrome activity," *Nucleic Acids Research*, vol. 48, no. W1, W580–W585, Mar. 2020, issn: 1362-4962. doi: 10.1093/nar/gkaa166.

[311] J. Wicker, K. Fenner, L. Ellis, L. Wackett, and S. Kramer, "Predicting biodegradation products and pathways: A hybrid knowledge- and machine learning-based approach," *Bioinformatics*, vol. 26, no. 6, pp. 814–821, Jan. 2010, issn: 1367-4803. doi: 10.1093/bioinformatics/btq024.

[312] Q. Zhao, Y. Chen, W. Huang, H. Zhou, and W. Zhang, "Drug-microbiota interactions: An emerging priority for precision medicine," *Signal Transduction and Targeted Therapy*, vol. 8, no. 1, Oct. 2023, issn: 2059-3635. doi: 10.1038/s41392-023-01619-w.

[313] Y. Imoto, Y. Mino, T. Naito, T. Ono, and J. Kawakami, "Simultaneous determination of itraconazole and its cyp3a4-mediated metabolites including n-desalkyl itraconazole in human plasma

using liquid chromatography-tandem mass spectrometry and its clinical application," *Journal of Pharmaceutical Health Care and Sciences*, vol. 6, no. 1, May 2020, issn: 2055-0294. doi: 10.1186/s40780-020-00167-7.

[314] L. M. M. Vermeer, C. D. Isringhausen, B. W. Ogilvie, and D. B. Buckley, "Evaluation of ketoconazole and its alternative clinical cyp3a4/5 inhibitors as inhibitors of drug transporters: The in vitro effects of ketoconazole, ritonavir, clarithromycin, and itraconazole on 13 clinically-relevant drug transporters," *Drug Metabolism and Disposition*, vol. 44, no. 3, pp. 453–459, Dec. 2015, issn: 1521-009X. doi: 10.1124/dmd.115.067744.

[315] N. Isoherranen, K. L. Kunze, K. E. Allen, W. L. Nelson, and K. E. Thummel, "Role of itraconazole metabolites in cyp3a4 inhibition," *Drug Metabolism and Disposition*, vol. 32, no. 10, pp. 1121–1131, Jul. 2004, issn: 1521-009X. doi: 10.1124/dmd.104.000315.

[316] Z. Ni, A. E. Stine, K. E. Tyo, and L. J. Broadbelt, "Curating a comprehensive set of enzymatic reaction rules for efficient novel biosynthetic pathway design," *Metabolic Engineering*, vol. 65, pp. 79–87, May 2021, issn: 1096-7176. doi: 10.1016/j.ymben.2021.02.006.

[317] A. Chakraborty, A. Thakkar, A. C. Vaucher, and T. Laino, "Data-driven reaction template fingerprints," Nov. 2022. doi: 10.26434/chemrxiv-2022-4kzp1.

[318] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of mdl keys for use in drug discovery," *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 6, pp. 1273–1280, Sep. 2002, issn: 0095-2338. doi: 10.1021/ci010132r.

[319] *Daylight Theory: Fingerprints — daylight.com*, https://www.daylight.com/dayhtml/doc/theory/theory.finger.html, [Accessed 27-07-2024].

[320] R. E. Carhart, D. H. Smith, and R. Venkataraghavan, "Atom pairs as molecular features in structure-activity studies: Definition and applications," *Journal of Chemical Information and Computer Sciences*, vol. 25, no. 2, pp. 64–73, May 1985, issn: 1520-5142. doi: 10.1021/ci00046a002.

[321] R. Todeschini and V. Consonni, *Descriptors from molecular geometry*, Aug. 2003. doi: 10.1002/9783527618279.ch37.

[322] N. C. Firth, N. Brown, and J. Blagg, "Plane of best fit: A novel method to characterize the three-dimensionality of molecules," *Journal of Chemical Information and Modeling*, vol. 52, no. 10, pp. 2516–2525, Sep. 2012, issn: 1549-960X. doi: 10.1021/ci300293f.

[323] G. A. Arteca, *Molecular shape descriptors*, Jan. 1996. doi: 10.1002/9780470125861.ch5.

[324] A. Baumgärtner, "Shapes of flexible vesicles at constant volume," *The Journal of Chemical Physics*, vol. 98, no. 9, pp. 7496–7501, May 1993, issn: 1089-7690. doi: 10.1063/1.464689.

[325] W. H. B. Sauer and M. K. Schwarz, "Molecular shape diversity of combinatorial libraries: a prerequisite for broad bioactivity," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 3, pp. 987–1003, Mar. 2003, issn: 0095-2338. doi: 10.1021/ci025599w.

[326] B. Chen, R. Barzilay, and T. Jaakkola, "Path-augmented graph transformer network," 2019. doi: 10.48550/ARXIV.1905.12712.

[327] G. B. Goh, C. Siegel, A. Vishnu, and N. Hodas, "Using rule-based labels for weak supervised learning: A chemnet for transferable chemical property prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, ACM, Jul. 2018. doi: 10.1145/3219819.3219838.

[328] K. Yang, K. Swanson, W. Jin, *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of Chemical Information and Modeling*, vol. 59, no. 8, pp. 3370–3388, Jul. 2019, issn: 1549-960X. doi: 10.1021/acs.jcim.9b00237.

[329] Ł. Maziarka, T. Danel, S. Mucha, K. Rataj, J. Tabor, and S. Jastrz□bski, "Molecule attention transformer," 2020. doi: 10.48550/ARXIV.2002.08264.

[330] *StandardScaler — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Accessed 27-07-2024].

[331] *MinMaxScaler — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html [Accessed 27-07-2024].

[332] *RobustScaler — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html [Accessed 27-07-2024].

[333] *PolynomialFeatures — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html [Accessed 27-07-2024].

[334] *Normalizer — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html [Accessed 27-07-2024].

[335] *Binarizer — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Binarizer.html [Accessed 27-07-2024].

[336] *KernelCenterer — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KernelCenterer.html [Accessed 27-07-2024].

[337] *QuantileTransformer — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html [Accessed 27-07-2024].

[338] *PowerTransformer — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html [Accessed 27-07-2024].

[339] *VarianceThreshold — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html [Accessed 27-07-2024].

[340] *SelectKBest — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html [Accessed 27-07-2024].

[341] *SelectPercentile — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html[Accessed 27-07-2024].

[342] *RFECV — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html [Accessed 27-07-2024].

[343] *SelectFromModel — scikit-learn.org*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html [Accessed 27-07-2024].

[344] G. W. Bemis and M. A. Murcko, "The properties of known drugs. 1. molecular frameworks," *Journal of Medicinal Chemistry*, vol. 39, no. 15, pp. 2887–2893, Jan. 1996, issn: 1520-4804. doi: 10.1021/jm9602928.

[345] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2017. doi: 10.48550/ARXIV.1710.10903.

[346] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2016. doi: 10.48550/ARXIV.1609.02907.

[347] Z. Xiong, D. Wang, X. Liu, *et al.*, "Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism," *Journal of Medicinal Chemistry*, vol. 63, no. 16, pp. 8749–8760, Aug. 2019, issn: 1520-4804. doi: 10.1021/acs.jmedchem.9b00959.

[348] M. Chen, C. J. .-. Ju, G. Zhou, *et al.*, "Multifaceted protein–protein interaction prediction based on siamese residual rcnn," *Bioinformatics*, vol. 35, no. 14, pp. i305–i314, Jul. 2019, issn: 1367-4811. doi: 10.1093/bioinformatics/btz328.

[349] O. Vinyals, S. Bengio, and M. Kudlur, *Order matters: Sequence to sequence for sets*, 2015. doi: 10.48550/ARXIV.1511.06391.

[350] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry*, 2017. doi: 10.48550/ARXIV.1704.01212.

[351] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong, "Graph networks as a universal machine learning framework for molecules and crystals," 2018. doi: 10.48550/ARXIV.1812.05055.

[352] S. J. Swamidass, C.-A. Azencott, T.-W. Lin, H. Gramajo, S.-C. Tsai, and P. Baldi, "Influence relevance voting: An accurate and interpretable virtual high throughput screening method," *Journal of Chemical Information and Modeling*, vol. 49, no. 4, pp. 756–766, Mar. 2009, issn: 1549-960X. doi: 10.1021/ci8004379.

[353] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, *et al.*, *Progressive neural networks*, 2016. doi: 10.48550/ARXIV.1606.04671.

[354] C. W. Coley, L. Rogers, W. H. Green, and K. F. Jensen, "SCScore: Synthetic complexity learned from a reaction corpus," *Journal of Chemical Information and Modeling*, vol. 58, no. 2, pp. 252–261, Jan. 2018. doi: 10.1021/acs.jcim.7b00622.

[355] G. B. Goh, C. Siegel, A. Vishnu, N. O. Hodas, and N. Baker, *Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models*, 2017. doi: 10.48550/ARXIV.1706.06689.

[356] G. B. Goh, N. O. Hodas, C. Siegel, and A. Vishnu, *Smiles2vec: An interpretable general-purpose deep neural network for predicting chemical properties*, 2017. doi: 10.48550/ARXIV.1712.02034.

[357] G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik, *Objective-reinforced generative adversarial networks (organ) for sequence generation models*, 2017. doi: 10.48550/ARXIV.1705.10843.

[358] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature Communications*, vol. 8, no. 1, Jan. 2017, issn: 2041-1723. doi: 10.1038/ncomms13890.

A

Supplementary Data for Chapter 3

A.1 Supporting Information - DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry

A.1.1 Extract features

In ML, extracting features from molecules is a common task. Molecular features can be categorized into four types: 0D, 1D, 2D, 3D, and 4D. 0D features provide information about the entire molecule, including properties like atom count, bond count, and molecular weight. 1D features describe substructures within the molecule, such as molecular fingerprints and fragment keys. 2D features capture the molecular topology based on the graph representation, including the number of rings and rotatable bonds. 3D features capture the geometric descriptors of the molecule's three-dimensional structure. Lastly, 4D features introduce an additional dimension to capture interactions between the molecule and an active site or multiple conformational states, such as molecular dynamics.

As we move from 0D to 4D molecular descriptors, the computational cost of feature calculation increases. For instance, generating 3D features involves creating 3D conformers, which can be time-consuming for larger molecules. It's worth noting that certain features may not be applicable to all molecules; for example, 3D features cannot be calculated for molecules lacking a 3D structure.

DeepMol provides a wide set of 1D features, all provided by rdkit. DeepMol includes one of the most famous circular fingerprints, the ECFP, atom pair fingerprints that encode the presence or absence of pairs of atoms in a molecule, as well as the distance between them. Moreover, DeepMol includes layered fingerprints that find all possible paths or subgraphs of specified lengths in the molecule based on the input parameters and compute layers of structural and functional features per molecular subgraph. DeepMol also includes an RDKit-specific fingerprint inspired by public descriptions of the Daylight fingerprint. The fingerprinting algorithm generates molecular fingerprints by identifying subgraphs within a specified size range, hashing each subgraph to create a raw bit that is hashed to fit the fingerprint size. The default scheme for subgraph hashing involves considering factors such as atom types (based on atomic number

and aromaticity), atom degrees in the path, and bond types. Finally, DeepMol includes MACCS keys that encode the presence or absence of certain molecular fragments or substructures in a molecule as a binary bitstring. The fragments used are based on a predefined set of SMARTS patterns, which represent specific substructures or features of a molecule.

DeepMol also provides a set of 0D, a few 1D and 2D descriptors in only one class. Those are enumerated and described in Table 20 and at `https://deepmol.readthedocs.io/en/latest/deepmol_docs/featurization.html#d-1d-and-2d-descriptors`.

As mentioned above, 3D molecular conformations have to be generated or loaded prior to generating 3D descriptors. For this matter, the conformer generation process begins by utilizing the ETKDG algorithm, which is an extension of the Knowledge Distance Geometry method. ETKDG incorporates efficiency enhancements and knowledge-based rules to generate a diverse set of low-energy conformers for small organic molecules. This method combines random sampling and efficient energy evaluations to strike a balance between computational efficiency and conformational coverage. It is widely employed in molecular modelling and drug discovery applications. Subsequently, the Merck Molecular Force Field and Universal Force Field algorithms are employed. These force fields optimize the conformers by calculating the potential energy and atomic forces based on the molecule's geometry. This process guides the conformational search towards more stable conformations.

Once generated, the methods within DeepMol can be used to extract features from the conformers (Supplementary Table 22). These methods encompass *AutoCorr3D* that captures spatial autocorrelation patterns in a molecule, while the Radial Distribution Function characterizes the distribution of particles based on their distances from a reference particle. The plane of best fit determines the optimal plane that fits a set of points, and MORSE utilizes molecular transforms to derive information from atomic coordinates. WHIM descriptors provide a holistic representation of a molecule's structure, while the Radius of Gyration measures its spatial extent. The Inertial Shape Factor, Eccentricity, Asphericity, and Spherocity Index quantify the shape and symmetry of the molecule. Principal Moments of Inertia describe its rotational behavior, and Normalized Principal Moments Ratios provide insight into the relative magnitudes of the principal moments. These descriptors collectively contribute to a comprehensive understanding of a molecule's three-dimensional properties and structural characteristics.

A.1.2 Supplementary tables

Table 20: 0D, 1D and 2D descriptors integrated into only one class in Descriptors

Set of descriptors	**Descriptors**	**Description**
Molecular weight descriptors	MolWt	Molecular weight.
	HeavyAtomMolWt	Average molecular weight of the molecule, ignoring hydrogens.
	ExactMolWt	The exact molecular weight of the molecule.

0D, 1D and 2D descriptors integrated into only one class in Descriptors (**continued**)

Set of descriptors	**Descriptors**	**Description**
QED	QED	Quantitative estimation of drug-likeness - a computational algorithm used to quantitatively assess the drug-likeness of a molecule. It combines various molecular descriptors, including 2D properties, to generate a single numerical score that represents the overall drug-likeness of the molecule.
EState index descriptors	MaxAbsEStateIndex	The MAEstate specifically represents the highest absolute EState index value among all the atoms in a molecule. It indicates the atom with the largest charge magnitude, reflecting its potential reactivity or contribution to chemical properties.
	MaxEStateIndex	The MaxEStateIndex specifically represents the highest EState index value among all the atoms in a molecule. It indicates the atom with the largest charge or electronic density, reflecting its potential reactivity or significance in the molecule's properties.
	MinAbsEStateIndex	The MinAbsEStateIndex specifically represents the lowest absolute EState index value among all the atoms in a molecule.
	MinEStateIndex	The MinEStateIndex represents the lowest EState index value among all the atoms in a molecule.
Electron descriptors	NumValenceElectrons	The number of valence electrons the molecule has.
	NumRadicalElectrons	The number of radical electrons the molecule has.
Charge descriptors	MaxPartialCharge	Maximum partial charge
	MinPartialCharge	Minimum partial charge
	MaxAbsPartialCharge	Maximum absolute partial charge
	MinAbsPartialCharge	Minimum absolute partial charge
BertzCT	BertzCT	Bertz complexity index. It measures the topological complexity or branching of a molecule based on its structural connectivity.
Morgan fingerprint density	FpDensityMorgan1	Quantify the frequency of occurrence of specific substructures within the molecule at a local level, taking into account their immediate surroundings. Higher values of density indicate a higher density of unique substructures in the molecule, while lower values indicate fewer unique substructures or a more uniform distribution of substructures. Densities for Morgan radius 1, 2 and 3.
	FpDensityMorgan2	
	FpDensityMorgan3	
AvgIpc	AvgIpc	The average information content of the coefficients of the characteristic polynomial of the adjacency matrix of a hydrogen-suppressed graph of a molecule.
Ipc	Ipc	The information content of the coefficients of the characteristic polynomial of the adjacency matrix of a hydrogen-suppressed graph of a molecule.
BalabanJ	BalabanJ	Balaban's J index. It quantifies the molecular topological structure by considering the connectivity of atoms and bonds in the molecule.
HallKier-Alpha	HallKierAlpha	Hall-Kier alpha value. It describes the flexibility or rigidity of atoms in a molecule.

0D, 1D and 2D descriptors integrated into only one class in Descriptors (**continued**)

Set of descriptors	**Descriptors**	**Description**
BCUT2D descriptors	BCUT2D_MWHI	Incorporates atom masses in the Burden matrix - returns the highest eigenvalue.
	BCUT2D_MWLOW	Incorporates atom masses in the Burden matrix - returns the lowest eigenvalue.
	BCUT2D_CHGHI	Incorporates atom charges in the Burden matrix - returns the highest eigenvalue.
	BCUT2D_CHGLO	Incorporates atom charges in the Burden matrix - returns the lowest eigenvalue.
	BCUT2D_LOGPHI	Incorporates atom logarithms of the partition coefficient (logP) in the Burden matrix - returns the highest eigenvalue.
	BCUT2D_LOGPLOW	Incorporates atom logarithms of the partition coefficient (logP) in the Burden matrix - returns the lowest eigenvalue.
	BCUT2D_MRHI	Incorporates atom molar refractivity in the Burden matrix - returns the highest eigenvalue.
	BCUT2D_MRLOW	Incorporates atom molar refractivity in the Burden matrix - returns the lowest eigenvalue.
Chi descriptors	Chi descriptors	The Chi descriptors represent the count of specific path patterns in the molecule and are calculated based on the Hall-Kier delta values or on the deviation of an atom's valence electron count from the expected count based on its atomic number.
LabuteASA	LabuteASA	Labute's Approximate Surface Area. It estimates the solvent-accessible surface area of a molecule, which is relevant for its solubility and permeability properties.
MolLogP	MolLogP	Molar logarithm of the partition coefficient (logP). It quantifies the lipophilicity or hydrophobicity of a molecule, which is important for its distribution and permeability properties.
Kappa descriptors	Kappa1	Kappa shape indices. They describe the shape of a molecule based on the distribution of bond lengths and angles. These descriptors are derived from the Hall-Kier alpha descriptor and the number of paths of specific lengths in the molecule.
	Kappa2	
	Kappa3	
PEOE VSA descriptors	PEOE VSA descriptors	These descriptors calculate the PEOE (Partial Equalization of Orbital Electronegativity) VSA (surface area contributions of atoms or groups of atoms in a molecule) for a molecule by assigning atom contributions to predefined bins based on their Labute ASA and Gasteiger charge values.
SMR VSA descriptors	SMR VSA descriptors	Calculates the SMR (Molar Refractivity) VSA for a molecule by assigning atom contributions to predefined bins based on their Labute ASA and MR values.
SlogP VSA descriptors	SlogP VSA descriptors	Calculates the SlogP VSA for a molecule by assigning atom contributions to predefined bins based on their Labute ASA and SlogP values.

0D, 1D and 2D descriptors integrated into only one class in Descriptors (**continued**)

Set of descriptors	**Descriptors**	**Description**
EState VSA	EState VSA	Calculates the EState (E-State) VSA for a molecule by assigning atom contributions to predefined bins based on their Labute ASA and EState values.
FractionCSP3	FractionCSP3	Fraction of sp3-hybridized carbon atoms in the molecule.
TPSA	TPSA	Topological polar surface area. It estimates the surface area of a molecule that is involved in polar interactions, which is relevant for its solubility and biological activity.
RingCount	RingCount	Number of rings in the molecule. It indicates the level of molecular complexity and rigidity.
Count of structural groups	Count of structural groups	These descriptors count the number of hydrogen bond acceptor groups, hydrogen bond donor groups, heteroatoms (non-carbon atoms), etc., present in the molecule. They provide information about the potential for specific molecular interactions.
Frequency of functional groups	Frequency of functional groups	These descriptors represent the count of specific functional groups or substructures in the molecule. They provide information about the presence of particular chemical moieties.

Table 21: Molecular Fingerprints provided by DeepMol.

Molecular Fingerprint	**Description**	**References**
Morgan Fingerprint	Circular fingerprint based on the Morgan algorithm. This fingerprint is generated by considering the "circular" environment of each atom up to a given radius.	[256]
MACCSkeys Fingerprint	Uses the 166 public keys implemented as SMARTS. The fragment definitions for the MACCS 166 keys can be found in this document: `https://github.com/rdkit/rdkit/blob/master/rdkit/Chem/MACCSkeys.py`	[318]
RDK Fingerprint	This fingerprinting identifies all subgraphs in the molecule within a particular range of sizes and hashes each subgraph based on atom and bond types and degrees. Based on the Daylight fingerprint.	[319]
Layered Fingerprint	Uses the same subgraph enumeration algorithm used in the RDK Fingerprint. Sets bits based on different atom and bond type definitions (pure topology, bond order, atom types, aromaticity, etc).	-
AtomPair Fingerprint	This fingerprint is generated by encoding interactions between all possible atom pairs in a molecule using a hashing scheme. These interactions are encoded based on atomic environments and the shortest path between the atom pair.	[320]

Table 22: 3D descriptors provided by DeepMol.

3D Descriptor	**Description**	**References**
AutoCorr3D	These descriptors are derived from the autocorrelation of various physicochemical properties such as charge, mass, van der Waals volume, electronegativity, polarizability, ionization potential, and electron affinity associated with the atoms within a molecule. In total, there are 80 descriptors.	[321]
Radial Distribution Function	These descriptors are based on the radial distribution function, describing the likelihood of finding an atom at a specific distance from another atom. They provide information about the spatial distribution of atoms and their environments. In total, there are 210 descriptors.	[321]
Plane Of Best Fit	The Plane of Best Fit (PBF) is the geometric plane that minimizes the sum of squared distances between the atoms of a molecule and the plane itself. This descriptor indicates the average distance of all heavy atoms from the PBF, offering a quantitative measure of how much the molecule deviates from a 2D shape and providing insight into its 3D configuration.	[322]
MORSE	224 Molecular Surface Electrostatics derive from the electrostatic potentials present on the molecular surface. They offer insights into the charge distribution across the molecule's surface and contribute to understanding its three-dimensional shape. In total, there are 224 descriptors.	[321]
WHIM	Weighted Holistic Invariant Molecular (WHIM) descriptors are based on the principle of invariance, ensuring that they maintain consistency even after molecule transformations or rotations. They depend on statistical indexes obtained by projecting atoms along principal axes. WHIM descriptors encapsulate three-dimensional information regarding molecular size, shape, symmetry, and atom distribution, all in reference to invariant reference frames. In total, there are 114 descriptors.	[321]
Radius Of Gyration	The radius of gyration quantifies how atoms are distributed in a molecular structure regarding its center of mass. Put simply, it represents the average distance of a molecule's atoms from their center of mass.	[323]
Principal Moments Of Inertia	The principal moments of inertia refer to the three rotational inertia values around its principal axes. These axes are the mutually perpendicular axes through the center of mass of the molecule, and the moments of inertia represent the distribution of mass around each axis.	-
Inertial Shape Factor	The inertial shape factor is a measure that characterizes how mass is distributed around the principal axes of rotation. It is derived from the principal moments of inertia.	[321]
Eccentricity	Eccentricity refers to the extent to which its principal axes differ in length. It is calculated as the square root of the ratio of the difference between the squares of the longest and shortest principal moments of inertia to the square of the sum of all three principal moments of inertia.	[323]
Asphericity	The asphericity of a molecule is a measure that quantifies the deviation of its shape from a perfect sphere.	[324]

3D descriptors provided by DeepMol (**continued**).

3D Descriptor	**Description**	**References**
Spherocity Index	The spherocity index is an anisometry descriptor defined as a function of the eigenvalues of the covariance matrix of the atomic coordinates. It varies from zero for flat molecules, such as benzene, to unity for totally spherical molecules.	[321]
Normalized Principal Moments Ratios	Normalized ratios of principal moments of inertia measure the distribution of mass around the principal axes of rotation for a molecule, providing insights into its overall shape and symmetry. 2 descriptors.	[325]

Table 23: DeepChem featurizers provided by DeepMol.

Featurizer	**Description**	**References**
ConvMolFeat	Featurization to implement Duvenaud graph convolutions. It constructs a vector of local descriptors for each atom in a molecule.	[136]
PagtnMolGraphFeat	It creates a molecular graph connecting all atom pairs, considering interactions between every atom pair in the molecule. The default node representation includes features such as atom type, formal charge, degree, explicit and implicit valence, and aromaticity, resulting in a feature length of 94. The default edge representation, with a feature length of 42, considers bond type, conjugation, same ring membership, ring size, aromaticity, and distance between atom pairs based on the shortest path.	[326]
WeaveFeat	Featurization to implement Weave convolutions. In contrast to Duvenaud graph convolutions, Weave convolutions require a quadratic matrix of interaction descriptors for every atom pair, potentially offering enhanced descriptive capability but resulting in a larger featurized dataset.	[103]
MolGanFeat	This featurizer was originally designed for MolGAN de-novo molecular generation. It encapsulates two matrices containing atom and bond type information that can be used in predictive models.	[208]
MolGraphConvFeat	This serves as a featurizer for general graph convolution networks applied to molecules, with default node and edge representations based on the WeaveNet paper. The default node features encompass atom type, formal charge, hybridization, hydrogen bonding, aromaticity, degree, number of hydrogens, chirality, and partial charge, while the default edge features include bond type, same ring membership, conjugation, and stereo configuration. Users have the flexibility to customize their own representations.	[103]
CoulombFeat	This featurizer calculates Coulomb matrices for molecules, offering a representation of the electronic structure. The resulting Coulomb matrix is an N x N matrix, where N represents the number of atoms in the molecule, with each element indicating the strength of the electrostatic interaction between two atoms.	[209]
CoulombEigFeat	Same as CoulombFeat but it also calculates the eigenvalues of Coulomb matrices for molecules.	[209]

DeepChem featurizers provided by DeepMol (**continued**).

Featurizer	**Description**	**References**
SmileImage-Feat	The SmilesImageFeat featurizer transforms a SMILES string into an image. The default image size is 80 x 80, supporting two modes: std, a grayscale representation using atomic numbers for atom positions and a constant value for bonds, and engd, a 4-channel specification incorporating atom properties like hybridization, valency, charges, and bond type for enhanced visualization. Atom coordinates are computed, and lines between atoms indicate bonds, with channels reflecting specified property values.	[327]
SmilesSeq-Feat	The SmilesSeqFeat featurizer converts a SMILES string into a sequence. SMILES below a specified maximum length are padded, while longer ones are excluded. The resulting sequence of character indices, obtained through a character-to-index mapping, can serve as input for a predictive model.	[327]
DMPNNFeat	This serves as a featurizer for the implementation of Directed Message Passing Neural Network (D-MPNN). The default node features include atomic number, degree, formal charge, chirality, number of hydrogens, hybridization, aromaticity, and mass, resulting in a feature length of 133. Edge features encompass bond type, same-ring membership, conjugation, and stereo configuration, with a feature length of 14.	[103], [328]
MatFeat	This functions as a featurizer for the Molecule Attention Transformer, producing a numpy array containing molecular graph descriptions, including node features, adjacency matrix, and distance matrix.	[329]

Table 24: Scikit-Learn scalers available in DeepMol.

Featurizer	**Description**	**References**
Standard-Scaler	Standardize features by removing the mean and scaling to unit variance.	[330]
MinMax-Scaler	Transform features by scaling each one of them to a given range.	[331]
RobustScaler	Scales features by subtracting the median and adjusting data based on the quantile range. This scaler is robust to outliers.	[332]
Polynomial-Features	Creates polynomial and interaction features by generating a new feature matrix that includes all polynomial combinations of the original features up to a specified degree.	[333]
Normalizer	Normalizes samples individually to unit norm.	[334]
Binarizer	Binarizes data (0 or 1) according to a threshold.	[335]
Kernel-Centerer	Centrally aligns kernel matrices by subtracting the mean along each feature dimension, ensuring that the kernel's diagonal elements are centered around zero.	[336]
Quantile-Transformer	Adjusts features to follow a uniform or normal distribution, spreading out frequent values and mitigating the impact of outliers.	[337]
Power-Transformer	Applies a featurewise power transform to make the data more Gaussian-like.	[338]

Table 25: Scikit-Learn feature selection methods available in DeepMol.

Feature Selection Method	**Description**	**References**
LowVarianceFS	Removes all features with a variance lower than a threshold.	[339]
KbestFS	Selects the top k features based on their statistical significance with the target variable. It scores and retains the features with the highest correlation or dependency with the target.	[340]
PercentilFS	The PercentilFS method is a form of univariate feature selection that involves choosing features through univariate statistical tests. It removes all features except for a user-defined highest-scoring percentage.	[341]
RFECVFS	Selects features based on recursive feature elimination with cross-validation. It systematically eliminates less relevant features based on an estimator's performance through cross-validation to determine the optimal subset of features.	[342]
SelectFromModelFS	It selects important features based on the coefficients or importance scores derived from a trained estimator. It allows for automatic feature selection by considering features that meet a specified threshold of importance.	[343]

Table 26: Data splitters in DeepMol - all of them can perform train-test, train-validation-test and k-fold splitting.

Data Splitter	**Description**	**References**
RandomSplitter	Randomly splits the data.	-
SingletaskStratifiedSplitter	Splits single-task data on a stratified fashion.	-
SimilaritySplitter	Splits molecules based on fingerprint similarity using Tanimoto similarity over Morgan fingerprints. It can create both homogeneous and heterogeneous splits. In homogeneous splits, molecules are evenly distributed across sets, whereas in heterogeneous splits, the goal is to minimize the similarity between molecules in one set and those in the other sets.	-
ScaffoldSplitter	Splits molecules according to the Bemis-Murcko scaffold representation, which represents rings, linkers, frameworks (combinations of linkers and rings), and atomic properties like atom type, hybridization, and bond order within a molecular dataset. It can create both homogeneous and heterogeneous splits. In homogeneous splits, molecules are evenly distributed across sets, whereas in heterogeneous splits, the goal is to minimize the number of shared scaffolds between splits.	[344]
ButinaSplitter	Splits the molecules based on the butina clustering of a bulk tanimoto fingerprint matrix. It can create both homogeneous and heterogeneous splits. In homogeneous splits, molecules are evenly distributed across sets, whereas in heterogeneous splits, molecules from the same clusters are kept in the same split.	[213]
MultiTaskStratifiedSplitter	Splits multi-task data on a stratified fashion.	-

Table 27: Imbalanced learning techniques provided by DeepMol.

Unbalanced Learn Method	**Description**	**References**
Over-sampling techniques		
RandomOverSampler	Randomly samples, with replacement, molecules in the under-represented class(es).	-
SMOTE	Synthetic Minority Oversampling Technique, is a statistical method aimed at balancing imbalanced datasets by creating new instances for the minority class(es). It generates new examples by combining features from existing minority cases and their nearest neighbors in the feature space.	[218]
Under-sampling techniques		
RandomUnderSampler	Randomly under-samples, without replacement, molecules in the over-represented class(es).	-
ClusterCentroids	Under-samples data by generating centroids through clustering methods (KMeans algorithm by default). This algorithm keeps N majority samples using the coordinates of the N cluster centroids as the new majority samples.	-
Combination of over and under-sampling techniques		
SMOTEENN	Technique that combines over- and under-sampling using SMOTE and Edited Nearest Neighbours (ENN) respectively. ENN removes examples for which the majority class label conflicts with the labels of the majority of its three nearest neighbors.	[220]
SMOTETomek	Technique that combines over- and under-sampling using SMOTE and Tomek links respectively. Tomek links identify pairs of instances, one from the majority class and one from the minority class, that are close to each other but have different class labels. The instances from the majority class are then removed.	[220]

Table 28: Scikit-learn, Deepchem and Keras models provided by DeepMol by default.

Model	**Description**	**References**
Scikit-learn		[190]
82 models as in scikit-learn (v1.2.0)	DeepMol includes all machine learning models available through Scikit-learn for regression, classification and multioutput.	-
DeepChem		[192]
GatModel	A Graph Attention Network (GAT)-based model for predicting graph properties. It involves updating node representations using a GAT variant. The model computes the graph representation by combining weighted sums and max pooling of node representations, followed by concatenation, and utilizes a Multi-Layer Perceptron (MLP) for the final prediction. Works both for classification and regression.	[345]

Scikit-learn, Deepchem and Keras models provided by DeepMol by default (**continued**).

Model	**Description**	**References**
DeepChem		[192]
GCNModel	Graph Convolution Networks (GCN)-based model for graph property prediction. It involves updating node representations using GCN. It then computes each graph's representation through a weighted sum and max pooling of node representations, followed by concatenation of the outputs. The final prediction is done using a Multilayer Perceptron (MLP). Works both for classification and regression.	[346]
AttentiveFP-Model	Graph Property Prediction Model that involves combining node and edge features to initialize node representations through a round of message passing. Subsequently, node representations are updated with multiple rounds of message passing, and for each graph, its representation is computed by combining the representations of all nodes using a gated recurrent unit (GRU), followed by making the final prediction using a linear layer. Works both for classification and regression.	[347]
PagtnModel	Graph Property Prediction model employs a modified Graph Attention Network (GAT) to update node representations in graphs, utilizing a linear additive attention mechanism based on concatenating node and edge features. Multiple rounds of message passing are applied with residual connections between each layer, and the final molecular representation is obtained by aggregating node representations. The final prediction is obtained through a linear layer. Works both for classification and regression.	[348]
MPNNModel	Message Passing Neural Networks (MPNN) considers graph convolutional operations as a specific form of a broader message passing scheme, wherein nodes exchange "messages", leading to updates in their internal states. The ordering of structures in this model adheres to the principles outlined in [349]. Works both for classification and regression.	[349], [350]
MEGNetModel	MatErials Graph Network uses multiple layers of Graph Networks known as MEGNetBlocks for predicting properties in molecules and crystals. It integrates node and edge properties through a Set2Set layer, combining this information with global features to perform property prediction tasks for materials or molecules. Works both for classification and regression.	[351]
DMPNNModel	Directed Message Passing Neural Network (D-MPNN), consisting of two phases: message-passing and read-out. In the message-passing phase, the objective is to generate hidden states for all atoms in the molecule using encoders, followed by the read-out phase where features are input into a feed-forward neural network to obtain task-based predictions. Works both for classification and regression.	[328]
CNN	1, 2, or 3 dimensional Convolution Neural Network (CNN) comprising of a variable number of convolutional layers, followed by a global pooling layer (max pool or average pool), and a final fully connected layer for output computation. Works both for classification and regression.	-

Scikit-learn, Deepchem and Keras models provided by DeepMol by default (**continued**).

Model	**Description**	**References**
DeepChem		[192]
MultitaskClassifier	A fully connected multitask classification network that offers extensive customization of the model, including options for adjusting the number and widths of layers, activation functions, regularization methods, and more.	-
MultitaskIRVClassifier	The IRV is a low-parameter neural network that enhances a k-nearest neighbor classifier by nonlinearly combining the impacts of neighboring chemicals in the training set. This model is used for multitask classification.	[352]
ProgressiveMultitaskClassifier	Progressive networks facilitate multitask learning by assigning a new set of weights to each task, preventing exponential forgetting and ensuring that prior tasks are not disregarded. This approach prevents the issue of exponential forgetting, ensuring the retention of knowledge from previous tasks in multitask learning. Used for multitask classification.	[353]
ProgressiveMultitaskRegressor	The same as ProgressiveMultitaskClassifier but for multitask regression.	[353]
RobustMultitaskClassifier	This model's fundamental concept involves incorporating bypass layers that directly connect features to the task output, offering potential flexibility to navigate multitasking challenges affected by destructive interference. The inclusion of these bypass layers aims to facilitate adaptability in overcoming obstacles during multitasking. Used for multitask classification.	[158]
RobustMultitaskRegressor	The same as RobustMultitaskClassifier but for multitask regression.	[158]
ScScoreModel	The SCScore model is a neural network that predicts the synthetic complexity score (SCScore) for molecules and establishes a correlation with the number of reaction steps needed to synthesize the target molecule. The model was adapted for classification tasks.	[354]
ChemCeption	The ChemCeption model applies convolutional neural networks (CNNs) to predict molecular properties by utilizing an image-based representation of the molecule. In this representation, various atomic and bond properties are encoded as pixels. Works both for classification and regression.	[355]
DAGModel	Directed Acyclic Graph models are used for predicting molecular properties by representing molecules as a series of directed graphs. In this approach, each atom in the molecule is transformed into a directed acyclic graph with edges pointing "inwards" to it. Works both for classification and regression.	[161]
GraphConvModel	Graph Convolutional Models, based on Duvenaud's convolutions. The model starts with per-atom descriptors for each molecule, undergoing a series of convolutional layers that combine and recombine these descriptors.	[136]

Scikit-learn, Deepchem and Keras models provided by DeepMol by default (**continued**).

Model	**Description**	**References**
DeepChem		[192]
Smiles2Vec	The Smiles2Vec model is implemented to generate neural representations of SMILES strings for subsequent tasks involving molecular properties. Using an Embedding layer, the model transforms input SMILES strings into vector representations, potentially incorporating a 1D convolutional layer and RNN cells to capture temporal dependencies and molecular structure information, facilitating molecular property prediction. Works both for classification and regression.	[356]
TextCNN-Model	The model uses multiple 1D convolutional filters on padded strings, followed by max-over-time pooling, extracting individual features. After concatenating these features, the model undergoes transformations through hidden layers to make predictions. Works both for classification and regression.	[357]
WeaveModel	WeaveModel style convolutions differ from GraphConvModel style convolutions primarily in explicitly modeling bond features. This explicit modeling requires constructing an NxN matrix for bond interactions, potentially leading to scaling issues but offering the potential for more precise representation of subtle bond effects. Only for regression tasks.	[103]
DTNNModel	Deep Tensor Neural Network (DTNN), is a deep learning architecture designed specifically for understanding quantum-mechanical properties of molecular systems. DTNNs are particularly effective in capturing complex relationships within molecular structures, enabling insights into properties such as stability, atomic energies, and electronic structure. Only for regression tasks.	[358]
MATModel	Model based on the Molecular Attention Transformer. It works by decomposing each molecule into its Node Features matrix, adjacency matrix, and distance matrix. Subsequently, a mask tensor is computed for the batch, serving as input for the MATEmbedding, MATEncoder, and MATGenerator layers. Works for regression tasks.	[329]
Multitask-Regressor	The same as MultitaskClassifier but for regression.	-
Tensorflow/Keras		[191]
FCNN	Fully connected neural network (FCNN) that allows flexible specification of the model architecture and training parameters. It supports multiple tasks, customizable hidden layers with activations, regularizers, and dropouts, along with options for batch normalization. It constructs a model with input, shared hidden layers, and task-specific output layers, compiling it with specified optimizer, loss functions, and evaluation metrics. Suitable for both classification and regression.	-
1D CNN	1D convolutional neural network (1D CNN) model suitable for classification and regression. It allows flexible specification of convolutional layers with options for filters, kernel sizes, activations, dropouts, and batch normalizations. The model architecture includes Gaussian noise injection, convolutional layers, dense layers, and task-specific output layers, compiled with specified optimizer, loss functions, and evaluation metrics.	-

Scikit-learn, Deepchem and Keras models provided by DeepMol by default (**continued**).

Model	**Description**	**References**
Tensorflow/Keras		[191]
Tabular-Transformer	Tabular transformer model based on the transformer architecture, including embedding layers, multi-head attention layers, layer normalization, and dense layers. The model allows flexible specification of parameters such as attention layers, dropout rates, and activation functions for last layers, compiled with specified optimizer, loss functions, and evaluation metrics. Suitable for both classification and regression.	-
RNN	This Recurrent Neural Network (RNN) model that supports flexible specification of parameters including the number of LSTM and GRU layers, units in each layer, dropout rates, and activation functions for both dense and last layers. The model is compiled with specified optimizer, loss functions, and evaluation metrics. Suitable for both classification and regression.	-
Bidirectional-RNN	Bidirectional RNN model similar to the RNN model. However, it incorporates bidirectional LSTM and GRU layers, allowing the model to learn from both past and future information in the input sequences. It supports flexible specification of parameters such as the number of LSTM and GRU layers, units in each layer, dropout rates, and activation functions for both dense and last layers. The model is compiled with specified optimizer, loss functions, and evaluation metrics. Suitable for both classification and regression.	-

Table 29: Feature explainability using Shapley values (SHAP [221]) in DeepMol.

Explainer	**Description**
Permutation	Approximates SHAP values by systematically permuting input features.
Exact	Computes SHAP values by using optimized exact enumeration techniques. It is particularly suited for models with less than 15 features.
Additive	Computes SHAP values for generalized additive models.
Tree	Computes SHAP values for ensemble tree models.
GPU Tree	GPU accelerated version of TreeExplainer.
Partition	Recursively computes SHAP values through a hierarchy of features, accounting for correlations among features by grouping them together.
Linear	Computes SHAP values for linear models.
Sampling	Computes SHAP values under the assumption of feature independence.
Deep	Approximates SHAP values for deep learning models.
Kernel	Computes SHAP values by using a weighted linear regression to compute the importance of each feature.
Random	Returns random (normally distributed) feature attributions. Only for benchmark comparisons.

Table 30: Ensembles and voting pipelines.

Dataset	**Ensemble type**	**Models/Pipelines**
Bioav	Voting Pipeline	• 5 ChEMBL Standardizer + DMPNN
Lipo	Stacking Ensemble	• Linear Regression • SVM • Random Forest • Gradient Boosting • Final estimator: MLP
BBB	Voting Pipeline	• 3 Custom Standardizer + DMPNN • ChEMBL Standardizer + DMPNN; ChEMBL Standardizer + GCN
PPBR	Stacking Ensemble	• Linear Regression • SVM • Random Forest • Gradient Boosting • Final estimator: MLP
CL-Hepa	Voting Pipeline	• 3 ChEMBL Standardizer + DMPNN • GCN • ChEMBL Standardizer + GCN
CL-Micro	Voting Pipeline	• 5 ChEMBL Standardizer + TextCNN
Ames	Voting Pipeline	• 5 ChEMBL Standardizer + GCN
DILI	Voting Pipeline	• Basic Standardizer + Layered and Morgan FPs + 1D CNN • 4 Custom Standardizers + MACCS keys + Select from model + Gaussian Process classifier

Ensembles and voting pipelines (**continued**).

Dataset	**Ensemble type**	**Models/Pipelines**
LD50	Voting Regressor	• Linear Regression • SVM • Random Forest • Gradient Boosting • MLP
CYP2C9 Inhibition	Voting Pipeline	• Basic Standardizer + MorganFingerprint + SVC • Custom Standardizer + AtomPairFingerprint + LowVarianceFS + GradientBoosting • BasicStandardizer + MorganFingerprint + GradientBoosting • BasicStandardizer + MorganFingerprint + RidgeClassifierCV • AtomPairFingerprint + LowVarianceFS + GradientBoosting
CYP2C9 Substrate	Bagging Ensemble	• 150 Logistic Regression
CYP2D6 Inhibition	Stacking Ensemble	• LogisticRegression • SVC • RandomForest • GradientBoosting • Final estimator: MLP
CYP3A4 Inhibition	Voting Pipeline	• ChEMBL Standardizer + ConvMolFeat + GraphConvModel • 4 Basic Standardizer + ConvMolFeat + GraphConvModel
CYP3A4 Substrate	Bagging Ensemble	• 450 SVMs

A.1.3 Comparison with similar tools

In recent years, several chemoinformatics toolkits, including DeepChem, OpenChem, AMPL and ZairaChem have been developed. These open-source projects, written in Python, are utilized for machine learning in constructing pipelines for QSAR/QSPR systems. Regarding dataset loading, DeepMol and DeepChem support CSV and SDF formats, while OpenChem, AMPL and ZairaChem only handle CSV files, which limits their capability to import pre-computed 3D structures.

DeepMol offers various methods for molecule standardization, whereas OpenChem allows sanitization, padding sequences, and canonizing SMILES, and AMPL enables striping salt groups, neutralizing, kekulizing molecules, and replacing rare isotopes with the most common ones for each element. Only DeepChem supports molecule sanitization during dataset loading. ZairaChem applies the MELLODDY-Tuner protocol, which includes disconnecting metal atoms from the rest of the molecule, reionizing the molecule, adjusting charges and protonation states, assigning stereochemistry to the molecule, ensuring consistent representation of chiral centers, and additional standardization procedures. All tools support data splitting, except OpenChem, and only DeepMol can perform a stratified split specifically designed for molecular data and multitask classification.

B

Supplementary Data for Chapter 5

B.1 Supplementary Material - Combining Evolutionary Algorithms With Reaction Rules Towards Focused Molecular Design

Source Code: `https://github.com/BioSystemsUM/ReactEA`

Results and code for reproducibility: `https://zenodo.org/record/7630352`

B.1.1 Supplementary Figures:

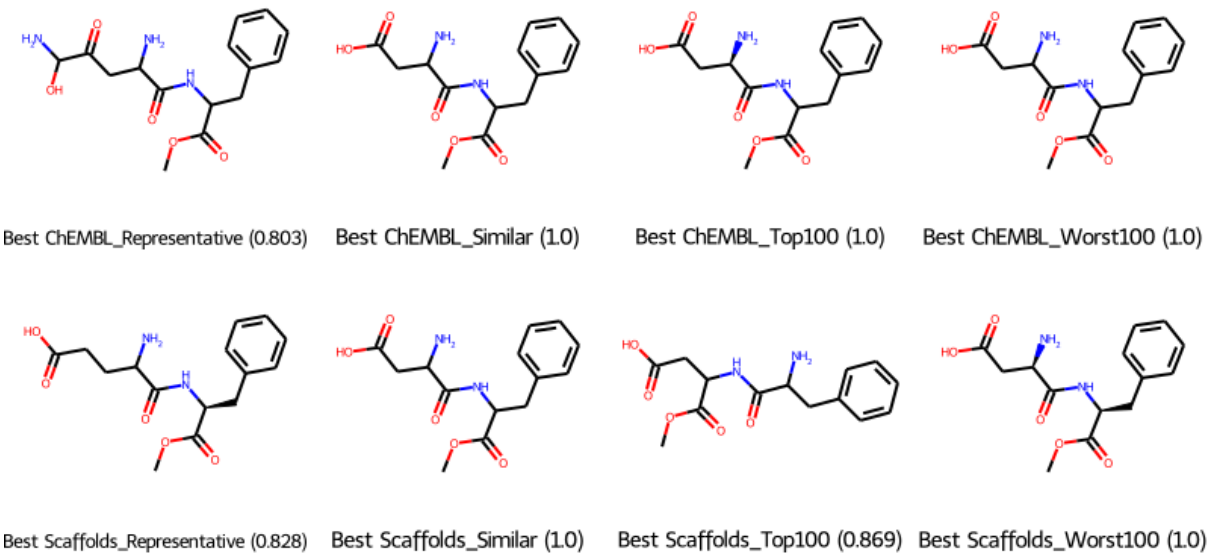


Figure 29: Best molecule according to its similarity to aspartame generated by each EA.

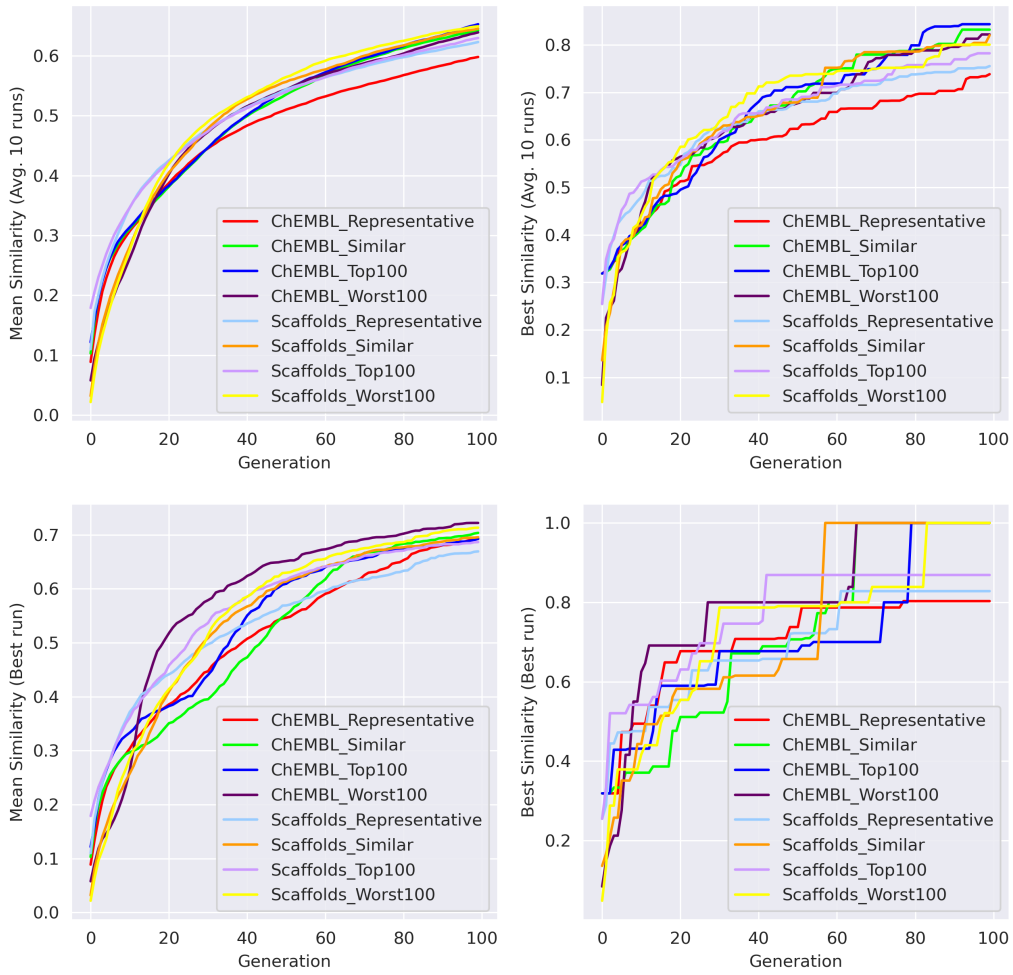


Figure 30: Evolution of solutions across generations for the different initial populations. Top left represents the average similarity to aspartame of all solutions (average 10 runs). Top right the best solution (average 10 runs). Bottom left the mean similarity of the best run only. Bottom right the best solution of the best run.

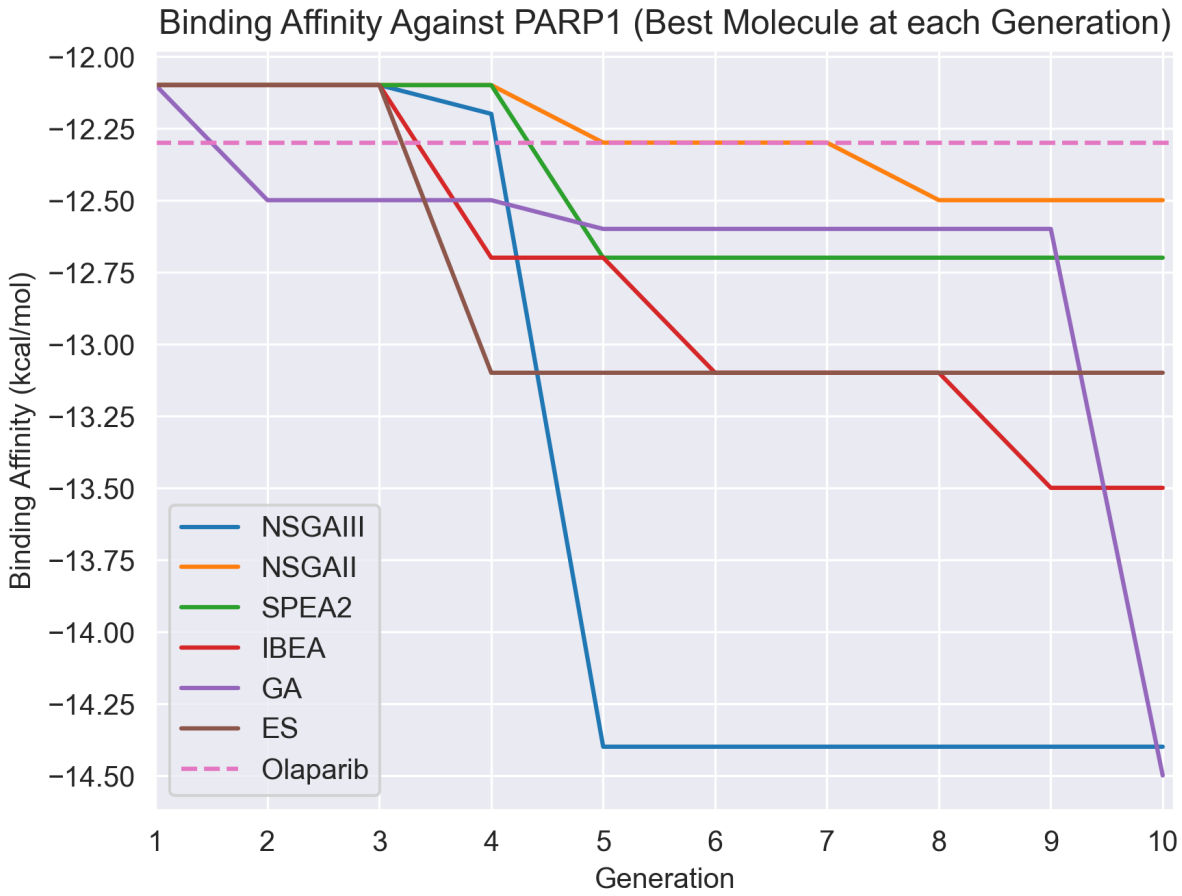


Figure 31: Binding affinity of the best molecule at each generation for the different EAs. The best inhibitor in the initial population, Olaparib, is shown for reference.

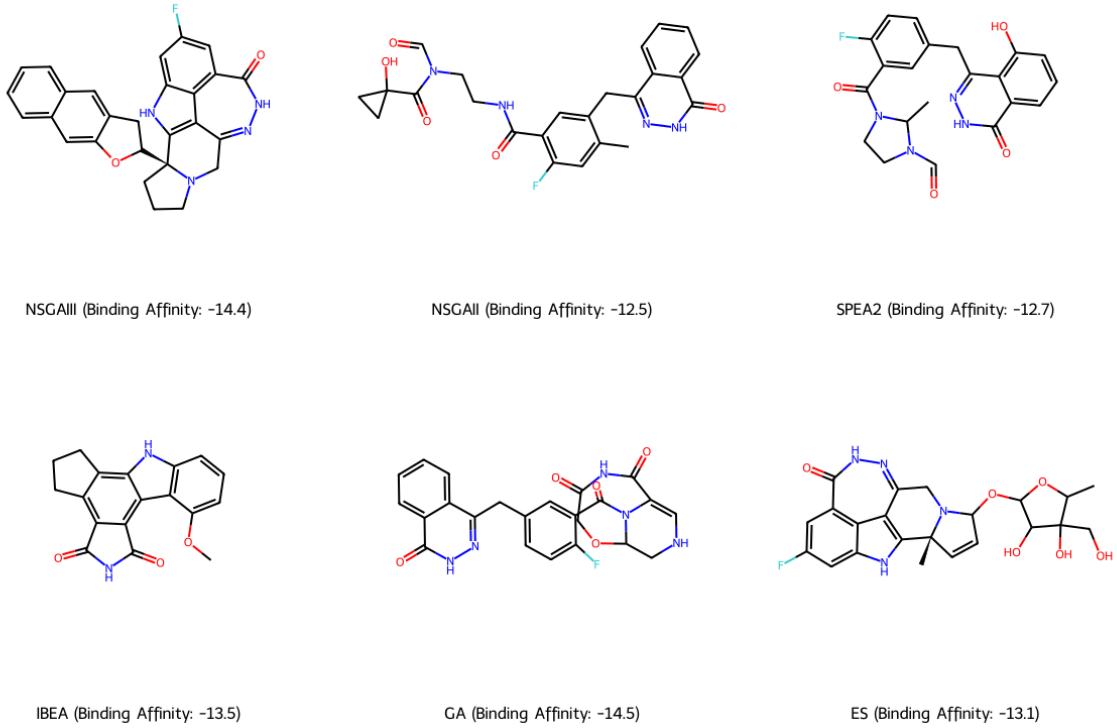


Figure 32: Best molecule and respective binding affinity of the best generated molecules by each EA.

B.1.2 Supplementary Tables:

Table 31: Results of the QED, pLogP, SAS and CLScore optimization.

	Initial Population				GA			
	Worst	Best	Mean	Std. Dev.	Worst	Best	Mean	Std. Dev.
QED	0.009	0.747	0.295	0.190	0.925	0.948	0.936	0.007
pLogP	30.000	4.658	-11.037	11.462	11.901	13.877	12.310	0.315
pLogP (norm)	-30.000	4.525	-9.773	12.156	9.254	11.194	9.615	0.291
SAS	10.000	1.211	3.939	1.439	1.416	1.000	1.287	0.109
SAS (norm)	0.000	0.977	0.674	0.160	0.952	1.000	0.968	0.013
CLScore	1.715	6.117	3.934	0.631	5.465	6.776	5.593	0.140

Note: The results for the SAS, SAS (norm) and CLScore generated with NSGAIII, NSGAII, SPEA2 and IBEA were not taken in consideration due to the high number of duplicated molecules.

Results of the QED, pLogP, SAS and CLScore optimization (**continued**).

	ES				NSGAIII			
	Worst	Best	Mean	Std. Dev.	Worst	Best	Mean	Std. Dev.
QED	0.942	0.948	0.944	0.002	0.940	0.948	0.944	0.002
pLogP	12.392	13.877	12.638	0.220	12.928	13.877	13.296	0.274
pLogP (norm)	9.690	11.194	9.932	0.200	10.429	11.194	10.747	0.226
SAS	1.354	1.000	1.250	0.096	1.000	1.000	1.000	0.000
SAS (norm)	0.962	1.000	0.974	0.011	1.000	1.000	1.000	0.000
CLScore	5.574	6.776	5.676	0.130	6.776	6.776	6.776	0.000

Note: The results for the SAS, SAS (norm) and CLScore generated with NSGAIII, NSGAII, SPEA2 and IBEA were not taken in consideration due to the high number of duplicated molecules.

Results of the QED, pLogP, SAS and CLScore optimization (**continued**).

	NSGAII				SPEA2			
	Worst	Best	Mean	Std. Dev.	Worst	Best	Mean	Std. Dev.
QED	0.942	0.948	0.944	0.002	0.939	0.948	0.943	0.002
pLogP	12.748	13.877	13.105	0.278	13.122	13.877	13.385	0.239
pLogP (norm)	10.619	11.194	10.847	0.206	10.429	11.194	10.828	0.234
SAS	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000
SAS (norm)	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000
CLScore	6.776	6.776	6.776	0.000	6.776	6.776	6.706	0.167

Note: The results for the SAS, SAS (norm) and CLScore generated with NSGAIII, NSGAII, SPEA2 and IBEA were not taken in consideration due to the high number of duplicated molecules.

Results of the QED, pLogP, SAS and CLScore optimization (**continued**).

	IBEA			
	Worst	Best	Mean	Std. Dev.
QED	0.937	0.948	0.941	0.003
pLogP	13.061	13.877	13.367	0.282
pLogP (norm)	10.429	11.194	10.780	0.240
SAS	1.000	1.000	1.000	0.000
SAS (norm)	1.000	1.000	1.000	0.000
CLScore	6.123	6.776	6.606	0.250

Note: The results for the SAS, SAS (norm) and CLScore generated with NSGAIII, NSGAII, SPEA2 and IBEA were not taken in consideration due to the high number of duplicated molecules.