Deep Learning Applications in Structure-Based Drug Discovery

Rocco Meli

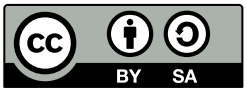
Linacre College

University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

October 1, 2022

This page intentionally contains only this sentence.



This page intentionally contains only this sentence.

To Laura & Edy.
To Pietro.
To Marta.

This page intentionally contains only this sentence.

Abstract

In recent years, machine learning and deep learning applications have permeated all fields of science thanks to rapid algorithmic advances and computer hardware developments. A very active area of research is the use of deep learning in structure-based drug design, where the goal is to design effective drugs against a pharmacological target of interest.

In this work, we explored the use of deep learning in the early stages of drug discovery. In particular, we focussed on structure-based virtual screening, binding affinity prediction, and *de novo* drug design.

First, we enabled docking with flexible residues within GNINA, a state-of-the-art docking software based on convolutional neural networks, and we performed a large-scale cross-docking study of such methodology, outlining its strengths and weaknesses.

Second, we extracted the convolutional neural network scoring function from the docking software into a standalone package for fast prototyping. With the new software at hand, we explored different annotations for supervised learning to improve the convolutional neural network scoring function for docking with flexible residues.

Third, we developed a novel scoring function for binding affinity prediction based on a successful deep learning architecture used to develop machine learning force fields.

Finally, we carefully evaluate a generative model for *de novo* design for the application in industrial drug discovery pipelines. We outline the weaknesses of the method and the problems with current evaluations of generative models.

This page intentionally contains only this sentence.

Acknowledgments

The path toward a DPhil is a complex and difficult one, especially when it brings you far from home, and to work in yet another language that is not yours. If on top of the expected and exciting challenges of trying to do innovative research you add a global pandemic, things get very tricky. More than 50% of the research described in this work has been performed in the (scientific) solitude of my old room at my parent's house or the living room at my partner's house. Therefore, I have a lot of people to thank for all the help and support they provided over the years.

Philip Biggin—my main supervisor—for all the guidance and support provided during my DPhil, and for fostering a friendly and relaxed research group. I particularly appreciated the freedom I was given and his constant availability. Despite the large size of the research group, he has always been happy to discuss my progress and my blockers. His support has been instrumental during the lockdowns and allowed me to build a decent work environment at home.

Garrett Morris—my second supervisor—for additional guidance, suggestions, and the "pedantic" corrections (as he calls them himself), which greatly improved the wording of some manuscripts associated with this work.

Evotec for providing additional funding—particularly helpful during the SARS-CoV-2 pandemic, and the following cost of living crisis—, and its scientists for providing invaluable help and comments on the different projects.

Andrew Anighoro for closely following my DPhil progress during the first few years—until he left Evotec—, and for providing useful feedback and encouragement.

Lionel Colliandre and Dimitar Hristozov for making a fully remote internship with Evotec interesting, fun, and extremely valuable, despite the circumstances. They might not have realised it, but our daily stand-up meetings in the morning have been the highlight of my day for several months during the interminable second lockdown.

Michael Bodkin for overseeing everything Evotec-related, for constructive feedback, and for organising the internship at Evotec.

All past and present members of the Biggin lab for providing a fun yet serious

working environment, with enjoyable—and often noisy—lunch breaks. My gratitude extends to all past and present members of the Structural Biology and Computational Biochemistry (SBCB) unit for creating a great work environment, and especially to those who organised or turned up frequently to the weekly virtual coffee breaks during the lockdowns.

Irfan Alibay—a postdoc in the Biggin group, later turned senior research software engineer—deserves a very special mention for introducing me to the field of AI in drug discovery at the very beginning of my DPhil, for suggesting and encouraging me to apply to Google Summer of Code, for the very interesting discussions about Python and programming in general, and for keeping alive and well our research infrastructure (in-house high performance computing (HPC) cluster, and workstations). This work would certainly look very different without the countless interactions I had with him.

Esther Becker, Gail Preston, and the staff of the Interdisciplinary Bioscience Doctoral Training Partnership for fostering a nice community to exchange thoughts and ideas with fellow students, as well as for providing training and support. As someone who started off by studying physics, moved to theoretical and computational chemistry, and finally landed on computational drug discovery and machine learning, I am in a privileged position to fully appreciate the importance of interdisciplinarity in modern science.

David Koes and Jocelyn Sunseri for mentoring me during the Google Summer of Code program and beyond, which has been an extremely valuable experience and ended up influencing much of this work. My gratitude extends to the members of the Koes group with whom I had the pleasure to collaborate.

Tina Friedrich and Andrew Gittings, from the University of Oxford's Advanced Research Computing (ARC) group, for their blazing fast replies to all my tickets related to "problems" with the new University HPC cluster, which I had the privilege to use extensively as a beta user.

My good friends in Oxford, Bristol, and around the UK for providing much-needed distractions.

Giovanni Carù and Vladimir Olivero, for the great moments spent together—over a coffee, a beer, a dinner, or playing billiard at Linacre—and for making me feel closer to home.

Nyree Manoukian and Shahé Gregorian for the enjoyable time spent together. During the lockdowns, I greatly missed our weekly dinners at Rickety Press, as well as our Sunday lunches at Edamamé, together with Giovanni and Vladimir.

Tom and Louise Mitchell, for providing a stress-free renting experience—which is somewhat rare in Oxford—, and for having been extremely nice neighbours with whom it was always a pleasure to have a spontaneous chat. I will be forever grateful for their generous rent reduction while I was away from

Oxford during the second lockdown. I wish I will always have neighbours (and landlords) as kind, thoughtful, and cool as them.

All the open source developers out there that contributed indirectly to this work by developing (often) good and (often) reliable scientific software for other people to use.

My parents Laura and Edy, my brother Pietro, and my partner Marta for their unconditional love and support all along this long but fulfilling chapter of my life. For always being there during the ups and downs, and during the additional challenges brought by the SARS-CoV-2 pandemic. I could not be luckier to have all of them in my life.

This page intentionally contains only this sentence.

List of Publications

0000-0002-2845-3410

Publications Included in This Work

- **R. Meli**, G. M. Morris and P. C. Biggin, "Scoring Functions for Protein-Ligand Binding Affinity Prediction using Structure-Based Deep Learning: A Review", *Front. Bioinform.*, 2:885983 (2022). DOI: 10.3389/fbinf.2022.885983 (Meli, Morris, et al. 2022)

- **R. Meli**, A. Anighoro, M. Bodkin, G. M. Morris and P. C. Biggin, "Learning Protein-Ligand Binding Affinity with Atomic Environment Vectors", *J. Cheminform.* 13, 59 (2021). DOI: 10.1186/s13321-021-00536-w (Meli, Anighoro, et al. 2021)

- A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, **R. Meli**, M. Ragoza, J. Sunseri and D. R. Koes, "GNINA 1.0: molecular docking with deep learning", *J. Cheminform.* 13, 43 (2021). DOI: 10.1186/s13321-021-00522-2 (McNutt et al. 2021)

- **R. Meli** and P. C. Biggin, "spyrmsd: symmetry-corrected RMSD calculations in Python", *J. Cheminform.* 12, 49 (2020). DOI: 10.1186/s13321-020-00455-2 (Meli and Biggin 2020)

Preprints Not Included in This Work

- F. R. Manby, T. F. Miller III, P. J. Bygrave, F. Ding, T. Dresselhaus, F. Batista-Romero, A. Buccheri, C. Bungey, S. J. R. Lee, **R. Meli**, K. Miyamoto, C. Steinmann, T. Tsuchiya, M. Welborn, T. Wiles and Z. Williams, "entos: A Quantum Molecular Simulation Package", *ChemRxiv* (2019). DOI: 10.26434/chemrxiv.7762646.v2 (Manby et al. 2019)

Publications Preceding This Work

- E. Liberatore, **R. Meli**, U. Röthlisberger, "A versatile multiple time step scheme for efficient *ab initio* molecular dynamics simulations", *J. Chem. Theory Comput.* 14, 6 (2018). DOI 10.1021/acs.jctc.7b01189 (Liberatore et al. 2018)
- **R. Meli**, G. Miceli, A. Pasquarello, "Oxygen DX center in $In_{0.17}Al_{0.83}N$: nonradiative recombination and persistent photoconductivity", *Appl. Phys. Lett.* 110, 7 (2017). DOI: 10.1063/1.4975934 (Meli, Miceli, et al. 2017)
- Jacob Beal, Traci Haddock-Angelli, Markus Gershater, Kim De Mora, Meagan Lizarazo, Jim Hollenhorst, Randy Rettberg, **iGEM Interlab Study Contributors**, "Reproducibility of Fluorescent Expression from Engineered Biological Constructs in *E. coli*", PLoS ONE 11, 3 (2016). DOI: 10.1371/journal.pone.0150182 (Beal et al. 2016)

Open Source Code Related to This Work

Python Packages

- `AEScore`: github.com/RMeli/aescore
- `gnina-torch`: github.com/RMeli/gnina-torch
- `spyrmsd`: github.com/RMeli/spyrmsd

Scripts and Pipelines

- Density matching: github.com/RMeli/densitymatch
- Generative modelling with `liGAN`: github.com/RMeli/generative3d
- GNINA 1.0 publication: github.com/dkoes/GNINA-1.0
- GSoC 19: github.com/RMeli/gsoc19

Main Contributions

- `GNINA`: github.com/gnina/gnina
- `MDAnalysis`: github.com/MDAnalysis/mdanalysis

Contributions

Some work presented in this thesis has been published in peer-reviewed journals, together with other authors. Below, my contributions for each chapter are clearly stated:

Chapter 1 contains very few extracts from Meli, Morris, et al. (2022). The review was conceptualised together with Philip Biggin and Garrett Morris. I wrote the initial draft of the manuscript, while Philip Biggin and Garrett Morris contributed with comments, feedback, and guidance.

Chapter 2 contains extracts from Meli, Morris, et al. (2022). The review was conceptualised together with Philip Biggin and Garrett Morris. I wrote the initial draft of the manuscript, while Philip Biggin and Garrett Morris contributed with comments, feedback, and guidance.

Chapter 3 contains extracts from Meli, Morris, et al. (2022). The review was conceptualised together with Philip Biggin and Garrett Morris. I wrote the initial draft of the manuscript, while Philip Biggin and Garrett Morris contributed with comments, feedback, and guidance.

Chapter 4 started as a collaboration with the group of David Koes at the University of Pittsburgh thanks to the Google Summer of Code 2019 (GSoC 2019) program under the Open Chemistry organisation. During GSoC, the project was mentored by Jocelyn Sunseri and David Koes, which conceptualised the project. After GSoC, the project continued as a collaboration with the Koes's group. Parts of Chapter 4 are published in McNutt et al. (2021). The text, the data, and the images concerning docking with flexible residues are my own work and are therefore reported in this thesis. Data concerning rigid docking with GNINA's default model—included for comparison—was produced by Paul Francoeur and Andrew McNutt. Other authors contributed with technical discussions, and comments on the draft of the manuscript.

Chapter 5 is a continuation of Chapter 4 and does not contain any published work. The project was initially mentored by Jocelyn Sunseri and David

Koes as part of GSoC, and continued as a collaboration with David Koes. The weights of the original GNINA models were converted from Caffe to PyTorch by Andrew McNutt.

Chapter 6 was conceptualised together with Philip Biggin and Garrett Morris. I implemented the method, carried out the numerical experiments, and wrote the first draft of the manuscript. Andrew Anighoro and Michael Bodkin—my industrial supervisors from Evotec—provided feedback and suggestions on the project. Andrew Anighoro, Philip Biggin, and Garrett Morris provided substantial feedback on the final draft of the manuscript. The chapter is published in Meli, Anighoro, et al. (2021).

Chapter 7 was conceptualised together with Lionel Colliandre, Dimitar Hristozov, Michael Bodkin—my industrial supervisors from Evotec—, and Philip Biggin. The project started as part of an Industrial Cooperative Awards in Science & Technology (iCASE) internship (carried out remotely, due to the SARS-CoV-2 pandemic) and extended beyond the internship as a collaboration. Lionel and Dimitar provided constant feedback, ideas, and suggestions—especially during the internship, when meetings were very frequent.

Appendix A lists most code contributions to free and open-source software (FOSS) steaming from this work. Such contributions benefitted greatly from the code review process on pull requests, in the form of comments and feedback from other developers.

Appendix B was wholly conceptualised by myself, out of frustration for the lack of a simple and standalone Python tool for the calculation of root mean squared displacements (RMSDs) with symmetry correction based on graph isomorphism. I wrote the entirety of the code—now a Python package on PyPI and conda-forge—and the first draft of the manuscript. Philip Biggin contributed with comments and feedback on the draft of the manuscript. Irfan Alibay contributed indirectly with interesting and useful discussions about Python, packaging, and CI. The appendix is published in Meli and Biggin (2020).

Philip Biggin—my main supervisor—contributed with ideas, guidance, and feedback on all the projects and the whole of this thesis. Garrett Morris—my second supervisor—contribute with ideas, guidance, and feedback on several projects. Irfan Alibay—a postdoc in the Biggin group turned scientific software developer—contributed indirectly to the large amount of code related to this thesis with interesting and useful discussions about software and software engineering.

Contents

List of Figures

This page intentionally contains only this sentence.

List of Tables

This page intentionally contains only this sentence.

Acronyms

ABFE absolute binding free energy

ACSF atom-centred symmetry function

ADME absorption, distribution, metabolism, and excretion

ADMET absorption, distribution, metabolism, excretion, and toxicity

AE autoencoder

AEV atomic environment vector

AI artificial intelligence

ANN atomic neural network

API application programming interface

APR attach-pull-release

AUC area under the curve

BFGS Broyden-Fletcher-Goldfarb-Shanno

BRD bromodomain

BRD4 bromodomain-containing protein 4

CASF Comparative Assessment of Scoring Functions

CDK cyclin-dependent protein kinase

CDK2 cyclin-dependent protein kinase 2

CI continuous integration

CLI command line interface

CNN convolutional neural network

cryo-EM cryogenic electron microscopy

CSAR Community Structure Activity Resource

D3R Drug Design Data Resource

DL deep learning

DMTA design, make, test, and analyse

DNN deep neural network

DoF degrees of freedom

DTI drug-target interaction

EF enrichment factor

FBDD fragmemt-based drug discovery

FEP free energy perturbation

FF force-field

FOSS free and open-source software

FP fingerprint

FPHF fast point feature histogram

GAN generative adversarial network

GNN graph neural network

GPCR G-protein-coupled receptor

GPU graphics processing unit

HBA hydrogen bond acceptor

HBD hydrogen bond donor

HPC high performance computing

HTS high-throughtput screening

I/O input/output

ICP iterative closest point

IFD induced fit docking

JTVAE junction-tree variational autoencoder

KL Kullback-Leibler

LSTM long short-term memory

MC Monte Carlo

MCTS Monte Carlo tree search

MD molecular dynamics

ML machine learning

MLP multi-layer perceptron

MM molecular mechanics

NF normalising flow

NLP natural language processing

NN neural network

NNP neural network potential

PCA principal component analysis

PDB protein data bank

PES potential energy surface

PI predictive index

PSA polar surface area

QCP quaternion characteristic polynomial

QED quantitative estimate of drug-likeness

QSAR quantitative structure-activity relationship

RANSAC random sample consensus

ReLU rectified linear unit

RF random forest

RGN random number generator

RL reinforcement learning

RMSD root mean squared displacement

RMSE root mean squared error

RNN recurrent neural network

ROC receiver operating characteristic

SA synthetic accessibility

SAR structure activity relationship

SASA solvent accessible surface area

SBDD structure-based drug design

SBVS structure-based virtual screening

SF scoring function

SGD stochastic gradient descent

SMILES simplified molecular-input line-entry system

SOTA state-of-the-art

TBDD target-based drug discovery

UFF universal force-field

VAE variational autoencoder

VS virtual screening

1 Introduction

Contents

R. Meli, G. M. Morris and P. C. Biggin, "Scoring Functions for Protein-Ligand Binding Affinity Prediction using Structure-Based Deep Learning: A Review", *Front. Bioinform.*, 2:885983 (2022). DOI: 10.3389/fbinf.2022.885983 (Meli, Morris, et al. 2022)

The discovery and development of new small-molecule drugs is a very challenging and expensive process (Dickson and Gagnon 2004; Drews 2000; Schneider and Schneider 2016). Only a handful of new molecular entities are approved each year (Brown and Wobst 2021), which is minuscule compared to the vastness of chemical space (Reymond et al. 2010) and the billions of dollars poured into drug discovery campaigns (DiMasi et al. 2016). The discovery pipeline for small-molecule drugs usually starts with the identification of a

protein target against which a hit compound is identified by high throughput screening (HTS) (Macarron et al. 2011; Mayr and Bojanic 2009). The hit compound is subsequently optimised to obtain a lead compound with good potency, and favourable pharmacodynamics and pharmacokinetics properties. The drug discovery and drug development process is briefly reviewed in section 1.1.

Thanks to significant methodological and hardware advances, computer-aided drug discovery (CADD) has played an important role in the development of new small-molecule drugs over the past decades (Sliwoski et al. 2013). CADD has the potential to speed up the early stages of drug discovery—hit identification and hit-to-lead optimisation—, and helps lower the costs of these phases by reducing the time and experimental resources needed. CADD methods fall into two broad classes: (explicit) structure-based, and ligand-based (or implicit structure-based) methods. For the latter, similarities to known active molecules play an important role since either the protein target is unknown, or information about the protein target is either unavailable or not included. For structure-based methods, the target structure is known, and the additional information is exploited in the modelling and optimisation of drug-target interactions (DTIs).

One of the main goals of the computational elucidation of DTIs is the calculation of relative or absolute binding free energies to distinguish potent binders from weak binders (or non-binders) against a target of interest. A fast and accurate prediction of protein-ligand binding affinities would circumvent the need for many time-consuming, complex, and costly experiments. Rigorous computational methods based on all-atom molecular dynamics (MD) simulations in explicit solvent—such as free energy perturbation and thermodynamic integration (Adcock and McCammon 2006)—can compute reasonably accurate free energy differences as well as many other interesting properties. Unfortunately, such rigorous methods are computationally expensive. Methods treating the solvent implicitly, such as the Poisson-Boltzmann and generalised Born models (Genheden and Ryde 2015), can offer significant speed-ups at the expense of accuracy.

The great successes of deep learning (DL)—a subset of machine learning (ML) employing deep neural networks (NNs)—in the fields of computer vision (Voulodimos et al. 2018), natural language processing (NLP) (Young et al. 2018), and other fields of computer science in recent years kick-started the research and application of DL in many scientific disciplines including physics, chemistry, biology, and medicine (Baldi 2021). In the field of drug discovery, ML has been in use for a long time, and the potential usefulness of the use of DL in virtual screening was identified early on (Unterthiner et al. 2014). The application of modern DL architectures to all stages of the drug discovery pipeline is a very active area of research today (Brown 2021; Gaudelet et al. 2021; Jiménez-Luna

et al. 2021; Jing et al. 2018; Muratov et al. 2020). The main applications in small-molecule drug design consist in the prediction of DTIs, identification of binding sites (Aggarwal et al. 2021; Jiménez, Doerr, et al. 2017; Pu et al. 2019), generation of novel molecular entities (Meyers et al. 2021; Schneider and Clark 2019), and prediction of absorption, distribution, metabolism, excretion, and toxicity (ADMET) properties (Huang, Baber, et al. 2021).

Bioactivity prediction can be performed as a classification task—where molecules are classified as binders (active molecules) or non-binders (inactive molecules)—or as a regression task—where the binding affinity is directly predicted. ML and DL scoring functions (SFs) for the prediction of binding affinities (regression) are useful in lead optimisation, in contrast with SFs that try to identify binders amongst a large pool of non-binders (classification) and are used in virtual screening to identify a hit. Another task where SFs are commonly used is pose prediction, where near-native poses are distinguished from incorrect poses (classification). Pose prediction and binding affinity prediction are complementary tasks in molecular docking, where a pose is generated and subsequently scored according to the predicted binding affinity.

The development of DL methods for pose prediction and binding affinity prediction have the potential to speed up and transform the early stages of drug discovery, especially when combined with generative models for the generation of new and interesting chemical entities as well as ADMET predictions.

The development and assessment of methods for pose prediction, binding affinity prediction, and generative modelling for the generation of novel molecular entities are the main subjects of this work.

1.1 Drug Discovery and Drug Development

1.1.1 Target Identification

Target-based drug discovery (TBDD) starts with the identification of a target involved in the disease of interest (Croston 2017). Targets can be a broad range of biological entities. Common targets are proteins and nucleic acids. TBDD is different from phenotypic drug discovery which relies on measures of response on normal or disease physiology (Swinney 2013; Vincent, Nueda, et al. 2022).

Often, the structure of the target is resolved—using X-ray crystallography, high-throughput X-ray crystallography (Blundell and Patel 2004), or, more recently, cryogenic electron microscopy (cryo-EM) (Nannenga et al. 2014)—, and therefore drug discovery efforts will focus on the target structure, leading to structure-based drug design (SBDD).

The target identification step involves direct biochemical methods—such

as identification of binding using labelled probes—, genetic interaction or genomic methods—manipulating the target of interest in cells by changing their sensitivity—, and computational inference methods (Schenone et al. 2013).

Target identification and validation require a deep understanding of the role of the target under investigation, and its role in different biological pathways related to the disease of interest. For target validation, a wide range of techniques is employed, to increase the confidence that the identified target is indeed involved in disease pathways.

1.1.2 Hit Identification

Once the target for drug discovery has been identified and validated, the goal of small molecule drug discovery is to find potential drug candidates. Given the complexity of the task, the first step is to identify a *hit*—a molecule with reasonable properties that can be further optimised in subsequent steps of the drug discovery pipeline.

There are several techniques for hit identification. Arguably the most common one is high-throughtput screening (HTS), where a library of known compounds is screened against the target of interest to determine their bioactivity and find potential drug candidates. However, HTS remains costly, depending on the level of automation of the screening pipeline as well as the cost of the compound library. If the target is part of a well-known and well-studied target class—such as kinases or G-protein-coupled receptors (GPCRs)—*focussed screening* can be employed instead.

An alternative to HTS is *virtual screening (VS)*, where screening is performed *in silico* to reduce costs and increase the number of molecules that can be (virtually) tested. One of the main tools for VS in SBDD is molecular docking, which is discussed in some depth in Chapter 2. Docking relies on very crude approximations, but when combined with expert knowledge from experienced medicinal chemists it can provide interesting starting structures for further elaboration and optimisation.

In recent years, hit identification has benefitted much from *fragmemt-based drug discovery (FBDD)* (Erlanson et al. 2016; Murray and Rees 2009), where small fragments are screened against the target of interest, to identify important interactions in the potential binding sites. Fragments are smaller than drug-like molecules (Congreve et al. 2003; Jhoti et al. 2013), but can form clear and high-quality interactions with the target, and help to identify different parts of the binding site. Different fragments are subsequently grown within the binding site, merged or linked together, to obtain drug-like hit molecules (Kirsch et al. 2019). The application of computational methods in FBDD remains however

challenging, and there is a lot of room for improvement (Grosjean et al. 2022). The application of DL to FBDD is an active area of research, especially concerning fragment linking (Imrie, Bradley, van der Schaar, et al. 2020), and fragment elaboration (Hadfield, Imrie, et al. 2022).

1.1.3 Hit-to-Lead and Lead Optimisation

Once hit molecules are at hand, a lot of effort is spent in *hit-to-lead and lead optimisation*, with the ultimate goal of obtaining potent and selective compounds possessing desirable pharmacodynamic and pharmacokinetic properties—to be tested in *in vivo* models (Hughes et al. 2011).

Hits have reasonable efficiency—the binding energy per non-hydrogen atom—, but they are usually nowhere near the efficiency needed for a commercial product, and several efficiency metrics have to be further optimised (Hopkins et al. 2014). Additionally, drug discovery is an inherent multi-task optimisation process, since, in addition to ligand efficiency, other properties are critical for the development of new drugs. These properties—oral bioavailability, the ability to cross the blood-brain barrier, toxicity, and many others—determine the pharmacokinetic and pharmacodynamic properties of the drug, and are therefore as important as the ligand efficiency and the binding affinity themselves.

In the hit-to-lead transition, the structure activity relationship (SAR) of each hit compound and its modifications is explored, to identify compounds with good activity and selectivity. The compounds with the best potency and selectivity as well as desirable pharmacokinetic (absorption, distribution, metabolism, and excretion (ADME)) properties are usually tested in *in vivo* models at this stage—mainly in mice and rats (Hughes et al. 2011).

Finally, the best lead compounds are optimised, to get rid of possible remaining problems identified in the hit-to-lead transition. Lead optimisation is a complicated and expensive process, which requires several design, make, test, and analyse (DMTA) cycles (Plowright et al. 2012). The design step involves several actors, such as computational chemists and medicinal chemists working together to improve the properties of the molecules tested and analysed in the previous step. How to reduce the number of DMTA cycles using DL—especially with reinforcement learning and synthesis prediction—is a very active area of research (Schneider, Walters, et al. 2020) that can potentially transform the industry by significantly speeding up drug discovery through automation (Schneider 2018).

In SBDD, the knowledge of the structure of the target of interest strongly influences the design of new candidates. Given the knowledge of the target and its binding site—obtained during target identification and hit identification—, the

three-dimensional structure of the target as well as the possible interactions in the binding site rationally inform design decisions. This knowledge can be exploited by expert computational and medicinal chemists, as well as computational models, or a combination of both.

Once one or several lead compounds have been optimised, they go through *pre-clinical testing*. In pre-clinical tests, compounds are tested for their safety in animal models and more information about ADMET properties is collected. Finally, a lead compound is selected and goes through four phases of clinical trials: *phase I clinical trials* (dose-escalation or human pharmacology trials), *phase II clinical trials* (therapeutic exploratory trials), *phase III clinical trials*, and *phase IV trials* (therapeutic use or post-marketing trials).

Failures at any of the stages of drug development are extremely costly (Pretorius 2016). Therefore, there is a lot of interest in improving the early stages of drug discovery to minimise the attrition rate at later—more expensive—stages of drug development.

1.2 Computational Sciences in Drug Discovery

Computational sciences pervade the drug discovery and drug development pipeline, from the management and analysis of data acquired from experiments to actual *in silico* simulations and modelling. In principle, computational methods can reduce costs and increase the speed of several drug discovery steps, by reducing the number of experiments that need to be performed. Cheminformatics and physics-based simulations have been used in drug discovery and drug development for a long time.

Cheminformatics is central to modern drug discovery since the field deals with the representation of molecules and associated data in machine-readable formats. Given the large amount of data pertaining to molecular entities produced in the early stages of drug discovery, cheminformatics is essential to handle and make sense of such a large amount of data (Chen, Kogej, et al. 2018). Cheminformatics has several uses in the drug discovery pipeline (Muchmore et al. 2010). In the early stages of drug discovery, cheminformatics tools can be used to determine chemical similarity, design compound libraries, and perform database searches. In hit-to-lead and lead optimisation, cheminformatics tools combined with statistical methods can help to elucidate the SAR by crafting project-specific quantitative structure-activity relationship (QSAR) models. Finally, in the pre-clinical phase, cheminformatics tools can be useful to model ADME properties, as well as perform predictive toxicology (Raies and Bajic 2016). In recent years, standard statistical methods used in "old school" cheminformatics have been

mixed with ML and DL methods, and cheminformatics has been catapulted once again at the forefront of computational drug discovery research.

Molecular docking—the computational prediction of binding modes and crude estimation of binding affinities—, is another computational tool commonly employed in SBDD, especially for VS. Molecular docking is a central theme in this work, and therefore it is discussed in some detail in Chapter 2.

While docking works well to filter many compounds and provides crude QSAR models, more accurate calculations based on MD can provide further insights and more quantitative results. Since the first simulations of proteins (McCammon et al. 1977), MD has found several applications in drug discovery (Durrant and McCammon 2011a). These applications include identification of cryptic pockets and allosteric binding sites (Feher et al. 2014; Wagner et al. 2016), elucidation of binding mechanisms (Limongelli et al. 2012; Tiwary et al. 2015), accurate calculation of absolute binding free energies (Aldeghi, Bluck, et al. 2018; Aldeghi, Heifetz, et al. 2016), prediction of ligand selectivity (Albanese et al. 2020; Aldeghi, Heifetz, et al. 2017), and fragment elaboration (Alibay et al. 2022).

Unfortunately, such rigorous methods are computationally expensive and often require a lot of expert knowledge and domain expertise (Hahn et al. 2021; Mey et al. 2020). This is particularly the case for membrane proteins, where the lipid bilayer introduces several additional complications (Wu and Biggin 2022).

A recent review of MD simulations in drug discovery is provided by Salo-Ahen et al. (2020), starting from target validation—where MD can help elucidate the dynamic and function of the target—, through lead discovery and lead optimisation—where MD can help to determine binding kinetics and thermodynamic—, concluding with drug formulation—where MD can help elucidate the stability of amorphous or crystalline drugs, as well as drug solubility.

1.2.1 Artificial Intelligence in Drug Discovery

More recently, several ML and DL approaches—some of which are briefly introduced and discussed in Chapter 3—have been applied at different stages of drug discovery. For SBDD, there has been a lot of research on SFs for binding affinity prediction (Meli, Morris, et al. 2022), structure-based VS (Li, Sze, et al. 2021), and structure-based lead optimisation (Li, Sze, et al. 2020).

Beside structure-based DL methods, which are the main focus of this work and will be discussed in detail later on, a lot of research has been devoted to ligand-based models (Baskin 2020), synthesis prediction (Struble et al. 2020), accelerating quantum chemistry calculation (Dral 2020), accelerating MD simu-

lations (Poltavsky and Tkatchenko 2021; Unke et al. 2021; Wang, Ribeiro, et al. 2020), and studying protein folding and dynamics (Degiacomi 2019; Noé et al. 2020; Ramaswamy et al. 2021). Many applications of DL in chemistry (Mater and Coote 2019) have profound implications on computational drug discovery, and drug discovery and drug development in general.

Synthesis prediction—both forward reaction prediction and retrosynthetic prediction—has seen great improvements in recent years (Struble et al. 2020), especially thanks to performant language models applied to text-based representations of reactions (Schwaller, Laino, et al. 2019; Schwaller, Probst, et al. 2021).

Another very active and very exciting area of research is the development of DL-based force-fields (FFs) (Unke et al. 2021), which has the potential to revolutionise MD modelling in drug discovery by providing descriptions of biological systems—including reactivity—with an accuracy comparable to quantum chemistry calculations. ML FFs are orders of magnitude faster than accurate quantum mechanical calculations, but they remain slower than classical FFs.

Gawehn et al. (2016) present a good overview of different applications of DL methods in drug discovery, while Jiménez-Luna et al. (2021) present more recent advances and discuss future perspectives. While ML and DL provide vast opportunities in accelerating and improving the drug discovery process, the impact of such tools in real drug-discovery applications remains to be demonstrated, and Schneider, Walters, et al. (2020) suggest a "curious but cautious" approach to artificial intelligence (AI) applications in drug discovery, with the ultimate goal of increasing efficiency across the whole drug discovery pipeline.

1.3 Protein-Ligand Binding

Protein-ligand binding is a dynamic process that can be described by the following reaction

$$P + L \underset{k_{off}}{\overset{k_{on}}{\rightleftharpoons}} PL,$$

where P represents the protein (substrate) and L the ligand. The binding process is characterised by the rate constant k_{on} while the unbinding process is characterised by the rate constant k_{off}. The differential equation governing this reaction is therefore given by

$$\frac{d[PL]}{dt} = k_{on}[P][L] - k_{off}[PL], \tag{1.1}$$

where [PL] is the concentration of protein-ligand complexes, while [P] and [L] are the protein and ligand concentrations, respectively.

k_{off}—with units of s^{-1}—represents the probability of unbinding per unit time so that $k_{off}[PL]$ is the total number of unbinding events per unit time and per unit volume. k_{on}–with units of $s^{-1}\,M^{-1}$—is defined so that the total number of binding events per unit time and per unit volume is given by $k_{on}[P][L]$, which reflects the fact that a binding event is a two-step process: diffusive collision and complexation (Zuckerman 2010).

1.3.1 Dissociation Constant K_d

At equilibrium, the concentration of protein-ligand complexes is constant, and we can therefore define the *dissociation constant* K_d as

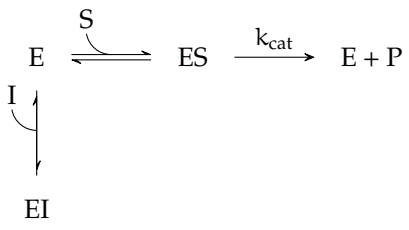
$$K_d \equiv \frac{k_{off}}{k_{on}} = \frac{[P][L]}{[PL]}. \tag{1.2}$$

1.3.2 Inhibition Constant K_i and IC_{50}

The function of an enzyme E can be inhibited by an inhibitor I, which prevents the substrate S to be transformed into a product P. Inhibition can be categorised in three different ways:

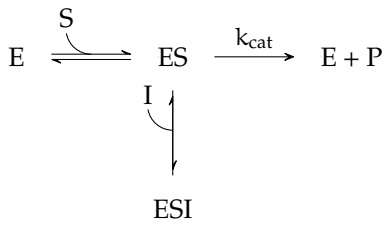
- competitive inhibition
- uncompetitive inhibition
- mixed inhibition

In *competitive inhibition* the inhibitor I competes directly with the substrate by binding in the same binding pocket. Competitive inhibition can be described by the following reaction mechanisms:

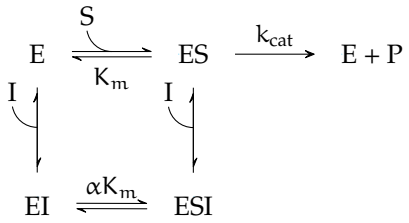


In *uncompetitive inhibition*, the inhibitor I only binds to enzyme-substrate complex ES and prevents the reaction transforming the substrate S into the

products P:



In *mixed inhibition*, the inhibitor can bind to both the free enzyme—preventing the substrate to bind—or the enzyme-substrate complex—preventing the ES $\xrightarrow{k_{cat}}$ E + P reaction. Therefore, this is a combination of competitive inhibition and uncompetitive inhibition:



When $\alpha = 1$, mixed inhibition is referred to as *non-competitive inhibition*.

K_i represents a dissociation constant, but specifically for the binding of an inhibitor to an enzyme. The term K_i is used when the dissociation constant is measured through the inhibition kinetics while K_d is used when binding is measured more directly (using fluorescence quenching or isothermal titration calorimetry (Leavitt and Freire 2001), for example).

A quantity related to the inhibition constant K_i is the *half maximal inhibitory concentration* IC_{50}, which measures the potency of the inhibitor in suppressing the biological function of the enzyme. The IC_{50} is related to K_i by the Cheng-Prusoff equation (Cheng and Prusoff 1973)

$$K_i = \frac{IC_{50}}{1 + \frac{[S]}{K_m}}, \tag{1.3}$$

where $[S]$ is the concentration of the substrate in the assay and K_m is the Michaelis constant. Therefore, the value of the inhibition constant obtained from IC_{50} values depends on the experimental settings. Assuming that all binding events result in effective protein inhibition, then $K_i \approx K_d$ (Hahn et al. 2021).

1.3.3 Binding Free Energy

The Gibbs free energy of binding

$$\Delta G_0^{\text{binding}} = G_0^{\text{bound}} - G_0^{\text{unbound}} \tag{1.4}$$

can be related to the dissociation constant by the Gibbs equation:

$$\Delta G_0^{\text{binding}} = -k_B T \ln\left(\frac{K_a}{c_0}\right) = k_B T \ln\left(\frac{K_d}{c_0}\right), \tag{1.5}$$

where $K_a = K_d^{-1}$ is the *association constant*, k_B is the Boltzmann constant, T is the temperature, and c_0 is the standard concentration.

In many applications, the dissociation constant K_d and the inhibition constant K_i—as well as, sometimes, IC_{50} values—are used interchangeably. Given that equilibrium constants span several orders of magnitude, it is common to consider the negative of their logarithm defined by

$$pK = -\log_{10} K. \tag{1.6}$$

In terms of pK, the Gibbs free energy of binding per mole can be simply expressed as

$$\Delta G_0^{\text{binding}} = -\ln(10) RT pK, \tag{1.7}$$

where R is the ideal gas constant. Eq. (1.7) can be used to readily convert between binding affinities expressed in kcal mol^{-1} and binding affinities expressed in pK units (dimensionless).

1.4 Challenges for ML and DL in SBDD

ML and DL (see Chapter 3 for a short introduction) have been very successful in several areas of computer science and science in general. However, drug discovery—and SBDD in particular—is a peculiar field of application, characterised by very high costs, noisy data, as well as publication and knowledge bias.

In terms of costs, it is extremely expensive to generate new data. The synthesis of new compounds can be a slow and expensive process—although ML and DL algorithms for retrosynthetic analysis might help to considerably lower costs in the future (Johansson et al. 2019)—, and X-ray crystallography for structure determination requires dedicated facilities, and it is also a very time-consuming and expensive process. In contrast with other fields, the data necessary to train and evaluate ML and DL pipelines can't be easily and cheaply generated or easily annotated or labelled.[1] Annotations/labels represent the ground truth that supervised ML algorithms are trained to learn. Chapter 3 describes supervised ML algorithms and how they are trained.

In terms of knowledge and publication bias, it is often the case that only

[1]Data annotation in some fields can be controversial and ethically questionable (Gurav et al. 2019).

good drug candidates appear in the literature and patents, while the large number of failed experiments leading to such candidates only live in proprietary databases of pharma companies or research labs. This is a very unfortunate bias since negative examples are important to train better ML and DL models, and to significantly increase the coverage of chemical space. Chemical space coverage is also subject to bias since when good hits are identified, they are often modified in a step-wise manner to improve desirable properties. This process only produces compounds with similarities, and it is also biased by the prior knowledge or expectations of the computational and medicinal chemists suggesting such modifications. Scaffold hopping (Böhm, Flohr, et al. 2004) helps to palliate some bias coming from hit elaboration, but given the vastness of drug-like chemical space, bias remains.

While resources like the PDB collect a large amount of invaluable data, the quality of the data—and the associated annotations, such as the binding affinity for protein-ligand complexes—might contain several errors, and comes from heterogeneous experimental methods. Additionally, while the PDB database is constantly growing, the number of protein-ligand complexes remains orders of magnitude smaller than the large-scale data sets employed in other DL applications. One-shot learning, few-shot learning (Altae-Tran et al. 2017), and transfer learning (Cai et al. 2020) are therefore essential tools in drug discovery applications. The collection of data from different sources and biological essays is inherently noisy, and the noise is often not quantified nor properly understood.

Finally, while there is a lot of academic research on ML and DL methods for drug discovery—including in this thesis—, it remains difficult and very costly to perspectively verify different models. This verification remains outside the capabilities of single academic groups. While public-private initiatives to properly benchmark computational methods are starting to appear (Ackloo et al. 2022) and crowdsourcing drug discovery approaches are emerging (The COVID Moonshot Consortium et al. 2020), the capabilities of evaluating computational methods within an end-to-end drug discovery pipeline remain an exclusive of (some) pharma companies. Moreover, given the lengths and complexity of the drug discovery pipeline, it is very difficult to evaluate the impact of computational methods—often employed in the early stages of drug discovery—on the performance of pre-clinical candidates in clinical trials.

1.5 Outline

In this work we extend, develop, and analyse different DL methods for SBDD. The work is organised as follows:

Chapter 2 discusses molecular docking, the associated SFs for pose prediction and binding affinity prediction, as well as the data sets employed in this work. Parts of Chapter 2 are reproduced from Meli, Morris, et al. (2022).

Chapter 3 presents a broad overview of ML and DL. The overview focuses on the methods employed in this work, namely NNs, convolutional neural networks (CNNs), and generative models. Besides generative models, we focus mainly on supervised methods for classification and regression. Parts of Chapter 3 are reproduced from Meli, Morris, et al. (2022).

Chapter 4 describes the implementation of docking with flexible residues in Gnina. The implementation has been used for a large-scale analysis of cross-docking with flexible residues, as well as a more focussed analysis outlining the strengths and limitations of the method. Parts of Chapter 4 are reproduced from McNutt et al. (2021).

Chapter 5 describes the development of a CNN SF specifically designed for docking with flexible residues. Additionally, the chapter describes our re-implementation of Gnina's SF in PyTorch, as well as the implementation and evaluation of a multi-task architecture for both ligand and flexible residues pose prediction.

Chapter 6 describes the implementation and evaluation of a novel DL SF for binding affinity prediction, that is translationally and rotationally invariant by construction and is built with ideas from DL models employed in quantum chemistry. Chapter 6 is published in Meli, Anighoro, et al. (2021).

Chapter 7 discusses the evaluation of one of the first structure-based generative models—developed by Masuda et al. (2020)—with an eye on possible uses within Evotec's drug discovery pipeline. We outline the severe limitations of the method, test possible solutions to circumvent the major problems, and discuss possible future directions of the field—with a focus on model evaluation.

Appendix A lists the main contributions to widely used FOSS—such as Gnina, MDAnalysis, Open Babel, and TorchANI—originating from this work, which are not fully described in the main text but represent nonetheless an important and tangible outcome.

Appendix B describes the implementation of a simple Python tool for the calculation of symmetry-corrected RMSDs using graph-isomorphism, widely used across this work. Appendix B in published in Meli and Biggin (2020).

This page intentionally contains only this sentence.

2

Molecular Docking and Data Sets

Contents

2.1 Docking and Scoring Functions

Molecular docking is a computational method for the prediction of the binding pose—and associated binding affinity—of a ligand within the binding site (or sites) of a target of interest. Docking is useful in drug discovery because it allows screening large libraries of compounds given its relatively low computational cost; it is therefore used for VS.

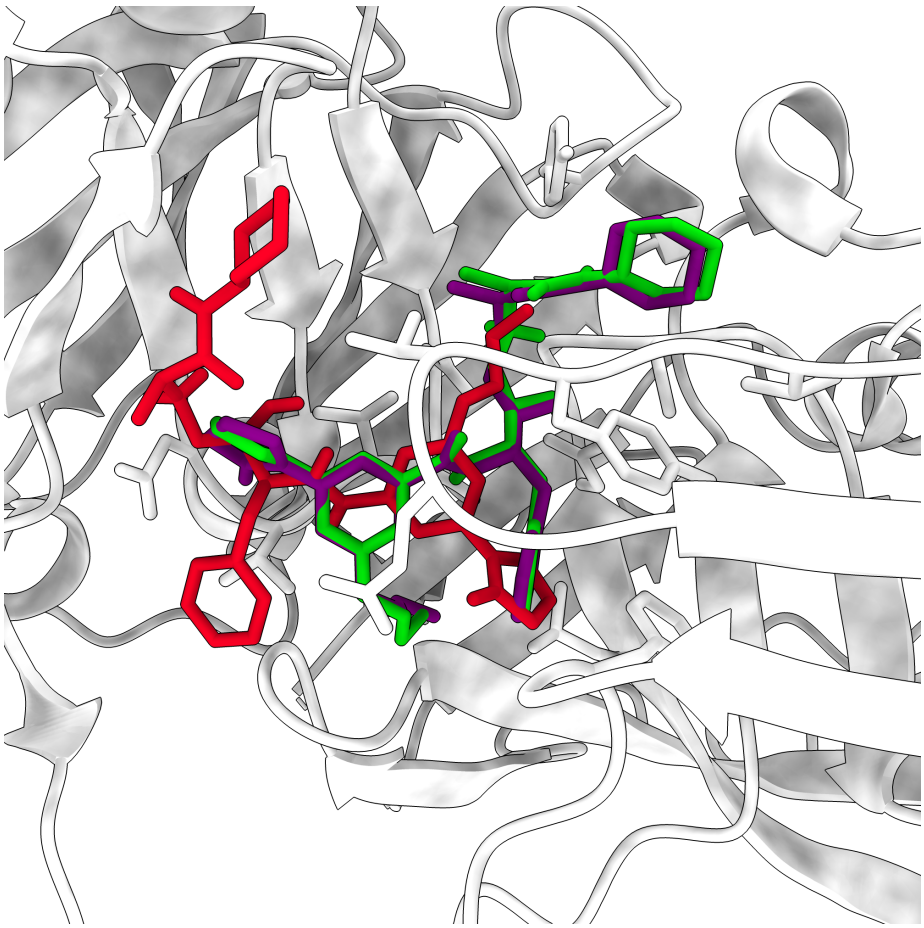


Figure 2.1: Partial results of a docking calculation with SMINA for the protein-ligand complex 2VIZ. The crystallographic pose is shown in purple, the top-ranked pose is shown in green, while a pose very different from the crystallographic one is shown in red.

Fig. 2.1 shows an example of the results of re-docking the ligand of the protein-ligand complex 2VIZ to its cognate receptor. The correct (crystallographic) pose is shown in purple, while two docking poses are shown in green and red. The green pose is very close to the crystal pose in terms of atomic positions, and therefore it is considered a "good" docking pose. In the literature—and in this work—a pose with RMSD smaller than 2.0 Å is considered good (Bursulaya et al. 2003). In contrast, the red pose is very different from the crystallographic pose, and it is therefore considered a "bad" pose (RMSD $>$ 2.0 Å).

Docking tries to strike a balance between computational cost and accuracy. It is therefore much quicker than MD simulations in explicit or implicit solvent, but computational efficiency comes at the cost of accuracy. While MD-based

free energy calculations allow accurate prediction of protein-ligand binding affinities (Aldeghi, Heifetz, et al. 2016) and are predictive of ligand selectivity (Aldeghi, Heifetz, et al. 2017), they remain too computationally expensive for VS and can't deal with large systems routinely. In contrast, docking is usually very fast but relies on several approximations such as neglecting the solvent as well as the flexibility of the receptor.

Docking software uses SFs to guide the conformational search of the binding pose as well as predict the protein-ligand binding affinity. The same SF can be used for both pose prediction and binding affinity prediction, or different SFs designed specifically for each task can be used instead.

SFs usually present two major sources of errors: the limited or absent description of protein flexibility, and implicit treatment of the solvent (Guvench and MacKerell 2009). Given the importance of water molecules in some systems—where they mediate protein-ligand interactions (Ladbury 1996; Lemieux 1996)—, protocols to accurately predict their position and orientation have been developed (Ross et al. 2012; Sridhar et al. 2017). However, taking into account protein flexibility remains challenging.

Historically, SFs for binding affinity prediction and virtual screening have been classified into three categories: FF-based SFs, empirical SFs, and knowledge-based SFs (Böhm and Stahl 2002; Muegge and Rarey 2001). However, recently Liu and Wang (2015) argued that this historical classification overlooks more recent developments in the field and thus proposed an updated classification scheme with four classes of SFs: FF-based or physics-based, empirical or regression-based, knowledge-based or potential of mean force-based, and descriptor-based or ML-based.

This classification is useful to distinguish different methodologies and ideas appearing in the development of SFs. However, some SFs can't be precisely assigned to only one category and the boundary between the four different classes remains rather fuzzy.

In this section we will briefly discuss the first three classes of SFs, often termed "classical" SFs. A good overview of the different SFs can be found in Liu and Wang (2015)—which proposed the current classification of SFs—and a more recent overview of different SFs used in protein-ligand docking is provided by Li, Fu, et al. (2019). While classical SFs are still actively developed and refined today, the research focus has certainly shifted to ML- and DL-based SFs.

2.1.1 Physics-Based (Force-Field-Based) Scoring Functions

Physics-based (or FF-based) SFs use energy terms of a molecular mechanics (MM) FF—whose parameters are determined to reproduce experimental observables

or *ab initio* quantum mechanical calculations (Monticelli and Tieleman 2012)—to evaluate protein-ligand interactions. The non-covalent interaction energy between protein and ligand atoms is usually expressed as the sum of van der Waals and electrostatic interaction terms. In their simplest form, such pairwise interactions are represented by a Lennard-Jones potential—which also includes a repulsive term to describe repulsion at short distances due to the overlap between atomic orbitals (Pauli exclusion principle)—and Coulomb interaction between point charges.

Different physics-based SFs use different potentials to describe van der Waals and electrostatic interactions, depending on the design of the underlying FF. For example, the dielectric constant can be distance-dependent to take into account electrostatic screening due to the solvent and the lower dielectric constant in protein-ligand binding sites (Gilson and Honig 1986; Hingerty et al. 1985; Huang, Grinter, et al. 2010).

Often, additional shorter-range—and, sometimes, directional—terms are added to account for hydrogen bonding as well as solvation energy. Therefore, physics-based SFs can take the following form:

$$\Delta G_{\text{binding}} = \Delta E_{\text{vdW}} + \Delta E_{\text{el}} + \Delta E_{\text{H-bond}} + \Delta G_{\text{sol}}. \tag{2.1}$$

ΔE_{vdW}, ΔE_{el}, and $\Delta E_{\text{H-bond}}$ capture the van der Waals, electrostatic and hydrogen bonding contributions to the binding free energy, respectively.

The solvation energy term ΔG_{sol} can take into account both polar and non-polar contributions. The former accounts for the loss of polar interactions between charged groups and water, while the latter accounts for the de-solvation of hydrophobic groups upon binding.

Finally, empirical terms accounting for the loss of torsional degrees of freedom upon complexation can also be included. Often, simple approximations based on the number of rotatable bonds are used (Böhm 1994; Chang et al. 2007; Huang and Zou 2010; Huey et al. 2007), although more advanced treatments have been suggested (Guedes, Barreto, et al. 2021). The same corrections are applied to empirical and knowledge-based SFs, discussed below.

FF-based SFs are attractive because of their physical origin and because they can leverage advances in FF developments, including the latest advances in ML FF (Unke et al. 2021). However, describing solvent effects in ligand binding remains an outstanding challenge (Darby et al. 2019; Limongelli et al. 2012; Ross et al. 2012).

2.1.2 Empirical (Regression-Based) Scoring Functions

Empirical or regression-based SFs are based on regression analysis to determine the coefficient of different pre-defined terms based on experimental data. This is also what ML (or descriptor-based) SFs do. However, in empirical or regression-based SFs the functional form of the SF is predetermined, and it is often quite simple (such as a linear combination of different contributions) (Ain et al. 2015). As we mentioned previously, the line between the four different classes of SFs suggested by Li, Fu, et al. (2019) is sometimes blurry.

Empirical SFs assuming a linear functional form take the following form (Guedes, Pereira, et al. 2018):

$$\Delta G_{\text{binding}} = w_0 + w_1 \Delta G_{\text{vdW}} + w_2 \Delta G_{\text{H-bond}} + w_3 \Delta G_{\text{entropy}}. \tag{2.2}$$

The functional form of empirical SFs is similar to physics-based SFs. However, in empirical SFs the parameters $\boldsymbol{w}$ are determined by regression analysis—usually multivariate linear regression or partial least squares (Li, Fu, et al. 2019)—to reproduce experimentally determined values.

Often, the different terms in empirical SFs are simple reward or penalty scores. For example, the ChemScore SF has the following functional form (Eldridge et al. 1997; Verdonk et al. 2003):

$$\begin{aligned} &\text{ChemScore} = \\ &w_0 + w_1 S_{\text{H-bond}} + w_2 S_{\text{metal}} + w_3 S_{\text{lipo}} + w_4 H_{\text{rot}} + E_{\text{int}} + E_{\text{clash}} + E_{\text{cov}} \end{aligned} \tag{2.3}$$

where $S_{\text{H-bond}}$ is the score assigned to hydrogen bonds, S_{metal} scores acceptor-metal interactions, S_{lipo} scores lipophilic interactions, H_{rot} describes the loss in conformational entropy upon complexation, E_{int} is the ligand's internal energy, E_{cov} is the covalent energy term—only present when the ligand is covalently bound to the protein target—, and E_{clash} represents the energetic penalty of clashes between protein and ligand atoms.

A fairly recent review of empirical SFs for structure-based virtual screening is provided by Guedes, Pereira, et al. (2018).

2.1.3 Knowledge-Based (Potential-Based) Scoring Functions

Knowledge-based or potential-based SFs are based on pairwise statistical potentials of the form

$$S = \sum_{i \in \text{lig}} \sum_{j \in \text{prot}} \omega_{ij}(r), \tag{2.4}$$

where the distance-dependent pairwise potential $\omega_{ij}(r)$ is given by

$$\omega_{ij}(r) = -k_b T \ln\left(\frac{\rho_{ij}(r)}{\rho^0_{ij}}\right). \tag{2.5}$$

$\rho_{ij}(r)$ is the number density of pairs of type i-j at distance r while ρ^0_{ij} is the same quantity for a reference state where there is no interaction between types i and j (Muegge and Martin 1999). Therefore, if $\rho_{ij}(r)$ is larger than the reference state ρ^0_{ij} it contributes favourably to the SF while if $\rho_{ij}(r)$ is smaller than the reference state ρ^0_{ij} then it contributes unfavourably to the SF. The pairwise potentials $\omega_{ij}(r)$ are obtained from the analysis of interactions in a large data set of protein-ligand complexes. Usually, only pairs of protein and ligand atoms within a certain cutoff are considered ($r \leqslant r_c$).

One of the advantages of knowledge-based SFs is that entropic and solvation contributions are taken into account implicitly (Muegge and Martin 1999). However, some knowledge-based SFs include solvation and entropic effects explicitly (Huang and Zou 2010).

2.1.4 AutoDock Vina and Smina

AutoDock Vina (Trott and Olson 2009) is a SF designed to be faster and more accurate than the AutoDock 4 SF (Morris, Goodsell, et al. 1998; Morris, Huey, et al. 2009), and has been applied successfully in many studies. In this work, we will often consider the AutoDock Vina SF as a baseline.

The AutoDock Vina SF has the following functional form (Trott and Olson 2009):

$$c = c_{inter} + c_{intra} = \sum_{i<j} f_{t_i,t_j}(r_{ij}), \tag{2.6}$$

where $f_{t_i t_j}$ represents the pairwise interaction between atom i of type t_i and atom j or type t_j, where both intermolecular and intramolecular contributions are taken into account. During docking, the value of c is optimised using Monte Carlo search to obtain low-scoring binding modes.

The predicted binding free energy s_0 for the optimal binding mode c_0 is computed as a function g of the intermolecular interactions as

$$s_0 = g(c_0 - c_{intra,0}) = g(c_{inter,0}). \tag{2.7}$$

Other scoring conformations are scored similarly as $s_i = g(c_i - c_{intra,0})$ where $c_{intra,0}$ for the lowest-scoring conformation is used to retain the correct ranking.

Atom types t_i describe the type of atoms involved in the pairwise interaction, and they often combine information about the element and its environment.

Atom type	Ligand	Receptor
AliphaticCarbonXSHydrophobe	Y	Y
AliphaticCarbonXSNonHydrophobe	Y	Y
AromaticCarbonXSHydrophobe	Y	Y
AromaticCarbonXSNonHydrophobe	Y	Y
Bromine	Y	N
Calcium	N	Y
Chlorine	Y	N
Fluorine	Y	N
Iodine	Y	N
Iron	N	Y
Magnesium	N	Y
Nitrogen	Y	Y
NitrogenXSAcceptor	Y	Y
NitrogenXSDonor	Y	Y
NitrogenXSDonorAcceptor	Y	Y
Oxygen	Y	N
OxygenXSAcceptor	Y	Y
OxygenXSDonorAcceptor	Y	Y
Phosphorus	Y	Y
Sulfur	Y	Y
SulfurAcceptor	Y	N
Zinc	N	Y

Table 2.1: AutoDock Vina/Smina atom types. Some types are defined only for ligand or protein atoms, resulting in a total of 18 ligand types and 16 protein types.

The AutoDock Vina atom types are reported in Tab. 2.1. Hydrogen atoms are used to determine the types of other atoms, but they are not explicitly included otherwise. Pairwise interactions are defined in terms of surface distances, therefore $f_{t_i,t_j} = h_{t_i,t_j}(d_{ij})$ where $d_{ij} = r_{ij} - R_i^{vdW} - R_j^{vdW}$—with R_i^{vdW} being the van der Waals radius of atom i.

h_{t_i,t_j} is a weighted sum of different interactions such as steric interactions, hydrophobic interactions, and hydrogen bonds. The actual functional form of the different terms is reported in section C.1. Because the weights are determined to reproduce empirical data from the PDBbind database, the authors classify their SFs as ML-based more than physics-based.

The optimisation of the conformation-dependent SF c is performed using an iterated local search algorithm, which consists of random mutations—global rotations, translation, and changes in rotatable bonds—followed by local optimisation using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Broyden 1970; Fletcher 1970; Goldfarb 1970; Shanno 1970). Therefore, AutoDock Vina generates new conformations using a combination of genetic algorithm and

local gradient-based optimisation.

Smina (Koes, Baumgartner, et al. 2013) is a fork of AutoDock Vina (Trott and Olson 2009) developed specifically for high-throughput scoring, and to make it easier to design custom SFs. The software also expands the support of input/output data formats thanks to the integration of the Open Babel library (O'Boyle, Morley, et al. 2008), which can also be used to compute partial charges (not used by the AutoDock Vina SF, but useful to design different SFs).

2.2 Evaluation of Docking Scoring Functions

The performance of docking SFs can be assessed on different tasks and with different metrics. Here we describe four fundamental tasks used in the Comparative Assessment of Scoring Functions (CASF) assessments, namely: scoring, ranking, docking, and screening. The CASF-2013 and CASF-2016 benchmark data sets are described in section 2.3.2. They are arguably the most used benchmarks for the development and comparison of SFs.

The CASF benchmark provides pre-defined poses to assess all SFs on the same ground, and to assess the ability of the SF to capture relevant protein-ligand interactions and convert them into a meaningful score (a proxy of the binding affinity). This approach allows disentangling the performance of the SF from the details of the sampling algorithm used to generate the docking pose, which might vary widely from software to software. While sampling is an important component of docking, here we focus mainly on the scoring problem. However, it is worth mentioning that DL-based docking protocols that circumvent sampling entirely are emerging (Stärk et al. 2022), and it is currently an active area of ML and DL research.

2.2.1 Scoring

The *scoring power* of a docking SF measures the ability of a SF to compute or predict binding affinities that correlate linearly with the corresponding experimental values.

In the CASF-2016 benchmark, arguably one of the most used benchmarks for the development of SFs, the scoring power of a SF is measured in terms of (linear) correlation between experimental and predicted values. This correlation is measured quantitatively using *Pearson's correlation coefficient* r, defined as

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}}, \tag{2.8}$$

where (x_i, y_i) are the predicted and experimental values of the binding affinity, while $\bar{x}$ and $\bar{y}$ are the corresponding averages on the whole data set. A Pearson's r of 1.0 indicates perfect correlation, while a Pearson's r of 0.0 indicates no correlation. The Pearson's correlation coefficient is often accompanied by the root mean squared error

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_i^N (x_i - y_i)^2}, \tag{2.9}$$

or the mean absolute error (MAE)

$$\text{MAE} = \frac{1}{N}\sum_i^N |x_i - y_i|, \tag{2.10}$$

where N is the total number of samples in the test set.

2.2.2 Ranking

The *ranking power* of a docking SF measures the ability of ranking compounds targeting a specific protein according to their binding affinity. This means that a method can do well even if there is no linear correlation between the predicted and the experimental binding affinity, as long as the compounds are ranked correctly. Given that predicting experimental binding affinities is a very challenging task, a method that can correctly predict the ranking of compounds against a given target is still useful to prioritise the best compounds.

Rank correlation can be quantitatively assessed with several metrics. In CASF-2016, the following metrics are considered: *Spearman's rank correlation coefficient* ρ (Spearman 1961), *Kendall's rank correlation coefficient* τ (Kendall 1938), and *predictive index (PI)* (Pearlman and Charifson 2001).

Spearman's rank correlation coefficient is given by (Spearman 1961)

$$\rho = \frac{\sum_i (r_{x_i} - \bar{r_x})(r_{y_i} - \bar{r_y})}{\sqrt{\sum_i (r_{x_i} - \bar{r_x})^2}\sqrt{\sum_i (r_{y_i} - \bar{r_y})^2}}, \tag{2.11}$$

which is similar to Pearson's r but uses the predicted and experimental ranks (r_{x_i}, r_{y_i})—and the corresponding sample averages—instead of using directly the predicted and experimental values (x_i, y_i). Spearman's rank correlation coefficient essentially measures how well the relationship between the predicted binding affinity and the experimental binding affinity can be described by a monotonic function. The sums run over the number of samples per complex (3 ligands per target for the CASF-2013 benchmark, 5 ligands per target for the

CASF-2016 benchmark; see section 2.3.2 for details).

The PI is given by (Pearlman and Charifson 2001)

$$\mathrm{PI} = \frac{\sum_i \sum_{j>i} W_{ij} C_{ij}}{\sum_i \sum_{j>i} W_{ij}} \tag{2.12}$$

where $W_{ij} = |y_i - y_j|$ is the absolute difference between the experimental binding data of ligands i and j and where C_{ij} is defined as (Pearlman and Charifson 2001)

$$C_{ij} = \begin{cases} 1 & \text{if } \frac{y_j - y_i}{x_j - x_i} < 0, \\ -1 & \text{if } \frac{y_j - y_i}{x_j - x_i} > 0, \\ 0 & \text{if } x_j - x_i = 0. \end{cases} \tag{2.13}$$

The weights W_{ij} reflect the fact that ranking incorrectly compounds with similar experimental binding affinities is less detrimental than ranking incorrectly compounds with vastly different binding affinities. As for Spearman's and Kendall's rank correlation coefficients, the PI is bound on the interval $[-1, 1]$ (with 0 indicating random predictions).

2.2.3 Docking

The *docking power* of a SF measures the ability to distinguish or classify low RMSD poses (native poses) from high RMSD poses (decoys). In the ideal case, the native ligand pose should be the one with the best score, i.e. the top-ranking pose. Unfortunately, this is sometimes not the case due to the approximate nature of docking.

The docking power is measured in terms of success rate, that is, the percentage of targets for which the native ligand pose is ranked amongst the top N poses (with $N = \{1, 2, 3\}$ for the CASF benchmark).

It is important to note that in the CASF-2016 docking benchmark, RMSDs are calculated with the Hungarian algorithm to account for the symmetries of the ligand. As we demonstrated in Meli and Biggin (2020) (see Appendix B), the Hungarian algorithm can lead to artificially low RMSD values. However, to compare with the original benchmark and other work in the literature, we did not re-compute the correct RMSDs, and we used the ones provided as part of the benchmark.

2.2.4 Screening

The *screening power* of a SF measures the ability to distinguish binders from non-binders (carefully selected decoys or random molecules). In *forward screening*

the goal is to identify a binder amongst a large pool of molecules for a given target of interest, while in *reverse screening* the goal is to identify a target for a given binder.

The screening power of a SF can be quantitatively assessed by the success rate of placing the best binder amongst the top $\alpha\%$ molecules (with $\alpha = \{1, 5, 10\}$ for the CASF benchmark) or by the *enrichment factor*

$$\mathrm{EF}_\alpha = \frac{\mathrm{NTB}_\alpha}{\alpha \mathrm{NTB}_{\mathrm{tot}}}, \tag{2.14}$$

where NTB_α denotes the number of true binders amongst the $\alpha\%$ top-ranked molecules, while $\mathrm{NTB}_{\mathrm{tot}}$ denotes the total number of true binders. In the CASF benchmark, the average enrichment factor across all targets is reported.

2.3 Data Sets for Protein-Ligand Binding

The success of ML and DL applications strongly depends on the quality and the size of the data sets used for training. Here we discuss in some depth the data sets used in this work. Such data sets are mostly a curated collection of data coming from different sources, and spanning many years. Therefore, such data is inherently noisy, contains experimental errors, and might contain subtle errors introduced during the data collection. The impact of experimental errors in public data sets of binding affinities is discussed by Kramer et al. (2012).

2.3.1 PDBbind

The PDBbind data set (Wang, Fang, Lu, and Wang 2004) is a curated subset of the PDB. It is arguably one of the most common data sets used to train ML and DL SFs for protein-ligand binding affinity prediction. The data set also contains protein-protein and ligand-nucleic acid complexes.

The origin of the data set can be traced back to 2004, when Wang, Fang, Lu, and Wang (2004) collected protein-ligand complexes from the PDB (release 103, January 2003) and screened the primary references of the identified complexes to extract binding affinity data (K_d, K_i, and IC_{50}).

To train ML and DL SFs, high-quality data is essential—although it has been demonstrated that, in the context of SBDD, including lower quality data can improve performance (Francoeur et al. 2020; Li, Leung, et al. 2015). Therefore, the PDBbind data set is split into a "refined" set and a "general" set (Wang, Fang, Lu, and Wang 2004; Wang, Fang, Lu, Yang, et al. 2005). The "refined" set is a selection of protein-ligand crystal structures with a resolution of 2.5 Å or lower, where there is a single ligand that is non-covalently bound without significant

steric clashes (Wang, Fang, Lu, Yang, et al. 2005). Only systems with associated equilibrium constants K_i and K_d are included in the refined set—IC_{50} values depend on the design of the binding assay (see section 1.3.2)—and complexes are filtered to only contain common organic elements.

The same approach was used to build the PDBbind refined set version 2007 (Cheng, Li, et al. 2009), but it was improved to produce the PDBbind refined set 2013 and subsequent versions (Li, Liu, et al. 2014; Liu, Li, et al. 2014; Liu, Su, et al. 2017). In addition to the previous criteria used to compile the PDBbind refined set 2007, the complexes added to the PDBbind refined set 2013 satisfy the following additional criteria (Li, Liu, et al. 2014): no missing backbone or side chain fragments within 8 Å from the ligand, no extreme values of binding affinity ($1\,\text{pM} < K < 10\,\text{mM}$, where $K = \{K_i, K_d\}$), no multiple binding sites with significantly different binding affinities (> 10 folds difference), no non-standard amino acids within 5 Å from the ligand, and no shallow binders ($< 15\%$ of buried ligand surface). The rules for selecting protein-ligand complexes into the PDBbind refined set 2013, together with their rationale and the difference with the rules used for the PDBbind refined set 2007, are very clearly summarised by Li, Liu, et al. (2014).

The PDBbind data set can be downloaded from `pdbbind.org.cn` (last accessed October 1, 2022). The current release, PDBbind 2020, collects binding affinities and structural data for 23 496 biomolecular complexes, 19 443 of which are protein-ligand complexes.

2.3.2 CASF

The CASF benchmarks are a series of comparative assessments of SFs originally introduced by Cheng, Li, et al. (2009). They evaluate different SFs for their performance on scoring, ranking, docking, and screening on a diverse and high-quality set of protein-ligand complexes. Originally employed to compare mostly classical SFs, the CASF benchmark has become the *de facto* standard for an initial evaluation of ML and DL SFs—especially for protein-ligand binding affinity prediction.

To test different SFs on diverse and high-quality protein-ligand complexes, a data set is extracted from the PDBbind refined set, where high-quality complexes have already been identified. The PDBbind refined set is clustered according to sequence similarity using BLAST (Altschul et al. 1990), with a similarity threshold of 90% (Cheng, Li, et al. 2009). This means that proteins with a sequence similarity higher than 90% are collected in the same cluster since they are likely to represent the same protein or the same protein family.

Once proteins from the PDBbind refined set are clustered by sequence

similarity, clusters containing at least four complexes are retained (Cheng, Li, et al. 2009). This results in a total of 65 clusters, from which three complexes are sampled: the complex with lower binding affinity, the complex with higher binding affinity, and the complex with binding affinity closer to the mean between the highest and lowest binding affinities (Cheng, Li, et al. 2009). This clustered sub-sampling of the PDBbind refined set (often called PDBbind core set) results in a total of $65 \times 3 = 195$ protein-ligand complexes used for the first comparative assessment of SFs (CASF-2007).

For the CASF-2013 comparative assessment of SFs (Li, Han, et al. 2014), the construction of the PDBbind core set was improved by retaining only clusters with five (and not four) proteins (Li, Liu, et al. 2014). Additionally, the best binding affinity has to differ at least 10-fold from the median binding affinity, and the median binding affinity has to differ at least 10-fold from the poorest binding affinity (Li, Liu, et al. 2014). The electron density maps of the remaining complexes were visually assessed; if a complex failed at this step, the next best candidate was selected amongst the same cluster (Li, Liu, et al. 2014). The final PDBbind core set 2013 still consists of 195 protein-ligand complexes from 65 protein clusters (Li, Liu, et al. 2014).

The core set for CASF-2016 (Su, Yang, et al. 2018) brought additional refinements and more data. As usual, the systems within the high-quality benchmark set are selected from the 4057 protein-ligand complexes in the PDBbind refined set (version 2016). The clustering of complexes based on protein sequence similarity remains the same. However, for CASF-2016, five representatives of each cluster were selected instead of the three selected for CASF-2007 and CASF-2013 (Su, Yang, et al. 2018). The representative complexes were selected according to their binding affinity: the complex with the lowest binding affinity, the complex with the highest binding affinity, and three complexes distributed as evenly as possible between the lowest and highest binding affinity (Su, Yang, et al. 2018). The lowest and highest binding affinities differ at least 100-fold (two logarithm units) and the difference between consecutive binding affinities is at least one-fold (that is, $\log(2)$). All ligands were inspected to ensure that there are no identical ligands or stereoisomers (Su, Yang, et al. 2018). The final PDBbind core set (CASF-2016 benchmark set) consists of $57 \times 5 = 285$ protein-ligand complexes, and it is arguably one of the test sets encountered more frequently in the development of ML and DL SFs.

Unlike the PDBbind data set, the CASF benchmark is not updated annually and therefore the latest release to date remains CASF-2016. The CASF benchmark packages can be downloaded from `pdbbind.org.cn/casf.php` (last accessed October 1, 2022).

2.3.3 Subsets of the PDBbind Refined Set

It is very common for ML and DL SFs to be trained on the PDBbind refined or general sets and subsequently tested on the CASF benchmark set. Recently, non-redundant subsets of the PDBbind refined set were introduced by Boyles et al. (2019) and Su, Feng, et al. (2020) to evaluate the ability of ML and DL SFs to generalise when removing increasingly dissimilar examples from the training set that have some similarities with the CASF benchmark set.

In the work of Su, Feng, et al. (2020) the similarity between the training and test sets is measured by three metrics: similarity between protein sequences, similarity between ligand shapes, and similarity between binding pockets. If two protein-ligand complexes—one in the training set, the other in the test set—have all three similarity metrics above a given threshold they are considered redundant. All redundant complexes are removed from the training set with an iterative procedure until the remaining complexes form a representative, non-redundant training set for the given similarity threshold.

2.3.4 CASF Decoys

The decoys for the docking power test and the screening power tests in the CASF benchmark consist of a series of binding poses obtained with three different docking software—with some differences in the docking pipeline between CASF-2013 and CASF-2016.

The preparation of the decoys for the docking power test consists of the production of 1000 poses with RMSD lower than 10 Å split into different RMSD bins of 1 Å width and clustered by conformational similarity into 10 clusters. The member with the lowest strain energy—obtained with a FF—was selected as the representative of each cluster, thus resulting in at most $10 \times 10 = 100$ binding poses for each binding site.

For the preparation of decoys for the screening power test, the protocol used is similar to the one described above where all ligands in the CASF set are docked to all protein targets (57 for CASF-2016). However, due to the size of the problem ($57 \times 285 = 16\,245$ protein-ligand pairs for CASF-2016) only 500 poses were generated with the three docking software and the generated poses were directly clustered by conformational similarity in 100 clusters. As for the docking decoys data set, the ligand with the lowest strain energy was selected as the representative member of the cluster, thus resulting in 100 decoys for each protein-ligand pair.

2.3.5 Cross-Docking Benchmark

Many docking benchmarks are based on *re-docking*, where the docking software is used to dock a ligand within its associated receptor (*cognate receptor*). The ligand is extracted from the crystal structure of the protein-ligand complex and re-docked in the same binding site. This means that the receptor conformation is already a bound conformation (*holo* structure). While assessment of the binding pose is easy in this case—RMSD between the docking pose and the crystal structure—the docking process is easier than in a real-case scenario because the receptor is already in the correct (bound) conformation.

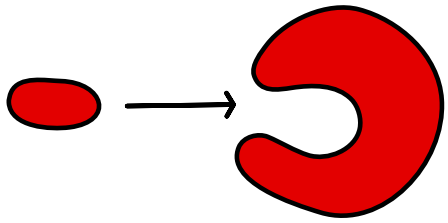
Cross-docking is a significantly more challenging problem, closer to a real-case scenario of *de novo* design, where the crystal structure of the protein-ligand complex is not known and the ligand, therefore, has to be docked to the *apo* structure or another bound structure (bound to a different ligand).

Fig. 2.2 shows schematically the difference between re-docking and cross-docking.

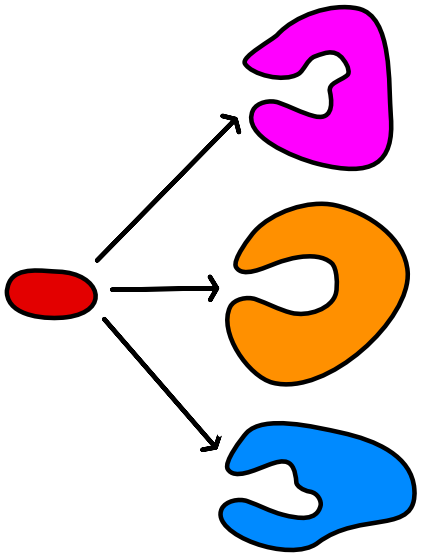
Recently, a novel benchmark set specifically designed for cross-docking was introduced by Wierbowski et al. (2019). The benchmark includes a subset of targets from the DUD-E data set (Huang, Shoichet, et al. 2006; Mysinger et al. 2012) and 4399 ligands for docking.

To build the cross-docking benchmark data set, Wierbowski et al. (2019) screened the PDB for structures with 90% sequence similarity compared to the reference target from the DUD-E data set, and collected affinity data from the Binding Database (Chen, Lin, et al. 2002; Chen, Liu, et al. 2001; Gilson, Liu, et al. 2015; Liu, Lin, et al. 2007), the PDBbind database (Cheng, Li, et al. 2009; Li, Han, et al. 2014; Li, Liu, et al. 2014; Liu, Li, et al. 2014; Liu, Su, et al. 2017; Wang, Fang, Lu, and Wang 2004; Wang, Fang, Lu, Yang, et al. 2005) and the Binding MOAD (Ahmed et al. 2014; Hu et al. 2005; Smith, Nebgen, Zubatyuk, et al. 2019) whenever available. Each structure was aligned to the reference structure and discarded if the protein alignment RMSD was higher than 4 Å or the ligand was not within 4 Å from the reference ligand (Wierbowski et al. 2019). Structures were trimmed by retaining only the chains within 10 Å from the ligand, while cofactors and water molecules within 5 Å were stored separately (Wierbowski et al. 2019). The final data set is composed of 94 unique binding pockets and 4399 ligands, with an average of 46 ligands per target.

The cross-docking benchmark introduced by Wierbowski et al. (2019) was further processed by McNutt et al. (2021) to ensure that the protein-ligand structures could be parsed by GNINA (McNutt et al. 2021) and SMINA (Koes, Baumgartner, et al. 2013). RDKit (Landrum 2022) was used to filter ligands with masses greater than 150 Da and lower than 1000 Da and ligands that could not



(a) Re-docking



(b) Cross-docking

Figure 2.2: Schematic representation of re-docking and cross-docking. In re-docking, the ligand is docked into its cognate receptor (*holo* structure) while in cross-docking the ligand is docked into non-cognate receptors (or the *apo* structure). Cross-docking is a more challenging problem than re-docking, since the receptor is not in the correct binding conformation for the ligand being docked.

be parsed by RDKit were removed (McNutt et al. 2021).

Given the size of the cross-docking benchmark (820 280 protein-ligand pairs), the data set was downsampled (McNutt et al. 2021). The downsampling consists of the random subsampling to 100 protein-ligand pairs with the same binding pocket—no subsampling occurs for pockets with less than 100 associated pairs—resulting in 7970 protein-ligand pairs across 92 pockets.

The final data set from McNutt et al. (2021) is the data set that will be used for cross-docking, unless otherwise stated.

This page intentionally contains only this sentence.

3

Machine Learning and Deep Learning

Contents

R. Meli, G. M. Morris and P. C. Biggin, "Scoring Functions for Protein-Ligand Binding Affinity Prediction using Structure-Based Deep Learning: A Review", *Front. Bioinform.*, 2:885983 (2022). DOI: 10.3389/fbinf.2022.885983 (Meli, Morris, et al. 2022)

Machine learning refers to a broad set of algorithms allowing computers to learn from data. A computer program is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$ (Mitchell 1983).

There are two tasks $\mathcal{T}$ for a ML algorithm that we will focus on: *classification* and *regression*. In a classification task with k classes, we want an algorithm that

can learn the mapping $f\colon \mathbb{R}^n \to \{1, \dots, k\}$ of an input vector $\boldsymbol{x} \in \mathbb{R}^n$ to a class $y \in \{1, \dots, k\}$. In a regression task, we want an algorithm capable of learning the mapping $f\colon \mathbb{R}^n \to \mathbb{R}$ of an input vector $\boldsymbol{x} \in \mathbb{R}^n$ to a real number $y \in \mathbb{R}$.

The performance measure $\mathcal{P}$ allows assessing quantitatively the performance of a model. The functional form of $\mathcal{P}$, often called *loss function*, depends on the tasks $\mathcal{T}$ at hand.

The experience $\mathcal{E}$ is represented by the data the ML algorithm can learn from. There are two broad classes of algorithms: *supervised* and *unsupervised*. Supervised algorithms learn from labelled data where each input vector $\boldsymbol{x} \in \mathbb{R}^n$ is associated with a label or a target, while unsupervised algorithms try to extract useful features from the input vector $\boldsymbol{x} \in \mathbb{R}^n$ alone. In other words, unsupervised algorithms try to estimate the probability distribution $p(\boldsymbol{x})$—or some interesting properties of such distribution—from several examples, while supervised algorithms try to learn the mapping $\boldsymbol{x} \mapsto y$ by estimating the conditional probability distribution $p(y|\boldsymbol{x})$ (Goodfellow, Bengio, et al. 2016).

In this chapter, we will briefly discuss the theory of supervised ML algorithms for classification, regression, and generative modelling, which are used extensively in the remainder of this work.

3.1 Linear Regression

The re-branding of *linear regression* as a ML method might seem a bit of a stretch. However, from the definition given above, linear regression is indeed a ML algorithm. In linear regression, the functional form of the mapping between the input vector $\boldsymbol{x} \in \mathbb{R}^n$ and the predicted output $\hat{y} \in \mathbb{R}$—to be distinguished from the true value $y \in \mathbb{R}$—is given by

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b \tag{3.1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of *parameters* or *weights* and b is a *bias* term. The weights and the bias need to be determined from the experience $\mathcal{E}$ to optimise the performance $\mathcal{P}$.[1]

3.1.1 Closed-Form Solution

A linear regression problem has a closed-form solution that can be obtained using the method of least squares. If one defines the residual $r_n \in \mathbb{R}$ as the

[1] Sometimes the bias term b is absorbed into the weight vector $\boldsymbol{w}' = (b, \boldsymbol{w})$ and the input is re-defined as $\boldsymbol{x}' = (1, \boldsymbol{x})^T$, so that $\hat{y} = \boldsymbol{w}'^T \boldsymbol{x}' = \langle \boldsymbol{w}', \boldsymbol{x}' \rangle$, where $\langle \cdot, \cdot \rangle$ is the scalar product between two vectors. In the reminder of this chapter, we assume—for simplicity and without loss of generality—that the weights vector incorporates the bias term.

difference between true and predicted values

$$r_n = \hat{y}_n - y_n = f(\boldsymbol{x}_n; \boldsymbol{w}) - y_n, \tag{3.2}$$

the method of least squares consists in minimising the sum of squared residuals

$$\mathcal{L} = \sum_n r_n^2 \tag{3.3}$$

to find the weights $\boldsymbol{w}$. The derivative of $\mathcal{L}$—often called *error function* or loss function—with respect to the weight w_i can be computed using the chain rule, and it is given by

$$\frac{\partial \mathcal{L}}{\partial w_i} = 2 \sum_n \left(\boldsymbol{w}^T \boldsymbol{x}_n - y_n \right) (\boldsymbol{x}_n)_i \,, \tag{3.4}$$

where $(\boldsymbol{x}_n)_i$ denotes the i-th component of the n-th observation. The zeros of the derivatives of the loss function with respect to weights and biases can be determined by solving the linear system of the corresponding equations. This gives a closed-form solution to the problem.

By defining the matrix of independent variables $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T$ and the vector of dependent variables $\mathbf{Y} = (y_1, \ldots, y_N)^T$, and by setting all partial derivatives of the loss function with respect to the weights to zero, we can solve explicitly for $\boldsymbol{w}$ to obtain[2]

$$\boldsymbol{w}^* = \underbrace{\left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T}_{\mathbf{X}^+} \mathbf{Y}, \tag{3.5}$$

which is a closed form solution for $\boldsymbol{w}$ which minimises $\mathcal{L}$.

Determining the optimal weights for the model that minimise the loss function can be formally expressed as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L} \left(\{\boldsymbol{x}_i\}, \{y_i\}; \boldsymbol{w} \right). \tag{3.6}$$

3.1.2 Iterative Solution

With linear regression, we can obtain a closed-form solution because of the linearity of the model with respect to the weights and thanks to the convexity of the loss function. However, in general, there is no closed-form solution for an arbitrarily complex model, and non-convex loss functions. Therefore, a different and more general approach is needed.

Since the goal is to determine the model parameters that minimise the loss

[2] This expression is only valid when $\mathbf{X}$ has linearly independent rows—that is, if $\mathbf{X}^T \mathbf{X}$ is invertible. If this is not the case, the Moore-Penrose pseudo-inverse $\mathbf{X}^+$ needs to be computed instead.

function, one can cast the problem in terms of numerical optimisation as in Eq. (3.6). When the gradient of the loss function with respect to the model parameters is available, one can use gradient-based optimisation techniques to optimise the loss function and find the model parameters corresponding to a (local) minimum. In the case of linear regression, the optimisation problem is convex and therefore the one and only minimum—the global minimum—can be found explicitly.

Using the steepest gradient descent method (Nocedal and Wright 1999), it is possible to update the weights and biases of the model iteratively as

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}, \tag{3.7}$$

where η is called *learning rate*, and it is a *hyperparameter* to be optimised.

3.1.3 Stochastic Gradient Descent

The steepest gradient descent method discussed above requires the predictions for the whole dataset to be computed, to determine $\nabla_{\boldsymbol{w}} \mathcal{L}$. For very large data sets, this is a problem since iterative training becomes prohibitively expensive. Therefore, training usually occurs in batches: the examples in the data set are randomly sampled and the weights and biases are updated according to the gradient computed only with such samples. Batches are quite small compared to the data set—typical sizes are low powers of 2, such as 32 or 64, therefore they are often called *mini batches*—and every batch corresponds to an iteration, while a complete pass over the training set is called an *epoch*.

The fact that random subsets of the training set are used at each iteration introduces stochasticity to gradient descent. In the context of optimisation, this stochasticity is often considered a good thing since the variability of gradients can prevent the method to get stuck in a "bad" local minimum of the loss function.

Optimisation methods are often combined with the *momentum* method (Polyak 1964), where

$$\Delta \boldsymbol{w}^{(k+1)} = \beta \Delta \boldsymbol{w}^{(k)} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}, \tag{3.8}$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \Delta \boldsymbol{w}^{(k+1)}, \tag{3.9}$$

with β an exponential decay factor (in the interval $[0, 1)$) determining the contribution of $\Delta \boldsymbol{w}^{(k)}$ to the current update. The momentum method helps to accelerate convergence (Sutskever et al. 2013). The contribution $\Delta \boldsymbol{w}^{(k)}$ can be interpreted as a considerable velocity if $\Delta \boldsymbol{w}^{(k)}$ is important in the initial steps, and converges to the standard negative gradient when β becomes small.

3.2 Logistic Regression

Logistic regression is one of the simplest models for classification. In *binary classification* the model has two output classes ($y_n \in \{0, 1\}$), but in *multi-class classification* the model has multiple output classes.

For binary classification with multiple explanatory variables, the logistic function is defined as

$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^\intercal \boldsymbol{x} + b)}}, \tag{3.10}$$

and it can be interpreted as the probability of the dependent variable being equal to the positive class: $p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{x})$.

Binary classification is often implemented as a multi-class classification task with only two classes. In such cases, a normalised probability distribution is obtained from the output of the different classes using the *softmax function*[3]

$$\text{softmax}_i(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{3.11}$$

Therefore, the softmax function takes an input vector and returns an output vector whose elements are in the range $[0, 1]$ and sum to one, and therefore represents a normalised probability distribution over the classes.

The weights and bias of the logistic function for a binary classification problem are determined by minimising the *binary cross-entropy loss* (or *negative log-likelihood loss*)

$$\mathcal{L} = -\sum_n y_n \log\left(\sigma(\boldsymbol{x}_n)\right) - \sum_n (1 - y_n) \log\left(1 - \sigma\left(\boldsymbol{x}_n\right)\right), \tag{3.12}$$

which measures the *cross-entropy* between target and input probabilities. In practice, the average over the samples—instead of the sum—is often considered.

Unlike linear regression where there is a closed-form solution for the parameters of the models that minimise the loss function, the parameters of a logistic regression model need to be determined iteratively—using the Newton method to solve $\nabla_{\boldsymbol{w}} \mathcal{L} = 0$, or using standard optimisation techniques to minimise $\mathcal{L}$.

3.2.1 Confusion Matrix, Precision, and Recall

The *confusion matrix* is a useful tool to evaluate the performance of classifiers, where the rows of the matrix represent the actual classes while the columns represent the predicted classes and the elements of the matrix encode the number of predictions for each combination. For a binary classifier, the confusion matrix

[3] In practice, the log-softmax function is used instead, which provides better numerical stability since the negative log-likelihood loss can use log-probabilities directly.

is a 2×2 matrix where the diagonal contains the number of *true positives* and the number of *true negatives*, while the off-diagonal elements contain the number of *false negatives* and the number of *false positives*. From the confusion matrix, it is possible to compute several informative metrics.

The *precision* is defined as the accuracy of the positive predictions

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.13}$$

which measures the ratio of true positives amongst all positive predictions.

The *recall*—or *sensitivity*, or *true positive rate* (TPR)—is defined as the ratio of positive instances that are correctly classified:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3.14}$$

There is a trade-off between precision and recall. Since $\sigma(\mathbf{x})$ gives the probability of the positive class, $p(y = 1|\mathbf{x})$, actual predictions are determined using a *decision threshold*—usually set at $1/2$, that is, the positive class is predicted if $p(y = 1|\mathbf{x}) > 1/2$. This trade-off is shown clearly in Fig. 3.1. If the threshold is brought to higher values, some false positives become true negatives thereby increasing the precision, while some true positives become false negatives, thus decreasing the recall. Conversely, lowering the decision threshold reduces precision and increases recall.

Precision-recall curves show the precision plotted against the recall for varying values of the decision threshold. *receiver operating characteristic (ROC) curves* plot the recall (or true positive rate) against the false positive rate (FPR) for a varying decision threshold. The ROC curve of a random classifier is a straight line between $(0, 0)$ and $(1, 1)$. Again, there is a trade-off between TPR and FPR since, for an imperfect classifier, an increase in TPR implies an increase in FPR as well.

A common metric to compare two classifiers is to employ the ROC *area under the curve (AUC)*. Given the shape of a random classifier ROC curve described above, the AUC for a random classifier is 0.5.

3.3 Feed-Forward Neural Networks

Feed-forward NNs (also known as multi-layer perceptrons (MLPs), fully-connected NNs, artificial NNs, or simply NNs) consist in a series of linear layers combined with point-wise non-linearities called *activation functions* (Bishop 2006). Originally, feed-forward NNs were inspired by the way neurons in the brain work (McCulloch and Pitts 1943; Rosenblatt 1962; Widrow and Hoff 1960),

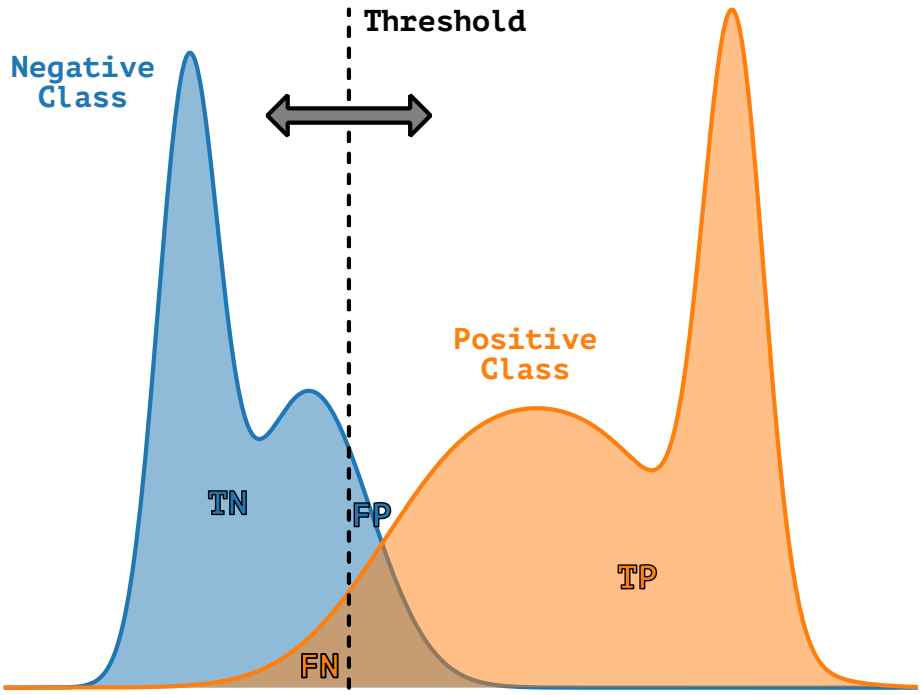


Figure 3.1: Frequency of the different class probabilities (between 0 and 1) for examples in the negative and positive classes. The value of the threshold determines which class probabilities are assigned to each class. A threshold of 0.5 is commonly used in binary classification, but it might be suboptimal.

but the comparison is nowadays quite loose, especially for more advanced architectures.

The basic unit of a NN is a "neuron" (perceptron, or node) and the neurons in a NN are clustered in different layers that are stacked. The neuron j in layer k takes an input vector $\boldsymbol{x} \in \mathbb{R}^N$ and returns an output

$$z_j^{(k)} = g\left(\sum_i^N w_{ji}^{(k)} x_i + b_j^{(k)}\right), \tag{3.15}$$

where $w_{ji}^{(k)}$ (weights) and $b_j^{(k)}$ (bias) for neuron j in layer k are learnable parameters to be determined during training, and where g is a non-linear function—an element-wise activation function. NNs are very expressive and can be regarded as universal approximators (Hornik et al. 1989), provided a large enough number of hidden neurons and some classes of activation functions (Bishop 2006).

Initially, NNs were composed only of few neurons with a single layer (*hidden layer*) between the input layer and the output layer but thanks to the development of algorithms able to train NNs with multiple layers in a simple and efficient way (Rumelhart, Hinton, et al. 1986) NNs became deeper and deeper (now called deep neural networks (DNNs)) by staking together multiple hidden layers.

3.3.1 Backpropagation

To perform gradient-based optimisation of the parameters of a NN so that they minimise the loss function, the derivatives of the loss function with respect to all parameters of the network are required. Computing such derivatives analytically quickly becomes tedious even for simple models.

Automatic differentiation is a technique allowing to compute the gradients of the loss function automatically. There are two flavours of automatic differentiation: *forward-mode* differentiation and *backward-mode* differentiation. The former is suited to differentiate functions with few inputs and many outputs, while the latter is suited to differentiate functions with many inputs and few outputs.

The *backpropagation algorithm* (Rumelhart, Hinton, et al. 1986) is a special case of backward-mode automatic differentiation where there is a single output. Since the loss function is a real-valued function, backward-mode automatic differentiation is commonly employed in modern DL frameworks. The backpropagation algorithm computes derivatives using the chain rule of calculus, in an order that is highly efficient (Goodfellow, Bengio, et al. 2016). The chain rule states that if $\boldsymbol{y} = g(\boldsymbol{x})$ and $z = f(\boldsymbol{y})$—where $g : \mathbb{R}^m \mapsto \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$—, the

derivative of the output z with respect to the input $\boldsymbol{x}$ is given by

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}, \tag{3.16}$$

or, in vector notation,

$$\nabla_x z = \mathbf{J}_g^T \nabla_y z, \tag{3.17}$$

where $\mathbf{J}_g$ is the $n \times m$ *Jacobian matrix* of g ($J_{ij} = \partial y_i / \partial x_j$). The backpropagation algorithm consists in performing the Jacobian-gradient product for each operation in the computational graph (Goodfellow, Bengio, et al. 2016).

If we consider a simple example of a two-layer NN $y = f(x)$ with a single input and a single output and not bias units we have

$$z = \sigma(w_1 x), \tag{3.18}$$

$$y = \sigma(w_2 z), \tag{3.19}$$

and therefore the backpropagation algorithm consists of computing

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial w_2}, \tag{3.20}$$

and then

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_1}. \tag{3.21}$$

From the simple expression above, we see that the term $\partial \mathcal{L} / \partial y$ appears in both expressions and therefore needs to be computed only once.

3.3.2 Neural-Network Potentials

A common application of fitting multidimensional functions with NNs is the creation of neural network potentials (NNPs) to reproduce the potential energy surface (PES) of molecular systems. The idea is to decompose short-range contributions to the total energy E into atomic contributions as (Bartók, Kondor, et al. 2013)

$$E = \sum_i \epsilon_i \left(\mathbf{q}^{(i)} \right) \tag{3.22}$$

where ϵ_i is the atomic contribution for atom i depending on a descriptor $\mathbf{q}^{(i)}$ of the local atomic environment of atom i. The atomic contribution ϵ can be fitted to the available data (usually obtained from computationally expensive quantum mechanical calculations). Fitting ϵ results is a PES given by (Bartók, Kondor, et al. 2013)

$$\epsilon(\mathbf{q}) = \sum_j \alpha_j K \left(\mathbf{q}, \mathbf{q}^{(j)} \right), \tag{3.23}$$

where K is the kernel and where the coefficients $\{\alpha_j\}$ are determined by the fitting procedure. The role of the kernel is to capture the degree of similarity between the atomic environment described by $\mathbf{q}$ and $\mathbf{q}^{(k)}$.

A simple kernel is the dot-product kernel $K_{KD} = \langle \mathbf{q}, \mathbf{q}^{(k)} \rangle$ which results in ϵ being determined by a linear fit of the elements of the descriptor $\mathbf{q}$ (Bartók, Kondor, et al. 2013):

$$\epsilon(\mathbf{q}) = \sum_j \alpha_j \sum_k q_k q_k^{(j)} = \sum_k q_k \underbrace{\sum_j \alpha_j q_k^{(j)}}_{\beta_k} = \langle \boldsymbol{\beta}, \mathbf{q} \rangle. \tag{3.24}$$

There are many choices for the kernel K and for the fitting algorithm used to determine the parameters $\{\alpha_j\}$. Behler and Parrinello (2007) suggested fitting the atomic contributions $\epsilon(\mathbf{q})$ using a MLP. For a MLP with a single hidden layer with n_H neurons we have

$$\epsilon(\mathbf{q}) = b + \sum_i^{n_h} w_i h(\mathbf{q}, \mathbf{u}_i), \tag{3.25}$$

where b, w_i and n_h are parameters of the NN. For $n_h \to \infty$ and $h(\mathbf{q}) = \tanh(u_0 + \sum_j q_j u_j)$ it is possible to show that the latter equation can be written as Eq. (3.23) with $K_{NN}(\mathbf{q}, \mathbf{q}') \sim -|q - q|^2 + c$ (Neal 1996). Additional layers in the NN essentially perform a non-linear transformation of the input descriptor while the last layer can be regarded as performing the regression task (Bartók and Csányi 2015).

3.4 Convolutional Neural Networks

CNNs (Fukushima 1980; Krizhevsky et al. 2017; Le Cun et al. 1989; Lecun et al. 1998) are a class of NNs that tries to overcome some limitations of feed-forward NNs—large number of parameters, and loss of spatial locality and spatial correlations when vectorising the input—, by using convolution operations instead of matrix multiplications in some of their layers (Goodfellow, Bengio, et al. 2016). Feed-forward NNs use a one-dimensional vector as input which prevents the encoding of spatial relationships, and use many parameters. In contrast, CNNs are based on three main concepts (Bishop 2006): local receptive fields (inspired by the structure of the visual cortex (Hubel 1959; Hubel and Wiesel 1959))—allowing the retention of spatial features—, weight sharing—allowing the reduction of the number of parameters—, and subsampling.

Local receptive fields are implemented in convolutional layers, where neurons in a layer do not receive the output of all neurons in the previous layer (as in

fully-connected NNs) but only the ones in their local receptive field (Géron 2019). For two-dimensional grid-based inputs—such as images—, the output of the neuron at location (i, j) of feature map k of the convolutional layer l is given by (Géron 2019)

$$z_{i,j,k}^{(l)} = b_k^{(l)} + \sum_{u=1}^{f_h^{(l)}} \sum_{v=1}^{f_w^{(l)}} \sum_{k^{(l-1)}=1}^{f_n^{(l-1)}} x_{i^{(l-1)},j^{(l-1)},k^{(l-1)}} \cdot w_{u,v,k^{(l-1)},k}^{(l)}, \tag{3.26}$$

with

$$\begin{cases} i^{(l-1)} = u s_h^{(l)} + f_h^{(l)} - 1, \\ j^{(l-1)} = u s_w^{(l)} + f_w^{(l)} - 1. \end{cases} \tag{3.27}$$

f_h and f_w are the height and the width of the receptive field—the size of the 2D *convolutional kernel*—while s_h and s_w represent the strides—the size of the displacement of the receptive field. $f_n^{(l-1)}$ denotes the number of feature maps in the previous layer $(l-1)$. b_k^l is a bias term associated to feature map k while $w_{u,v,k',k}^{(l)}$ denotes the weight term associated to the connection between the input located at (u, v) in feature map $k^{(l-1)}$ (relative to the neuron's receptive field) and the neuron in feature map k of layer l. Both b_k^l and $w_{u,v,k',k}^{(l)}$ are learnable parameters to be determined during training. Fig. 3.2 shows schematically the action of a 2D convolutional kernel.

More intuitively, a convolutional filter is a (learnable) tensor applied to the input of the convolutional layer as a convolution—the sum over all elements of an element-wise multiplication—between said matrix and a sub-sample of the input with the same dimensions, to produce a single number. Hand-crafted convolutional filters have been used in computer vision for a long time (see, for example, the Sobel-Feldman filter for edge detection), but in CNNs the convolutional kernel is learned, to extract relevant features from the input.

Parameter sharing in a convolutional network comes from the fact that each weight $w_{u,v,k',k}^{(l)}$ of the kernel is used at every position of the input, avoiding the need to learn a parameter for each input element as it is the case in MLPs. Parameter sharing does not reduce the computational complexity of the forward pass, but significantly reduces the number of parameters in the network (when the size of the convolutional kernel is much smaller than the size of the input) and, therefore, the associated memory footprint (Goodfellow, Bengio, et al. 2016).

Pooling layers—such as maximum pooling (Zhou, Chellappa, et al. 1988), and average pooling—are often inserted after (activated) convolutional layers to make the representation approximately invariant to small translations (Goodfellow, Bengio, et al. 2016). Additionally, they reduce the size of the input of the next

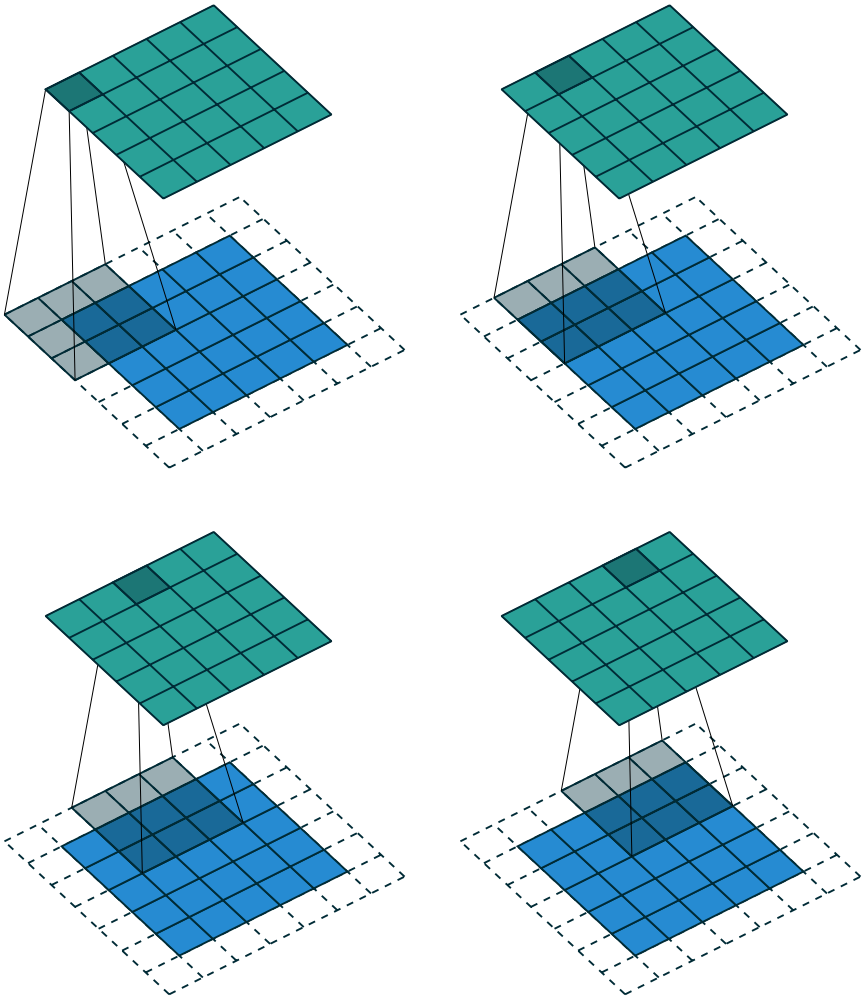


Figure 3.2: Application of a 3×3 convolutional kernel (grey) to an input feature map (blue). By choosing a padding of one (dashed lines) and a stride of one, the output feature map (cyan) retains the same spatial dimensions of the input feature map. Images reproduced from Dumoulin and Visin (2016) under the MIT licence.

layer thus increasing the computational efficiency of the CNN, and are essential for dealing with inputs of varying size (Goodfellow, Bengio, et al. 2016). Pooling layers are applied to each feature map independently so that the depth of the tensor representation is conserved—only spatial dimensions are reduced. Pooling layers are often of size 2^n (where n is the number of spatial dimensions) and applied with a stride of 2, so that each spatial dimension is reduced by half.

CNNs have been very successfully applied to different tasks in computer vision such as image classification (Krizhevsky et al. 2017) in the ImageNet challenge (Deng et al. 2009; Russakovsky et al. 2015). Their impressive performance has been partially attributed to their ability to build a hierarchical representation of features, from simple features—for shallow layers—to complex patterns—for deep layers (Zeiler and Fergus 2013).

3.4.1 CNNs in Drug Discovery

Thanks to the great success in computer vision, CNNs have been applied to drug discovery as well. Since CNNs are well suited to work with dense grid-based representations, in SBDD several authors have developed density-based representations of protein-ligand binding sites for binding site prediction (Jiménez, Doerr, et al. 2017), pose prediction (Ragoza, Hochuli, et al. 2017), and binding affinity prediction (Jiménez, Škalič, et al. 2018; Stepniewska-Dziubinska et al. 2018). One such representation, developed by Ragoza, Hochuli, et al. (2017) and used in parts of this work, is discussed in detail in section 4.2.1.

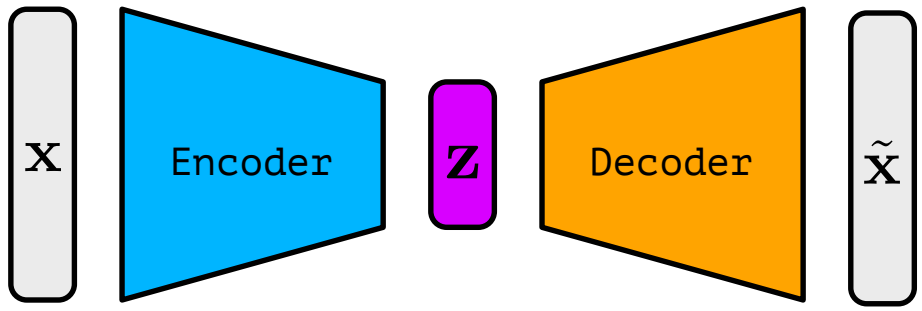
3.5 Architectures for Generative Modelling

There are several architectures for generative modelling, such as autoencoders (AEs), variational autoencoders (VAEs), generative adversarial networks (GANs), recurrent neural networks (RNNs), normalising flows (NFs), and diffusion models. Here we focus on AEs, VAEs, and GANs, which are the architectures used in Chapter 7.

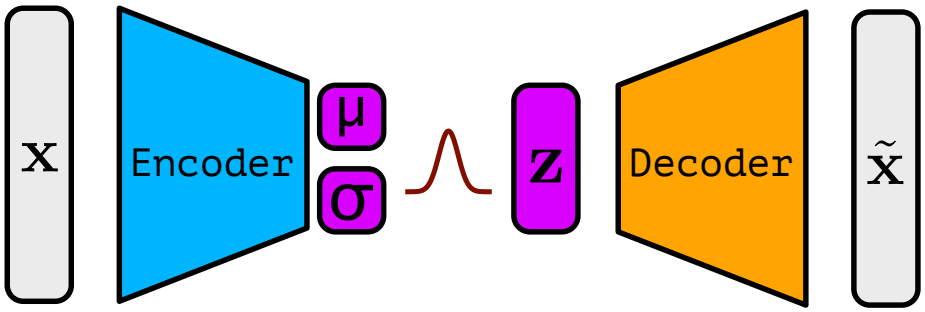
3.5.1 Autoencoders

AEs (Baldi 2012; Rumelhart, McClelland, et al. 1986) are ML or DL architectures designed to learn a dense representation of an input space with unsupervised training (Géron 2019). The dense representation, also called *latent representation*, is usually of much lower dimensionality that the input space.

Fig. 3.3a shows a schematic of an AE architecture, composed of an encoder and a decoder. The input is compressed by the encoder into a lower dimensionality



(a) AE



(b) VAE

Figure 3.3: AE and VAE architectures. In an AE (a) the latent space representation is encoded directly, while in a VAE (b) the latent space representation is sampled from a probability distribution (represented here by a Gaussian distribution with encoded mean μ and variance σ). The actual architecture of the encoder and the decoder depends on the task at hand.

representation (latent representation) which can then be decoded back to the input space by the decoder. When an AE is trained to minimise the $\mathcal{L}_2$ loss between input $\mathbf{x}$ and output $\tilde{\mathbf{x}}$, it learns to copy the input to the output. Thanks to the information bottleneck enforced by the lower dimensionality of the latent representation, however, only important features are retained in the latent space, which constitutes an efficient data representation. The latent representation is therefore a by-product of learning the identity operator through an information bottleneck (Géron 2019).

AEs are useful for dimensionality reduction (Baldi and Hornik 1989; Bourlard and Kamp 1988; Plaut 2018), but have many other applications such as feature detection, de-noising (Vincent 2011; Vincent, Larochelle, et al. 2008), and generative modelling (Vincent 2011). For dimensionality reduction, an autoencoder where encoder and decoder networks are a single fully-connected layer without activation functions is equivalent to principal component analysis (PCA) (Plaut 2018).

Unfortunately, the latent space representation learned by AE does not have

any particular structure—none is enforced—and it is, therefore, difficult to sample latent space points that decode into meaningful representations. The distribution of latent space points is undefined, and no constraints are forcing the autoencoder to decode meaningful representations since the latent space is not enforced to be structured or smooth (Foster 2019).

3.5.2 Variational Autoencoders

VAEs (Kingma and Ba 2014) have a similar architecture to AEs (see Fig. 3.3), but have been designed to generate new data $\tilde{\mathbf{x}}$ that is similar to the examples in the training set $\mathbf{x}$ by enforcing continuity in the latent space. Despite the similar architecture, VAEs are effectively a different class of models compared to AEs since they are probabilistic.

The generative model in VAEs is represented by $p_\theta(z)p_\theta(\tilde{x}|z)$, which indicates that data $\tilde{x}$ are generated by a random process involving an unobserved random variable z in two steps: z is sampled from the *prior distribution* $p_\theta(z)$ and then $\tilde{x}$ is generated from the conditional distribution $p_\theta(\tilde{x}|z)$ (Kingma and Ba 2014). The prior distribution and the conditional distribution are parametric functions of θ and are assumed to be differentiable with respect to both θ and z. With complicated likelihood functions $p_\theta(\tilde{x}|z)$ the *posterior distribution* given by *Bayes' theorem*

$$p_\theta(z|\tilde{x}) = \frac{p_\theta(\tilde{x}|z)p_\theta(z)}{p_\theta(\tilde{x})} \tag{3.28}$$

is intractable because the marginal likelihood

$$p_\theta(\tilde{x}) = \int p_\theta(\tilde{x}, z)\, dz = \int p_\theta(\tilde{x}|z)p_\theta(z)\, dz \tag{3.29}$$

is impossible to compute analytically (Kingma and Ba 2014). To solve this problem, an approximation $q_\phi(z|\tilde{x})$ to the intractable $p_\theta(z|\tilde{x})$ is introduced (Kingma and Ba 2014). Essentially, the idea is that $p_\theta(z|\tilde{x})$ will be close to zero over most of the latent space—given that $\tilde{x}$ can be generated only from some values of z—and, therefore, it does not contribute much to $p_\theta(\tilde{x})$ in Eq. (3.29). Therefore, the approximation $q_\phi(z|\tilde{x})$ to the posterior distribution can be constructed in such a way that for a given $\tilde{x}$ the distribution is over values of z likely to produce $\tilde{x}$ (Doersch 2021).

The unobserved random variables z can be interpreted as latent space representations. The model $q_\phi(z|\tilde{x})$ is a *probabilistic encoder* (given $\tilde{x}$, it produces a probability distribution over all possible values of z from which $\tilde{x}$ could have been generated), while $p_\theta(\tilde{x}|z)$ is a *probabilistic decoder* (given z, it produces a probability distribution over all possible corresponding values of $\tilde{x}$).

Fig. 3.3b shows the architecture of a VAE. The probabilistic encoder $q_\phi(z|\tilde{x})$—

an approximation to the posterior distribution of the generative model $p_\theta(z|\tilde{x})$—is a NN multivariate Gaussian distribution with diagonal covariance ($q_\phi(z|\tilde{x}) = \mathcal{N}(z; \mu, \sigma^2)$, see section D.1), while the prior over the latent space variables is a centred isotropic multivariate Gaussian distribution $p_\theta(z) = \mathcal{N}(z; 0, 1)$. The parameters θ and ϕ of the model, defining the encoder and decoder networks, are obtained by optimising the following VAE loss:

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) + \mathcal{D}_{\text{KL}} \left[q_\phi(z|x)||p_\theta(z)\right], \tag{3.30}$$

where $\mathcal{D}_{\text{KL}}$ is the Kullback-Leibler (KL) divergence (see section D.2).

VAEs have a stochastic component at the core of the network, where sampling from a Gaussian distribution of mean μ and variance σ is performed to obtain a latent space vector $z \sim \mathcal{N}(\mu, \sigma^2)$. This poses problems for back-propagation. Using the *reparametrisation trick* (Kingma and Ba 2014), the sampling process can be re-written as

$$z = \mu + \sigma\, \mathcal{N}(0, 1), \tag{3.31}$$

and therefore the sampling of the latent space vector z can be regarded as a deterministic computation with an extra random variable whose distribution does not depend on the parameters with respect to which derivatives are computed during backpropagation (Goodfellow, Bengio, et al. 2016).

The standard deviation σ for the distribution from which latent space points z are sampled is produced by the encoder. However, it is possible to manually control the standard deviation with an additional parameter η, called *variability factor*:

$$z = \mu + \eta\sigma\, \mathcal{N}(0, 1). \tag{3.32}$$

η multiplies the standard deviation σ and therefore gives some control over the posterior sampling once the model is trained. Generative models are usually trained with $\eta = 1$, and the parameter is only changed when sampling the learned distribution.

3.5.3 Generative Adversarial Networks

GANs are another architecture for generative modelling introduced by Goodfellow, Pouget-Abadie, et al. (2014), where a generator network is trained to generate new samples starting from a random distribution—which is essentially the latent representation—to fool a discriminator network that is trained to distinguish between generated and real samples.

The generator and the discriminator networks are trained together so that the generator has to come up with better and better samples while the discriminator

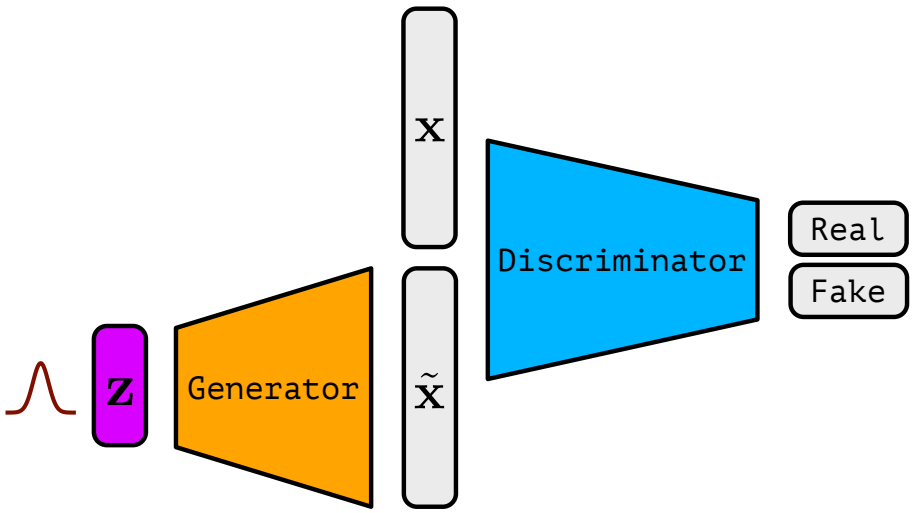


Figure 3.4: GAN architecture. The actual architecture of the generator and of the discriminator networks depends on the task at hand.

tries to improve in its discrimination task. This corresponds to a zero-sum game between two players, where an improvement for one player corresponds to a loss for the other.

Fig. 3.4 shows a typical GAN architecture: a latent vector $\mathbf{z}$ is sampled from a random distribution and is used by the generator network to generate a sample $\tilde{\mathbf{x}}$. The discriminator then tries to distinguish generated samples $\tilde{\mathbf{x}}$ from real samples $\mathbf{x}$. Once training is converged, the discriminator network is no longer useful and the generator network can be used to generate new samples $\tilde{\mathbf{x}}$ from random samples $\mathbf{z}$.

Because of the competing networks, training happens in two phases (Géron 2019): in the first phase, the discriminator is trained to distinguish real from generated samples, while in the second phase the generator is trained by generating samples that are labelled as real so that the discriminator will believe them to be real. The generator is never confronted with real samples since it is only sampling from a random distribution, but as training progresses the weights are tuned—with backpropagation—so that the generated samples can fool the discriminator (by being similar to real samples).

GANs are notoriously hard to train (Goodfellow 2017; Salimans et al. 2016). In particular, GANs suffer from mode collapse—where the generator output becomes less and less diverse—and instabilities—where the parameters start oscillating during training and become unstable. Therefore, training GANs requires very careful hyperparameter tuning.

3.5.4 Generative Models in Drug Discovery

Applications of generative models in drug discovery mainly concern *de novo* design. *De novo* design is not a new concept (Schneider and Fechner 2005), but recent advances in generative models are permeating the field (Sanchez-Lengeling and Aspuru-Guzik 2018). For example, Gómez-Bombarelli et al. (2018) developed a VAE to learn a continuous representation of molecules, exploitable for automated chemical design. A similar model was used for the identification of potent kinase inhibitors (Zhavoronkov et al. 2019). However, some argued that the best-generated compounds were very similar to a marketed drug and that the novelty and significance of the compounds obtained from generative models remain to be proven (Walters 2019; Walters and Murcko 2020a).

Architectures that allow the generation of molecules but are not discussed here are RNNs. RNNs can learn the grammar of SMILES—a one-dimensional text-based representation of the molecular graph—and subsequently used to generate new SMILES strings, corresponding to novel molecular entities (Gupta et al. 2018).

How to best evaluate generative models for drug discovery applications is still an open question. In this work, after testing and evaluating a novel structure-based generative model developed by Masuda et al. (2020), we will discuss some problems with current evaluations and the need for better and more stringent benchmarks (see Chapter 7).

4

Flexible Docking with GNINA

Contents

> A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, **R. Meli**, M. Ragoza, J. Sunseri and D. R. Koes, "GNINA 1.0: molecular docking with deep learning", *J. Cheminform.* 13, 43 (2021). DOI: 10.1186/s13321-021-00522-2 (McNutt et al. 2021)

The code associated to this chapter is available in the following GitHub repositories: RMeli/gsoc19, RMeli/ifd, gnina/gnina (some contributions), gnina/scripts (some contributions), and dkoes/GNINA-1.0 (some contributions).

In this chapter, we describe the implementation and evaluation of docking with flexible residues in GNINA, a FOSS DL framework for molecular docking (McNutt et al. 2021). First, we identify and fix several problems with the

implementation of docking with flexible residues inherited from Smina (Koes, Baumgartner, et al. 2013)—the classical docking software based on AutoDock Vina (Trott and Olson 2009) from which Gnina has been forked. Then, we describe the implementation of CNN SFs for docked poses with flexible residues. Finally, we evaluate the performance of docking with flexible residues with Gnina's default CNN SF, to recommend the best possible default parameters for the end-user in version 1.0 of the software, which is supposed to "just work" out of the box. The evaluation is performed on a very large cross-docking data set as well as the small curated data set used to validate the induced fit docking (IFD) protocol of Sherman et al. (2005).

4.1 Docking with Flexible Residues

Receptor flexibility is important in protein-ligand binding (Feixas et al. 2014) since biomolecular recognition relies on effects such as conformational selection—the change in the distribution of receptor conformations upon binding (Kumar, Ma, et al. 2008; Ma et al. 1999; Tsai et al. 1999)—and induced fit (Koshland 1958)—the conformational change of the protein induced by ligand binding.

Unfortunately, the receptor flexibility is often neglected in large structure-based virtual screening (SBVS) campaigns because of the additional computational resources and modelling difficulties (Scior et al. 2012). However, different techniques have been developed to tackle this problem, and they can be pooled into two distinct categories: methods that account for local receptor flexibility (within the binding site), and methods that account for global receptor flexibility (Durrant and McCammon 2010).

4.1.1 Ensemble Docking

The idea behind *ensemble docking* is to use several conformations of the receptor—either from experimental data or generated *in silico* using MD or Monte Carlo (MC) simulations—and dock ligands to all of them (Amaro, Baudry, et al. 2018). This methodology stems from the fact that the receptor exists in solution in several conformations whose distribution changes upon binding, due to a shift in the equilibrium now favouring the bound conformation.

Ensemble docking has been applied successfully in several studies (Amaro, Baudry, et al. 2018). Arguably the most famous ensemble docking method is the *relaxed-complex scheme* (Amaro, Baron, et al. 2008; Lin et al. 2002), where long MD simulations of the unliganded receptor are used to sample the conformational space of the receptor, followed by docking of different ligands into the large ensemble of conformations obtained from the simulations.

Some drawbacks of ensemble docking are that experimental data might be scarce (there might be no or few alternative conformations for the same target), the computational resources required to generate the alternative conformations might be prohibitive—especially for slow conformational changes—, and the method is only suitable in cases dominated by conformational selection.

Additionally, it is tricky to combine the results of docking on different receptor conformations and how to best combine the results into a single SF remains an open question. Originally, consensus scoring strategies were employed (Bajusz et al. 2019) but nowadays interesting applications of ML methods to optimally combine the docking results are emerging (Mohammadi et al. 2022; Ricci-Lopez et al. 2021).

Finally, the presence of many ligands and receptor conformations might lead to a prohibitive combinatorial explosion in the number of generated poses.

4.1.2 Flexible Side Chains

Methods that only consider the local flexibility of the receptor instead of the global conformation are more approximate—global conformational changes might be important for binding (Grant, Gorfe, et al. 2010), as discussed above—but computationally more tractable. Additionally, only a single starting conformation of the receptor is used, thus avoiding the question of how to combine results from docking to different conformations altogether.

There are different methods to treat the local flexibility of the receptor within the binding site. One such method is IFD developed by Sherman et al. (2005), where rigid receptor docking is combined with protein structure prediction techniques to take into account induced fit effects. In this method, the ligand is initially docked into a rigid receptor structure where highly flexible residues in the binding site have been mutated to alanine, and the docking process is performed with scaled-down van der Waals interactions—so that steric clashes with the receptor are partially allowed. In a second stage, the top 20 poses are energy minimised to allow the binding site residues—reverted to their original form—and the protein backbone to re-arrange. Finally, the ligand is docked again into the relaxed receptor structure to obtain a final score.

While the IFD procedure has been introduced several years ago and had been used successfully in several projects, the efficient inclusion of receptor flexibility within docking protocols remains an active area of research. Recently, the IFD method was improved using MD resulting in a IFD-MD protocol that appears to perform better than IFD, but at an increased computational cost (Miller et al. 2021).

Another method to treat local receptor flexibility consists of sampling side

chain conformations on the same footing as the ligand conformational sampling—using the same SF. This method is simpler to implement and allows the simultaneous sampling of the ligand and protein side chains conformational spaces. This latter method is the method implemented in Smina—from which Gnina is forked—and it is, therefore, the method that we will consider here.

4.2 CNNs for Docking

CNNs have been developed in the field of computer vision and different architectures have shown fantastic performance on challenging image recognition benchmarks such as ImageNet (Deng et al. 2009; He et al. 2016; Krizhevsky et al. 2017; Russakovsky et al. 2015; Szegedy et al. 2015). CNNs can automatically extract high-level features from an image, and therefore they are well-suited for learning from discrete spatial data (Zeiler and Fergus 2013). Refer to Chapter 3 for a brief introduction of CNNs.

Two-dimensional convolutions can be generalised to three dimensions. Therefore, CNNs can be applied to volumetric data. This is particular interesting in biomedical applications, where volumetric data is often collected in clinical practice using techniques such as magnetic resonance imaging (MRI) (Vlaardingerbroek and den Boer 1996), computed tomography (CT) (Buzug 2011), or positron emission tomography (PET) (Bailey et al. 2005). One of the first applications of three-dimensional CNNs on medical volumetric data was for the segmentation of MRI volumes (Milletari et al. 2016). Given the successes of CNNs in computer vision and biomedical applications, there was a strong interest in applying the same architectures and ideas to SBDD. For example, Jiménez, Doerr, et al. (2017) introduced a volumetric representation of protein-ligand complexes to train a CNN for the prediction of protein-ligand binding sites.

In this chapter, we will focus on the seminal CNN architecture developed by Ragoza, Hochuli, et al. (2017) and implemented in the FOSS Gnina (McNutt et al. 2021). Gnina is a fork of the classical docking software Smina (Koes, Baumgartner, et al. 2013) and includes a custom version of the Caffe (Jia et al. 2014) DL framework for the development and use of CNN-based SFs (Fig. 4.1).

4.2.1 Atomic Density Grids

CNNs were developed originally for image recognition. To use the same state-of-the-art (SOTA) architectures that are very successful in computer vision for SBDD, the protein-ligand binding site needs to be mapped to a similar representation. Images are discrete two-dimensional grids of pixels representing a colour

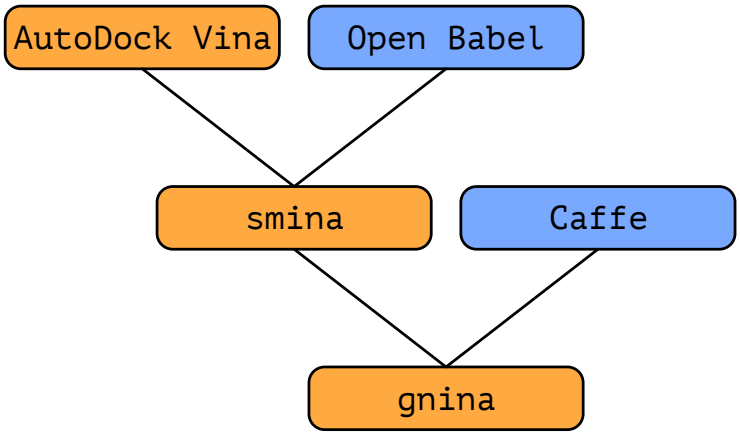


Figure 4.1: Development history of Gnina. Gnina is a fork of Smina integrating the Caffe DL library. Smina is in turn a fork of AutoDock Vina integrating Open Babel for molecular input and output, as well as functionality to develop custom SFs.

intensity and have one such grid for three different fundamental colours (red, green, and blue; RGB). Ragoza, Hochuli, et al. (2017) introduced a discretisation of the protein-ligand binding site into a collection of three-dimensional grids. Each atom of type t is encoded into a density distribution $A(r; R_t)$ around the atom centre given by

$$A(r; R_t) = \begin{cases} e^{-2r^2/R_t^2} & 0 \leqslant r < R_t, \\ \frac{4}{e^2 R_t^2} r^2 - \frac{12}{e^2 R_t} r + \frac{9}{e^2} & R_t \leqslant r < 1.5R_t, \\ 0 & r \geqslant 1.5R_t, \end{cases} \tag{4.1}$$

where r is the distance from the atom centre and R_t is the Van der Waals radius of the atom of type t. Ligand and protein atoms have different types encoding information about the element and its chemical environment (for example "aliphatic carbon" or "aromatic carbon"), corresponding to Smina atom types (Koes, Baumgartner, et al. 2013). The density of each atom type is encoded in a different grid (or *channel*, analogous to RGB channels in images), and all n_t densities coming from atoms of the same type are summed together so that the total atomic density g at grid point $\mathbf{r}_g$ for atom type t is given by:

$$g_t(\mathbf{r}_g) = \sum_i^{n_t} A\left(\|\mathbf{r}_i - \mathbf{r}_g\|; R_t\right). \tag{4.2}$$

There is one such 3D grid for each atom type t. The 18 ligand types and 16 protein atom types supported by Gnina are reported in Tab. 2.1. Some atom

types of Tab. 2.1 are fused together in Gnina, to reduce the number of channels to 14 for ligand and protein atoms. The actual channels are reported in Tab. E.1.

The default resolution of the discretisation of atomic density on a grid is 0.5 Å and the default box size is 23.5 Å. This means that each grid contains $48 \times 48 \times 48$ values and therefore the protein-ligand binding site can be represented by a 4-dimensional tensor of size $28 \times 48 \times 48 \times 48$, where 28 is the total number of ligand and protein atom type channels. Since Gnina uses single precision floating point numbers (32 bits), a single grid requires about 13 MB.

Using atomic densities discretised on a grid for DL applications has several advantages. The main advantage is the similarity with images, with the difference that data is three-dimensional instead of two-dimensional and channels encode chemical information instead of information about colours. This similarity allows using the same DL architectures that have been very successfully applied in computer vision directly in the drug discovery domain. Additionally, the computation of the atomic density of Eq. (4.2) on a grid can be parallelised using graphics processing units (GPUs); this is essential to perform on-the-fly data augmentation—random translations and rotations of the systems—which is essential to prevent overfitting (Ragoza, Hochuli, et al. 2017).

To build the atomic density grids, only atom types and coordinates are needed. The `gninatyper` utility shipped with Gnina allows extracting such information using Open Babel (O'Boyle, Morley, et al. 2008) from common molecular file formats, and store it into a binary file with lower input/output (I/O) footprint. Such files can be subsequently squashed together into a `.molcache` (or `.molcache2`) file which allows reducing I/O overhead even further. This is particularly helpful to speed up training.

The computation of atomic densities on a grid, with the corresponding atom types, has been extracted into a standalone library called `libmolgrid` (Sunseri and Koes 2020). `libmolgrid` is compatible with Caffe, as well as more recent DL frameworks such as PyTorch (Paszke et al. 2019) and Keras/TensorFlow (Abadi et al. 2015; Chollet et al. 2015).

4.2.2 CNN Architectures

Gnina comes with pre-trained CNN architectures, but users can define and train different Caffe models. For example, Imrie, Bradley, van der Schaar, et al. (2018) trained the DenseNet architecture (Huang, Liu, Van Der Maaten, et al. 2017), and using transfer learning they developed protein family-specific SFs for structure-based VS.

In Gnina there are a series of pre-trained ensembles of models currently available: `redock_default2018`, `general_default2018`, `crossdock_default2018`, and

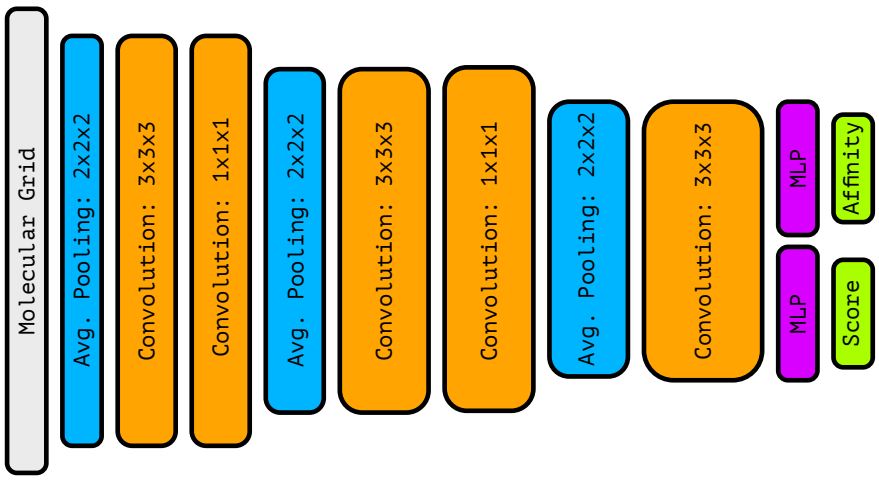


Figure 4.2: Schematic of the GNINA `default2018` architecture. Average pooling layers reduce the spatial resolution without changing the number of channels, while convolutional layers increase the number of channels (features) leaving spatial dimensions unchanged. After the last convolutional layer, the tensor representation is flattened and fed to two different MLPs, predicting the binding affinity and computing a pose score (between 0 and 1).

`dense`. Each ensemble of models—five identical model architectures trained on the same data set but with different initial weights—corresponds to different model architectures and/or has been trained on different data sets (McNutt et al. 2021).

The different CNN architectures and data sets used for training are described in Ragoza, Hochuli, et al. (2017) and Francoeur et al. (2020). The `default2018` CNN architecture consists of a series of $2 \times 2 \times 2$ average pooling layers, combined with $3 \times 3 \times 3$ and $1 \times 1 \times 1$ convolutional layers followed by two fully connected MLPs predicting either the protein-ligand binding affinity or a pose score to distinguishing between low- and high-RMSD poses. Fig. 4.2 shows a schematic of the `default2018` architecture.

The `dense` model consists of maximum pooling layers, combined with $3 \times 3 \times 3$ and $1 \times 1 \times 1$ convolutional layers and dense blocks—where a dense block corresponds to a series of four $3 \times 3 \times 3$ convolutional layers with skip connections (Huang, Liu, Van Der Maaten, et al. 2017; Imrie, Bradley, van der Schaar, et al. 2018). A schematic of the `dense` model is shown in Francoeur et al. (2020).

A single docking calculation with GNINA's default model ensemble (see section 4.4.1 for details) requires about 3 GB of GPU memory.

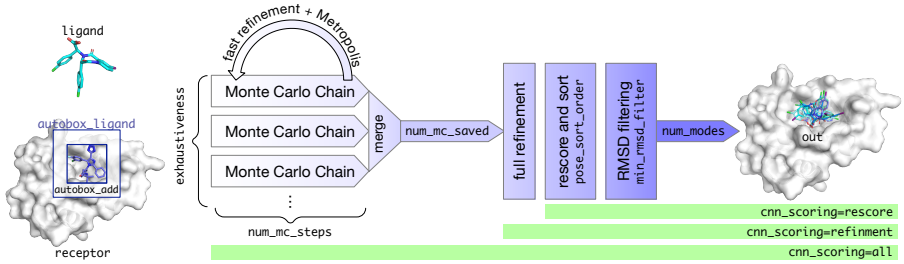


Figure 4.3: Gnina docking pipeline. Image reproduced—without changes—from McNutt et al. (2021) under the CC BY 4.0 licence.

4.2.3 Docking Pipeline

The docking pipeline of Gnina is shown in Fig. 4.3. Given a protein-ligand pair and a definition of the binding site, poses are generated by sampling the translational and torsional degrees of freedom using a MC algorithm with the Metropolis acceptance criteria. The number of MC chains is controlled with the `--exhaustiveness` command-line option, while the number of MC steps is determined heuristically based on the number of degrees of freedom (DoF). The candidate poses obtained are subsequently refined, re-scored and sorted.

MC sampling, refinement, and re-scoring can be performed with standard SFs—such as AutoDock Vina (Trott and Olson 2009) or Vinardo (Quiroga and Villarreal 2016)—as well as with CNN SFs. However, given the large number of evaluations performed during sampling, using CNN SFs at this stage is discouraged (McNutt et al. 2021).

4.3 Implementation of Flexible Docking

Docking with flexible residues was already loosely implemented in Smina (Koes, Baumgartner, et al. 2013) but has never been extensively tested.

Smina—and therefore Gnina—allows the sampling of side chains' conformational space, while keeping the backbone fixed. More specifically, side chain torsion angles are randomly set as part of the MC search, on the same footing as the ligand torsion angles. The energetic contribution of the intramolecular interactions between flexible residues—and not only the ones between receptor and ligand—are included in the final score.

4.3.1 Bugfixes

To make sure that flexible docking as implemented in Smina works correctly, a large re-docking study based on the PDBbind 2016 was performed (see

section 2.3.1 for information about PDBbind), where we re-docked all 16 126 complexes while treating side chains within 3.5 Å from the ligand as flexible. From this study, two problems emerged: proline residues were treated as flexible thus leading to an erroneous ring-opening, and residues very far away from the ligand were selected when using the `--flexdist` and `--flexdist_ligand` options (see section 4.3.3 for more details on the different options available for docking with flexible residues).

The first problem, clearly outlined in Fig. E.1, was easy to fix (see smina/MR#3, last accessed October 1, 2022): proline residues, as well as alanine and glycine residues, are now always excluded from the list of residues whose side chains are treated as flexible.

The second problem, where residues further away from the ligand than the threshold distance `--flexdist` were incorrectly treated as flexible, was present only in systems where residues identifiers contained insertion codes. PDB insertion codes are used to denote insertions in the sequence of the protein of interest with respect to a reference—usually the wild type. While deletions are represented by missing residue numbers, a new field is needed to represent insertions. The insertion code is a single character (at position 27 on the `ATOM` field in PDB files) added to differentiate residues with the same residue number. The following is an extract of the 1K22 PDB file, where insertion codes are highlighted in red:

```
ATOM    267  CG  ASP L  14      -2.280   6.992  21.697  1.00 18.10           C
ATOM    268  OD1 ASP L  14      -1.109   7.431  21.698  1.00 18.97           O
ATOM    269  OD2 ASP L  14      -2.735   6.263  22.603  1.00 19.18           O
ATOM    270  N   LYS L  14A     -5.804   6.060  21.469  1.00 20.85           N
ATOM    271  H   LYS L  14A     -5.589   5.759  20.497  1.00  0.00           H
ATOM    272  CA  LYS L  14A     -6.654   5.209  22.296  1.00 22.86           C
```

The PDB file format specification states that "the combination of residue numbering and insertion code defines the unique residue". Unfortunately, many tools do not handle insertion codes, and those who do often do not handle them carefully enough.

In Open Babel, PDB insertion codes were read from an input file and stored in the `class OBMolecule`, but they were not included in copies of such class. This is now fixed by copying all residue attributes—including insertion codes—when copying an `OBMolecule` and by fixing the `class OBResidue` copy-constructor to copy the insertion code field as well (see openbabel/PR#1998, last accessed October 1, 2022).

4.3.2 Enabling CNN Scoring of Flexible Residues

In the original implementation of Gnina (Ragoza, Hochuli, et al. 2017) ligand and receptor atoms were distinguished by the fact that ligand atoms were movable while receptor atoms were always fixed. The movable and fixed atoms were stored in different `std::vector<atom>` data structures in the `struct model` from the AutoDock Vina implementation (Trott and Olson 2009) and their corresponding atomic densities were assigned to different ligand and receptor channels according to this distinction.

With flexible docking enabled, the atoms of the receptor side chains treated as flexible are movable as well, on the same footing of ligand atoms. Therefore, the distinction between ligand and receptor based on the movability of different atoms is no longer appropriate. To correctly split the ligand and receptor atoms in the corresponding channels, we introduced new attributes in the `class CNNScorer` that allow to store ligand atom types and coordinates—the information needed to compute the atomic density grids—separately from receptor atom types and coordinates. We then implemented two functions `CNNScorer::setLigand` and `CNNScorer::setReceptor` that automatically fill such attributes by distinguishing receptor atoms (both movable and fixed) from ligand atoms (always movable).

Finally, the custom `class MolGridDataLayer` inherited from Caffe's `class BaseDataLayer` was modified to get the ligand and receptor information from the new data structures, uniforming the signature of `MolGridDataLayer::setLigand` and `MolGridDataLayer::setReceptor`.

For geometry optimisation according to the CNN SF (Ragoza, Masuda, et al. 2018), the gradient of the loss function—using the desired good pose label as reference—was computed only with respect to ligand atomic coordinates. Therefore, we modified the computation of the gradient of the loss function with respect to atomic coordinates to include the atoms of flexible side chains as well. This allows using the CNN SF to refine poses by local geometry optimisation.

All the modifications described in this section have been merged into Gnina's codebase (see gnina/gnina#73, last accessed October 1, 2022).

4.3.3 Usage of Flexible Docking in Gnina

Gnina—like Smina—allows the sampling of side chain conformational space while keeping the backbone fixed. Side chains' flexibility can be specified manually or semi-automatically in several ways:

- flexible side chains can be defined in a PDBQT file (`flex` parameter),
- they can be selected using a comma-separated list of residue identifiers

(chain, residue number, and—optionally—insertion code; `flexres` parameter),

- or they can be selected based on their distance from a given ligand (`flexdist` and `flexdist_ligand` parameters).

The argument of the `flexres` option allows to manually specify the residues to be treated as flexible, to exploit domain knowledge about the target of interest; it is simply a comma-separated list where each element is a pair of chain identifier and residue number, or a triplet of chain identifier, residue number and insertion code.

The `flexdist` parameter allows the specification of a threshold distance from the `flexdist_ligand`. If a residue has any side chain atoms that are within this distance of the specified ligand, then the entire residue side chain is marked as flexible.

Flexible side chains are selected at the very beginning of the docking procedure and the selection is not updated during sampling. When using `autobox_ligand` to automatically define the search space, flexible side chains are included in the calculation of the box bounds. The usage of CNN models for docking with flexible side chains can be selected by the user in the same way it is done for docking with a rigid receptor.

4.4 Evaluation of Default CNN Model for Flexible Docking

Before developing a SF trained explicitly for flexible docking (see Chapter 5), we want to evaluate the performance of the default model—described below—, which has not been explicitly trained for docking with flexible residues but is readily available in Gnina.

4.4.1 Gnina's Default CNN Model

Over the years, different DL architectures for Gnina have been trained on different data sets (Francoeur et al. 2020; Ragoza, Hochuli, et al. 2017). The default model ensemble of McNutt et al. (2021) was obtained by greedily selecting pre-trained models, to maximise performance and minimise inference time/computational costs.

The default model ensemble consists of five CNN architectures—two of which are `dense` architectures—and outperforms any single model in ranking

low-RMSD poses (McNutt et al. 2021).[1] The default model ensemble also outperforms all ensembles of five models sharing the same architecture and the same training set (McNutt et al. 2021). The performance of this ensemble of five models is essentially equivalent to an ensemble of all the available pre-trained models—a total of 25 models—, at a fraction of the computational cost (McNutt et al. 2021).

4.4.2 Comparison with Rigid Docking

The data concerning docking with a rigid receptor—used here for comparative analysis—has been produced by Andrew McNutt and Paul Francoeur as part of our collaboration for McNutt et al. (2021). The data is available at dkoes/GNINA-1.0 (last accessed October 1, 2022).

We now test the performance of Gnina for docking with flexible side chains. Since receptor flexibility is only useful in the context of cross-docking, we limit our tests to the cross-docking data set (see 2.3.5 for details). Allowing side-chain re-arrangements would only deteriorate the performance in re-docking, given that the receptor is already in the correct *holo* conformation.

Flexible docking is computationally more expensive than docking with a rigid receptor because of the larger number of DoF to be sampled (ligand DoF and flexible side chains' DoF). For this reason, we use the default parameters carefully selected in McNutt et al. (2021), which maximise performance while minimising computational cost. To define the side chains to be treated as flexible, the cognate ligand used to define the binding site (with the `autobox_ligand` option) is also used as `flexdist_ligand` and `flexdist` is set to 3.5Å, which gives a reasonable representation of the protein-ligand binding site (from visual inspection). Therefore, conformations for all side chains with at least one atom within 3.5Å from the cognate ligand are sampled during docking of the non-cognate ligand.

Out of the 7970 systems in the cross-docking data set, flexible side chains are not identified for 24 protein-ligand complexes (see Tab. E.2 for details). Such systems are discarded from the following analysis of flexible docking since they are equivalent to rigid docking. Additionally, nine systems were problematic when computing RMSDs for the flexible residues because of connectivity issues. Disulfide bonds between cysteine residues are allowed to break during sampling—with a software warning—, which results in different connectivity

[1] There is a small error in McNutt et al. (2021), which lists the following models for the default ensemble: `dense`, `general_default2018_3`, `dense_3`, `crossdock_default2018`, `redock_default2018`. The actual ensemble of models used in Gnina consists of `dense`, `general_default2018_3`, `dense_3`, `crossdock_default2018`, `redock_default2018_2`. This discrepancy was discovered when implementing `gnina-torch` (see Chapter 5), and shows the value of FOSS, where all implementation details are available.

in the output file for eight systems. In the remaining system, the connectivity between flexible residues in the input and output file was found to be different, due to failures in Open Babel's automatic bond perception. All nine systems with connectivity issues were also removed from the analysis, resulting in a data set of 7937 cross-docked complexes.

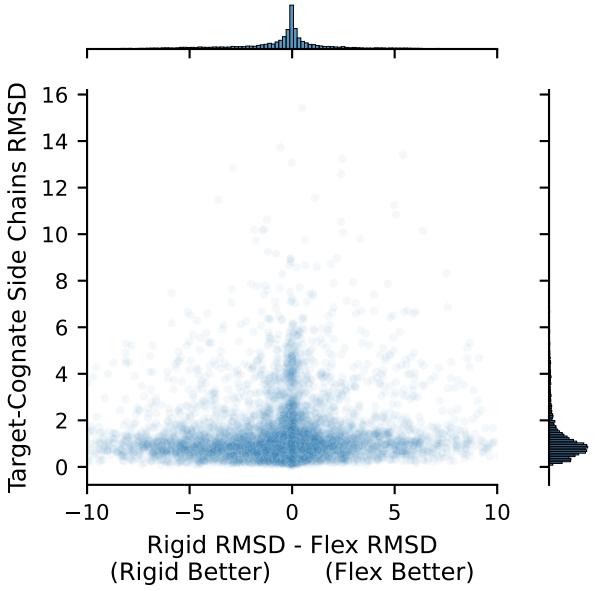
To assess the performance of flexible docking compared to rigid docking, we look at RMSD differences between ligands docked with both methods in relation to the similarity between binding pockets of the cognate and target receptors. The similarity of the binding pockets is assessed via side chains RMSD between the target (non-cognate receptor of the ligand being docked) and the cognate receptor, which we denote target-cognate RMSD. This is distinct from either the side chain RMSD between docking input and output (target-pose RMSD) and the side chain RMSD between cognate receptor and docking output (cognate-pose RMSD). Fig. 5.2 shows schematically the different RMSD types. The target-cognate RMSD is computed by finding the best match between residues in the target and cognate receptors that are within 3.5 Å from the ligand being docked using ProDy (Bakan, Meireles, et al. 2011; Zhang et al. 2021).

Fig. 4.4a shows the difference in RMSDs for the ligand top pose between flexible and rigid docking versus the target-cognate RMSD. For higher target-cognate RMSDs, indicating differences in the binding pocket between the target and cognate receptor, one would expect flexible docking to perform better. However, the difference in ligand RMSD between flexible and rigid docking for the top pose varies widely between systems and there is no clear advantage for flexible docking. The overall RMSD distributions for the ligand top poses (see Fig. E.2) are fairly similar, with slightly more systems with low RMSD for rigid docking than flexible docking.

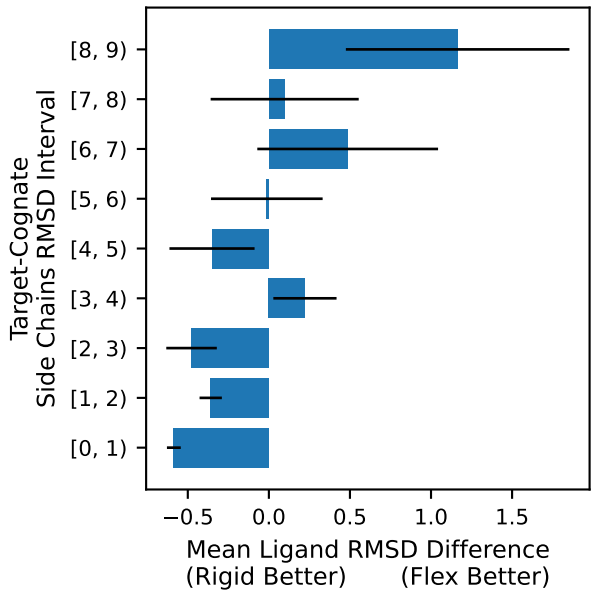
From visual inspection, the very high target-cognate RMSDs of Fig. 4.4a are due to the presence of very flexible protein loops within or close to the binding site, that assume very different conformations in the different crystal structures. In the cognate receptor the loop interacts with the ligand, while in the non-cognate receptor the loop assumes a completely different conformation, leading to very high RMSD values. Such high RMSD values indicate that the binding site is indeed very different, because of the presence/absence of the loop.[2]

Fig. 4.4b shows the average RMSD difference between rigid and flexible docking for different 1 Å intervals of target-cognate side chains' RMSD. For low target-cognate RMSDs—which correspond to a highly similar binding site and therefore a situation akin to re-docking—rigid docking seems to be advantageous

[2]The highest target-cognate RMSD of 15.42 Å is obtained between the binding site residues of protein 4DEI and protein 4IWD.



(a)



(b)

Figure 4.4: Comparison between rigid and flexible docking with the default GNINA parameters on the cross-docking data set. (a) Ligand RMSD differences between rigid and flexible docking versus target-cognate side chain RMSD. (b) Average ligand RMSD difference for different 1 Å intervals of target-cognate side chains RMSD.

on average, as expected. On the other hand, for higher target-cognate side chains RMSDs, indicating a lower similarity between binding sites, the situation is less clear. Docking with flexible residues seems to be equivalent or slightly more advantageous, on average, especially at higher target-cognate side chains RMSDs. However, as it can be inferred from Fig. 4.4a, the number of systems with target-cognate side chains RMSD higher than 6 Å is low and therefore the apparent improvement in ligand RMSD for flexible docking is inconclusive (resulting in large standard deviations).

When performing flexible docking, the side chains identified as flexible are included in the calculation of the bounds for the box defining the search space. This results in a larger search space for flexible docking compared to rigid docking, which in turn might require higher exhaustiveness for better sampling (although, by default, the number of Monte Carlo steps is already proportional to the number of degrees of freedom, hence the increased computational cost of flexible docking). It is also worth stressing that we used the default CNN model, which was not explicitly trained for flexible docking.

Given the much higher computational cost of flexible docking, the optimisation of Gnina default parameters and training of new CNN models for this specific task will be addressed in future versions of Gnina and are the subject of Chapter 5. However, it is clear that improvements for flexible docking are system-dependent and, therefore, accounting for the increased computational cost, it is reasonable to use rigid docking as the default docking method.

In terms of speed, runtimes of docking to a rigid receptor are usually of the order of seconds while runtimes of docking with a flexible receptor are of the order of minutes. The actual runtime is highly system-dependent since it depends on the number of degrees of freedom.

4.4.3 Comparison with Induced Fit Docking

In the previous section, we compared the results of docking to a rigid receptor and docking to a receptor where side chains' conformations are sampled as well on a very large cross-docking data set. On such a large data set, we see no clear difference between the two methodologies, with rigid docking being slightly better, on average, when the difference between the binding sites of the cognate and target receptors—measured in terms of target-cognate RMSD of matching residues—is low.

To apply rigid docking on systems where allowing some receptor flexibility has been deemed essential, we compare the performance of Gnina's default model—which has not been trained explicitly for flexible docking—with the

results obtained using the IFD method of Sherman et al. (2005).[3] To perform this comparison, we downloaded and aligned most of the systems used in the original study. Two systems—targets 1DBA and 1DM2—were discarded because they did not contain a co-crystallised ligand, which is used in our protocol to automatically define the binding site and automatically select the residues to be treated as flexible.

Alignment of the different structures is performed with ProDy (Bakan, Meireles, et al. 2011; Zhang et al. 2021), as already described for the target-cognate RMSD calculation. The alignment is then used to superimpose the structures to minimise the C_α RMSD. All structures were manually inspected to ensure the correctness of the ligand selection and the alignment, and they were visually compared with the structures of Sherman et al. (2005).

For rigid docking, the binding site was defined according to the cognate ligand of the receptor under investigation. The same cognate ligand was used to automatically select the residues to be treated as flexible within 3.5 Å from the ligand. This is the same value we used in the previous study, but it is lower than the threshold of 5 Å used in the IFD protocol (Sherman et al. 2005). Our value of 3.5 Å was chosen by visual inspection and provides a good balance between the representation of the binding site and computational speed; in contrast to Sherman et al. (2005) where side chains are relaxed, our method requires the sampling of the conformational space and using a higher distance would significantly increase the number of DoF.

Fig. 4.5 shows the ligand RMSD obtained with docking with flexible side chains using Gnina, compared with the original result of the IFD protocol (Sherman et al. 2005). For 11 systems, Gnina obtains low-RMSD poses close to the ones obtained with the IFD protocol. For the remaining 8 systems, docking with flexible side chains using Gnina fails to produce low-RMSD poses. For three systems, the IFD protocol fails as well, but for the remaining five the IFD protocol considerably outperforms Gnina.

For the eleven systems where Gnina finds a low-RMSD pose, the results of docking with flexible side chains are comparable with the results obtained with the more sophisticated IFD protocol, which takes into account backbone relaxation as well.

Rigid docking with Gnina only achieves a ligand RMSD smaller than 2 Å for six out of 19 systems (see Fig. E.3 for a comparison between rigid docking and docking with flexible residues using Gnina). Docking with flexible residues, however, remains inferior to the IFD protocol, for which 16 ligands were correctly docked (compared to four for their baseline).

[3] The IFD method has been developed by Schrödinger Inc. and we don't have access to their platform for a comparison of speed. The IFD results are taken directly from the original reference.

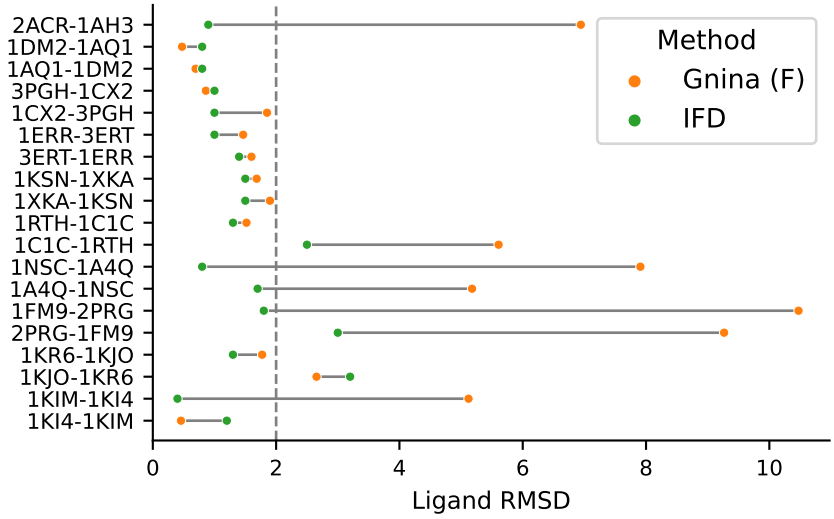


Figure 4.5: Comparison of ligand RMSD for poses obtained with Gnina (docking with flexible side chains) and with the IFD protocol. For 11 system, Gnina obtains low-RMSD poses close to the ones obtained with the IFD protocol. However, for 5 systems where the IFD protocol is successful, Gnina fails to find a low-RMSD pose and the poses found have significantly larger RMSD.

For some systems, it might be impossible to find good binding modes without backbone flexibility. For example, for the system 1KIM-1KI4 the residues from 1KIM identified to clash with the non-cognate ligand from 1KI4 are Y132 and A168. Since the side chain of alanine residues consists only of a methyl group, it is not affected by docking with flexible side chains. Therefore, a protocol allowing for backbone relaxation has a clear advantage in this scenario. Indeed, the IFD protocol produces a very good binding pose for this system (RMSD of 0.4 Å), while Gnina fails to find a good pose.

Interestingly, comparing the Glide baseline (Friesner, Banks, et al. 2004; Friesner, Murphy, et al. 2006; Halgren et al. 2004) with rigid docking using Gnina (see Fig. E.4), it appears that Gnina provides a much stronger baseline. Not only Gnina provides lower RMSD poses as top pose, but in five additional cases the poses generated by Gnina are sub-2 Å RMSD poses. This indicates that for these systems, a low RMSD pose can be obtained without considering the receptor flexibility.[4] This is in addition to the 4 poses for which this was already the case in Sherman et al. (2005). For two additional systems, Gnina gets very close to the 2 Å RMSD threshold, significantly improving over the

[4]It is not clear how the RMSD was computed in Sherman et al. (2005). In our study, we applied symmetry corrections based on graph isomorphisms (Meli and Biggin 2020). Without symmetry correction, the RMSD of some poses might have been artificially inflated (see Appendix B).

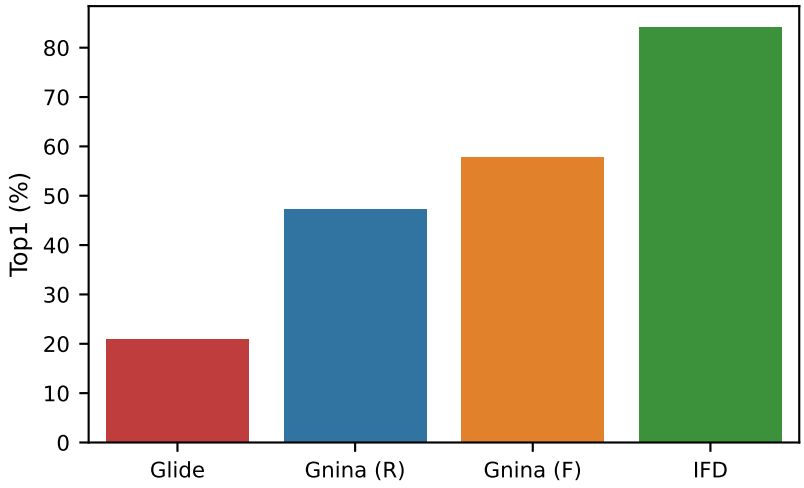


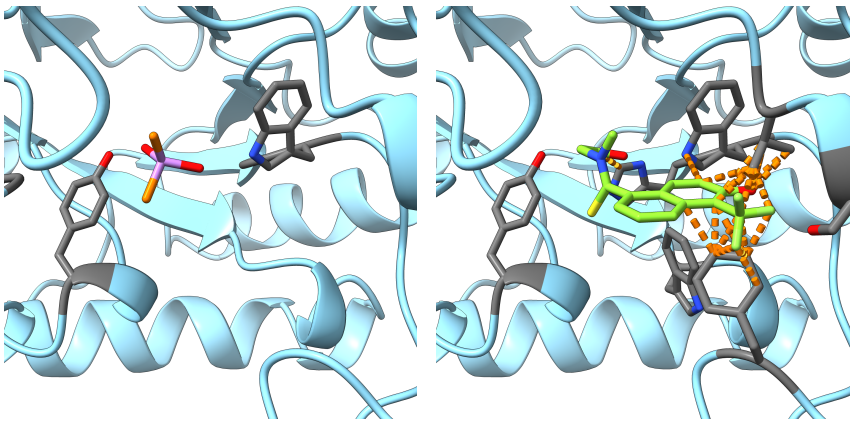
Figure 4.6: Top1 metric—the percentage of targets with a good pose (RMSD smaller than 2 Å) ranked as top—for different docking methods applied to the IFD data set. Flexible residues in Gnina are automatically selected.

poses obtained with Glide. The higher success of the IFD protocol in such cases might therefore be attributed to increased and different sampling, instead of the receptor flexibility itself.

Fig. 4.6 compares the performance of the different methods—Glide, Gnina, Gnina docking with flexible residues, and IFD—on the IFD set. The Top1 metric—the percentage of targets with a good pose (RMSD smaller than 2 Å) ranked as top—is reported. We can see that Gnina is significantly better than Glide, but Gnina using docking with flexible residues is not quite as good as the IFD protocol when residues treated as flexible are automatically selected as discussed above.

4.4.4 Docking with Selected Flexible Residues

In the previous section, we automatically selected the flexible residues based on the distance of said residues from a given ligand. The ligand used for this selection (as well as to define the search space) is the cognate ligand for the receptor under investigation. This choice might however be suboptimal because non-cognate ligands might be very different from cognate ligands (see for example Fig. 4.7). Additionally, since the search space needs to contain the residues selected as flexible, such automated selection might result in a search space that is too large and therefore decreases the chances of finding a good



(a) (b)

Figure 4.7: Receptor 2ACR (aldose reductase) shown with its cognate ligand (dimethylarsinic acid, $(CH_3)_2As(O)OH$) and the non-cognate ligand (tolrestat) from 1AH3. (a) Receptor side chains (grey) within 3.5 Å from the cognate ligand (orange), and (b) receptor side chains (grey) within 3.5 Å from the non-cognate ligand (green). The automatic selection based on the cognate ligand fails to identify the residues actually clashing with the non-cognate ligand (clashes identified by ChimeraX, using default parameters, are outlined in orange). This is an extreme case of the limitation of the automatic selection of residues to be treated as flexible due to the difference in ligand sizes.

binding mode (RMSD smaller than 2 Å).

Sherman et al. (2005) provide a list of residues that clash with the non-cognate ligand, reported in Tab. E.3. This list is of course available only because we have the binding pose of the non-cognate ligand—obtained by aligning the receptor under investigation and the one associated with the non-cognate ligand. However, experienced computational chemists and medicinal chemists with a good knowledge of the target of interest should in principle be able to choose reasonable residues to be treated as flexible.

Fig. 4.8 shows the results of docking with selected flexible residues compared to the results of the IFD protocol. Comparing these results with the ones presented in Fig. 4.5 it is clear that manually selecting the residues producing clashes with the non-cognate ligand substantially increases the performance of our method. A good binding pose is found for 5 additional systems compared to the automated flexible side chain selection method employed in section 4.4.3.

The IFD protocol finds slightly lower RMSDs for most systems, but the results obtained with Gnina are very much comparable. Of the 4 systems for which Gnina fails to find a low RMSD pose, 3 are the ones where the IFD protocol fails as well (although for one system the top pose found by Gnina is considerably worse). More importantly, as shown in Fig. E.5, docking with selected flexible

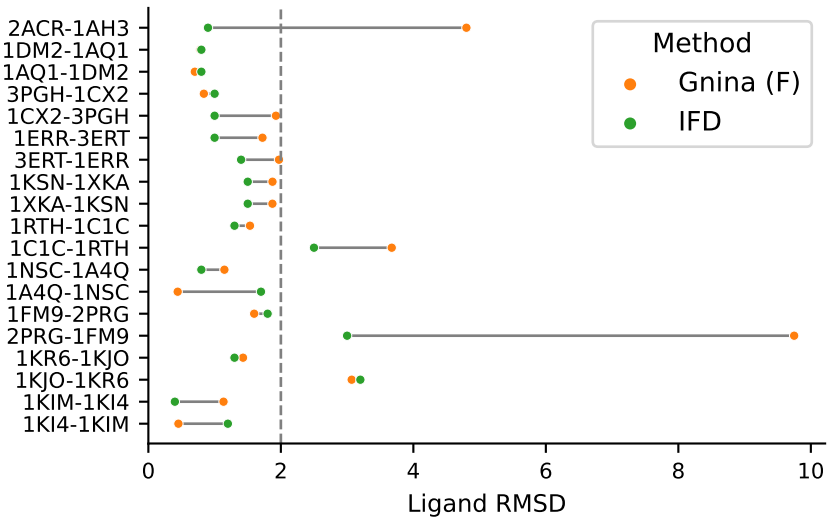


Figure 4.8: Comparison of ligand RMSD for poses obtained with Gnina (with flexible side chains, manual selection from Tab. E.3) and with the IFD protocol. For 15 systems, Gnina obtains low-RMSD poses close to the ones obtained with the IFD protocol. Of the remaining four systems, three are systems where the IFD protocol fails as well to find a low RMSD pose.

residues does not significantly deteriorate the performance of rigid docking.

Fig. 4.9 compares the performance of the different methods—Glide, Gnina, Gnina docking with flexible residues, and IFD—on the IFD set. The Top1 metric—the percentage of targets with a good pose (RMSD smaller than 2 Å) ranked as top—is reported. We can see that the results of Gnina using docking with flexible residues are comparable to the IFD protocol when residues treated as flexible are carefully selected (compare with Fig. 4.6).

4.5 Discussion and Conclusions

In this chapter, we described the implementation and evaluation of docking with flexible residues in Gnina (McNutt et al. 2021). Thanks to a large-scale re-docking study we identified several problems that have been fixed in both Gnina and Smina, making them usable for docking studies with flexible residues.

Once the classical docking engine for docking with flexible residues was fixed and validated, we modified Gnina to allow the use of the integrated CNN SF when docking with flexible residues. This required several changes in the way Gnina deals with movable atoms internally.

After building the capabilities to perform docking with flexible residues

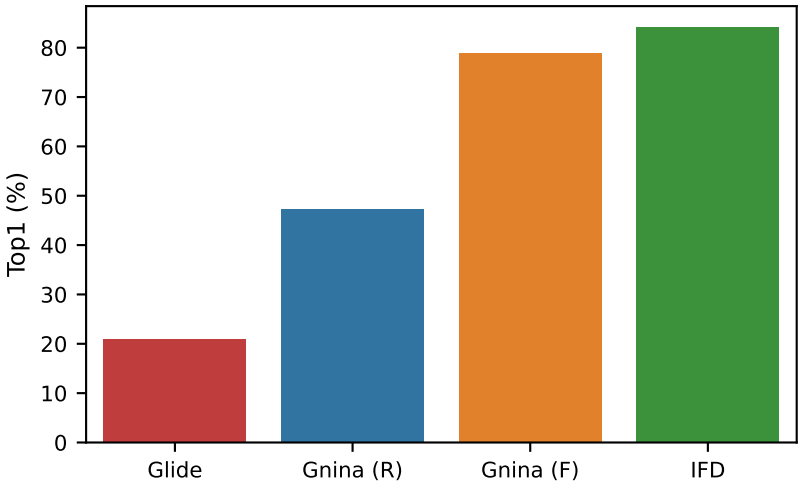


Figure 4.9: Top1 metric—the percentage of targets with a good pose (RMSD smaller than 2 Å) ranked as top—for different docking methods applied to the IFD data set. Flexible residues in Gnina are manually selected.

using Gnina's CNN SF we performed a large-scale comparison of cross-docking with a rigid receptor versus receptor with flexible residues. Given the size of the data set—7937 protein-ligand complexes—we used automated detection of the flexible residues based on the cognate ligand and a distance criteria. Our results show that for very similar pockets—measured using the RMSD between matching flexible residues—rigid docking has a slight advantage, as expected. For more dissimilar pockets the situation is less clear and docking with flexible residues seems to perform slightly better on average but the number of structures with highly dissimilar pockets is limited. Based on this study and the significantly increased computational cost of docking with flexible residues, the default behaviour of Gnina 1.0 was set to rigid docking and docking with flexible side chains needs to be explicitly and consciously enabled by the user.

Finally, we performed a comparison between docking with flexible residues using Gnina and the IFD protocol developed by Sherman et al. (2005), for which a reasonably sized and well-described data set was introduced. We show that docking with flexible residues in Gnina works better than rigid docking on this data set. In the cases where docking with flexible residues fails to find a good top one pose, it does not deteriorate the performance of rigid docking either. Compared with the IFD protocol, Gnina combined with automatic selection of the flexible residues to be treated as flexible performs worse. This is mostly due to the automated selection, which might not select all the relevant residues to

be treated as flexible and selecting more residues than necessary complicates the MC search (with a bigger search space and more DoF). When selecting the specific residues clashing with the non-cognate ligand —something and experience user might be able to do by applying domain knowledge—, the performance of GNINA approaches the one of the IFD protocol. Additionally, selecting specific residues to be treated as flexible reduces computational time.

In future iterations of the software, it would be interesting to investigate and include better schemes for the automatic selection of residues to be treated as flexible. While we don't expect experimental temperature factors to be a good choice—different rotamers in different structures might have low experimental temperature factors if they do not interconvert—, it would be interesting to explore and implement simple heuristic rules for this selection (Anderson et al. 2001). Additionally, only sampling relevant rotamers for the flexible side chains from rotamer libraries could significantly MC speed up sampling (Dunbrack Jr 2002).

In conclusion, we implemented and validated docking with flexible residues in GNINA using the default CNN SF. The method works well in cross-docking scenarios when there are significant differences between the binding site of the cognate and non-cognate receptors and especially when combined with expert knowledge about the system to select relevant residues to be treated as flexible.

5

A Deep Learning Scoring Function for Flexible Docking

Contents

All code associated to this chapter is available in the following GitHub repositories: RMeli/gsoc19, RMeli/gnina-torch, gnina/gnina (some contributions), and gnina/scripts (some contributions).

In Chapter 4 we worked on the implementation of docking with flexible side chains in Gnina. We fixed several bugs with the implementation inherited from Smina, and we refactored the CNN SF to work flawlessly and correctly with flexible side chains. With these new capabilities at hand, we first performed a large-scale study of cross-docking with flexible residues, from which we concluded that the benefits of taking into account (local) receptor flexibility are very much system-dependent and therefore rigid docking remains a good default for large-scale docking studies, given the significantly higher computational

costs of docking with flexible residues. However, we also focussed on a smaller data set where taking into account receptor flexibility has been shown to be essential to obtain good docking poses. Our study on this smaller data set outlined the importance of carefully selecting the side chains to be treated as flexible, in contrast with the automatic selection used for the large-scale study. However, we also demonstrated that docking with flexible side chains using Gnina can lead to very competitive results, compared to more involved protocols such as the IFD protocol of Sherman et al. (2005).

Here we want to investigate if re-training the CNN-SF specifically for docking with flexible residues can improve the results obtained in the previous chapter, where the default model—an ensemble of models greedily selected to optimise performance and computational cost—has been used instead.

5.1 Flexible Docking Data Set

To develop a new CNN SF for flexible docking we need a suitable data set on which the CNN can be trained. The ten poses per complex obtained by docking the downsampled cross-docking data set with Gnina for the evaluation of Gnina's default CNN model (Chapter 4) might not be enough to train a robust SF. Therefore, we performed docking with flexible residues using the AutoDock Vina SF, producing 20 poses per complex instead. For convenience, we used Gnina without CNN scoring instead of Smina to perform the docking, since the two software have equivalent performance (McNutt et al. 2021). To obtain more low-RMSD poses for training, we also performed a local optimisation of the crystal pose according to the AutoDock Vina SF—including the flexible residues. Evaluating the docking results show that eleven systems appear to be problematic (see Tab. F.1). Again, we have four systems with different connectivity because of broken disulfide bonds. The remaining problematic systems mostly overlap with the ones already identified in Chapter 4 (compare Tab. E.2 and Tab. F.1), while the remaining differences can be attributed to the different number of poses (20 instead of ten).

The systems with a disulfide bond or a spurious bond between residues in the crystal structure (see Tab. F.1) are removed from the data set since such bonds often break during the minimisation of the crystal pose (and restraints on inter-residue bonds are currently not supported). For the remainder of the problematic systems (reported in Tab. F.1), a spurious bond is added to some poses, due to Open Babel automatic bond perception. For such systems, we only removed the few poses where the spurious bond appears. Additionally, all systems for which no flexible residues were identified have been removed. All

the retained systems and poses form the basis of our training and test sets.

5.1.1 Receptor Reconstruction

When performing docking with a rigid receptor, the same rigid structure is associated with all docking poses obtained. When docking with flexible residues, however, a different receptor structure—or receptor pose—is associated with each docked ligand pose. The side chains of the receptor assume a different conformation that is correlated with the binding pose of the ligand.

While only side chains within the binding site are treated as flexible, the grid used for discretisation of the protein-ligand binding site can also contain additional side chains not treated as flexible, as well as backbone atoms. Given that GNINA only outputs the conformation of the flexible side chains—to minimise I/O—, and to avoid bias introduced by missing rigid side chains and backbone atoms, we need to reconstruct the receptor structure as a whole. The `makeflex.py` script—provided with GNINA—allows reconstructing the whole receptor corresponding to a given binding pose, given the original receptor structure and the PDB file containing the new pose of the side chains treated as flexible.

Therefore, to build the training set for a CNN SF geared toward docking with flexible side chains, docking with flexible side chains is performed first using the AutoDock Vina SF and, subsequently, the poses of the flexible residues are inserted into the original PDB files. This considerably increases the disk memory requirements for the flexible docking data set compared to rigid docking, since the protein PDB file is duplicated as many times as the number of ligand poses.[1]

The `makeflex.py` script, which is central to building the training set for our CNN SF for docking with flexible residues also suffered from the same PDB insertion code-related problems described in Chapter 4, and was therefore amended to correctly reconstruct the whole receptor—with the new poses for the flexible side chains—when insertion codes are present (see gnina/gnina#65, last accessed October 1, 2022). Other fixes to this script, necessary to perform the receptor reconstruction correctly, are shortly listed in Appendix A.3.

[1]Since GNINA only outputs the pose of the side chains treated as flexible, an improvement would be to use those as input alongside the original receptor and do the reconstruction of the whole protein within GNINA. This approach, however, requires several changes to both GNINA and the associated training/evaluation pipelines.

5.2 CNN Model for Flexible Docking

5.2.1 Training and Test Datasets Annotation

The CNN SF is trained to distinguish low- from high-RMSD binding poses given a protein-ligand complex. Therefore, it is a binary classifier for which two suitable classes have to be defined. In docking studies, a binding pose with RMSD lower than 2 Å from the true (known) pose is considered a success (Bursulaya et al. 2003). Therefore, poses with RMSD lower than 2 Å are usually assigned to the positive class (label 1) while poses with RMSD higher than 2 Å are assigned to the negative class (label 0). This class definition is used by Francoeur et al. (2020), while the original development of CNN SFs used to leave a gap between the positive and negative classes by discarding all poses with a RMSD in the interval $[2, 4]$Å (Ragoza, Hochuli, et al. 2017).

For flexible docking, the receptor is no longer rigid since side chains' conformations are sampled as well. Therefore, it is possible to define a global RMSD of the receptor (RMSD of all side chains)—denoted $\text{RMSD}_{\text{flex}}$—as well as a RMSD for each individual side chain and the corresponding maximum—denoted $\text{RMSD}_{\text{flexMAX}}$. Fig. 5.1 shows the RMSD distribution of the data set described in section 5.1 for different ranks as well as the minimised crystal pose. RMSDs for the ligand as well as for the flexible side chains were computed using graph isomorphism-based symmetry corrections as implemented in Open Babel's `obrms` (O'Boyle, Morley, et al. 2008) or `spyrmsd` (Meli and Biggin 2020).[2]

We can see from Fig. 5.1 that defining a good pose as a pose with RMSD lower than 2 Å is a sensible choice since most minimised crystal poses have a RMSD lower than this threshold and the distribution of top-ranked poses also has a (local) maximum below said threshold.

Since we want to develop a CNN SF suitable for docking with flexible residues, we can use both the ligand RMSD and the RMSD of the flexible side chains for annotation. The RMSD of interest for annotation is the cognate-pose RMSD, since we consider the position of the shared side chains in the cognate receptor as the correct conformation. Obviously, this is an approximation, and cross-docking annotations are inherently more noisy than the ones employed in re-docking, where the experimentally determined pose is known, and the receptor is considered fixed.

For the total RMSD of flexible residues—$\text{RMSD}_{\text{flex}}$—, a threshold of 1 Å captures the minimised crystal pose peak but cuts the peaks for ranks 1-3

[2] `obrms` is used to compute the RMSD of the ligand, while `spyrmsd` is used to compute both the RMSD of the ligand and of the flexible residues. The comparison of ligand RMSD between `obrms` and `spyrmsd` served as further validation of `spyrmsd`, in addition to the validation presented in Appendix B.

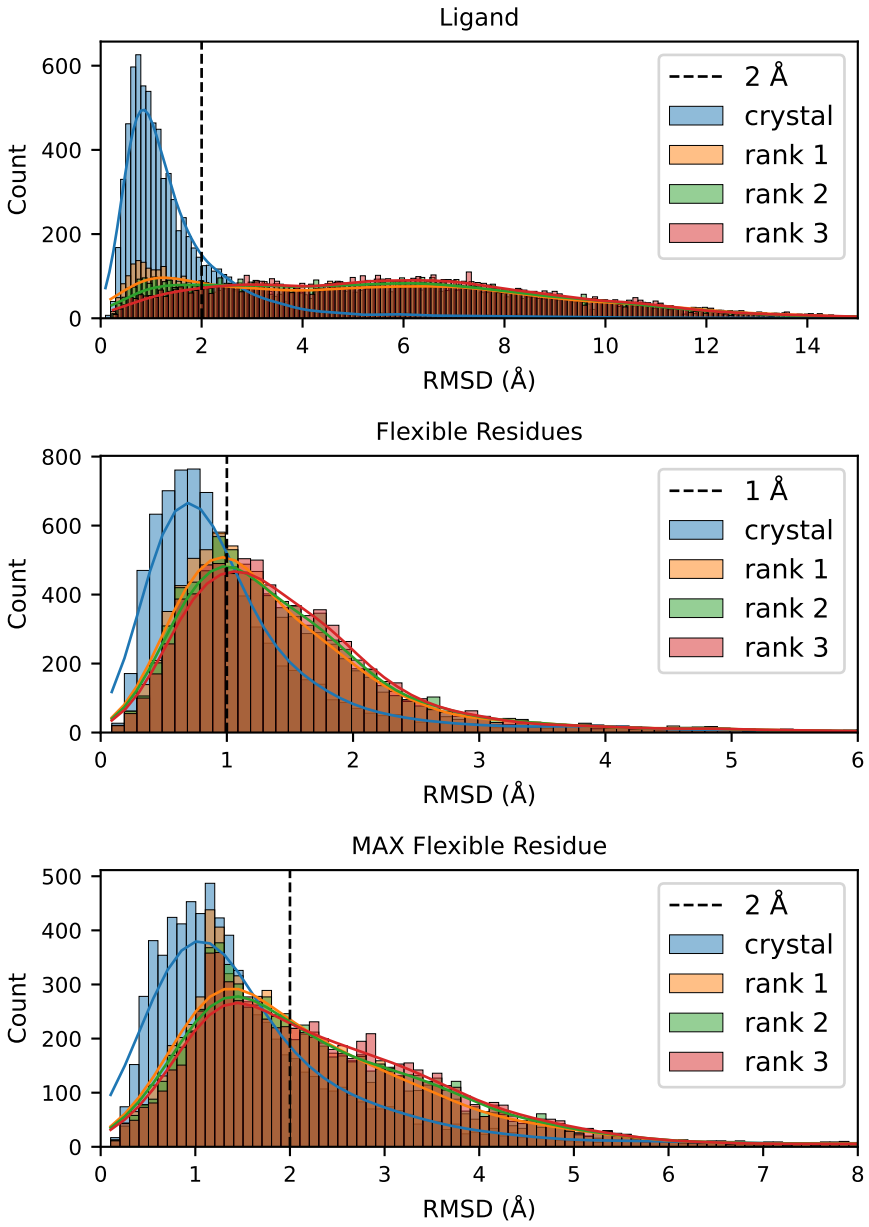


Figure 5.1: RMSD distributions for the ligand and flexible residues poses with respect to the crystal structure for ranks 1-3 and for the minimised crystal pose. Flexible residues' RMSD is computed for all residues together or separately (in which case the maximum RMSD is retained). Poses are generated with GNINA with the AutoDock Vina SF. Residues within 3.5 Å from the ligand are treated as flexible.

right in the middle. A threshold of 2 Å is however able to capture wholly the distribution of minimised crystal poses and most of the peaks corresponding to ranks 1-3. For the maximum RMSD amongst all side chains treated as flexible—$\text{RMSD}_{\text{flexMAX}}$—, a threshold of 2 Å captures both the peaks of poses 1-3 and the peak of the minimised crystal pose. However, it is worth stressing that in contrast with the ligand pose, where the distribution of the first rank is bimodal—providing a rather natural threshold—, the RMSD distributions for the flexible residues are unimodal, and therefore it is more difficult to define a sensible threshold (if any).

Therefore, we try different receptor-based thresholds—in addition to the usual ligand-based threshold—for which a pose is labelled as positive (low-RMSD):

- $\text{RMSD}_{\text{lig}} < 2$ Å (no receptor-based annotation),
- $\text{RMSD}_{\text{lig}} < 2$ Å and $\text{RMSD}_{\text{flex}} < 1$ Å (denoted `flex1`),
- $\text{RMSD}_{\text{lig}} < 2$ Å and $\text{RMSD}_{\text{flex}} < 2$ Å (denoted `flex2`),
- $\text{RMSD}_{\text{lig}} < 2$ Å and $\text{RMSD}_{\text{flexMAX}} < 2$ Å (denoted `max2`).

As mentioned previously, in a cross-docking scenario different types of RMSDs that can be computed: cognate-target, cognate-pose and target-pose. As for the ligand RMSD, where the reference configuration is the one in complex with the aligned cognate receptor, we are interested here in the cognate-pose RMSD between flexible residues. Fig. 5.2 shows a schematic of the different possible RMSD calculations between receptor residues in the cross-docking scenario.

To compute the cognate-pose RMSD, we use the same approach described in section 4.4.2, where an alignment of the cognate and target protein sequences is performed with ProDy (Bakan, Dutta, et al. 2014; Bakan, Meireles, et al. 2011; Zhang et al. 2021). Here, however, only the matching side chains that are treated as flexible are used to compute the cognate-pose RMSD. Additionally, from initial tests, it was recognised the importance of symmetry corrections when side chain conformations are sampled. To compute symmetry-corrected RMSD with `spyrmsd`, bonds are automatically inferred using ProDy—which uses a simple distance criteria, and it is therefore imperfect. For the rare occasions where the inferred connectivity differs between the cognate receptor and the pose receptor (8220 poses out of 166 370, or about 5%), the standard RMSD—without symmetry corrections—is computed instead.

Systems for which the RMSD could not be computed between the pose obtained with GNINA and the cognate receptor are reported in Tab. F.1 and are discarded from the data set.

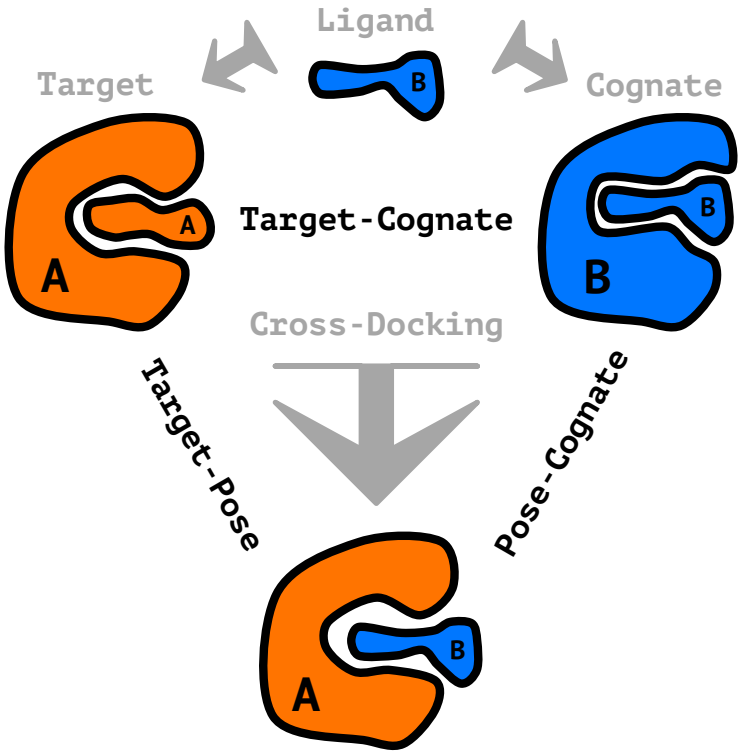


Figure 5.2: Schematic of cross-docking and the involved configurations. When ligand B is docked into receptor A, the "correct" RMSD for the ligand is between the pose docked to receptor A and the (crystal) pose in receptor B (aligned to receptor A). For receptor A aligned to receptor B we have three possible RMSDs for the flexible residues: target-cognate RMSD—which measures the similarity of the binding site—, target-pose RMSD—which captures the changes in the binding site during docking—, and pose-cognate RMSD—which is the RMSD used to evaluate cross-docking, as for the ligand.

The different annotations of the data set described above result in a heavily unbalanced data set, as shown in Fig. F.1. This imbalance is caused by the low success rate of cross-docking and highlights the importance of including minimised crystal poses in the training data to inflate the number of poses in the positive (low-RMSD) class.

Imbalanced data sets are common in ML applications, especially so in drug discovery applications, where data is scarce and good data is difficult and costly to obtain. There are several pre-processing methods to deal with data set imbalance (Garcia et al. 2012) such as undersampling the majority class (Anand et al. 2010; Kumar, Rao, et al. 2014) or oversampling the minority class (Cao et al. 2013, 2011; Nekooeimehr and Lai-Yuen 2016). More recently, a simple but effective post-processing method based on thresholding has been developed and shown to be very effective for the classification of active and inactive compounds (Esposito et al. 2021).

Here we use the minority class oversampling method as implemented in `libmolgrid` (Sunseri and Koes 2020), which has proven to be successful in training CNN SFs (Francoeur et al. 2020; Ragoza, Hochuli, et al. 2017). However, it would be interesting to explore the method suggested by Esposito et al. (2021) in the context of CNN-based SFs in future work.

5.2.2 Cross-Validation Splits

In the cross-docking benchmark, described in section 2.3.5, proteins are clustered by pocket. Therefore, to create cross-validation splits containing different pockets, we only need to compute the similarity between said pockets. Following Francoeur et al. (2020), we use ProBiS (Konc, Depolli, et al. 2012; Konc and Janežič 2010) to compute the similarity between binding pockets. A short description of ProBiS is provided in Appendix F.1.

A representative of each pocket is selected by taking the cognate receptor corresponding to the ligand with the largest molecular mass. Following Francoeur et al. (2020), residues within 7 Å from the ligands are selected to perform the binding pocket alignments and a z-score of 3.5 is used as the threshold to define similar and dissimilar proteins. The computed z-scores for all alignments are shown in Fig. F.2.

With this threshold, some pockets are considered both similar and dissimilar depending on how the alignment has been performed. Aligning A to B can give slightly different results than aligning B to A. The pocket codes and z-scores for such pockets are reported in Tab. F.2 and given that the z-score threshold has been selected reasonably high, pockets with at least one alignment passing the threshold are considered similar.

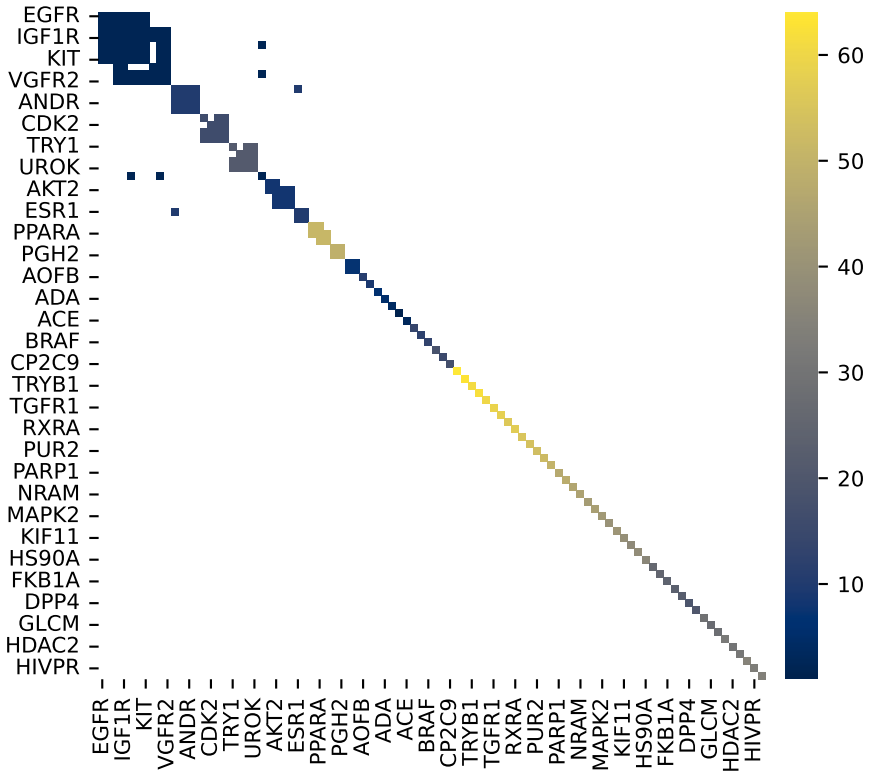


Figure 5.3: Pairwise similarity between pockets, colour-coded by cluster. White indicates that there is no similarity between pockets. There are a total of 62 clusters (for the 92 pockets).

With the pairwise similarity between pockets, a graph whose edges represent the similarity between pockets is constructed. Clusters are obtained by finding all the connected components of such graph using NetworkX (Hagberg et al. 2008). The results of pocket clustering are shown in Fig. 5.3, where each of the 62 clusters corresponds to a different colour.

Cross-validation splits are created with `scikit-learn`'s `GroupKFold`, so that similar pockets—pockets in the same cluster—are assigned to the same cross-validation fold.

5.2.3 Training and Data Augmentation

The CNN models are trained using stochastic gradient descent (SGD) with a momentum of 0.9, and a weight decay of 0.001. The initial learning rate is set

to 0.01, which is dynamically scaled in response to the training process. The models are trained for 500 epochs with early stopping, so that the set of weights with the best performance on the validation set is retained.

During training, the examples are randomly rotated (uniform random rotation described by a uniform random quaternion)[3] and randomly translated (by at most ± 6 Å along each Cartesian axis). This on-the-fly data augmentation procedure is essential to avoid overfitting (Ragoza, Hochuli, et al. 2017) and allows palliating for the lack of rotational invariance of standard CNNs.

It has been observed that CNN-based SFs—as well as other ML and DL SFs—are prone to rely too much on ligand information (Francoeur et al. 2020) and an effective data augmentation procedure to correct for this problem has been recently proposed (Scantlebury et al. 2020). Here we do not use the latter data augmentation—to remain consistent with the original data set preparation—but it is expected to provide a more generalisable model, and it will be explored in future work.

5.2.4 Evaluation of CNN SFs for Flexible Docking

We trained three different model architectures—`default2017`, `default2018`, and `dense`—with three-fold cross-validation. The data sets, as described above, consist of different annotations for the protein pose as well as clustering by pocket similarity. As in McNutt et al. (2021), we are mainly interested in the TopN metric, which is the percentage of targets with a good pose amongst the top N poses, as ranked by the SF. Since in the cross-docking data set we are dealing with systems subdivided by pocket and since each pocket can have a different number of protein-ligand pairs, the TopN metric is computed for the targets within a pocket, and then averaged across pockets—as in McNutt et al. (2021). Since we are ultimately interested in the ligand pose, the best possible case is where poses are ranked according to the ligand RMSD.[4]

Fig. 5.4 shows the performance of the CNN with the three different annotations for the flexible side chains (the annotation for the ligand remains the same), and with and without clustering by pocket similarity. The performance is computed by including or excluding minimised crystal poses from the test set

[3] The three-dimensional rotation group, $SO(3)$, can be sampled uniformly by generating three random numbers $u_1, u_2, u_3 \in [0, 1]$ and constructing the normalised quaternion

$$Q = \left(\sqrt{1-u_1}\sin(2\pi u_2), \sqrt{1-u_1}\cos(2\pi u_2), \sqrt{u_1}\sin(2\pi u_3), \sqrt{u_1}\cos(2\pi u_3)\right)^T,$$

which represents a rotation in three dimensions (Shoemake 1992).

[4] For the ligand-only annotation the best TopN metric is constant, while for ligand and receptor annotation the best TopN metric is not constant since a low-RMSD ligand pose does not necessarily correspond to a good pose—the global annotation can be "bad" for a "good" ligand pose because of a "bad" receptor conformation.

(minimised crystal poses are always included in the training set).

We see that when the crystal pose is included in the test set, the performance of the CNN SF is consistently and significantly better than the performance of the AutoDock Vina SF. This means that the CNN SF can distinguish near-native poses from other poses better than AutoDock Vina. This is true when systems are not clustered by pocket similarity as well as in the more stringent case where similar pockets are assigned to the same cross-validation fold.

When minimised crystal poses are removed from the test set, the performance of the CNN SF drops significantly, but so does the "best" performance. Since the best performance also drops significantly, this drop is attributed to the considerable lack of good poses when minimised crystal poses are removed. However, the gap between the CNN performance and the AutoDock Vina performance almost disappears. Only for the `flex2` annotation ($\text{RMSD}_{\text{lig}} < 2$ Å and $\text{RMSD}_{\text{flexMAX}} < 2$ Å) we observe a partial recovery of the performance gap for large values of N.

The latter observation also applies to the case where only the ligand annotation is employed ($\text{RMSD}_{\text{lig}} < 2$ Å, see Fig. 5.5a), indicating that the `flex1` and `max2` annotations might be too strict and therefore introduce confounding variables. Since the annotation for the ligand and the annotation for the flexible side chains are combined into a single annotation with an AND gate, a too stringent annotation for the flexible residues might cause too many good ligand poses to be annotated as "bad". This problem will be addressed in section 5.3.

Fig. 5.6 shows the most extreme case of this situation, where the 11th ranked pose is a very good pose—with a RMSD of only 0.41 Å—but it is annotated as a "bad" pose because of the high RMSD for the flexible residues (4.23 Å). We see that in a region reasonably far from the ligand being docked, the two protein structures diverge. In particular, we can see that the phenylalanine residues in the foreground are in completely different locations, due to the divergence of the backbone. This causes the pose of the residues treated as flexible to be annotated as "bad", and therefore the whole annotation is "bad" despite the very good pose for the ligand. Again, this problem is associated with the large size mismatch between the ligand being docked and the cognate ligand of the receptor target: the flexible loop in the binding site is in a very different conformation. Cases such as this one are essentially hopeless for our protocol, which uses the cognate ligand to select residues to be treated as flexible, and does not allow for backbone re-arrangements or relaxation.

This observation indicates that for some systems, the correlation between a good ligand pose and a good pose for the flexible residues is weak, and therefore combining them creates a noisy annotation.

We can see that the flexible residues contributing to the high RMSD are

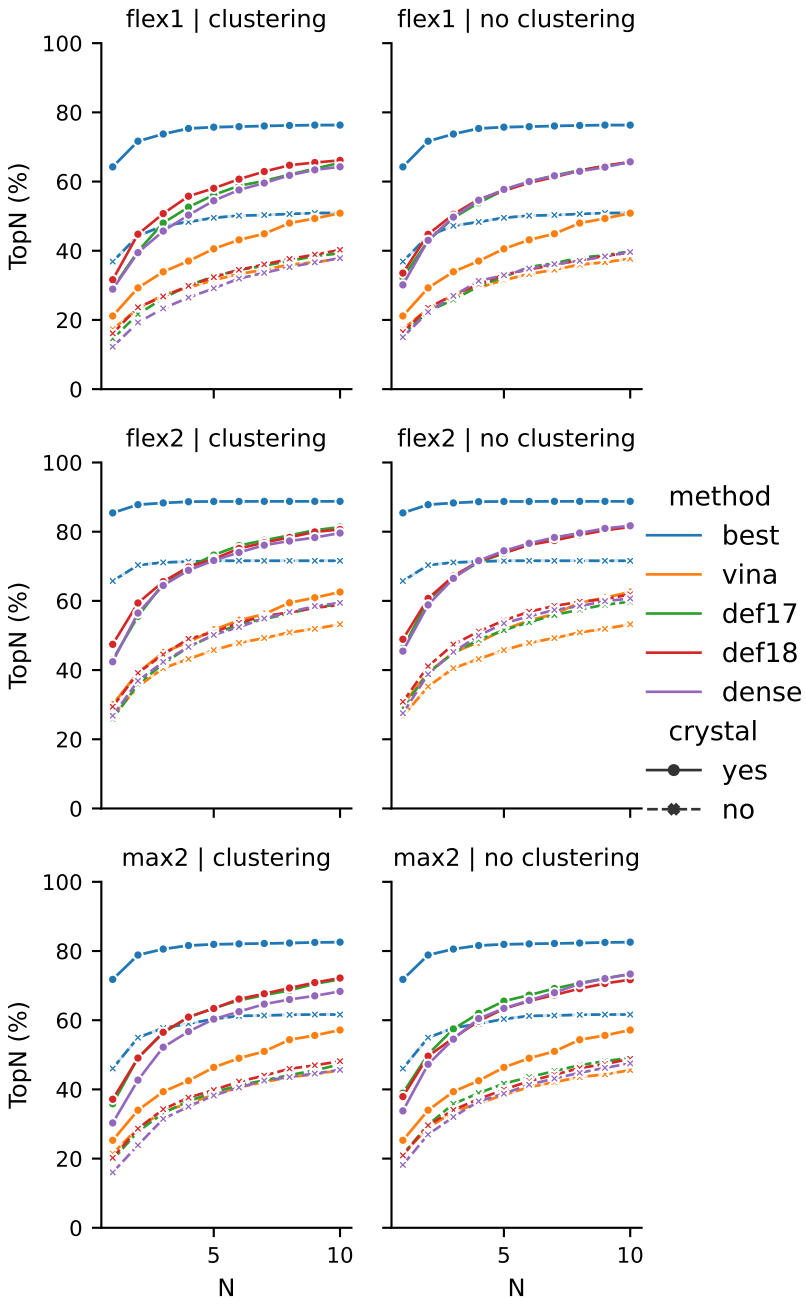
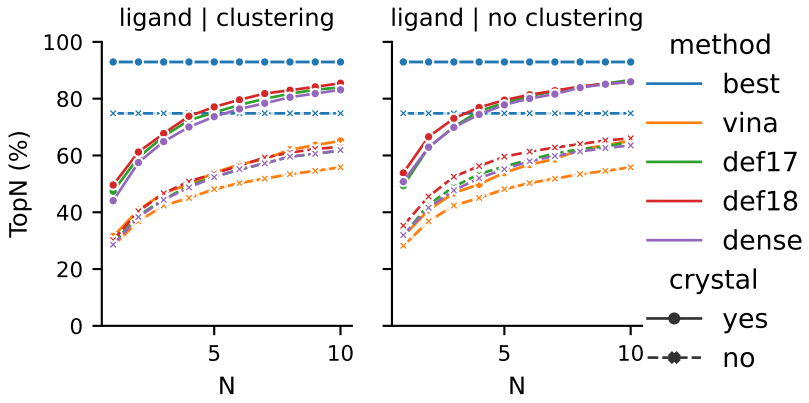
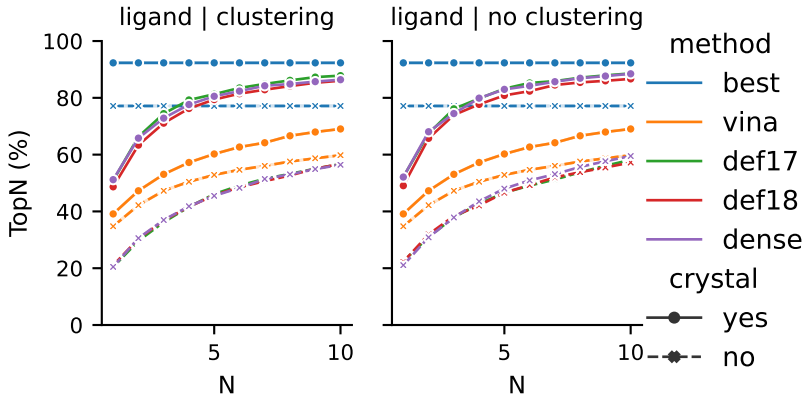


Figure 5.4: TopN performance for CNN SFs trained for flexible docking with different combined annotations for the ligand and the flexible side chains.



(a)



(b)

Figure 5.5: TopN performance for CNN SFs (a) trained and tested for flexible docking using only the ligand annotation, and (b) trained on rigid docking (ligand annotation only) and tested for flexible docking.

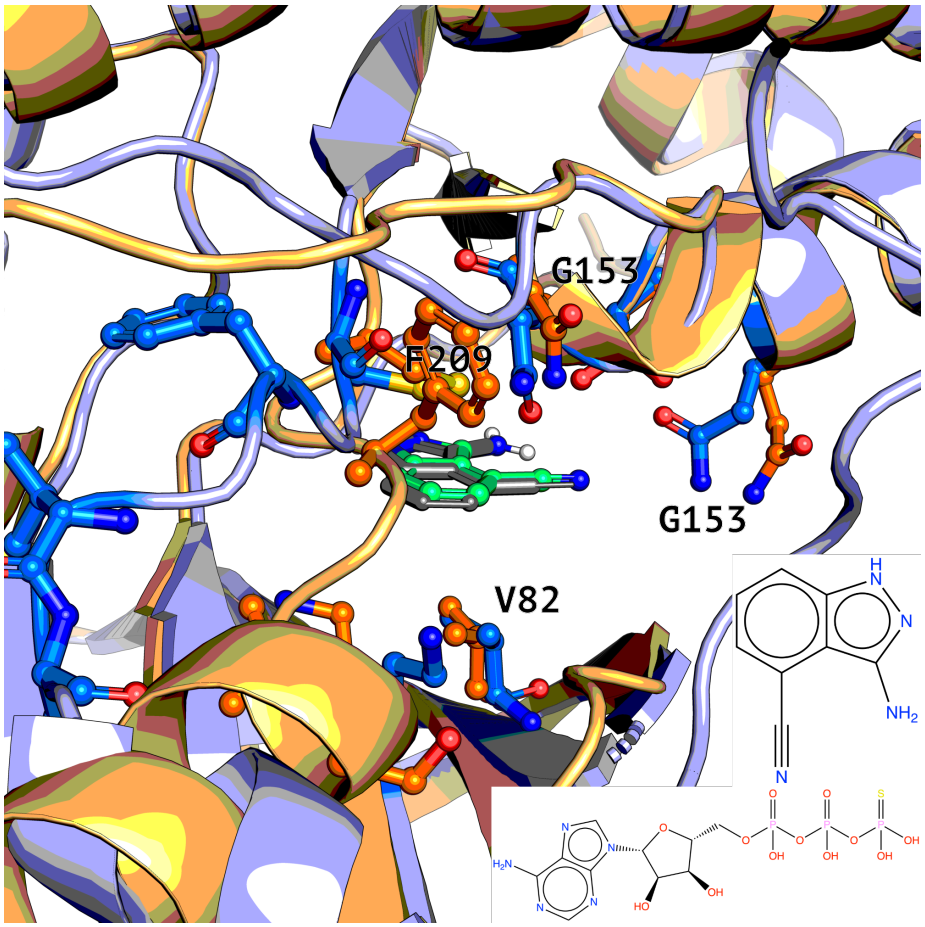


Figure 5.6: Cross-docked pose (rank 11) for ligand 3ZLY docked into target 3W8Q (human mitogen-activated protein kinase 1, MEK1) for the MP2K1 pocket of the cross-docking data set, shown together with the skeletal depiction of the docked ligand (top) and the cognate ligand used to automatically select the side chains to be treated as flexible (bottom). We see that the backbone in some regions of the binding site is very different between the cognate (orange) and target (blue) receptors, and the phenylalanine residues in the forefront have therefore completely different orientations. This causes the whole pose to be annotated as "bad" even if the ligand pose is "good".

reasonably far from the ligand (see Fig. F.3 for a different view compared to Fig. 5.6). This is related to the issue already discussed in relation to Fig. 4.7 and again outlines the problems related to a large size mismatch between the cognate ligand—employed here to automatically select the important residues in the binding site treated as flexible—and the ligand being docked. When there is a large mismatch between the two ligands there are more chances for the annotation to be noisy—because of differences in the backbone, which is not sampled, as well as differences in sampling.

This is a drawback of the automated selection of the flexible residues based on the cognate ligand of the docking target. Unfortunately, in a real cross-docking scenario, this is the only information that is likely available (but in some cases only the *apo* structure might be available).

For comparison, Fig. 5.7 shows a case where both the ligand pose and the pose of the flexible residues are very good—with RMSDs of 0.28 Å and 0.41 Å, respectively. We can see that in this case the backbone atoms are very well aligned for the whole of the binding site and docking with flexible residues can recover or maintain the correct conformation of the residues treated as flexible. In this case, the residues selected as flexible are reasonable thanks to the small size difference between the ligand being docked and the cognate ligand of the target. This situation, where the binding sites are in a very similar conformation, is more akin to re-docking and therefore easier for our cross-docking protocol.

Finally, it is interesting to determine how a SF trained on rigid receptor docking performs on docking to a flexible receptor. In Chapter 4 we used Gnina's default model—trained on rigid receptor docking—to perform docking with flexible residues and determined that if the flexible residues are carefully selected, the CNN model performs well. However, it is not clear *a priori* why we obtained a good performance despite the model not being trained explicitly for flexible docking. Therefore, we performed rigid docking with the AutoDock Vina SF on the cross-docking data set used here and trained CNN SFs on the same training and test splits used to train the CNN SFs for docking with flexible residues. Fig. 5.5b shows the performance of the CNN SFs trained on systems obtained by rigid docking (ligand annotation only) and tested on systems obtained using docking with flexible residues. Interestingly, we see that the SFs trained on rigid docking perform well on the flexible docking test set when minimised crystal poses are retained. In contrast, when minimised crystal poses are removed from the test set, not only the performance deteriorates, but it becomes considerably worse than the one of the AutoDock Vina SF.

This is an interesting observation that explains the good results obtained in Chapter 4, where the CNN SF was employed within the docking pipeline. Poses obtained by minimising the protein-ligand crystal structure considering

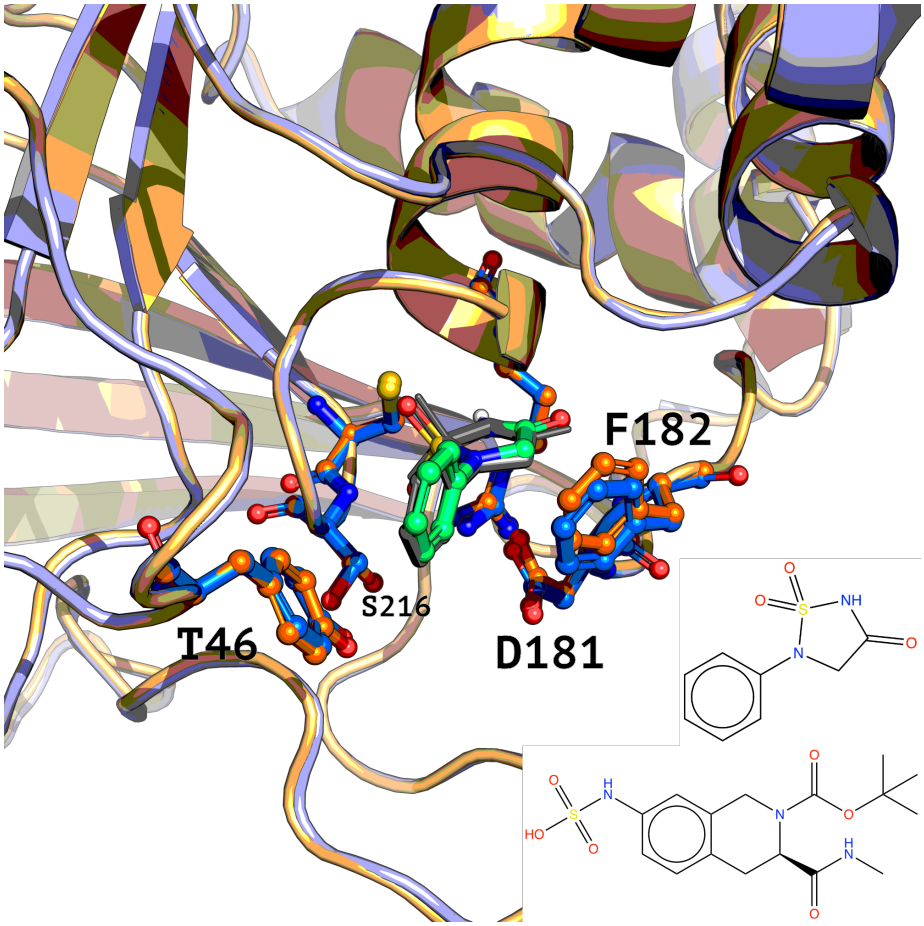


Figure 5.7: Cross-docked pose (rank one) for ligand 2BGE docked into target 2F6Y (protein tyrosine phosphatase 1B) for the PTN1 pocket of the cross-docking data set, shown together with the skeletal depictions of the docked ligand (top) and the cognate ligand used to automatically select the side chains to be treated as flexible (bottom). We see that the backbone is very well superimposed between the two structures.

side chains as flexible are rather similar to the poses obtained by minimising the same structures while retaining the receptor rigid. Since the minimised crystal structures make up the majority of examples in the positive class, the CNN model can exploit this similarity. This explains why CNN SFs trained on docking to a rigid receptor can perform well when docking with flexible side chains: when the docking software generates a near-native pose for the ligand, the CNN SF is able to recognise it, no matter how it was obtained (docking with or without flexible side chains). However, poses not obtained by minimising the crystal structure are somewhat different from the ones obtained with rigid docking—mostly because of the difference in side chain conformations—, and therefore the CNN SF trained on rigid docking struggles to classify them. This confirms that docking with flexible side chains provides additional signals to the CNN SF and that it is beneficial to train a CNN SF explicitly for docking with flexible residues. This is important to ensure the similarity between the training set and a real application scenario, and to avoid out-of-distribution samples.

However, as we demonstrated above, combining a RMSD-based annotation for both the ligand pose and the pose of the residues treated as flexible is not a good strategy since the annotation becomes rather noisy. Therefore, in the next section, we describe a PyTorch implementation of the Gnina SF that allows splitting the annotation for the ligand pose and the pose of the flexible residues.

5.3 Separate Annotation for Ligand and Flexible Side Chains

As mentioned above, combining the annotation for the ligand pose and for the pose of the flexible side chains into a global annotation with the AND gate might introduce too much noise. The noise comes from the fact that a good ligand pose might be annotated as a bad pose because the side chains are in a bad conformation, or because the backbone between the target and cognate receptor is significantly different. Therefore, it would be interesting to separate the two annotations.

Unfortunately, Gnina's codebase is not designed for flexibility and fast prototyping. As a monolithic C++/CUDA code combining the original code of AutoDock Vina, Open Babel, and a custom version of the discontinued Caffe DL framework, the code is rather complex to navigate and change, and the design is tailored toward speed rather than "playability". For this reason, it is not possible to easily play with different model architectures that require input other than a pose label and, eventually, the associated binding affinity.

Fortunately, the algorithms for featurising the protein-ligand binding site

with the density representation of Eq. (4.2) have been extracted into `libmolgrid`, a standalone software package that plays nicely with Caffe, as well as modern DL frameworks such as TensorFlow and PyTorch (Sunseri and Koes 2020). Therefore, we re-implemented the Gnina SF on top of `libmolgrid` in a well-designed and well-tested standalone Python package, with extensibility and flexibility for future research as design goal.[5]

5.3.1 PyTorch Implementation of Gnina Scoring Function

The weights of the pre-trained Gnina models—converted from Caffe to PyTorch—were kindly provided by Andrew McNutt (see RMeli/gnina-torch#34 and RMeli/gnina-torch#43, last accessed October 1, 2022).

Thanks to `libmolgrid`, it is possible to use the grid-based featurisation of the protein-ligand binding site with PyTorch and prototype different model architectures rather quickly. However, since there is no reference PyTorch implementation of the Gnina SF, we decided to build a new software package from the ground up, to provide such reference implementation, while keeping the design as flexible as possible for expansion toward new research topics—such as the separate pose annotation for the ligand and the flexible side chains we are interested in.

Our implementation consists of a Python package which provides training and inference scripts, as well as a script to reproduce the results of Gnina pre-trained models. The command line interface (CLI) of such scripts is kept as similar as possible to the training and inference scripts of the original Caffe implementation (available at gnina/scripts, last accessed October 1, 2022)—so that they can be used almost as a drop-in replacement—, and to Gnina's CLI. Additionally, the code is designed to be used as a library, whose components can be integrated within existing Python pipelines or to build web applications around the SF. The CLI provided is indeed built from such reusable components.

To significantly reduce the code to maintain and minimise the chances of introducing bugs, we use PyTorch Ignite to implement training, inference and logging (Fomin et al. 2020). PyTorch Ignite is a high-level library built on top of PyTorch allowing to write less boilerplate code—such as the training loop—while maintaining all the control needed. By design, the library mixes well with pure PyTorch code and therefore everything can be completely customised according to different needs.

Our implementation, which we call `gnina-torch`, tries to follow software engineering best practices as tightly as possible: the code comes with extensive

[5]The `gnina-torch` Python package is available at RMeli/gnina-torch (last accessed October 1, 2022).

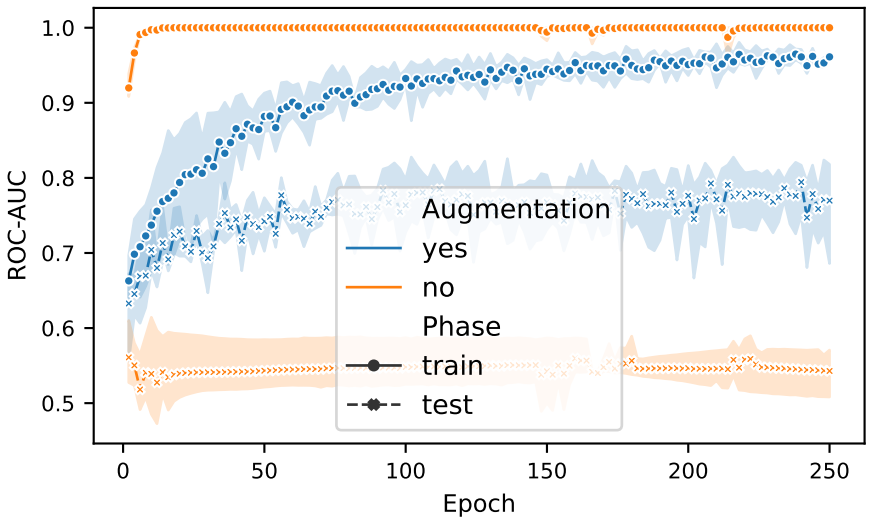


Figure 5.8: ROC AUC for pose classification on the CSAR data set in function of the number of training epochs, with and without data augmentation. Data augmentation is essential to prevent overfitting in CNN SFs.

unit tests (currently covering 99% of the codebase), as well as static linting tools for both code formatting and type hinting. All the checks are run automatically via continuous integration (CI) using GitHub Actions. Additionally, the code is now available on the Python Package Index (PyPI) (last accessed October 1, 2022), thus providing easy access to the whole library and its CLI.

Fig. 5.8 shows the ROC AUC for a pose classifier with the `default2017` model architecture trained on the CSAR data set as a function of the training epoch. The figure reproduces well Fig. 3 in Ragoza, Hochuli, et al. (2017). Worth noticing is the fact that the extensive data augmentation provided by random rotations and translations of the system is essential to obtain good performance on the test set. Without data augmentation, the model quickly overfits the training set, and it is unable to generalise to the test set.

Fig. F.4 shows the performance of the `default2018` model for binding affinity prediction trained on crystal poses from the PDBbind data set, while Fig. F.5 shows the performance of the same model trained on docked poses. The performance obtained with our PyTorch implementation is comparable to the performance of the original Gnina implementation, as it can be inferred by comparing the results of Fig. F.4 and Fig. F.5 with Fig. 2 in Francoeur et al. (2020).[6]

[6]Francoeur et al. (2020) describe the use of a pseudo-Huber loss function for binding affinity prediction. However, the actual implementation (see gnina/models, last accessed October 1, 2022), uses a standard $\mathcal{L}_2$ loss for the `default2018` and `dense` models. The `default2017` model does use

The extensive suite of unit tests, combined with the reproducibility of previously published results from Ragoza, Hochuli, et al. (2017) and Francoeur et al. (2020)—for which we did not perform any hyperparameter tuning—is a strong indicator that our independent PyTorch implementation works as intended and can therefore be used as a starting point for further research. The implementation is available on GitHub at RMeli/gnina-torch (last accessed October 1, 2022) under a permissive MIT licence (Meli and McNutt 2022).

Thanks to Andrew McNutt, who provided pre-trained weights of the original GNINA's Caffe models converted to PyTorch, `gnina-torch` can also reproduce all the results of GNINA previously obtained with the `default2017`, `default2018`, and `dense` models—including model ensembles, and the `default` ensemble. This further validates our PyTorch implementation, and provides users with more accessible pre-trained GNINA models within a `pip`-installable Python package that can be easily integrated into existing pipelines (such as `liGAN`, discussed in Chapter 7). The `gnina-torch` package, while still being under active development, has the potential to become the reference PyTorch implementation for GNINA's CNN-based SFs.

5.3.2 Architecture for Multitask Classification

To exploit a separate annotation for the ligand pose and the flexible side chains, we resort to multitask classification. As for the multitask prediction of ligand pose and binding affinity, we use the same convolutional layers—which automatically extract relevant features of the protein-ligand binding site—and two different fully connected layers. For the CNN layers, we use the same architectures introduced in Francoeur et al. (2020), and re-implemented in `gnina-torch`. A schematic of the `default2018` architecture for the classification of ligand and flexible side chains is reported in Fig. 5.9.

The loss function for the multitask classification is given by

$$\mathcal{L} = \mathcal{L}_{\text{ligand}} + \alpha \mathcal{L}_{\text{flex}}, \tag{5.1}$$

where $\mathcal{L}_{\text{ligand}}$ and $\mathcal{L}_{\text{flex}}$ correspond to binary cross-entropy losses for binary classification (low RMSD pose versus high RMSD pose) and α is a hyperparameter controlling the relative weight of the two loss functions.

Since no binding affinities are easily available for the cross-docking data set we are using here for training and validation, binding affinity prediction is not included in our implementation. However, thanks to the flexibility of our implementation and following the re-implementation of the original GNINA

the pseudo-Huber loss.

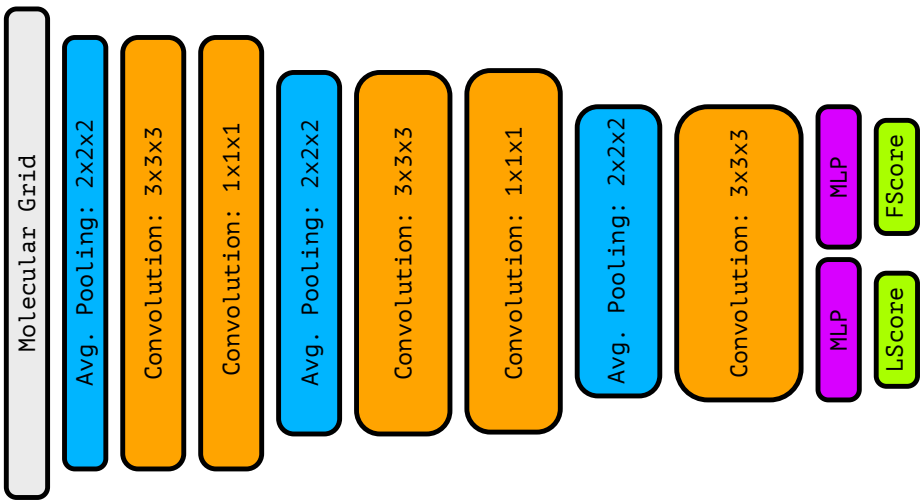


Figure 5.9: Schematic of GNINA `default2018` architecture for multitask ligand pose, and flexible side chains pose prediction.

SF described in section 5.3.1, it would be trivial to subclass the multi-task classification architecture to add binding affinity prediction as well.

5.3.3 Evaluation of Separate Annotation

Our PyTorch implementation of GNINA's SFs allows playing around with different model architectures. Here, we are interested in splitting the annotation of the ligand pose from the annotation concerning the flexible side chains. Fig. 5.10 shows the joint distribution of RMSDs for the ligand pose and the flexible pose, where we can see that combining the two annotations creates several examples in the data set where the ligand pose is good, but it is annotated as a bad pose because of the RMSD of the flexible residues. This is especially true when only poses with flexible side chains' RMSD smaller than 1 Å are annotated as positive. A similar figure showing the maximum flexible side chains' RMSDs is reported in Fig. F.6.

For completeness, Fig. F.7 and Fig. F.8 show the RMSD distributions in the case where minimised crystal poses are removed from the data set. We see that the bimodal distribution obtained when minimised crystal poses are present—where one of the modes represents low-RMSD poses—completely disappears and the ligand RMSD distribution becomes unimodal. This observation stresses again the importance of including minimised crystal poses to obtain a reasonable number of positive examples in the training set (although the data set remains highly unbalanced).

From Fig. 5.10 it is clear that it is important to distinguish the annotation

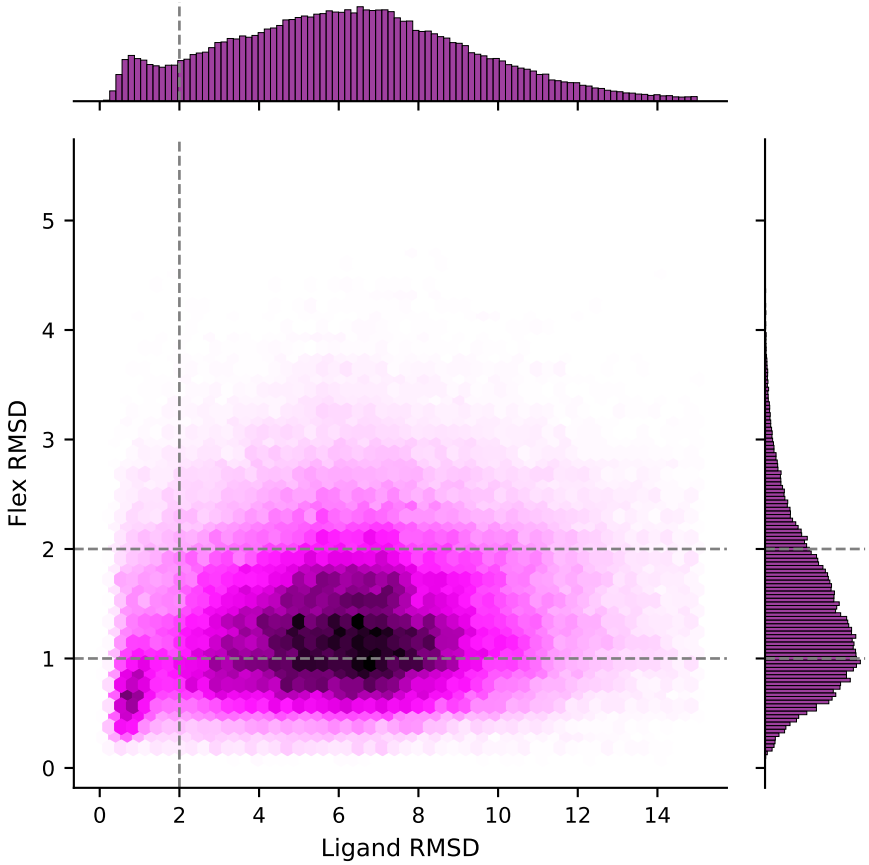


Figure 5.10: RMSD distribution for the ligand poses versus (pose-cognate) RMSD distribution of the flexible side chains. Dashed lines represent the boundaries for the different annotations.

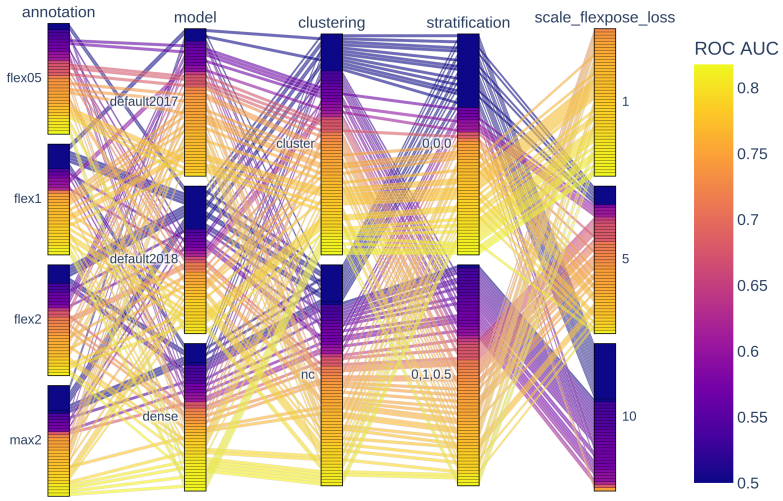
of the ligand pose from the annotation of the flexible side chains. This is now possible thanks to our flexible and extensible PyTorch implementation of the GNINA SF described above.

Fig. 5.11 shows the results of an extensive hyperparameter optimisation using 3-fold cross-validation, where a total of 144 hyperparameter combinations have been tested. The figure contains parallel coordinate plots, which can be interpreted as follows. Every line in the graph corresponds to a training run with different hyperparameters. Each line goes through the different hyperparameter classes (RMSD annotation, model, clustering, stratification, and scale α of the flexible residues' loss)—indicating which hyperparameters have been used—and the colour coding represents the ROC AUC—which is a measure of the performance of the classifier.

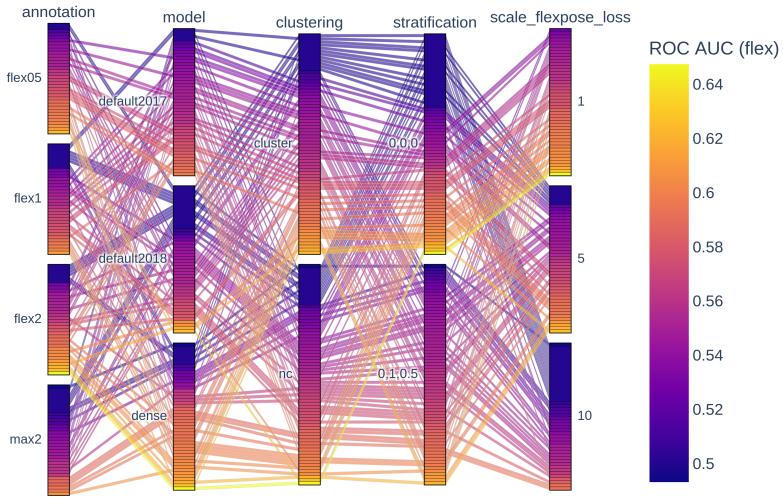
In addition to the annotations used above for the ligand and the flexible residues—which are now split into two different tasks—we introduced an additional annotation where the pose of the flexible residues is considered good if $\text{RMSD}_{\text{flex}} < 0.5\,\text{Å}$ (`flex05`). Additionally, now that ligand and receptor poses are separate, the oversampling of the minority class should be based on both the ligand pose and the pose of the flexible residues. Therefore, we had to use `libmolgrid` stratification functionality—originally developed to stratify by binding affinity—in an unorthodox manner to balance the sampling of both ligand and receptor poses. In Fig. 5.11, the hyperparameter $(0, 0, 0)$ for stratification indicates that no stratification was performed (i.e. only the ligand pose class is balanced), while the hyperparameter $(0, 1, 0.5)$ indicates that stratification is performed (i.e. both ligand pose and flexible side chains classes are balanced).

From Fig. 5.11 we can see that there are several hyperparameter combinations where the performance on ligand pose prediction is high, as measured by the ROC AUC. However, the performance on the classification of flexible side chains remains low for most hyperparameters, and it is close to or equivalent to the performance of a random classifier in several instances. It is clear from Fig. 5.11b that the near-random performance is obtained mostly when the loss function for the classification of flexible side chains is dominant. When $\alpha = 10$ the classification of the ligand pose becomes impossible—because the contribution of the flexible side chains is dominant—but the performance of the side chains pose classifier is random or near-random. This observation, combined with the extensive hyperparameter optimisation for which no satisfying performance is obtained in the classification of flexible side chains' poses, indicates that learning to classify the pose of the side chains based on RMSDs is very tricky.

There are multiple reasons behind this problem, which we mostly touched upon already. First, the RMSD of the flexible side chains used for annotation



(a) Ligand pose classification



(b) Flexible side chains pose classification

Figure 5.11: Parallel coordinate plot for the ROC AUC of CNN SFs trained on separate annotations for the ligand pose and the pose of flexible residues. Training and inference are performed with our custom PyTorch implementation.

also leads to an aggregated annotation, since some side chains might be in the correct conformation while others are not. Second, as already discussed in Fig. 4.7, for some systems the side chains selected as flexible based on the pose of the cognate ligand might not be necessarily the ones contributing to binding with the non-cognate ligand. Third, for the cases where the backbone differs significantly between the target and cognate receptor (see Fig. 5.6), the RMSD is dominated by differences in the backbone and, therefore, it is no longer a measure of the correctness of the pose.

5.4 Discussion and Conclusions

In this chapter, we extended the study of CNN SFs for flexible docking started in Chapter 4, by training several CNN SFs specifically designed for docking with flexible residues. We first built a large data set of cross-docked poses obtained from the cross-docking data set using the AutoDock Vina SF—as implemented in Gnina—and the automated selection of flexible residues already introduced in Chapter 4—in which we also discussed the limitations of this approach.

Using the large data set obtained by cross-docking with flexible residues, we trained different CNN models with different annotations, and compared the TopN performance with AutoDock Vina. Interestingly, we showed that when minimised crystal poses are included in the test set—they are always included in the training set, otherwise the data set would be too unbalanced—Gnina significantly outperforms AutoDock Vina. However, when minimised crystal poses are removed from the test set, then the difference in performance becomes rather small. This outlines the good performance of CNN SFs in recognising near-native poses, and explains why the default CNN model evaluated in Chapter 4 performs well for flexible docking, even if it has not been trained for that specific task.

By looking at the RMSD distributions of the data set obtained, it was clear that a single annotation for both the ligand pose and the pose of the flexible residues is suboptimal. While minimised crystal poses often end up with low RMSDs for both the ligand and the flexible residues, some poses can have a low RMSD for the ligand and a high RMSD for the flexible residues, and the other way around. Additionally, in some cases, the backbone differs significantly between the cognate and target receptors within the binding site. Using an AND gate to combine the two annotations does therefore introduce some confounding variables, which render the annotation noisy.

Unfortunately, Gnina only supports a single annotation, and given the complexity of the code it is difficult to experiment with different ideas. There-

fore, we re-implemented the GNINA SF—and only the SF, not the docking engine[7]—in a Python package based on PyTorch and PyTorch-Ignite. Having a Python implementation of the SF, called `gnina-torch`, based on a modern DL framework—instead of a discontinued one—allows for fast prototyping and exploration of new ideas. Additionally, thanks to the original model weights converted from Caffe to PyTorch and provided by Andrew McNutt, `gnina-torch` has the potential of becoming a semi-official implementation of GNINA CNN-based SFs, that can be easily integrated into different Python pipelines, as well as web applications for drug discovery.

Using the `gnina-torch` implementation we trained several CNN SFs with separate annotation—based on the RMSD—for the ligand pose and the pose of the flexible residues. Despite an extensive hyperparameter search and the use of several annotations, it appears that the model can learn the ligand pose but not the pose of the flexible residues—for which the classification performance remains close to random. This is again attributed to the noisiness of the annotation for flexible residues, which might have several residues in the correct pose—especially the ones interacting with the ligand—but might have one or a few in an incorrect conformation and with a high RMSD. This problem is certainly related to the automatic selection of the residues to be treated as flexible since when ligands' sizes are very different it is more likely that flexible residues far from the ligand can end up with incorrect conformations despite the ligand being in a good pose. Distant residues can also suffer from divergences in the backbone conformation, as we have seen. Therefore, we conclude that while the RMSD annotation for the ligand works well, annotating the side chains as flexible also using the RMSD is not informative enough.

Going forward, it would be interesting to explore different annotations for the flexible residues that are more robust to the actual pose of the flexible residues, such as the number of conserved contacts with the ligand, or different interaction classes. Thanks to the new `gnina-torch` implementation, exploration of new ideas is likely to be orders of magnitude easier, and we hope the code can enable new research directions in CNN-based SFs. Our implementation also outlines the potential for an extensible docking framework in Python, in which several of the new ML and DL SFs can easily be plugged for testing.

Additionally, it is worth noticing that the focus of DL applications in docking has recently shifted to the sampling problem—in contrast with the scoring problem considered here, where sampling is performed according to a classical SF. Interesting ideas for the generation of docked poses are starting to appear (Masters et al. 2022; Stärk et al. 2022). These new ideas use a coarse representation

[7] An end-to-end differentiable docking engine with Python bindings would be extremely useful and an interesting project in itself.

of the receptor or the binding site, therefore it would be very interesting to evaluate their performance on cross-docking compared to GNINA.

The model of Stärk et al. (2022) is limited to whole protein docking, and therefore can't be directly compared to GNINA for the use case described here—that is, cross-docking to a specific binding site. However, we expect future iterations to be able to generate poses within a well-defined binding site thus opening direct comparisons with GNINA. The model of Masters et al. (2022) does compare cross-docking with AutoDock Vina—and it shows an advantage on the Top1 metric, which disappears for the Top5 metric—, but the code is not available, and therefore it is currently impossible for us to perform a direct comparison with GNINA—which has been shown to significantly outperform the AutoDock Vina SF (McNutt et al. 2021). In the near future, it would be interesting to evaluate novel DL models directly generating protein-ligand poses on cross-docking, and to see how they compare with current CNN-based SFs and future iterations based on a better annotation of the flexible receptor.

This page intentionally contains only this sentence.

6

Binding Affinity Prediction with Atomic Environment Vectors

Contents

R. Meli, A. Anighoro, M. Bodkin, G. M. Morris and P. C. Biggin, "Learning Protein-Ligand Binding Affinity with Atomic Environment Vectors", *J. Cheminform.* 13, 59 (2021). DOI: 10.1186/s13321-021-00536-w (Meli, Anighoro, et al. 2021)

The code associated with this chapter is available at the following GitHub repository: RMeli/aescore.

In Chapter 4 and Chapter 5 we used the grid-based representation of the protein-ligand binding site introduced by Ragoza, Hochuli, et al. (2017), and

worked with CNN SFs. As we discussed, such representation encodes clear spatial relationships, it is easy to compute with parallel algorithms on modern GPU accelerators, can be combined with standard architectures for computer vision, and provides good performance for docking SFs. However, grid-based methods are coordinate frame dependent. Therefore, they need to be combined with data augmentation techniques to generalise (Ragoza, Hochuli, et al. 2017). In Fig. 5.8 the importance of data augmentation to obtain good performance in pose prediction is clearly shown, and the same remains true for binding affinity prediction (Jiménez, Škalič, et al. 2018). The dependence on the coordinate frame is not a desired property, since it implies that translational and rotational invariance is not ensured.

Cartesian coordinates are generally not suited to compare different molecular structures: the order of the atoms in the list of atomic coordinates is arbitrary, and molecules with different Cartesian coordinates could be related by symmetry operations—rotation, reflection and translation (Bartók, Kondor, et al. 2013).[1]

For the generation of interatomic NNPs—useful for accurate simulations of large (bio)molecular systems on long timescales, at an accuracy comparable with quantum chemical calculations—researchers had to circumvent the problems inherent to Cartesian coordinates to build useful representations. Ideally, such representations should also be able to cope with a varying number of atoms—to ensure transferability across different systems—and be end-to-end differentiable—to compute atomic forces. Several methods to represent atomic environments have been published over the years (Bartók, Kondor, et al. 2013) and have been successfully employed to generate NNPs (Behler and Parrinello 2007), to fit many-body potentials (Bartók, Payne, et al. 2010), to fit atomisation energies (Rupp et al. 2012), and for many other interesting applications in chemistry and material science. The current state of the field of ML potentials has been recently reviewed by Unke et al. (2021).

Representations from quantum chemistry have been previously applied with success to virtual screening for the classification of binders and decoys against a particular target (Bartók, De, et al. 2017). Here, we explore the use of Behler-Parrinello atom-centred symmetry functions (ACSFs) (Behler and Parrinello 2007) and the associated atomic environment vectors (AEVs) (Smith, Isayev, et al. 2017)—in combination with a collection of feed-forward NNs—for the prediction of protein-ligand binding affinities. By construction, the architecture employed here is rotationally and translationally invariant, and therefore does not require extensive data augmentation by artificially rotating and translating the system on-the-fly during training.

[1] This problem also appears in the calculation of RMSDs, as discussed in Appendix B.

6.1 AEVs and NN architecture

To predict the binding affinity of a ligand to a target of interest, we need a description of the protein-ligand binding site that allows the key protein-ligand interactions to be learned. Ideally, this representation should depend only on the relative positions of the ligand and the protein—the representation should be invariant under translation, rotation, and mirror operations. However, some ML and especially DL SFs employed in computational drug discovery do not satisfy such conditions: grid-based methods are not translationally or rotationally invariant and need extensive data augmentation (Ragoza, Hochuli, et al. 2017), while vector-based representations are often order-dependent.

Local representations of atomic environments satisfying the ideal properties outlined above have been employed with success in ML for quantum chemistry (Bartók, De, et al. 2017; Bartók, Kondor, et al. 2013; Bartók, Payne, et al. 2010; Behler and Parrinello 2007). In particular, the ACSFs originally introduced by Behler and Parrinello (2007) and further developed to build the Accurate NeurAl networK engINe for Molecular Energies (ANAKIN-ME or "ANI" for short) family of NNPs have been successful in producing accurate molecular properties (Behler 2011; Smith, Isayev, et al. 2017; Smith, Nebgen, Lubbers, et al. 2018).

Here we employ the ACSFs defined for the ANI family of NNPs to represent the protein-ligand binding site, where protein residues with at least one atom within a distance d from the ligand are considered.

For each atom i of element X in the system, its chemical environment can be represented by combining radial ($G^R_{i;\alpha,m}$) and angular ($G^A_{i;\alpha,\beta,m}$) ACSFs in a one dimensional vector $\mathbf{G}^X_i = \{G^R_{i;\alpha_1,m_1}, \ldots, G^A_{i;\alpha_1,\beta_1,m_1}, \ldots\}$, called AEV. X corresponds to the element of the atom for which the AEV is being computed, while α and β denote the elements of the neighbours within a cutoff radius, R_c. ACSFs capture the atom's radial and angular chemical environment (Smith, Isayev, et al. 2017), and their locality is ensured by a cutoff function:

$$f_c(R_{ij}) = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & R_{ij} \leqslant R_c, \\ 0 & R_{ij} > R_c. \end{cases} \tag{6.1}$$

Radial symmetry functions are given by (Behler and Parrinello 2007; Smith, Isayev, et al. 2017)

$$G^R_{i;\alpha,m} = \sum_{\substack{j \neq i \\ j \in \alpha}} e^{-\eta_R (R_{ij} - R_s)^2} f_c(R_{ij}), \tag{6.2}$$

where the index m runs over the set of parameters $\{\{R_s\}, \{\eta_R\}\}$ and the summation over j runs over all the atoms of element α. η_R controls the width of the radial

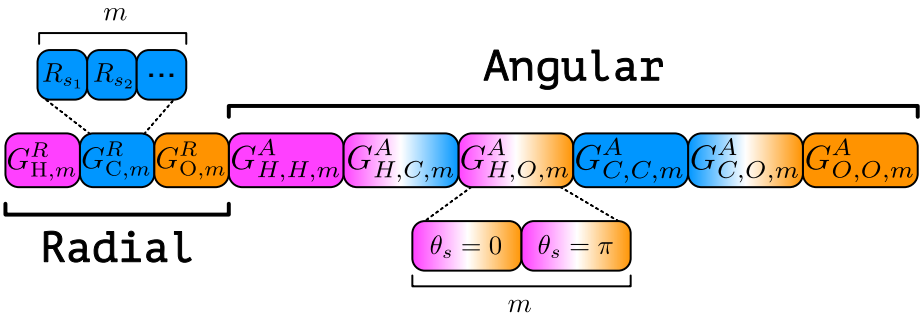


Figure 6.1: AEV constructed using ACSFs (Behler and Parrinello 2007; Smith, Isayev, et al. 2017) (with $R_s = 0$ and $\{\theta_s\} = \{0, \pi\}$ for angular symmetry functions) for an atom in a system composed only of the elements H, C and O. The radial and angular symmetry functions, $G^R_{\alpha,m}$ and $G^A_{\alpha,\beta,m}$, respectively, are given for the elements α and β, and iterate over the parameters m. Loosely adapted from Gao, Ramezanghorbani, et al. (2020).

Gaussian distributions, while R_s controls their radial shift.

Angular symmetry functions are defined as (Smith, Isayev, et al. 2017)

$$G^A_{i;\alpha,\beta,m} = \\ = 2^{1-\zeta} \sum_{\substack{j,k\neq i \\ j\in\alpha k\in\beta}} [1 + \cos(\theta_{ijk} - \theta_s)]^\zeta \, e^{-\eta_A \left(\frac{R_{ij}+R_{ik}}{2} - R_s\right)^2} f_c(R_{ij}) f_c(R_{ik}), \tag{6.3}$$

where the index m runs over the set of parameters $\{\{R_s\}, \{\theta_s\}, \{\eta_A\}, \{\zeta\}\}$ and the summation runs over pairs of atoms of elements α and β. η_A and R_s have the same role of η_R and R_s in the radial symmetry function described above, with θ_s capturing different regions of the angular environment, while ζ controls the width of the peaks of the ACSF in the angular environment (Smith, Isayev, et al. 2017).

The AEVs $\mathbf{G}^X_i$ of atom i of element X—composed of different ACSFs in a single vector—encodes the neighbour-dependent local atomic environment of atom i of element X. This corresponds essentially to fine-grained and flexible atom typing, in contrast to the static and arbitrary atom types employed in standard SFs.

Fig. 6.1 shows schematically the components of an AEV for an atom in a system composed only of the elements H, C, and O. By construction, this vector is translationally and rotationally invariant as well as invariant under the exchange of two atoms of the same element. A few examples of calculations of ACSFs and AEVs for simple systems are reported in section G.1.

To keep the size of the AEVs reasonably small, we restrict the parameters of $G^A_{\alpha,\beta,m}$ to those of the original Behler-Parrinello formulation: $\{\theta_s\} = \{0, \pi\}$ and

$R_s = 0$. All other parameters are the same as those employed in the ANI-1x NNP (Smith, Isayev, et al. 2017), which results in an AEVs size of 200 (for each atom). AEVs are built using the `AEVComputer` as implemented in TorchANI (Gao, Ramezanghorbani, et al. 2020).[2]

6.1.1 Neural Network Architecture

The NN architecture is implemented using PyTorch (Paszke et al. 2019),[3] loosely following the original work of Behler and Parrinello (Behler and Parrinello 2007), the ANI family of NNPs (Smith, Isayev, et al. 2017), and the TorchANI implementation (Gao, Ramezanghorbani, et al. 2020). It consists of n_e atomic neural networks (ANNs), where n_e is the number of elements in the data set. The ANNs are standard feed-forward NNs with rectified linear unit (ReLU) activation functions and dropout layers. The outputs of the atomic NNs are then summed together to obtain the final estimate of the binding affinity.

Fig. 6.2 shows a schematic representation of the model for a hypothetical system composed of two hydrogen atoms, one carbon atom, and one oxygen atom (such as formaldehyde). The AEVs $\mathbf{G}_i^X$ corresponding to atoms of the same element X are propagated through the same ANNs (with the same weights). All atomic contributions are summed together to get the final prediction.

The idea behind the decomposition of the binding affinity into atomic contributions is essentially the one that has been proven useful for short-range energy decomposition in NNPs. The negative logarithm of the binding affinity $pK = -\log_{10}(K/c_0)$ is proportional to the Gibbs free energy of binding

$$pK = -\frac{1}{\ln(10)} \frac{\Delta G_0^{\text{bind}}}{RT} \tag{6.4}$$

and therefore decomposing pK into atomic contributions corresponds to a decomposition of the Gibbs free energy. As for the total energy in NNPs, this decomposition allows the description of local contributions only (Bartók, Kondor, et al. 2013), but it is very effective in practice—as demonstrated by the success of NNPs in fitting high-dimensional potential energy surfaces (Bartók, De, et al. 2017; Behler and Parrinello 2007; Smith, Isayev, et al. 2017; Smith, Nebgen, Lubbers, et al. 2018; Smith, Nebgen, Zubatyuk, et al. 2019). This decomposition also appears to be very effective in generalisation and transferability, since it works for systems much larger than the ones included in the training set (Smith, Isayev, et al. 2017).

[2] Originally, TorchANI 2.1 was used. However, the software has been kept up to date—partially thanks to CI—and it now works with TorchANI 2.2.

[3] Originally, PyTorch 1.7 was used. However, the software has been kept up to date—partially thanks to CI—and it now works with PyTorch 1.11.

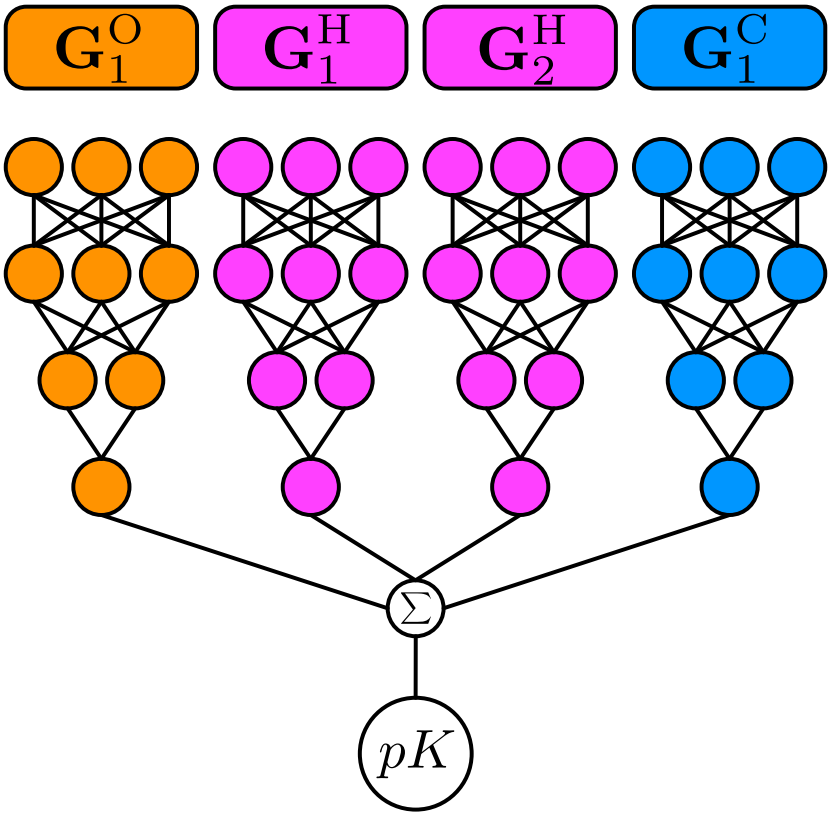


Figure 6.2: Propagation of AEVs $\mathbf{G}_i^X$ through ANNs for a hypothetical system composed of two hydrogen atoms, one carbon atom, and one oxygen atom—such as formaldehyde. The AEVs $\mathbf{G}_i^X$ are constructed for each atom i of element X as described in the main text and propagated through the atomic NN of the corresponding element (NNs with the same colour have the same weights). All atomic contributions are finally summed together to obtain the pK prediction. This image is loosely adapted from Smith, Isayev, et al. (2017).

6.1.2 Δ-Learning

Δ-*learning* is a powerful ML approach where the model is trained to predict the corrections to a baseline towards the target value, instead of predicting the target value itself (Ramakrishnan et al. 2015). This approach has been applied successfully to the prediction of molecular properties from quantum mechanical calculations (Ramakrishnan et al. 2015) as well as to binding affinity predictions (Lu et al. 2019; Wang and Zhang 2016). In the context of docking SFs, a Δ-learning approach has the advantage of retaining the good docking power of traditional methods while significantly improving the SF for binding affinity prediction (Wang and Zhang 2016).

In this work we explored the use of Δ-learning in combination with the AutoDock Vina SF (Trott and Olson 2009). The Δ-AEScore SF is therefore given by:

$$\Delta\text{-AEScore} = S + \Delta \tag{6.5}$$

where S is the standard AutoDock Vina score (converted to pK units) and Δ is the learned correction.

6.1.3 Consensus Scoring

To compensate for the variability introduced by random initialisation of model's weights and their stochastic optimisation, we investigated the use of *consensus scoring* to evaluate our models. Consensus scoring has been shown, in some cases, to improve performance across targets in structure-based virtual screening (Ericksen et al. 2017; Francoeur et al. 2020; Oda et al. 2006).

For consensus scoring, five models were randomly initialised and independently trained. Final predictions were obtained as the average protein-ligand binding affinity of the models. This technique also allows the computation of the standard deviation associated with each prediction. The benefits of consensus scoring are analysed retrospectively below.

6.2 Training and Test Data Sets

In this work, the PDBbind 2016 Refined set is used for training and validation while the CASF-2013 and CASF-2016 data sets are used for testing and comparison with other ML and DL models, as well as classical SFs (Li, Han, et al. 2014; Li, Liu, et al. 2014; Li, Su, et al. 2018; Liu, Su, et al. 2017; Su, Yang, et al. 2018). These data sets are described in Chapter 2.

The PDBbind 2016 Refined set is randomly split into training and validation sets with a 90/10 ratio. Systems present in both PDBbind and CASF data sets

are removed from the training and validation sets and used only for testing. This procedure ensures that there is no exact overlap ("hard overlap") of protein-ligand complexes between the PDBbind (training/validation) and CASF (test) data sets, although some overlap with similar targets and ligands remains (Li, Han, et al. 2014; Su, Feng, et al. 2020; Su, Yang, et al. 2018). To assess this remaining "soft overlap" between training and test sets—arising from similar proteins, similar binding sites, and similar ligands—we use the subset of the PDBbind 2016 data set proposed by Su, Feng, et al. (2020).

Ligand SDF or MOL2 files from the data sets were either converted to PDB files using Open Babel and parsed using MDAnalysis (for scoring and ranking) or parsed directly with Open Babel's Python bindings (for docking and screening). Protein PDB files were discarded when the element column was absent or could not be parsed correctly by MDAnalysis (this never occurred for the test set). All water molecules were removed from the data set. All the systems in the PDBbind and CASF data set were automatically protonated using Open Babel, and given the size of the data set, the protonation state was not further assessed.

The complexity of the NN model grows quickly with the number of atomic species present in the data set since every element requires its own atomic NN. Therefore, we selected only protein and ligand atoms (retaining protein residues with at least one atom within distance d from the ligand), thus neglecting water and other molecules. Additionally, we removed the few selenoproteins present in the training or validation sets. The following elements remained (in order of abundance for the ligands, see Fig. G.3): H, C, O, N, S, P, F, Cl, Br, I. This resulted in a total of 10 atomic NNs, one for each element.

When "hard overlaps" with CASF-2016 were removed, the final training set consisted of 3377 complexes while the validation set consisted of 376 complexes, while when "hard overlaps" with CASF-2013 were removed, the final training set consisted of 3464 complexes while the validation set consisted of 385 complexes. The CASF test sets are left unchanged.

6.3 Results

6.3.1 AEScore

Hyperparameters Optimisation

The hyperparameters of our model—the number and size of layers in the elemental NNs, dropout probability, batch size, and protein-ligand distance d—were optimised with a grid-based method and manually fine-tuned to maximize Pearson's correlation coefficient between the predicted and experimental binding

affinities on the validation set.

We found that a protein-ligand distance $d = 3.5$ Å and 256-128-64-1 feed-forward NNs performed best when combined with a batch size of 64 and a dropout probability of 25%.

Tab. G.2 shows the performance of the model—with consensus scoring—on the validation test for different values of d. Using a distance of $d = 4.0$ Å does not change the performance, compared to $d = 3.5$ Å. However, the larger number of protein atoms causes the computational time to be increased (see Tab. G.2 for the average time per epoch during training). Visual inspection of a selection of systems showed that the $d = 3.5$ Å selects the important residues in the binding site.

The model's weights were optimised using the ADAM optimizer (Kingma and Ba 2014) with a learning rate of 1×10^{-4}, and using PyTorch's default parameters ($\beta_1 = 0.9$ and $\beta_2 = 0.999$).

Dropout layers are usually not employed in NNPs, but our hyperparameter search shows that they increase the performance of our model by decreasing overfitting on the training set, thus improving transferability.[4]

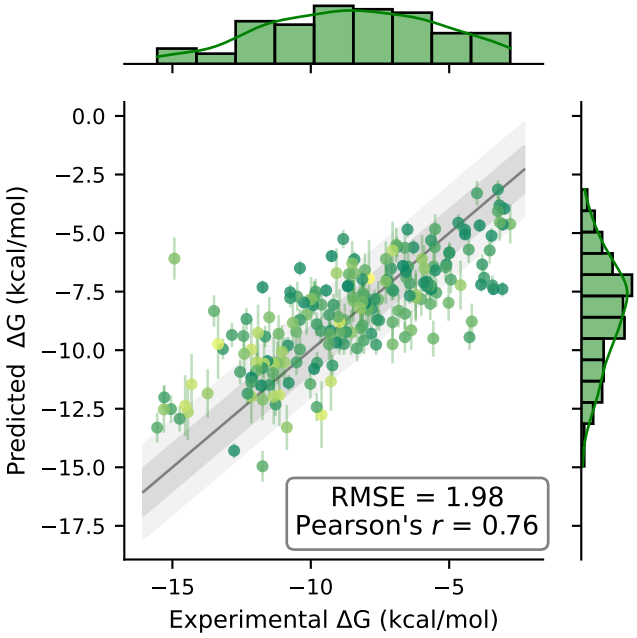
Scoring Power

Fig. 6.3 shows the predictions of our model versus the experimental values of the binding affinity for the CASF-2013 and CASF-2016 benchmark data sets—when only protein and ligand atoms are considered. Our model achieves an RMSE of 1.30 pK units and a Pearson's correlation coefficient of 0.80 on the CASF-2016 test set, and an RMSE of 1.46 pK units and a Pearson's correlation coefficient of 0.76 on the CASF-2013 test set. Error bars show the standard deviation of the predictions obtained with consensus scoring (average over five independently trained models).

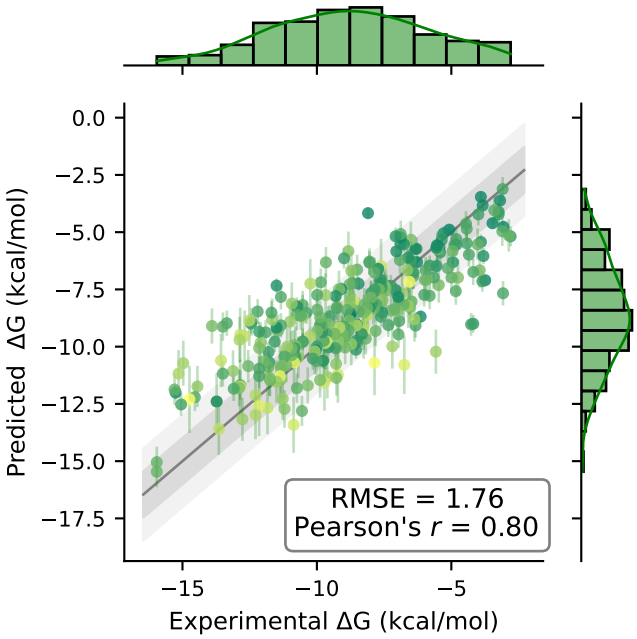
Confidence intervals (CIs) for the correlation coefficient can be obtained by bootstrapping (with 10 000 bootstrap replicates), as described in the CASFs evaluation (Su, Yang, et al. 2018). The 90% CI for Pearson's correlation coefficient for the CASF-2016 test set is $[0.76, 0.83]_{\text{CI }90\%}$, while for the CASF-2013 test set it is $[0.68, 0.81]_{\text{CI }90\%}$.

Fig. 6.4 shows a breakdown of Pearson's correlation coefficient (and the RMSE) for each protein class in the CASF-2016 benchmark data set. We see that the performance of AEScore is class-dependent and there is no clear correlation between Pearson's correlation coefficient and the RMSE (by comparing class 1 and class 55, for example). For the majority of targets, the predicted binding

[4] NNPs are trained on results of quantum chemistry calculations, therefore the training and test data sets are not noisy, in contrast to experimental binding affinities.



(a) CASF-2013



(b) CASF-2016

Figure 6.3: Predicted versus experimental binding affinities for AEScore, expressed in kcal mol^{-1}. Data points are colour-coded according to their corresponding standard deviation.

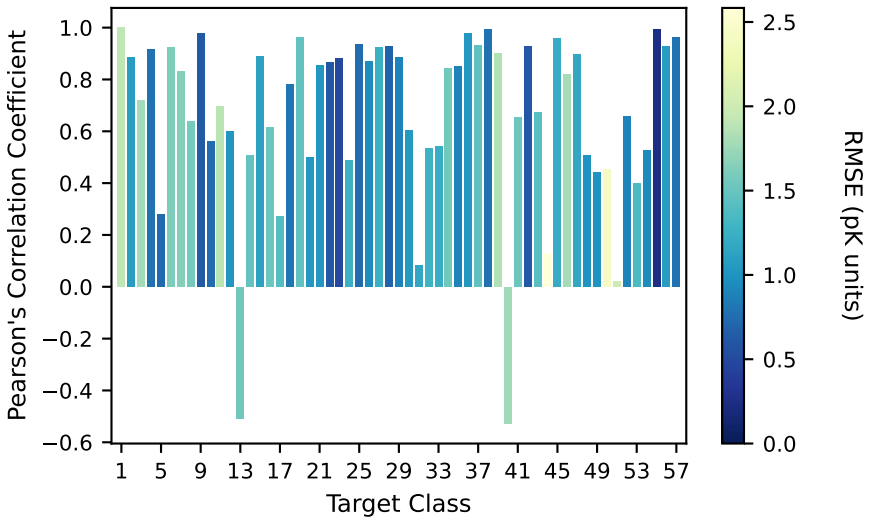


Figure 6.4: Per-class Pearson's correlation coefficient colour-coded by RMSE in pK units, for the 57 classes of the CASF-2016 data set.

affinity is well correlated with the corresponding experimental value. Only a few classes have a low correlation coefficient and two classes show a negative correlation. The average and median Pearson's correlation coefficients across all target classes are 0.67 and 0.82, respectively.

Fig. G.4 compares per-class Pearson's correlation coefficient obtained with AEScore (and reported in Fig. 6.4) with the results obtained with Gnina (McNutt et al. 2021), the CNN SF described in Chapter 4. We see that for most classes, the Pearson's correlation coefficient obtained with both methods is similar. However, there are some classes where the difference between the two methods is larger than 0.2 and, in such cases, Gnina shows a better correlation in most cases (15 out of 21).

This protein class-dependence opens up the scope for protein-specific models or fine-tuning—for example by refining the model using transfer learning—which are likely to improve per-class performance (Imrie, Bradley, van der Schaar, et al. 2018; Ross et al. 2013).

Consensus Scoring

In the previous section, we employed consensus scoring—with five independently trained models—since it has previously been shown to improve performance (Ericksen et al. 2017; Francoeur et al. 2020). A small performance boost is also obtained in our case, as it can be verified retrospectively.

If we consider the CASF-2016 data set, the average correlation coefficient of the five independent models is 0.77 (minimum 0.77, maximum 0.78) while consensus scoring reaches 0.80—better than the best-performing individual model among the five. The same observation is true for the RMSE on the same test set. The average RMSE is 1.38 pK units (minimum 1.35, maximum 1.42) while the consensus scoring has a RMSE of 1.30 pK units—which is lower than the best-performing model among the five.

Implicit Hydrogen Atoms

To assess the impact of automatic protonation using Open Babel (O'Boyle, Morley, et al. 2008) we also trained AEScore without hydrogen atoms for both the protein and the ligand. This results in the removal of one atomic NN, thus decreasing the number of parameters in the model.

Training the model without hydrogen atoms does not consistently affect the performance of our model: we observe a small decrease in performance with the CASF-2013 test set and a small gain with the CASF-2016 test set. For the CASF-2013 test set, we obtain a Pearson's correlation coefficient of $0.75 \in [0.69, 0.80]_{\text{CI }90\%}$ and a RMSE of 1.48 pK units while for the CASF-2016 test set we obtain a Pearson's correlation coefficient of $0.81 \in [0.77, 0.84]_{\text{CI }90\%}$ and a RMSE of 1.28 pK units.

Per-class Pearson's correlation coefficient (and RMSE) for the CASF-2016 test set for the model trained without hydrogen atoms is shown in Fig. G.5. Again, there is no clear relationship between Pearson's correlation coefficient and RMSE. In this case, the average Pearson's correlation coefficient is 0.69 while the median is 0.85.

The model trained without hydrogen atoms is less memory intensive–thanks to one less ANN—and faster—thanks to the lack of several forward passes through the hydrogen ANN.

Similarity Between Training and Test Sets

As mentioned above, we removed the systems appearing in the CASF-2016 and CASF-2013 benchmark data sets from the training sets, thus removing the so-called "hard overlap". However, some "soft overlap"—arising from similar proteins, similar binding sites, and similar ligands—between the training and test sets remains and could therefore artificially inflate the results. This is a known problem as shown by Boyles et al. (2019) and, more recently, by Su, Feng, et al. (2020) who both proposed non-redundant subsets of the PDBbind refined set with decreasing similarity with respect to the CASF-2016 test set. Such non-redundant data sets allow assessing how SFs behave when the "soft

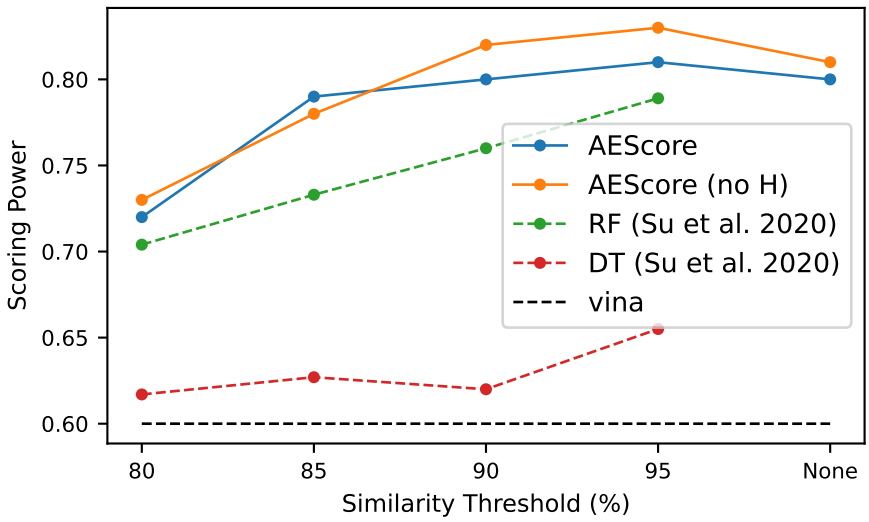


Figure 6.5: Scoring power of AEScore (with and without hydrogen atoms) as a function of the similarity threshold between the training and test sets, as defined by Su, Feng, et al. (2020). The raw data for the RF and DT SFs was kindly provided by Su, Feng, et al. (2020) upon request. RF and DT are respectively the best and worst performing models (at the 95% similarity threshold) presented in Su, Feng, et al. (2020) and are consistently outperformed by AEScore.

overlap" between the training and test sets is incrementally reduced. The non-redundant training sets developed by Su, Feng, et al. (2020) are shortly described in section 2.3.3.

Fig. 6.5 shows the performance of our model on the CASF-2016 data set when trained on the non-redundant training sets proposed by Su, Feng, et al. (2020), with different similarity thresholds ("None" indicates that only the "hard overlap" between training and test sets is removed). We see that as the overlap threshold between the training and test sets increases, the performance of our model also increases. Interestingly, a similarity threshold of 95% does not negatively affect our SF, in contrast with other ML SFs (Su, Feng, et al. 2020). This trend is similar to the RF model of Su, Feng, et al. (2020), which is consistently outperformed by our model. Other ML SFs evaluated by Su, Feng, et al. (2020) are effectively negatively affected by removing structurally redundant samples already at high thresholds.

We also found that the model with a similarity threshold of 95% (denoted AEScore_{95} hereafter) seems to perform slightly better than the model trained by only removing the "hard overlap". This could be attributed to the removal of some inconsistencies in the training set, introduced by experimental errors, or simply to the variability of the training procedure (mini-batches, dropouts,

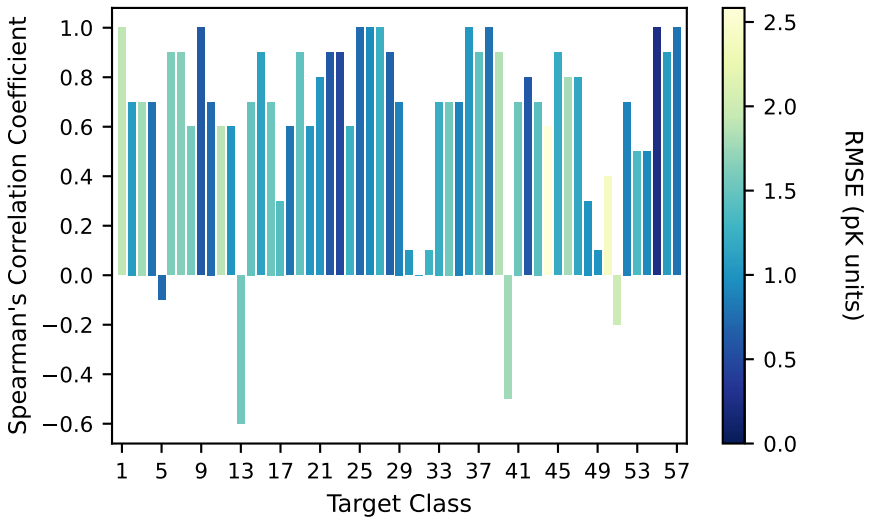


Figure 6.6: Per-class Spearman's correlation coefficient colour-coded by RMSE in pK units, for the 57 classes of the CASF-2016 data set.

etc.). The AEScore_{95} model is our best performing model on the CASF-2016 test set (Pearson's correlation coefficient of $0.83 \in [0.79, 0.86]_{\text{CI }90\%}$, RMSE of 1.22 pK units) and it performs very well compared to other SOTA SFs (see discussion of Fig. 6.9)—although differences with other SOTA methods might not be statistically significant.

Ranking Power

Our SF, AEScore, has an average Spearman's correlation coefficient of $0.64 \in [0.54, 0.71]_{\text{CI }90\%}$. This is similar to the best classical SF evaluated in the CASF-2016 (Su, Yang, et al. 2018), although it is within the 90% confidence interval. The same observation remains true for the average Kendall's correlation coefficient of $0.55 \in [0.47, 0.62]_{\text{CI }90\%}$ and for the PI of $0.67 \in [0.58, 0.73]_{\text{CI }90\%}$.

Interestingly, if hydrogen atoms are removed the ranking power does not change. When hydrogen atoms are ignored, the Spearman's correlation coefficient becomes $0.63 \in [0.54, 0.71]_{\text{CI }90\%}$, the Kendall's correlation coefficient becomes $0.56 \in [0.48, 0.63]_{\text{CI }90\%}$, and the PI becomes $0.66 \in [0.57, 0.74]_{\text{CI }90\%}$.

Fig. 6.6 shows per-class Spearman's rank-correlation coefficients, while per-class Kendall's correlation coefficients are reported in Fig. G.6. For Spearman's correlation coefficients we now have four classes with negative correlation. For Kendall's correlation coefficients we have only three classes with negative correlation. A few other classes have no correlation.

Docking Power

AEScore has been developed with the intent of predicting the binding affinity of a given protein-ligand complex. However, SFs can also be used to determine correct binding poses. Therefore, we evaluate the docking power of AEScore using the docking decoys provided in the CASF-2016 data set (Su, Yang, et al. 2018).

If we consider a correct binding pose as one with a RMSD from the crystallographic binding mode that is smaller than 2 Å, we can define the docking success rate as the percentage of targets with a good pose ranked among the top one, top two, or top three poses.

AEScore has a success rate of $35.8\% \in [30.9\%, 40.4\%]_{90\%\,\mathrm{CI}}$ for the top one pose, a success rate of $54.4\% \in [48.8\%, 58.6\%]_{90\%\,\mathrm{CI}}$ for the top two poses and a success rate of $60.4\% \in [54.7\%, 64.2\%]_{90\%\,\mathrm{CI}}$ for the top three poses. Such low success rates are comparable with the worst classical SFs evaluated on the CASF-2016 benchmark (Su, Yang, et al. 2018). This low success rate is also observed with AK-score, a CNN-based SF, which reports a top one success rate of 34.9% (single model) or 36.0% (ensemble of models) (Kwon et al. 2020).

These results are not surprising, since AEScore has been trained to predict the experimental binding affinity given a protein-ligand complex and has therefore never been exposed to high-RMSD binding poses (decoys). To use the SF to determine low-RMSD poses one has to train for such task. One way to train a SF for docking is to train a pose classifier—distinguishing low RMSD poses from high RMSD poses—, but this requires a change in the model architecture. Another way to tailor a ML SF for docking is to train on docking scores as done for AGL-Score (Nguyen and Wei 2019). A third way to improve binding affinity predictions while retaining the good docking and screening power of some classical SFs is to use Δ-learning (Wang and Zhang 2016). In this work, we explore the latter approach.

6.3.2 Δ-AEScore

Δ-Learning with AutoDock Vina

The use of AEVs combined with a collection of feed-forward NNs has proven successful to predict protein-ligand binding affinities on the CASF-2013 and CASF-2016 benchmark data sets using exclusively elements and atomic coordinates, as demonstrated above. Unfortunately, the results of the docking power test were expectedly deceiving. However, it has been previously demonstrated that a Δ-learning approach can retain the good screening power of a SF while improving the performance in the docking and screening power tests (Wang

and Zhang 2016).

In the Δ-learning approach, a classical SF is used to obtain a crude prediction of the binding affinity, which is subsequently corrected with a ML or DL SF. If corrections to the AutoDock Vina SF can be learned by our model, combining such corrections with the docking power of AutoDock Vina would provide a SF with both good scoring and docking powers (Wang and Zhang 2016).

To combine AutoDock Vina and the experimental data of PDBbind, AutoDock Vina scores, S, are converted to pK values using

$$pK = -\log_{10}\left(e^{\frac{S}{RT}}\right), \tag{6.6}$$

where $T = 295\,K$ and R is the ideal gas constant.

Scoring Power

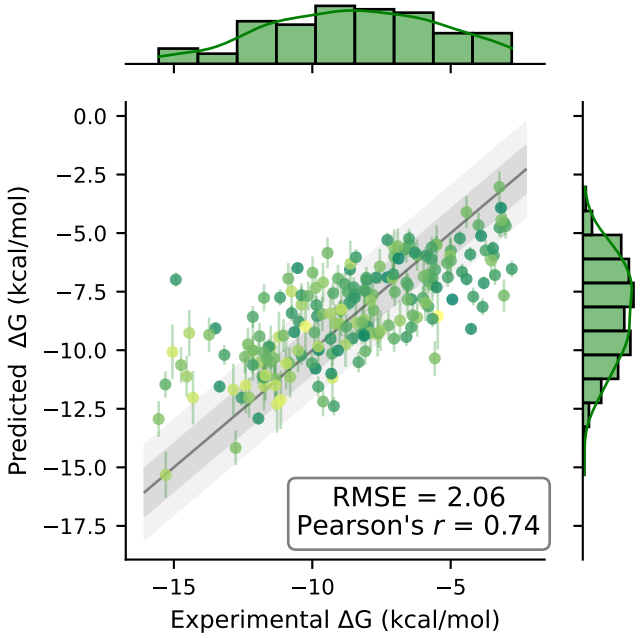
Fig. 6.7 shows the predictions of our model versus the experimental values of the binding affinity for the CASF-2013 and CASF-2016 benchmark data sets. Δ-AEScore achieves an RMSE of 1.53 pK units and a Pearson's correlation coefficient of $0.74 \in [0.67, 0.78]_{CI\,90\%}$ on the CASF-2013 test set and an RMSE of 1.34 pK units and a Pearson's correlation coefficient of $0.79 \in [0.75, 0.82]_{CI\,90\%}$ on the CASF-2016 test set. The performance is slightly worse than that of AEScore, indicating that corrections to the AutoDock Vina native SF are also difficult to learn. This is probably caused by the approximate nature of classical SFs.

Tab. 6.1 compares our Δ-learning results on the CASF-2013 and CASF-2016 data sets with the Δ_{vina}RF SF, arguably the most successful implementation of this approach (Wang and Zhang 2016). Our model performs better than Δ_{vina}RF on the CASF-2013 data set and comparably on the CASF-2016.

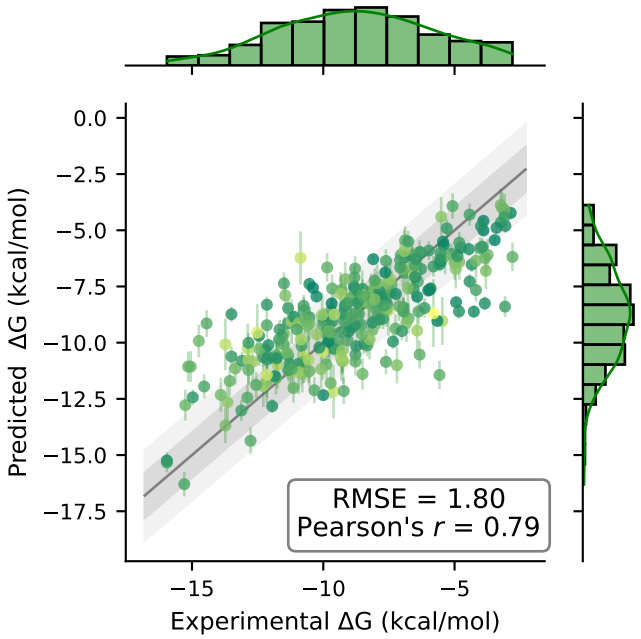
It is worth noting that Δ_{vina}RF is the best SF on the scoring and ranking power tests for the CASF-2016 benchmark, and it is ranking consistently among the top SFs for the docking and screening power tests. However, Δ_{vina}RF is trained on protein-ligand complexes from the PDBbind 2013, which overlaps with ~50% of the CASF-2016 test set. Therefore, its performance might have been artificially enhanced by the large overlap between the training and test sets (Su, Yang, et al. 2018). Both Δ_{vina}RF and Δ-AEScore outperform the classical SF AutoDock Vina in the scoring power test, by a large margin (Su, Yang, et al. 2018).

Ranking Power

In terms of ranking power Δ-AEScore has a Spearman's correlation coefficient of $0.59 \in [0.47, 0.68]_{90\%\,CI}$, a Kendall's correlation coefficient of $0.52 \in$



(a) CASF-2013



(b) CASF-2016

Figure 6.7: Predicted versus experimental binding affinities using the Δ-learning approach with Δ-AEScore, expressed in kcal mol^{-1}. Data points are colour-coded according to their corresponding standard deviation.

Table 6.1: Performance of Δ-AEScore compared to the Δ_{vina}RF for affinity prediction on the CASF-2013 and CASF-2016 benchmarks. For Δ-AEScore the "hard overlap" between the training and both test sets is removed while for Δ_{vina}RF only the "hard overlap" between the training set and CASF-2013 is removed. This leads to artificially inflated results for Δ_{vina}RF on CASF-2016. The best performance for each test set is underlined. RMSE values are given in pK units.

Model	Training Set	Test Set	RMSE	Pearson's r
Δ-AEScore†	Refined 2013	CASF-2013	1.53	<u>0.74</u>
Δ-AEScore†(no H)	Refined 2013	CASF-2013	<u>1.52</u>	<u>0.74</u>
Δ_{vina}RF	Refined 2013	CASF 2013	—	0.69
Vina (optim)	—	CASF-2013	1.82	0.61
Δ-AEScore†	Refined 2016	CASF-2016	1.34	0.79
Δ-AEScore†(no H)	Refined 2016	CASF-2016	1.32	0.80
Δ_{vina}RF	Refined 2013	CASF 2016	—	<u>0.81</u>
Vina (optim)	—	CASF-2016	1.75	0.59

† This work.

$[0.42, 0.60]_{90\%\,CI}$ and a PI of $0.61 \in [0.49, 0.69]_{90\%\,CI}$ on the CASF-2016 benchmark. For the CASF-2013 benchmark, Δ-AEScore has a Spearman's correlation coefficient of $0.61 \in [0.47, 0.71]_{90\%\,CI}$, a Kendall's correlation coefficient of $0.58 \in [0.44, 0.67]_{90\%\,CI}$ and a PI of $0.63 \in [0.49, 0.73]_{90\%\,CI}$.

The performance of Δ-AEScore in the ranking power test is lower than the performance of AEScore. This is to be attributed to the poor performance of AutoDock Vina on this benchmark, with a Spearman's correlation coefficient of $0.53 \in [0.43, 0.61]_{90\%\,CI}$ on the CASF-2016 benchmark. However, the use of AEScore on top of AutoDock Vina allows us to improve the performance of the latter in both scoring and ranking.

Docking Power

We next wanted to see if the corrections to the AutoDock Vina SF can be applied in the context of docking. Using the docking decoys of the CASF-2016 benchmark data set we obtain a top one success rate of $85.6\% \in [81.1\%, 88.1\%]_{90\%\,CI}$, a top two success rate of $94.4\% \in [90.9\%, 95.8\%]_{90\%\,CI}$ and a top three success rate of $95.8\% \in [92.6\%, 96.8\%]_{90\%\,CI}$. This is a very significant improvement on the previous results obtained with AEScore.

The top one performance is lower than AutoDock Vina itself, which performs extremely well in this benchmark with a top 1 success rate of $90.2\% \in [86.7\%, 92.6\%]_{90\%\,CI}$ (when the native ligand binding pose is included), and compared to the performance of Δ_{vina}RF, the second-best performing SF in CASF-2016 with a top 1 success rate of $89.1\% \in [85.6\%, 91.6\%]_{90\%\,CI}$ (Su, Yang,

et al. 2018). However, the much higher performance compared to AEScore indicates that the protein-ligand binding site representation and the model architecture used for AEScore are amenable to Δ-learning. We thus have good scoring power—significantly better than AutoDock Vina alone—while retaining the excellent docking power of AutoDock Vina.

Screening Power

Given the good success rate of Δ-AEScore in the docking power test, we wanted to evaluate Δ-AEScore in the context of virtual screening as well. The screening power test assesses the ability of a SF to identify true binders among a large pool of decoys.

For forward screening, Δ-AEScore ranks the best ligand among the top 1% of candidates with a success rate of $19.3\% \in [10.5\%, 26.3\%]_{90\%\,\mathrm{CI}}$. The top 5% success rate and the 10% success rates are $49.1\% \in [36.8\%, 57.9\%]_{90\%\,\mathrm{CI}}$ and $54.4\% \in [42.1\%, 63.2\%]_{90\%\,\mathrm{CI}}$, respectively. The top 1% success rate is rather low compared to AutoDock Vina ($29.8\% \in [19.3\%, 38.6\%]_{90\%\,\mathrm{CI}}$) and Δ_{vina}RF ($42.1\% \in [29.8\%, 50.9\%]_{90\%\,\mathrm{CI}}$), but top 5% and top 10% performances are in line with Δ_{vina}RF and better than AutoDock Vina itself (Su, Yang, et al. 2018). Again, it is worth re-iterating that the reported performance of Δ_{vina}RF on CASF-2016 might be artificially inflated by the overlap between training and test sets (Su, Yang, et al. 2018).

Another quantitative metric of the screening power is the enrichment factor (EF). Δ-AEScore has an average $\mathrm{EF}_{1\%}$ of $6.16 \in [4.14, 8.75]_{90\%\,\mathrm{CI}}$, an average $\mathrm{EF}_{5\%}$ of $3.76 \in [2.94, 4.63]_{90\%\,\mathrm{CI}}$, and an average $\mathrm{EF}_{10\%}$ of $2.48 \in [2.02, 3.00]_{90\%\,\mathrm{CI}}$. The EF are not too far from AutoDock Vina's EF on CASF-2016, with an $\mathrm{EF}_{1\%}$ of $7.7 \in [5.37, 10.97]_{90\%\,\mathrm{CI}}$ (Su, Yang, et al. 2018). Δ_{vina}RF is again among the top performing SFs on CASF-2016, notwithstanding the training/testing caveats discussed above; Δ_{vina}RF $\mathrm{EF}_{1\%}$ is $11.73 \in [8.84, 15.41]_{90\%\,\mathrm{CI}}$ (Su, Yang, et al. 2018).

For reverse screening on the CASF-2016 benchmark, we obtain a top 1% success rate of $11.9\% \in [8.8\%, 15.1\%]_{90\%\,\mathrm{CI}}$, a top 5% success rate of $19.3\% \in [15.4\%, 23.2\%]_{90\%\,\mathrm{CI}}$ and a top 10% success rate of $27.0\% \in [22.5\%, 30.9\%]_{90\%\,\mathrm{CI}}$. Again, the results are similar to AutoDock Vina ($13.7\% \in [10.5\%, 16.8\%]_{90\%\,\mathrm{CI}}$) and slightly worse than the optimistic values reported for Δ_{vina}RF—$15.1\% \in [11.6\%, 18.6\%]_{90\%\,\mathrm{CI}}$ (Su, Yang, et al. 2018).

6.3.3 Ligand-Only Affinity Prediction

To test the effect of protein information in the binding affinity prediction and to elucidate possible biases in the data set (Sieg et al. 2019), we also trained a

model with only ligand atoms ($d = 0$ Å). The AEVs parameters used to describe ligand atoms are left unchanged.

For the CASF-2013 data set, we obtained an RMSE of 1.65 pK units and a Pearson's correlation of 0.70, while for the CASF-2016 data set we obtained an RMSE of 1.49 pK units and a Pearson's correlation of 0.74 (when only protein and ligand atoms are kept and systems are automatically protonated). Fig. 6.8 also reports the results when hydrogen atoms are removed and when the model is trained on a data set with a protein/ligand/pocket similarity threshold of 95% similarity with the training set.

As shown in Fig. 6.8 (and, equivalently, in Tab. G.3), the performance of the model in absence of protein atoms (L) is always worse than that obtained when including both ligand and protein atoms (P + L). This indicates that the model can exploit the additional information about the binding site provided by the protein atoms to improve binding affinity predictions. However, the difference is not as striking as one might expect.

The same observations apply to the Δ-learning approach, although the difference between protein-ligand (P + L) and ligand-only (L) models is even less pronounced. This suggests that corrections to the AutoDock Vina SF mainly stem from the information about the ligand and that information about the protein target plays a minor role.

The fact that AEScore models using only information about the ligand already perform well is in line with recent work from Boyles et al. (2019) who showed that ligand features alone are predictive of the mean protein-ligand binding affinity in PDBbind (Boyles et al. 2019). Additionally, ligand information plays a significant role in affinity prediction in DL models as well (Chen, Cruz, et al. 2019; Francoeur et al. 2020; Wallach and Heifets 2018). For ligand-only predictions, AEScore is essentially learning a conformation-dependent fingerprint (FP) of the active ligand and using such information to predict the mean binding affinity of said ligand. RDKit descriptors alone, combined with a random forest (RF) model, can already achieve a Pearson's correlation coefficient of 0.71 on CASF-2013 and of 0.76 on CASF-2016, as demonstrated by Boyles et al. (2019). Our results suggest that the AEScore model presented here can use AEVs as 3D ligand FPs and use such information to predict the average binding affinity of a ligand in the same way RDKit descriptors allow.

Work parallel to ours recently investigated the application of Smooth Overlap of Atomic Positions (SOAP) (Bartók, Kondor, et al. 2013)—another widely used and related structural representation for molecules and materials (Musil et al. 2021)—for 3D QSAR (McCorkindale et al. 2020). The method is shown to perform competitively with FP-based methods as well as SOTA graph neural networks (GNNs).

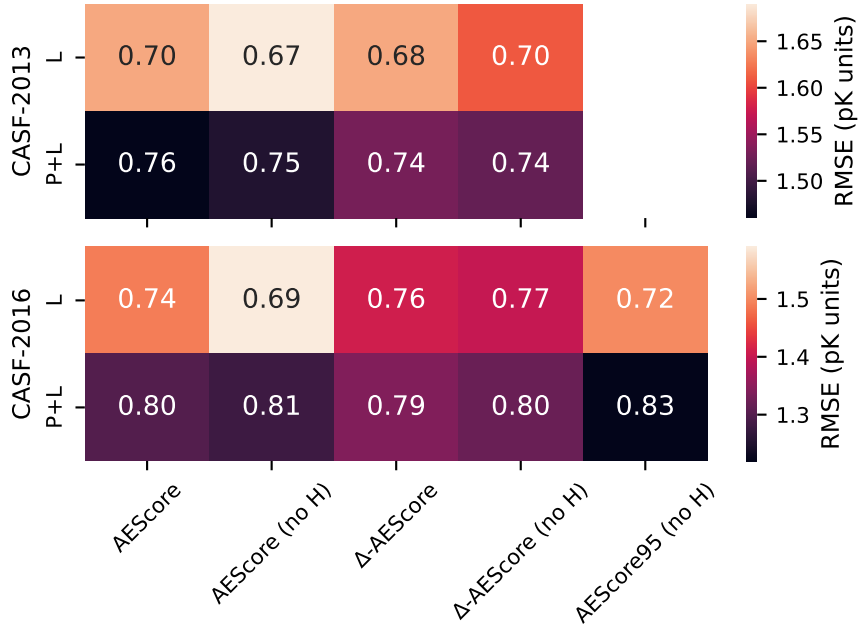


Figure 6.8: Pearson's correlation coefficients for different models incorporating atoms of the protein and the ligand (P + L, $d = 3.5$ Å) or only atoms of the ligand (L), for the CASF-2013 and CASF-2016 benchmarks. Each box is colour-coded by the corresponding RMSE (in pK units).

6.4 Discussion

Fig. 6.9 compares the performance of our model—denoted AEScore—in terms of binding affinity prediction for the CASF-2013 and CASF-2016 benchmark data sets with other SOTA ML and DL models. The performance of the other methods is taken directly from the original references. The same results are also reported in Tab. G.4, together with root mean squared errors (RMSEs) and additional information about models and training data sets. A more up-to-date and extended table is included in Meli, Morris, et al. (2022).

In the literature, there is some confusion about the CASF benchmark and the PDBbind Core set, as indicated on the PDBbind website (`pdbbind.org.cn`, last accessed October 1, 2022). In Tab. G.4 we indicate which data set has been used for testing. The CASF-2016 benchmark set contains 285 protein-ligand complexes while the PDBbind Core 2016 set contains 290 protein-ligand complexes.[5]

Our results compare favourably with other SOTA DL models based on feed-forward NNs, CNNs, and ML SFs on both the CASF-2016 and PDBbind Core 2016 test sets. However, a quantitative and statistically sound comparison with other methods is somewhat difficult because error bars and confidence intervals are often not reported.

One of the main advantages of the AEVs-based approach is that it is translationally and rotationally invariant, thus removing an additional source of variability. This is not the case for SFs based on standard CNNs, where random translations and rotations of the input protein-ligand systems give different results. Fig. G.7 shows the variation in CNN-based predictions as a function of the angle of rotation for a particular complex. Data augmentation with random translations and rotations has proved to be essential to prevent overfitting and significantly improve training in CNN-based SFs (Ragoza, Hochuli, et al. 2017), but this is computationally expensive.

In addition to being translationally and rotationally invariant, AEVs also require minimal information about the system. Only elements and atomic coordinates are needed by the model. Other methods often require additional information such as FF parameters or specific atom types and are therefore limited by these parameters and underlying assumptions.

Compared to "classical" ML SFs, our method performs similarly to RF Score and other RF-based SFs (Afifi and Al-Sadek 2018; Ballester and Mitchell 2010a). Despite recent advances in DL architectures, which consistently outperform "classical" ML algorithms in image recognition and natural language processing (Graves et al. 2013; Hinton et al. 2012; LeCun et al. 2015), RFs remain very

[5] Complexes 4MRW, 4MRZ, 4MSN, 5C1W, 4MSC, and 3CYX in PDBbind Core 2016 are not included in CASF-2016, while 1G2K is an additional complex not present in the Core set, according to Nguyen and Wei (2019).

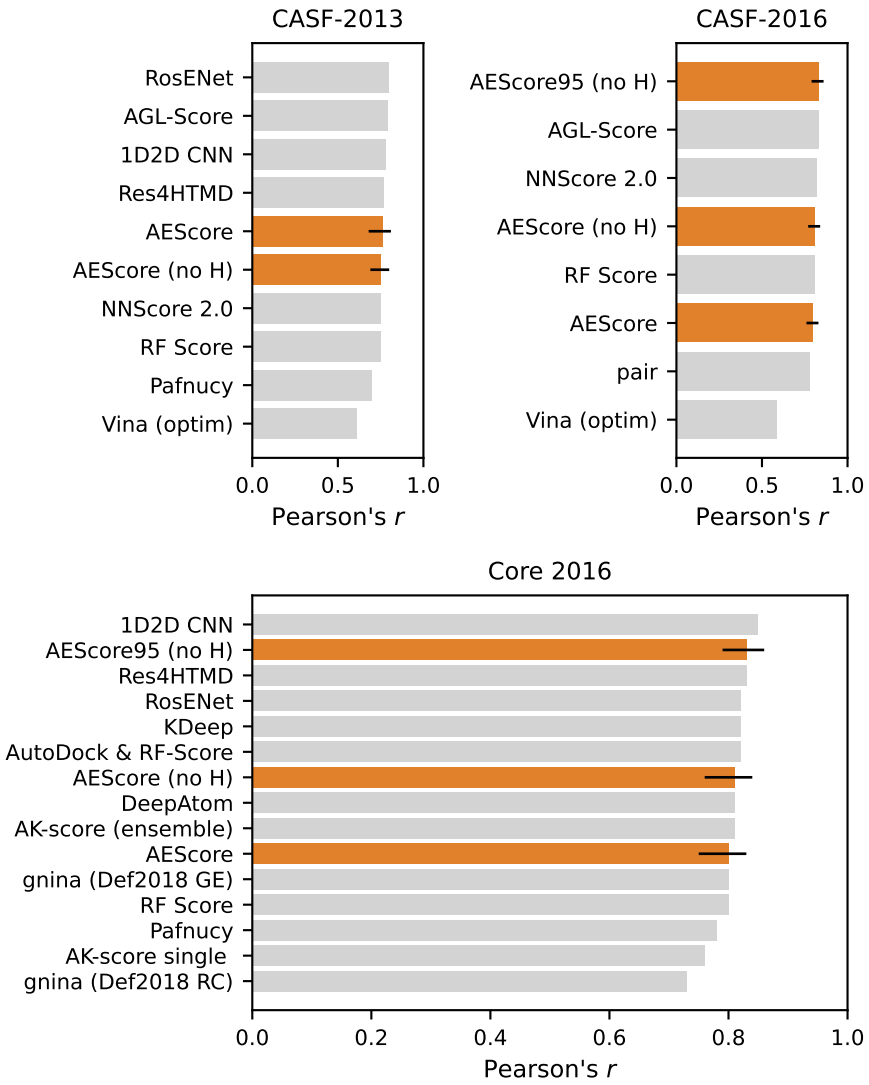


Figure 6.9: Performance of different ML and DL models for binding affinity prediction on the CASF-2013 and CASF-2016 benchmarks as well as for the Core 2016 set. Our results, shown in orange, include 90% confidence intervals. Numerical values for Pearson's correlation coefficients and the RMSE are reported in Tab. G.4, together with references for all the different methods.

competitive for binding affinity predictions. All top-performing ML and DL methods considered here achieve similar performance on the CASF benchmark—as measured by Pearson's correlation coefficient. This is likely due to the fact that errors in the experimental measurements of the binding affinity and the X-ray crystallographic coordinates of the protein-ligand complex set a theoretical upper limit on the maximal performance of SFs trained on such noisy data (Liu, Su, et al. 2017).

It is instructive to also compare the performance of our model with standard docking SFs. Here we used the AutoDock Vina (Trott and Olson 2009) SF as implemented in Smina (Koes, Baumgartner, et al. 2013) as a baseline. We see that our model outperforms the AutoDock Vina SF for protein-ligand binding affinity predictions, as do other ML and DL approaches. This is expected since previous studies show that standard SFs do not perform well in scoring and ranking power tests (Liu, Su, et al. 2017).

The removal of the systems in the CASF test set from the PDBbind Refined set used for training is common practice with ML and DL SFs and therefore ensures a fair comparison with other methods. However, it has been previously noted that the performance on the CASF set is not necessarily very indicative of a model's ability to generalise, since this data set samples the same regions of the chemical and target spaces as the PDBbind data set (Francoeur et al. 2020; Su, Feng, et al. 2020). To better evaluate the ability of a model to generalise, we tested its performance when trained on a recently developed non-redundant training set (Su, Feng, et al. 2020). We showed in Fig. 6.5 that the performance of AEScore deteriorates gradually when the similarity between the training set and the test set is reduced, in contrast with other ML SFs that are severely inhibited by removing structurally redundant samples from the training set (Su, Feng, et al. 2020).

When we tested AEScore for docking power we obtained poor results. This is not surprising since the model was trained to predict binding affinities given the correct binding pose, and it was not trained explicitly to distinguish low- from high-RMSD poses. However, we showed that combining AEScore with the classical SF AutoDock Vina using a Δ-learning approach improves the performance in terms of docking and screening while maintaining good scoring and ranking performance. As already demonstrated by Δ_{vina}RF, this is a good approach for developing a SF that works well on all four tasks: scoring, ranking, docking, and screening. Usually, ML and DL SFs work very well for scoring but not as well for docking and virtual screening, while classical SFs have the opposite behaviour. Fig. 6.10 collects most of the results of AEScore and Δ-AEScore on the CASF-2016 benchmark, together with the results for Δ_{vina}RF and AutoDock Vina (our baseline) as reported by Su, Yang, et al. (2018).

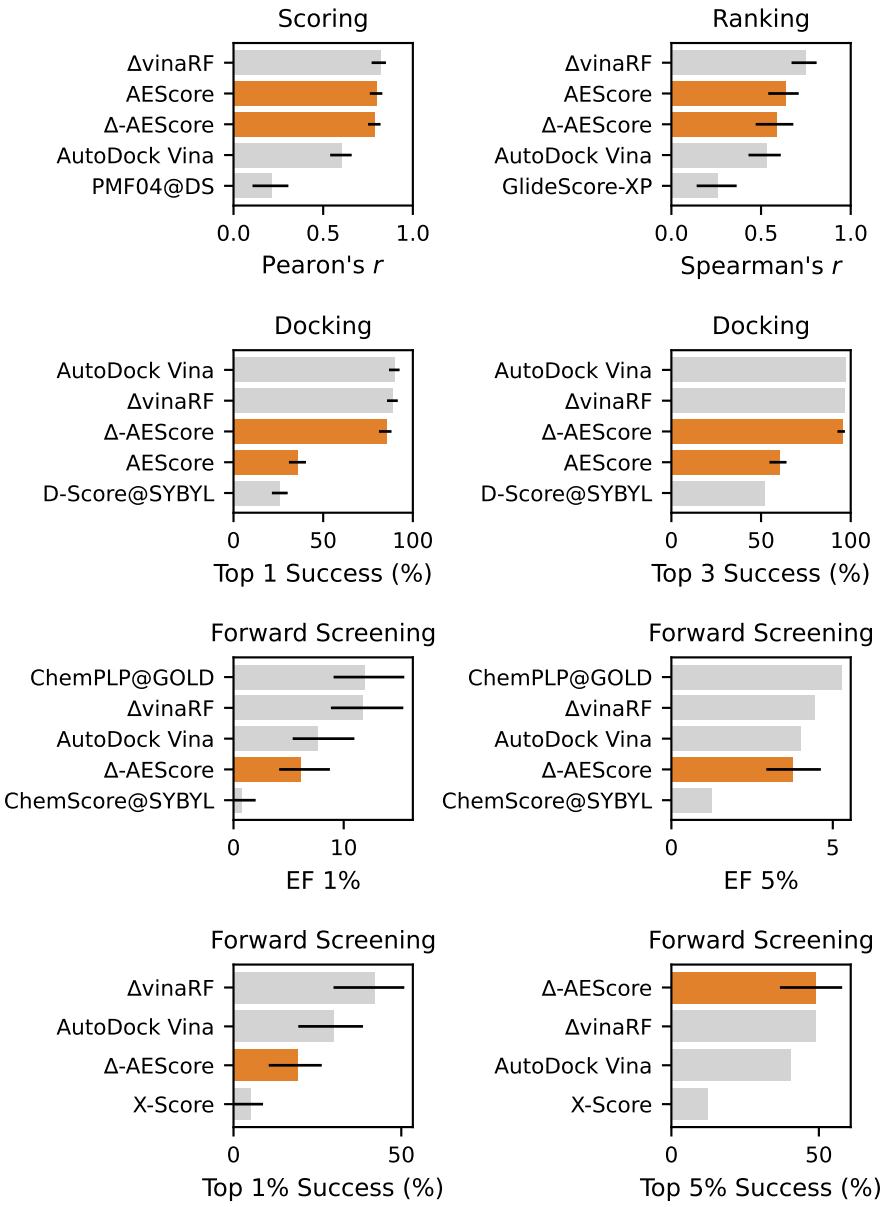


Figure 6.10: Performance of AEScore, Δ-AEScore, Δ_{vina}RF, and AutoDock Vina. The best- and worst-performing SFs evaluated as part of CASF-2016 are also added for comparison. The results include 90% confidence intervals (where they were available).

We also added the best- and worst-performing SFs for each of the CASF-2016 benchmarks reported in Su, Yang, et al. (2018), whenever these SFs were different from Δ_{vina}RF or AutoDock Vina. We see that both AEScore and Δ-AEScore perform well in scoring and ranking power tests, but AEScore performance for docking is low. However, the Δ-learning approach can recover a good docking power (similar to the AutoDock Vina baseline) while retaining a good performance in scoring and ranking. The performance of Δ-AEScore in forward screening is rather poor as measured by EF 1% or top 1% success rate but greatly improves for EF 5% and the top 5% success rate.

Given the good performance of our ligand-only model—which was nonetheless consistently worse than that of the protein-ligand model—it is clear that the model is extracting a lot of information from the ligand. Finding strategies to force the model to rely more on protein information could further improve the model and make it more transferable. This is a known problem (Chen, Cruz, et al. 2019; Sieg et al. 2019; Wallach and Heifets 2018) and strategies to force the model to rely more on the protein structure are an active area of research (Scantlebury et al. 2020).

The advantage of using an end-to-end differentiable model is that the gradient of the SF with respect to the input parameters can be readily obtained by backpropagation. Since the TorchANI `AEVComputer` is fully differentiable and its inputs are atomic coordinates (Gao, Ramezanghorbani, et al. 2020), the gradient of the SF with respect to atomic coordinates can be computed. This can be used for visualisation or for local geometry optimisation of the binding pose (Ragoza, Turner, et al. 2017).

Finally, it is worth noting that we exploited the representation and architecture commonly used to develop NNP to predict a different endpoint, namely the protein-ligand binding affinity, and corrections to classical SFs. However, given the success of NNPs (Smith, Isayev, et al. 2017; Smith, Nebgen, Lubbers, et al. 2018) one could use them in a MM/PBSA- or MM/GBSA-style approach (Genheden and Ryde 2015) to directly compute the free energy of binding on more physical grounds. Approaches to combine NNP with MD for drug discovery applications are already starting to appear (Cole et al. 2020; Lahey and Rowley 2020; Rufa et al. 2020).

6.5 Conclusions

We demonstrated that AEVs are a promising representation of the protein-ligand binding site (and of the ligand alone, for ligand-based model) amenable to ML-based predictions of the protein-ligand binding affinity, and corrections

to classical SFs. This representation is rotationally and translationally invariant and, in contrast to CNN-based SFs, does not require data augmentation. The results reported here for AEScore show similar or better performance than other SOTA ML and DL methods on the CASF-2013 and CASF-2016 benchmarks (as well as the Core 2016 set) in binding affinity prediction.

One of the major shortcomings of our model, however, is the over-reliance on ligand features as demonstrated by the good performance of the ligand-only model. This is a common problem with ML and DL SFs (Boyles et al. 2019; Chen, Cruz, et al. 2019; Francoeur et al. 2020; Wallach and Heifets 2018) and better strategies to force the model to rely more on protein and ligand atoms involved in binding need to be developed.

Using training sets with decreasing similarity to the test set, first introduced by Boyles et al. (2019) and later by Su, Feng, et al. (2020), we showed that our model is not completely hindered by the removal of systems with high similarity, but that AEScore's performance deteriorates only gradually. This is in contrast with other ML and DL SFs, where a performance drop is observed as soon as a similarity threshold is introduced (Boyles et al. 2019; Su, Feng, et al. 2020). This property could be useful in real drug discovery applications, where data on similar or related systems (such as a congeneric series of ligands) is acquired gradually.

In this work, we did not optimise the ANI parameters for radial and angular symmetry functions, and we did not explore the full flexibility of the angular symmetry functions. Bayesian optimisation of ACSFs' hyperparameter space could lead to further improvements of the SF.

We also showed that the AEScore model presented here can be exploited in tandem with standard docking SFs using a Δ-learning approach, to improve the performance in docking and virtual screening (in which AEScore does not perform well, since it has not been explicitly trained for such task). Δ-AEScore outperforms the Δ_{vina}RF SF by a good margin on the CASF-2013 test set and performs similarly on the CASF-2016 test set (notwithstanding the training/test set overlap in Δ_{vina}RF reported performance). Δ-learning has the advantage of partially retaining the good docking and screening power of standard SFs while improving affinity predictions using machine-learned corrections, allowing the development of a SF that works reasonably well on all four tasks of early-stage structure-based drug discovery applications.

This page intentionally contains only this sentence.

7

Exploring 3D Generative Models for Structure-Based Drug Design

Contents

The work presented in this chapter was initially carried out as part of an iCASE internship with Evotec, under the direct supervision of Lionel Colliandre and Dimitar Hristozov, and continued as a collaboration. Most of the code related to this chapter is available at RMeli/densitymatch and RMeli/generative3d.

In the previous chapters, we discussed discriminative models for SBDD, where given a protein-ligand pair the goal is to determine if the ligand is active/inactive against the given target, if a binding pose is near-native or not, or to predict the binding affinity. However, VS based on active/inactive classifiers or regression of the binding affinity has a major limitation: only the compounds already contained in the screening library are tested. Additionally, many compounds contained in large libraries are most likely irrelevant to the target of interest.

In this chapter, we discuss and explore generative models for SBDD. In particular, the goal is to assess the performance and limitations of two recently proposed models for the one-shot generation of 3D ligand conformations, based on atomic densities (Masuda et al. 2020; Ragoza, Masuda, et al. 2020). Originally, the goal of the project was to use the generative model of Masuda et al. (2020) to generate novel and interesting inhibitors against targets of interest, and subsequently evaluate the affinity of such compounds with GNINA to prioritise them for free energy perturbation (FEP) calculations. Unfortunately, as we will discuss in detail, most generated molecules are problematic. Therefore, we ended up exploring different methods to circumvent these problems.

7.1 Generative Models in Chemistry

The chemical space—the ensemble of all possible molecules—is huge. A commonly reported estimate of the number of drug-like molecules (below 500 Da) is 1×10^{60} (Reymond et al. 2010), and the subset of the drug-like chemical space with good pharmacokinetic properties is estimated to contain 2×10^6 molecules (Drew et al. 2011). Therefore, only a very small fraction of potential drug candidates have been discovered.

One method to discover new active compounds against a target of interest is HTS (Bajorath 2002). Unfortunately, HTS is expensive and, therefore, screening libraries remain rather small. A less expensive way to suggest new active compounds is to perform *in silico* HTS (or VS). However, both experimental and computational methods for screening compound libraries have an underlying limitation: only the compounds contained in the HTS or VS libraries are considered. Since it is impossible to enumerate the whole drug-like chemical space, this means that some compounds are not included and will never be tested. Additionally, many compounds contained in large screening libraries are likely to be irrelevant to the target of interest.

The idea behind generative models is that one can use SOTA ML and DL methods to generate new molecules, having desirable properties conditioned by

the problem at hand. Therefore, they allow the exploration of chemical space in a focused and efficient way. This is more advantageous than screening large compound libraries, in which many compounds might be irrelevant. *De novo* design is not a new idea, but recently a lot of effort has been devoted to develop and test new ML and DL generative models (Mouchlis et al. 2021).

Generative models in chemistry use one-dimensional, two-dimensional, or three-dimensional molecular representations, and recent DL methods use different architectures. Section 3.5 gives a brief introduction about VAEs and GANs, the architectures considered in this chapter.

One-dimensional representations are often based on the simplified molecular-input line-entry system (SMILES) (Weininger 1988; Weininger et al. 1989), although alternative representations have been suggested (Krenn et al. 2020; O'Boyle and Dalke 2018). Gómez-Bombarelli et al. (2018) used a SMILES representation of molecules combined with RNN-based and CNN-based VAEs to build a continuous (latent) representation of molecules that can be used for gradient-based optimisation of molecular properties. Popova et al. (2018) used reinforcement learning (RL) to fine-tune a generative and a predictive model, biasing the generation of novel compounds towards desired properties. Unfortunately, SMILES-based generative models are prone to generate invalid SMILES, and validity is often only assessed by valency rules therefore generated molecules might well be unstable (Bilodeau et al. 2022).

Two-dimensional representations often consist of molecular graphs. Molecular graphs can be generated in one shot by defining the atoms and the adjacency matrix directly (De Cao and Kipf 2018; Simonovsky and Komodakis 2018), or they can be generated atom-by-atoms or fragment-by-fragment (Jin et al. 2018; Li, Vinyals, et al. 2018; You et al. 2018).

Three-dimensional representations involve point clouds with 3D coordinates (Gebauer et al. 2022; Simm, Pinsler, et al. 2020; Simm and Hernández-Lobato 2019)—where each point in the point cloud represents an atom—or fictitious atomic densities such as the one used by Gnina (Ragoza, Masuda, et al. 2020; Skalic, Jiménez, et al. 2019). Three-dimensional representations are particularly interesting for drug discovery applications since it is often possible to incorporate information about the protein target of interest, to bias the generation of new inhibitors (Imrie, Hadfield, et al. 2021; Masuda et al. 2020; Ragoza, Masuda, et al. 2022; Skalic, Sabbadin, et al. 2019). Applications of three-dimensional generative models include the generation of property-matched decoys (Imrie, Bradley, and Deane 2021), and fragment elaboration (Hadfield, Imrie, et al. 2022). Unfortunately, while several structure-based generative models use a three-dimensional representation as input, the output is often a SMILES string which does not provide any information about the binding modes.

Recent reviews of *de novo* design and generative models in chemistry are provided by Meyers et al. (2021) and Bilodeau et al. (2022), while an interesting review on the use of AI in structure-based compound design is provided by Hadfield and Deane (2022).

7.2 `liGAN`

`liGAN` (Masuda et al. 2020; Ragoza, Masuda, et al. 2020) is a FOSS for training and evaluating deep generative models for *de novo* drug design based on 3D atomic density grids. The generative models implemented in `liGAN` are based on AEs and VAEs used as the generative component of a GAN.

The code combines molecular gridding algorithms from `libmolgrid` (Sunseri and Koes 2020) together with PyTorch automatic differentiation (Paszke et al. 2019), the Gnina fork of Caffe (Jia et al. 2014), and Open Babel (O'Boyle, Morley, et al. 2008) and RDKit (Landrum 2022) for molecular input and reconstruction. With two DL frameworks—one of which is discontinued—and two cheminformatics libraries—one of which can only be installed from source or using the `conda` package manager—, the codebase is rather complex. For this reason, it was necessary to build a Docker container to install all dependencies and ensure reproducibility across different platforms. The Dockerfile used to build the Docker container for `liGAN` is available at RMeli/containers (last accessed October 1, 2022).

While the Docker container is useful to run on a local machine or Evotec's cluster, it presents serious vulnerability issues (escalation of privileges), and therefore it is usually not allowed on University and national HPC clusters. For containerised HPC applications Singularity containers are a better tool since they do not present the same vulnerabilities. For this reason, we also built a Singularity container,[1] also available at RMeli/containers (last accessed October 1, 2022).

In this chapter we evaluate the pre-trained models presented in Ragoza, Masuda, et al. (2020) and Masuda et al. (2020), which are described below.[2] Most of the tests performed here are run on Evotec's HPC cluster with NVIDIA Tesla V100 or NVIDIA GeForce GTX 1080 Ti GPUs.

[1] We thank Sebastien Buchoux, Senior Research Scientist at Evotec, for the help in setting up the Singularity container.

[2] The weights of the pre-trained models presented in Ragoza, Masuda, et al. (2020) and Masuda et al. (2020) were kindly provided by Matthew Ragoza and David Koes. The weights are now freely available at `http://bits.csb.pitt.edu/files/ligan_weights/` (last accessed October 1, 2022).

7.2.1 Adversarial Training for Autoencoders

The calculation of atomic density grids from an input small molecule or protein-ligand complex is described in section 4.2.1. With Eq. (4.2) the molecular input—atomic coordinates and atom types—can be discretised on a series of three-dimensional grids, each encoding spatial information about different atom types. Such information is used as input of an AE or VAE which allows encoding the atomic density grids into a continuous low-dimensional representation (latent space) which can be subsequently decoded into a generated density. Atomic density grids are therefore the input and output of the deep generative models proposed by Ragoza, Masuda, et al. (2020) and Masuda et al. (2020). As for Gnina, the advantage of using atomic densities is that the same architectures used in computer vision can be applied directly.

`liGAN` uses adversarial training to train the AE and VAE models, where the autoencoders are trained as part of a GAN architecture. Essentially, the autoencoders of Fig. 3.3 are used as the generator network in the GAN architecture of Fig. 3.4, while the discriminator network is a standard CNN that learns to classify densities as real or generated.

The loss function used for training the VAEs by Ragoza, Masuda, et al. (2020) and Masuda et al. (2020) has three components:

- $\mathcal{L}_2$ loss between input (real) and output (generated) densities,
- $\mathcal{L}_{\text{GAN}}$ loss,
- KL divergence $\mathcal{D}_{\text{KL}}$ between latent space distribution and a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

The $\mathcal{L}_2$ loss enforces the model to minimise the error between (real) input and (generated) output—the *reconstruction error*—, the GAN loss enforces the model to generate densities from the same distribution as real densities—so that the discriminator network can be fooled—, and the KL divergence $\mathcal{D}_{\text{KL}}$ enforces the latent space distribution to a normal distribution—to allow sampling of the latent space. The AE is trained similarly, but without the KL divergence on the latent space, given that the latter is non-probabilistic.

Given the issues associated with the AEs for generative modelling, we only focus on the evaluation of the VAE models in the following sections.

7.2.2 Fitting Atoms to Densities

As discussed in Chapter 4, generating an atomic density grid from a molecule is straightforward using Eq. (4.2). Unfortunately, such a function is not invertible

and therefore there is no analytical expression to obtain the Cartesian coordinates corresponding to a given atomic density grid.

The problem of reconstructing a molecule from electron densities is a well-known problem in X-ray crystallography (Jones et al. 1991). However, the atomic densities used here do not correspond to experimental electron densities: there is additional information available in the form of different density channels—one for each atom type—, and generated densities might not be ideal, nor correspond to real molecules.

To avoid the problem of reconstructing molecules from generated densities, Skalic, Jiménez, et al. (2019) developed a captioning network composed of a CNN and a RNN to caption the generated atomic densities with a SMILES string representing the corresponding molecule. This DL solution to the problem presents a major drawback: the 3D information conveyed by the atomic densities is lost because the final output only represents the molecular graph. For structure-based generative models this means that generated molecules need to be re-docked into the receptor structure, which requires additional resources and can introduce errors—the docked pose might not correspond to the generated density.

To avoid loss of important and useful 3D information, Ragoza, Masuda, et al. (2020) developed an algorithm to fit atoms directly into generated atomic densities, where every atom type channel is treated separately. Their algorithm combines beam search with gradient descent on atomic coordinates.

For a given guess of the atomic coordinates $\mathbf{R}$, the corresponding atomic density $\mathbf{D}_{\text{fit}}(\mathbf{R})$ can be computed using Eq. (4.2). The atomic positions $\mathbf{R}_{\text{opt}}$ that best fit the target density $\mathbf{D}_{\text{target}}$ are then obtained by solving the optimisation problem

$$\mathbf{R}_{\text{opt}} = \arg\min_{\mathbf{R}} \|\mathbf{D}_{\text{target}} - \mathbf{D}_{\text{fit}}(\mathbf{R})\|^2, \tag{7.1}$$

where $\|\cdot\|^2$ represents the squared norm between the two tensors. `libmolgrid` (Sunseri and Koes 2020) is used to compute $\mathbf{D}_{\text{fit}}(\mathbf{R})$ and the norm of the difference between atomic densities, while PyTorch's automatic differentiation provides the gradient with respect to atomic coordinates by backpropagation, which is used for optimisation. The optimal atomic positions $\mathbf{R}_{\text{opt}}$ are therefore found iteratively by optimising the density overlap between the density corresponding to the current atomic positions and the generated density.

The optimisation of atomic positions is combined with *beam search*, a greedy heuristic search algorithm that expands the most promising candidates while discarding the others. According to Ragoza, Masuda, et al. (2020), the structures—atom types and coordinates—are expanded by detecting atoms in the remaining density, after subtracting the density of the structure from the reference density.

If adding a new atom decreases the loss, the structure is retained. The best structure is used as a starting point for the next iteration, and the process continues until no more atoms improving the loss can be added.

As we shall see in this chapter, the atom fitting procedure developed by Ragoza, Masuda, et al. (2020) is the major bottleneck to the generation of sensible molecular structures. We will attempt to develop new methods to circumvent the problem.

7.2.3 Molecule Reconstruction

The atom fitting algorithm of Ragoza, Masuda, et al. (2020) described above returns atom types—corresponding to the different density channels being fit—and Cartesian coordinates. Further processing is needed to obtain a valid molecule. Bonds between atoms (and their bond order) are inferred using a custom version of Open Babel's bond perception algorithm (O'Boyle, Morley, et al. 2008), which takes into account additional information about atom types.

According to the atom types described in Tab. E.1, the available information for molecular reconstruction includes aromaticity (for carbon atoms only), presence of bonds to hydrogen and heteroatoms, and formal charge.

The reconstructed molecule, complete with information about bonds, is minimised using the universal force-field (UFF) (Rappe et al. 1992) to produce reasonable molecular geometries. This optimisation step is essential since generated molecules are often far from an equilibrium conformation.

7.2.4 Molecule-to-Molecule Pipeline

The full pipeline of `liGAN` is depicted in Fig. 7.1. A molecule is taken as input, the atom types of Tab. E.1 are determined, and atomic density grids for every atom type are computed using Eq. (4.2). The atomic density grids for the original molecule are hereafter referred to as "real densities" as opposed to "generated densities" which are the output of a deep generative model.

The real density can be either used as input of a generative model—to obtain a generated density—or it can be fit directly. Fitting a real density directly allows assessing the performance of the fitting procedure described in section 7.2.2.

Once atoms are fitted to the real or generated atomic density, bonds are added to reconstruct a molecule.

A reconstructed molecule is considered valid if and only if it is composed of a single fragment and raises no error when parsed with RDKit (with sanitisation) (Landrum 2022). It is worth noticing, however, that this criterion is quite loose because the validity of a molecule parsed with RDKit does not imply that

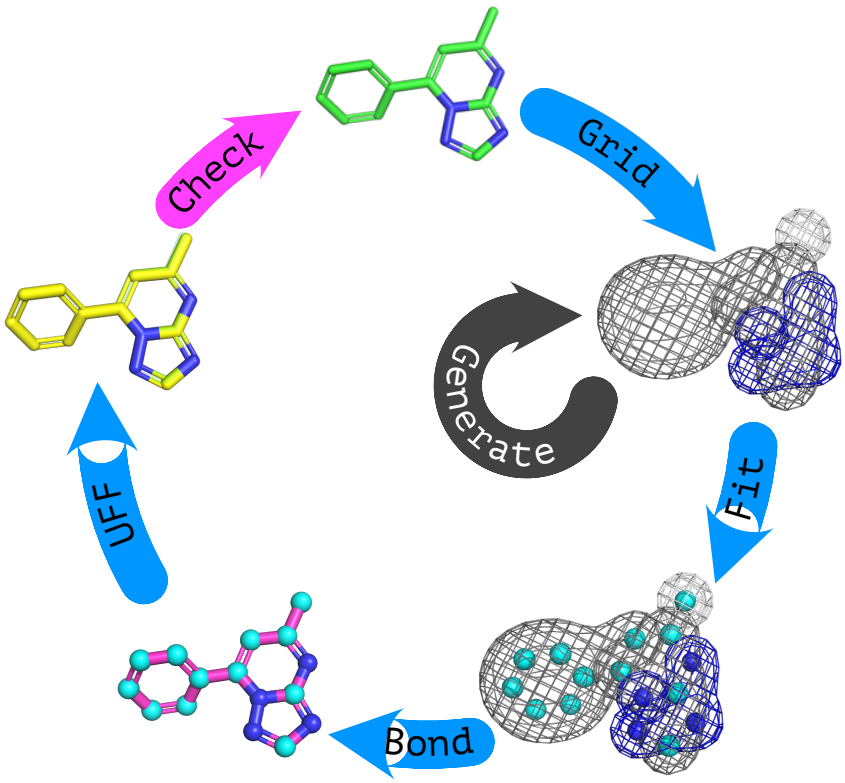


Figure 7.1: Molecule to molecule transformation via atomic density grids. A molecule (top, green) can be converted into an atomic density grid (aromatic carbon in grey, aliphatic carbon in white, and nitrogen in blue) using Eq. (4.2). The grid can be used as input for a deep generative model, which will return a generated density. Atoms are then fit to the original or generated grid to obtain their Cartesian coordinates, as described in section 7.2.2. With atomic positions and atom types at hand, bonds are added, and the molecule is finally optimised using RDKit's UFF.

the molecule is stable or synthetically accessible. Additionally, the molecular reconstruction is designed specifically to produce valid molecules. As we will see, many molecules generated by liGAN have unreasonable and/or highly strained structures, while being "valid".

7.2.5 Ligand Variational Autoencoder

The ligand VAE consists of a standard convolutional encoder/decoder architecture.

The encoder consists of a series of three convolutional layers with a $3 \times 3 \times 3$ kernel (with no stride and a padding of one voxel), each one associated with a leaky rectified linear unit (ReLU) activation function (with a negative slope of 0.1), followed by an average pooling layer with a $2 \times 2 \times 2$ kernel and a stride of two. The convolutional layers in blocks of three have 32, 64, 128, and 256 channels, respectively.

The results of the last convolutional layer (lacking of maximum pooling) are flattened into a one-dimensional vector that is used as input of two single-layer MLPs that produce an average vector and a standard deviation vector, both of dimension 1024. Using the average and standard deviation vectors outputted by the encoder, a latent space vector is sampled using the reparametrisation trick.

The encoder consists in a single-layer MLPs, which upsample the latent space vector, followed by inverse convolution (deconvolution) operations intercalated with nearest-neighbour upsampling. The architecture mirrors the encoder architecture, with blocks of three deconvolution layers with 256, 128, 64, and 32 channels, respectively, each followed by a leaky ReLU activation function.

The actual architecture graph, obtained with Caffe (Jia et al. 2014), is reported in Fig. H.1.

7.2.6 Receptor-Conditional Variational Autoencoder

The architecture of the receptor-conditional VAE is similar to the architecture of the ligand VAE described above. The notable difference is that there are two encoder networks, one encoding the ligand, the other encoding the binding site.

The latent space vector of size 1024 is obtained by concatenating the output of the ligand encoder—which is variational—and the output of the receptor encoder—which is not variational. The first 512 elements of the latent are sampled from a normal distribution with mean and standard deviation given by the ligand encoder, while the last 512 elements represent the encoding of the receptor (binding site).

An additional and very important difference between the ligand VAE of Ragoza, Masuda, et al. (2020) and the receptor-constrained VAE of Masuda et al.

(2020) is the presence of skip connections in the latter model, which allow to directly propagate some information from the receptor encoder to the decoder. The concept of skip connections was initially introduced in ResNET (He et al. 2016)—a deep CNN for image recognition—to address the vanishing gradient problem, and it was subsequently refined with DenseNet (Huang, Liu, Pleiss, et al. 2019). In `liGAN`, feature maps obtained from the first and the second convolution block are concatenated to the tensor representation before the last and the second last deconvolution block. These skip connections enforce the decoder to take into account some feature maps of the receptor—encoding structural information—, so that the resulting generated density is conditional on the binding site.

The actual architecture graph, obtained with Caffe (Jia et al. 2014), is reported in Fig. H.2.

7.2.7 Pre-Trained Models

All models used in this work are from Ragoza, Masuda, et al. (2020) and Masuda et al. (2020). The weights of the ligand VAE and the receptor-conditional VAE were kindly provided by David Koes and Matthew Ragoza, and are now available at `http://bits.csb.pitt.edu/files/ligan_weights/` (last accessed October 1, 2022).

In Ragoza, Masuda, et al. (2020) (ligand VAE) the training set consists in a subset of the commercially available stock compounds from the MolPort database. Molecules whose atoms could not be typed according to Tab E.1 were discarded (Ragoza, Masuda, et al. 2020). The data set was augmented using RDKit's conformer generator (Landrum 2022; Riniker and Landrum 2015) to obtain up to 20 different conformers for a given molecule (Ragoza, Masuda, et al. 2020).

In Masuda et al. (2020) (receptor-conditional VAE) the training set consists of low-RMSD docking poses (RMSD smaller than 2 Å) from the CrossDocked2020 data set (Francoeur et al. 2020) for a total of 725 048 poses from a single cross-validation split that clusters similar protein targets.

7.2.8 Evaluation Metrics

With a valid molecule at hand, it is possible to compute many evaluation metrics such as quantitative estimate of drug-likeness (QED), synthetic accessibility (SA) and RMSD. Molecular descriptors and metrics are computed with RDKit (Landrum 2022), while RMSDs are computed using `spyrmsd` (Meli and Biggin 2020).

QED (Bickerton et al. 2012) is a quantitative measure of drug-likeness based on the concept of desirability (Derringer and Suich 1980; Harrington 1965). Desirability functions—taking the value of 0 for completely undesirable properties and the value of 1 for completely desirable properties—are combined into a single dimensionless score, the QED score (Bickerton et al. 2012)

$$\mathrm{QED}_w = \exp\left(\frac{\sum_i w_i \ln d_i}{\sum_i w_i}\right), \tag{7.2}$$

where $\{d_i\}$ are desirability functions and $\{w_i\}$ the corresponding weights. In the case of the QED score, the desirability functions are defined empirically as asymmetric double sigmoidal functions[3] following the property distributions for a set of approved orally administered drugs (Bickerton et al. 2012). The following properties are used to define the desirability functions: molecular weight, octanol-water partition coefficient (Ghose and Crippen 1986), number of hydrogen bond donors (HBDs), number of hydrogen bond acceptors (HBAs), molecular polar surface area (PSA), number of rotatable bonds, number of aromatic rings, and number of structural alerts (Brenk et al. 2008). The weights are determined so that they maximise the information content (Shannon's entropy (Shannon 1948)).

Another interesting metric for generative models is the SA score, which estimates the ease of synthesis of drug-like molecules (Ertl and Schuffenhauer 2009). This is an important metric since the ultimate goal of generative models is to generate novel drug candidates that can be synthesized and experimentally validated against a target of interest. The SA score developed by Ertl and Schuffenhauer (2009)—which tries to strike a balance between fast complexity-based scores and computationally expensive retrosynthetic analysis—is defined as

$$\mathrm{SA}^{\dagger}_{\mathrm{score}} = S_{\mathrm{fragment}} - P_{\mathrm{complexity}}. \tag{7.3}$$

The fragment score S_{fragment} tries to capture the synthetic knowledge in the PubChem database (Kim et al. 2021) based on frequently occurring fragments—with the underlying assumption that frequently occurring fragments are synthetically accessible. The complexity penalty $P_{\mathrm{complexity}}$ captures the presence of complex structural features (spiro rings, ring fusions, stereo centres, and macrocycles). The actual SA score is obtained by multiplying $\mathrm{SA}^{\dagger}_{\mathrm{score}}$ by -1 and scaling to the range $[1, 10]$.

The similarity between molecules can be quantified by computing the

[3] The asymmetric double sigmoidal function defined used in (Bickerton et al. 2012) is given by:

$$d(x) = a + b\left[1 + \exp\left(-\frac{x - c + d}{e}\right)\right]^{-1} \cdot \left\{1 - \left[1 + \exp\left(-\frac{x - c - d}{f}\right)\right]^{-1}\right\}.$$

Tanimoto/Jaccard similarity (Jaccard 1912; Tanimoto 1958)

$$S(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\langle \mathbf{a}, \mathbf{a} \rangle + \langle \mathbf{b}, \mathbf{b} \rangle - \langle \mathbf{a}, \mathbf{b} \rangle} \tag{7.4}$$

between two molecular FPs $\mathbf{a}$ and $\mathbf{b}$ encoding the molecules being compared (Bajusz et al. 2015). Common FPs to compute molecular similarity are the Morgan—or extended connectivity—FPs (Morgan 1965; Rogers and Hahn 2010) and the MACCS FPs (Durant et al. 2002), although more modern and robust alternatives are available (Capecchi et al. 2020).

Given that the final molecules are optimised, another interesting metric is the RMSD between the molecule fitted to the density and the minimised molecule. This RMSD gives an idea of how well the molecule still fits the real or generated density after optimisation. For the receptor-constrained model, generated molecules are optimised first with RDKit's UFF (Rappe et al. 1992)—to obtain reasonable geometries—and later with AutoDock Vina (Trott and Olson 2009) within the binding pocket, to optimise protein-ligand interactions (according to the AutoDock Vina SF). In the following, we denote RMSD_{UFF} the RMSD between the molecule fitted to the density and the UFF-minimised molecule, and $\text{RMSD}_{\text{Vina}}$ the RMSD between the molecule fitted to the density and the UFF- and Vina-minimised molecule. Symmetry-corrected RMSDs are computed using `spyrmsd` (Meli and Biggin 2020), which takes into account molecular symmetry using graph isomorphism (see Appendix B for details).

7.3 Test Systems: BRD4 and CDK2

To test the new 3D generative models proposed by Ragoza, Masuda, et al. (2020) and Masuda et al. (2020) we selected two different targets of pharmaceutical interest, with associated sets of known inhibitors: bromodomain-containing protein 4 (BRD4) and cyclin-dependent protein kinase 2 (CDK2).

7.3.1 BRD4

Bromodomains (BRDs) are epigenetic mark readers that recognise acetylated lysine residues (Dhalluin et al. 1999). Acetylation of lysine residues (Fig. 7.2) is a post-translational modification of proteins that significantly alters the physicochemical properties of the modified residue due to charge neutralisation (Kouzarides 2000; Muller et al. 2011). This modification, performed by acetyl-transferase enzymes, plays important roles in gene expression, via chromatin regulation and transcriptional control (Muller et al. 2011).

BRD4 has been previously used to perform a retrospective analysis of the

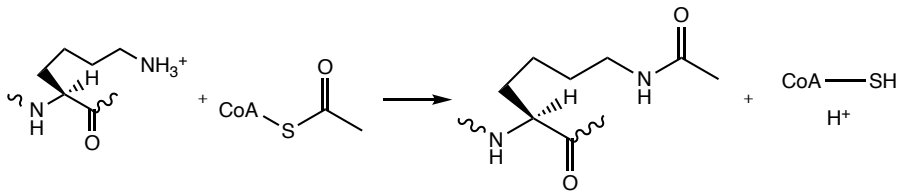


Figure 7.2: Acetylation of a lysine residue within a histone N-terminal tail. The transfer of the acetyl group from acetyl-CoA to the lysine residue is catalysed by histone acetyltransferases (Berg et al. 2019). Acetylated lysine residues are recognised by BRDs (Dhalluin et al. 1999).

performance of absolute binding free energy (ABFE) calculations based on classical MD (Aldeghi, Heifetz, et al. 2016)—in collaboration with Evotec. Parts of the data set used by Aldeghi, Heifetz, et al. (2016) were subsequently included in a benchmark data set for free energy calculations (Mobley and Gilson 2017; Mobley and Slochower 2017).

The BRD4 data set from Mobley and Slochower (2017) consists in an *apo* structure built from the crystal structure with PDB ID 4LYI as well as co-crystal structures with 9 inhibitors and an additional non-binder. Fig. 7.3 shows the 10 molecules included in the benchmark data set. Ligand 1, a non-binder, was obtained from PubChem and docked into the *apo* structure using AutoDock Vina. The ligands have been protonated with Open Babel. Ligands 4-9 were included in Aldeghi, Heifetz, et al. (2016) while ligands 3-4, ligands 6-8, and ligand 10 were employed in the first application of attach-pull-release (APR) calculations to protein-ligand binding, a method originally developed to study host-guest binding (Heinzelmann et al. 2017).

The 10 ligands reported in Fig. 7.3 consist of drug-like molecules with different physicochemical properties and functional groups. All the inhibitors are neutral and contain aromatic and fused ring systems (from 2 to 4 rings). The heavy-atoms molecular weights of the inhibitors ranges from 200 to 432 Da, the Wildman-Crippen $\log p$ (Wildman and Crippen 1999) ranges between 1.68 and 5.53, while the number of rotatable bonds varies from 0 to 6.

According to Mobley and Slochower (2017), ligand 1 is a non-binder (inactive at 250 µmol) while ligand 2 did not reach saturation (32% inhibition at 250 µmol). All other ligands are binders, with experimental binding affinities ranging from $-5.95\,\mathrm{kcal\,mol^{-1}}$ (ligand 4) to $-10.41\,\mathrm{kcal\,mol^{-1}}$ (ligand 10) (Filippakopoulos, Picaud, et al. 2012; Filippakopoulos, Qi, et al. 2010; Fish et al. 2012; Gehling et al. 2013; Lucas et al. 2013; Picaud et al. 2013).

Over the last years, several BRD4 inhibitors have been suggested and studied, and some compounds are currently in different stages of clinical trials as potential

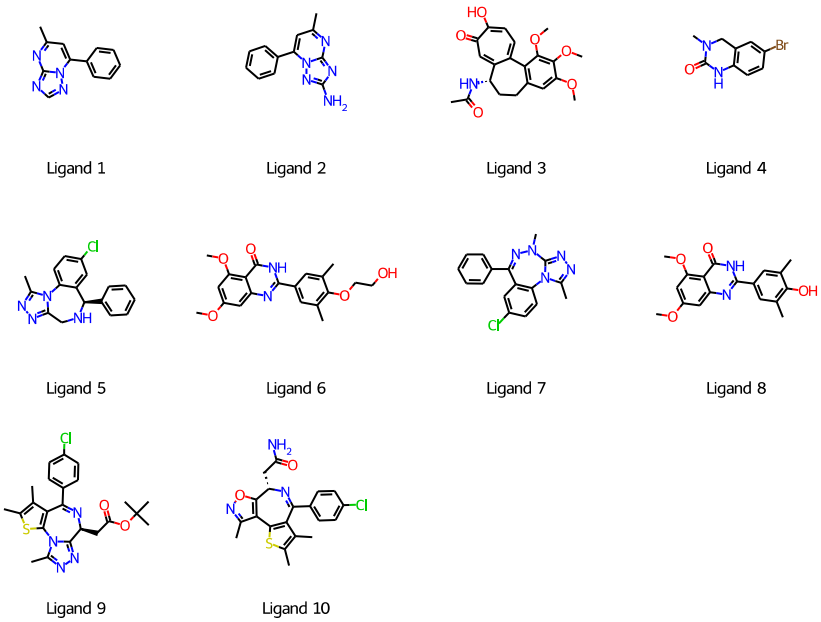


Figure 7.3: Set of 10 BRD4 inhibitors.

anticancer agents (Duan et al. 2018).

7.3.2 CDK2

Protein kinases are a class of proteins that control most aspects of cell life by selectively phosphorylating—adding a phosphoryl group to—other proteins. Aberrant behaviour of protein kinases—caused by alterations in their expression, or by mutations in the genes encoding for them—is involved in cancer and several other diseases—including inflammation, autoimmune disorders, and neurodegenerative disorders. Therefore, they are a very interesting target for drug discovery (Cohen et al. 2021; Kooistra and Volkamer 2017). Given their importance, several resources have been developed to navigate the wealth of data collected about such targets and their inhibitors (Kanev et al. 2020; Kooistra, Kanev, et al. 2015; Sydow et al. 2022; van Linden et al. 2013).

Cyclin-dependent protein kinases (CDKs) are a subset of protein kinases—specifically, serine-threonine kinases—involved in the regulation of the cell cycle. Cell division in eukaryotes occurs in 4 phases: DNA replication (phase S), synthesis of new proteins (phase G2), mitosis (M phase), and waiting period (phase G1) (Nelson and Cox 2013). Protein kinases phosphorylate specific proteins at different time intervals during the cell cycle and therefore ensure that the different phases occur in the correct order and with the correct timing

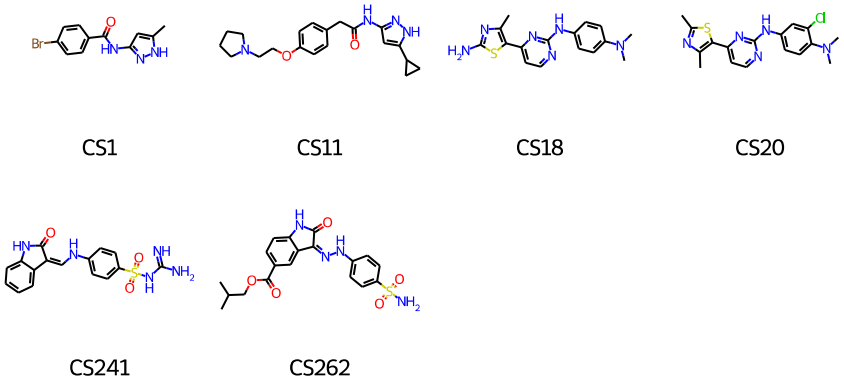


Figure 7.4: Set of 6 CKD2 inhibitors.

(Nelson and Cox 2013). Cyclin-dependent kinases are heterodimers with a regulatory subunit (cyclin) and a CDK subunit, whose active site becomes accessible upon cyclin binding (Jeffrey et al. 1995).

For our evaluation, we used the Community Structure Activity Resource (CSAR) CDK2 Kinase data set (Dunbar et al. 2013) available on the Drug Design Data Resource (D3R) website (`drugdesigndata.org`, last accessed October 1, 2022). The data set contains active molecules and decoys. We downloaded the active molecules together with the associate protein structure from the protein data bank (PDB), while we discarded all inactive molecules.

Ligand CS12 (PDB ID 4FKL) was discarded because lacking experimental binding affinity. Ligands CS2, CS5, CS13, CS14, CS15, CS17, CS244 and CS247 were discarded for lack of associated PDB ID. PDB ID 4FJK in the data set did not correspond to CDK2, therefore the correct system (PDB ID 4FKJ) was downloaded instead.

The CDK2 inhibitors retained come from three different chemical series. To reduce the number of systems to test, we selected the first and last compound of each series. The selected CDK2 inhibitors from the CSAR CDK2 Kinase data set are presented in Fig. 7.4.

The target structures are superimposed using ChimeraX's alignment tool (Goddard et al. 2017), and all solvent molecules, ions, and crystallisation co-factors are removed.

The 6 ligands reported in Fig. 7.4 consist of drug-like molecules with different physicochemical properties and functional groups, belonging to three different chemical series as can be inferred from the shared scaffolds. All the inhibitors are neutral and contain aromatic and/or fused ring systems (from 2 to 4 rings). The heavy-atoms molecular weights of the inhibitors ranges from 270 to 396 Da,

	η	Samples	Valid (%)	Minimised (%)	Matching Seed
BRD4	1.0	10 000	84.38	81.65	8
	5.0	10 000	89.22	80.82	0
CDK2	1.0	6000	89.72	86.68	3
	5.0	6000	88.50	79.40	0

Table 7.1: Percentages of valid and minimised molecules obtained by sampling the posterior distribution—with different variability factors η—of the ligand VAE seeded with BRD4 and CDK2 inhibitors. Valid molecules are molecules that can be parsed by RDKit, while minimised molecules are molecules obtained with optimisation using UFF.

the Wildman-Crippen $\log P$ (Wildman and Crippen 1999) ranges between 1.26 and 4.68, while the number of rotatable bonds varies from 2 to 8.

7.4 Ligand Variational Autoencoder

7.4.1 Posterior Sampling

Posterior sampling of the ligand VAE is preformed by encoding the known inhibitors of Fig. 7.3 and Fig. 7.4 (referred to as seeds) and sampling from the corresponding posterior distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\eta}^2\boldsymbol{\sigma}^2)$, with variability factors $\eta = (1.0, 5.0)$.

Tab. 7.1 reports the number of valid molecules obtained by sampling the ligand VAE posterior distribution with BRD4 and CDK2 inhibitors as seeds. Sampling is performed 1000 times per molecule, resulting in a total of 10 000 samples for BRD4 inhibitors and 6000 samples for CDK2 inhibitors. A high percentage of molecules obtained is valid, in the sense that they can be parsed by RDKit without errors. However, a lower number of molecules can however be optimised using the UFF, which indicates that some valid molecules can't be parametrised properly with the FF. Only a handful of generated molecules match the seed molecule, as determined by comparing canonical SMILES.

Fig. H.3 reports the distributions of some molecular properties for both BRD4 and CDK2 inhibitors and different variability factors. We observe that the QED for samples seeded with BRD4 inhibitors is spread over many possible values, while the QED for samples seeded with CDK2 inhibitors is skewed towards low values. For SA score and molecular weight, molecules obtained from BRD4 and CDK2 seeds have similar distributions. In particular, it is worth noticing that, in both cases, the distribution of SA scores obtained with $\eta = 5.0$ is skewed towards high values, which indicates synthetically inaccessible molecules. For calculated $\log p$, molecules generated from BRD4 seeds show

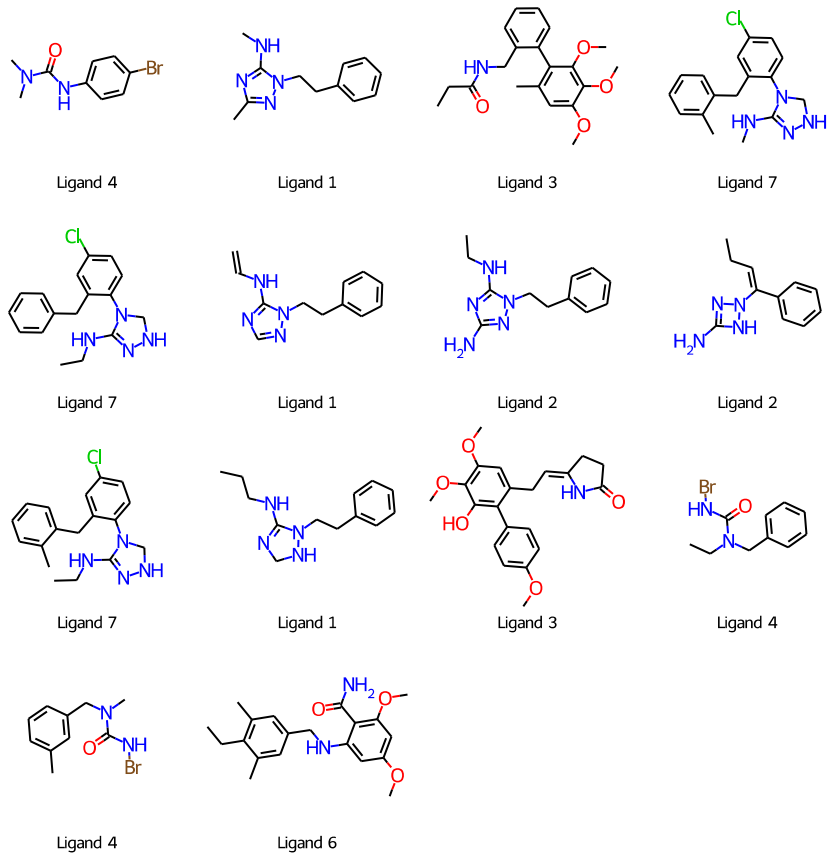
a bi-modal distribution for $\eta = 1.0$, which turns into a unimodal distribution when sampling is performed using a higher variability factor, showing that the influence of the seed diminishes when a higher variability factor is employed for sampling.

Fig. H.4 (BRD4) and Fig. H.5 (CDK2) show the distributions of the different properties disaggregated by ligand (1000 samples per ligand). For BRD4, we can see that ligand 4 has molecular properties' distributions that are very narrow around a particular value—with many outliers—when sampling with $\eta = 1.0$. However, the distribution of the similarity with the seed molecule is much broader than the one of other ligands. This is because ligand 4 is a small fragment-like ligand and therefore generated densities and subsequent fitting often lead to the same molecule or similar molecules. This is also the cause of the peaks at certain values of different properties that appear in Fig. H.3 only for BRD4.

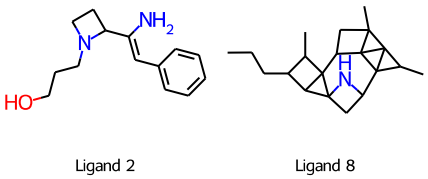
Most generated molecules have a high SA score, indicating synthetically intractable molecules. This is a common problem with generative models, where unrealistic molecular structures are often generated despite the reasonable performance on quantitative benchmarks (Gao and Coley 2020). In `liGAN`, the small number of systems with a low SA score is principally a consequence of the difficult grid-to-molecule conversion. Many generated molecules end up having several strained fused rings (see Fig. H.6 for some examples). It is therefore essential to filter the generated molecules according to some criteria, to obtain reasonable compounds.

For molecules generated with the ligand VAE we filter molecules using QED and SA, to obtain drug-like compounds with a reasonable (predicted) synthetic accessibility. Molecules are retained for QED $\in [0.8, 1.0]$ and SA $\in [1, 3]$. The molecules that pass the filters are shown in Fig. 7.5 (for BRD4 seeds) and Fig. 7.6 (for CDK2 seeds), ordered by increasing RMSD_{UFF}. Only 14 molecules obtained from BRD4 seeds with $\eta = 1.0$ (out of 10 000) pass the filters, and this number drops to 2 for $\eta = 5.0$. The situation is similar for molecule generated from CDK2 seeds: only 7 molecules remain after applying the filters for $\eta = 1.0$ and none survives for $\eta = 5.0$.

These filters are quite strict, but as it can be inferred from Fig. 7.5b they are nonetheless not enough to filter out all problematic systems—often containing several fused three- and four-member rings, coming from a poor reconstruction of widespread carbon densities.

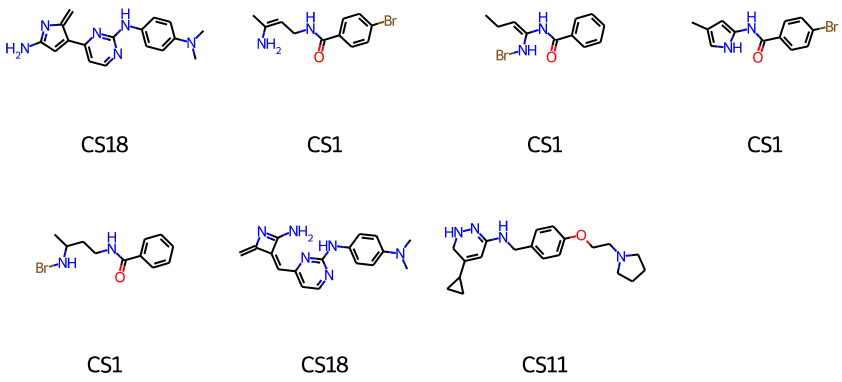


(a) $\eta = 1.0$



(b) $\eta = 5.0$

Figure 7.5: Molecules generated from BRD4 seeds using the ligand VAE, filtered according to the following criteria: QED $\in [0.8, 1.0]$ and SA $\in [1, 3]$. Molecules are ordered from lowest (top left) to highest (bottom right) RMSD_{UFF}.



(a) $\eta = 1.0$

Figure 7.6: Molecules generated from CDK2 seeds using the ligand VAE, filtered according to the following criteria: QED $\in [0.8, 1.0]$ and SA $\in [1, 3]$. Molecules are ordered from lowest (top left) to highest (bottom right) RMSD_{UFF}.

7.4.2 Prior Sampling

Prior sampling of the ligand VAE is performed by sampling latent space vectors from a normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\eta}^2)$—where $\eta = 1.0$ corresponds to a standard normal distribution—and decoding them into generated densities. This sampling from a normal distribution is possible because the KL divergence loss $\mathcal{D}_{\text{KL}}$ is employed during training, which forces the latent space distribution to follow a standard normal distribution.

Fig. H.7 shows the distribution of some molecular properties for latent space sampling with a variability factor $\eta = 1.0$. Since `liGAN` requires a seed for latent space sampling (even if the seed is not used), we performed the sampling twice, using either a BRD4 or CDK2 inhibitor as input. As expected from the fact that the seed molecule is ignored when sampling the prior distribution, the molecular properties follow the same distributions independently of the seed molecule.

From the distributions reported in Fig. H.7 it is clear that not many molecules would survive our previous filters. The number of generated molecules drops significantly at a QED of 0.7. The reason is that the QED score considers the number of rings, while molecules sampled from the prior distribution are mostly linear or branched molecules without rings, as shown in Fig. H.8. This problem was already observed by Ragoza, Masuda, et al. (2020). Fig. H.9 shows the number of aromatic and aliphatic rings present in all samples from the prior distribution. It is clear that most generated molecules have no ring systems, while only a small fraction of generated molecules have aliphatic rings. Aromatic

	η	Samples	Valid (%)	Minimised (%)	Matching Seed
BRD4	1.0	10 000	89.39	86.48	37
	5.0	10 000	92.70	82.01	0
CDK2	1.0	6000	93.28	90.78	0
	5.0	6000	93.13	89.25	0

Table 7.2: Percentages of valid, and UFF/Vina minimised molecules obtained by sampling the posterior distribution—with different variability factors η—of the receptor-constrained VAE seeded with BRD4 and CDK2 inhibitors. Valid molecules are molecules that can be parsed by RDKit, while minimised molecules are molecules obtained after optimisation using UFF and Vina.

rings, which are often used in medicinal chemistry (Aldeghi, Malhotra, et al. 2014; Ertl 2021; Roughley and Jordan 2011), are essentially never produced when sampling from the prior distribution. The complete lack of aromatic rings suggests that the latent space distribution does not encode drug-like molecules.

We also tested higher variability factors such as $\eta = 2.5$ and $\eta = 5.0$. Unfortunately, with such high variances `liGAN` often fails. This is not unexpected, since the KL divergence constrains the latent space distribution to a standard normal distribution, and therefore sampling with a higher variability factor will end up decoding points far from this ideal distribution. Given the results obtained with $\eta = 1.0$, however, we doubt that better handling of potential failures in the software would produce drastically different and more interesting results. If anything, we would expect to see worse connectivity issues, as is the case for samples from the posterior distribution.

7.5 Receptor-Conditional Variational Autoencoder

7.5.1 Posterior Sampling

Posterior sampling of the receptor-conditional VAE is performed by encoding the known inhibitors of Fig. 7.3 and Fig. 7.4, together with the corresponding protein structure, and sampling from the posterior distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\eta}^2\boldsymbol{\sigma}^2)$, with variability factors $\eta = (1.0, 5.0)$.

Tab. 7.2 reports the number of valid molecules obtained by sampling the receptor-conditional VAE posterior distribution with BRD4 and CDK2 inhibitors as seed. Sampling is performed 1000 times per molecule, resulting in a total of 10 000 samples for BRD4 inhibitors and 6000 samples for CDK2 inhibitors. A high percentage of molecules obtained is valid, in the sense that they can be parsed by RDKit without errors. The percentages of valid generated molecules are higher than the ones obtained with the ligand VAE, for both BRD4 and

CDK2. However, it is worth noticing that during initial tests we uncovered a bug preventing sulfonyl groups ($R_1-S(=O)_2-R_2$) to be correctly reconstructed, which has since been fixed.[4]

A lower number of molecules can however be optimised using the UFF, which indicates that some valid molecules can not be parametrised properly. For CDK2 and with a high variability factor of $\eta = 5.0$, some molecules were causing the optimisation of the generated ligand with the AutoDock Vina SF to fail due to "explosion" during the UFF minimisation; these systems—all involving sulfur atoms—have been manually removed so that batched optimisation with the AutoDock Vina SF could be performed.

Fig. H.10 reports the distributions of some molecular properties for both BRD4 and CDK2 and different variability factors. Compared to the ligand VAE (see Fig. H.3), we observe that the QED shifted towards higher values. This is especially evident for CDK2 inhibitors, and remains true for a high variability factor. The SA distribution, however, does not seem to be particularly different for $\eta = 1.0$, but is shifted towards lower (i.e. better) values for $\eta = 5.0$. These observations combined, hint that the quality of molecules generated by the ligand-constrained VAE is higher (predicted to be more drug-like and more synthetically accessible). However, it remains unclear if this is due to the receptor constraint, or other factors such as the different training set. The distribution of molecular weights and similarities between the generated molecule and the seed are similar to the ones obtained with the ligand VAE, although for $\eta = 5.0$ it seems that the receptor-constrained model produces slightly lighter (and thus smaller) molecules. Fig. H.11 (BRD4) and Fig. H.12 (CDK2) show the distributions of the different properties disaggregated by ligand (1000 samples per ligand).

Fig. 7.7 and Fig. 7.8 show the top 25 molecules, according to the Gnina affinity score, sampled from the posterior distribution using variability factor $\eta = 1.0$. For the receptor-conditional VAE, there are more molecules with QED $\in [0.8, 1.0]$ and SA $\in [1.0, 3.0]$ than there were for the ligand VAE—77 (BRD4) and 25 (CDK2). However, these numbers remain unsatisfactorily low, considering the large number of samples and the computational resources involved.

Fig. 7.9 shows the variation in RMSD distributions between the generated molecule and its conformation after optimisation, as well as the CNNscore (pose prediction) and the CNNaffinity (binding affinity prediction) of the protein-ligand complex obtained with Gnina. Fig. H.13 and Fig. H.14 report the same

[4] In earlier versions of `liGAN` the reconstruction of sulfonyl groups was incorrect (see liGAN/#15, last accessed October 1, 2022) and they are still not fully supported in certain parts of the code (see liGAN/#22, last accessed October 1, 2022).

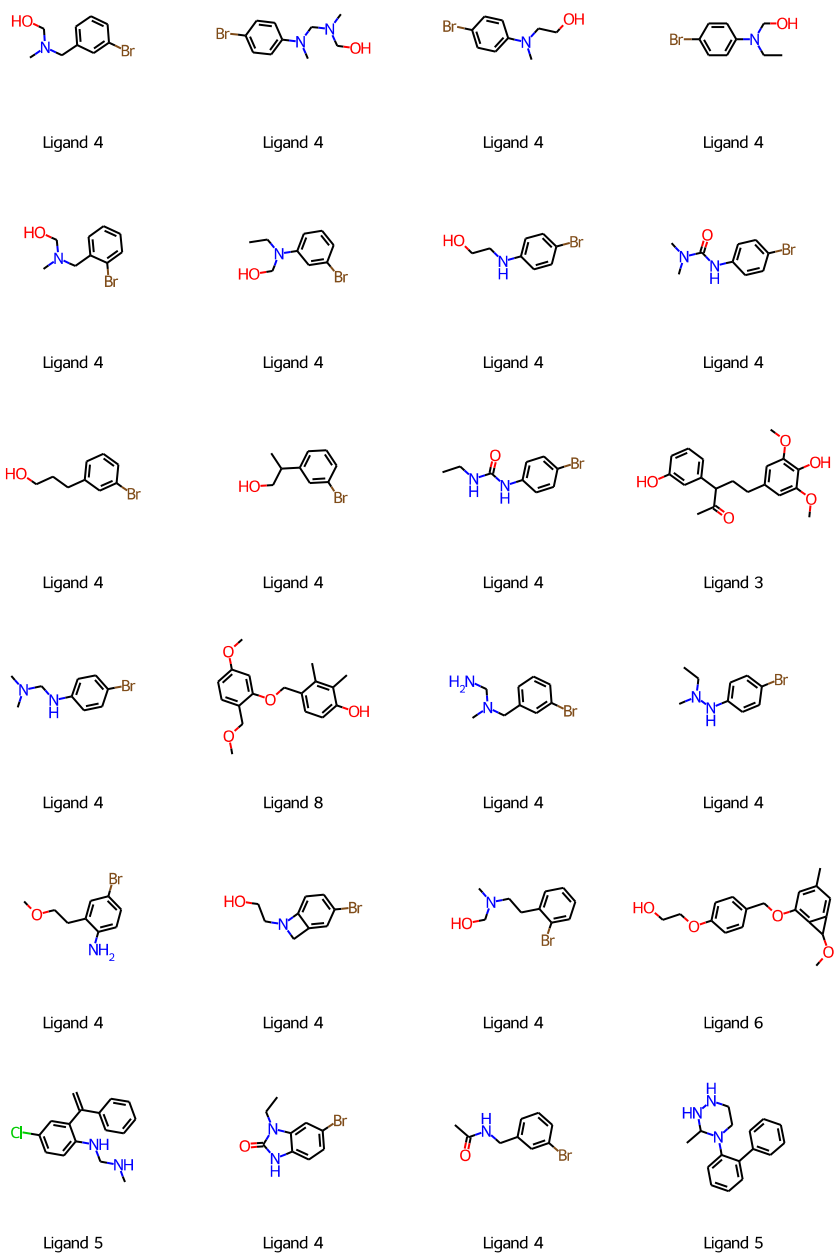


Figure 7.7: Top 24 filtered molecules—according to GNINA's CNNaffinity—generated with the receptor-conditional VAE sampling from the prior distribution conditioned by BRD4. The BRD4 inhibitors of Fig. 7.3 are used as seeds and sampling is performed with a variability factor of $\eta = 1.0$.

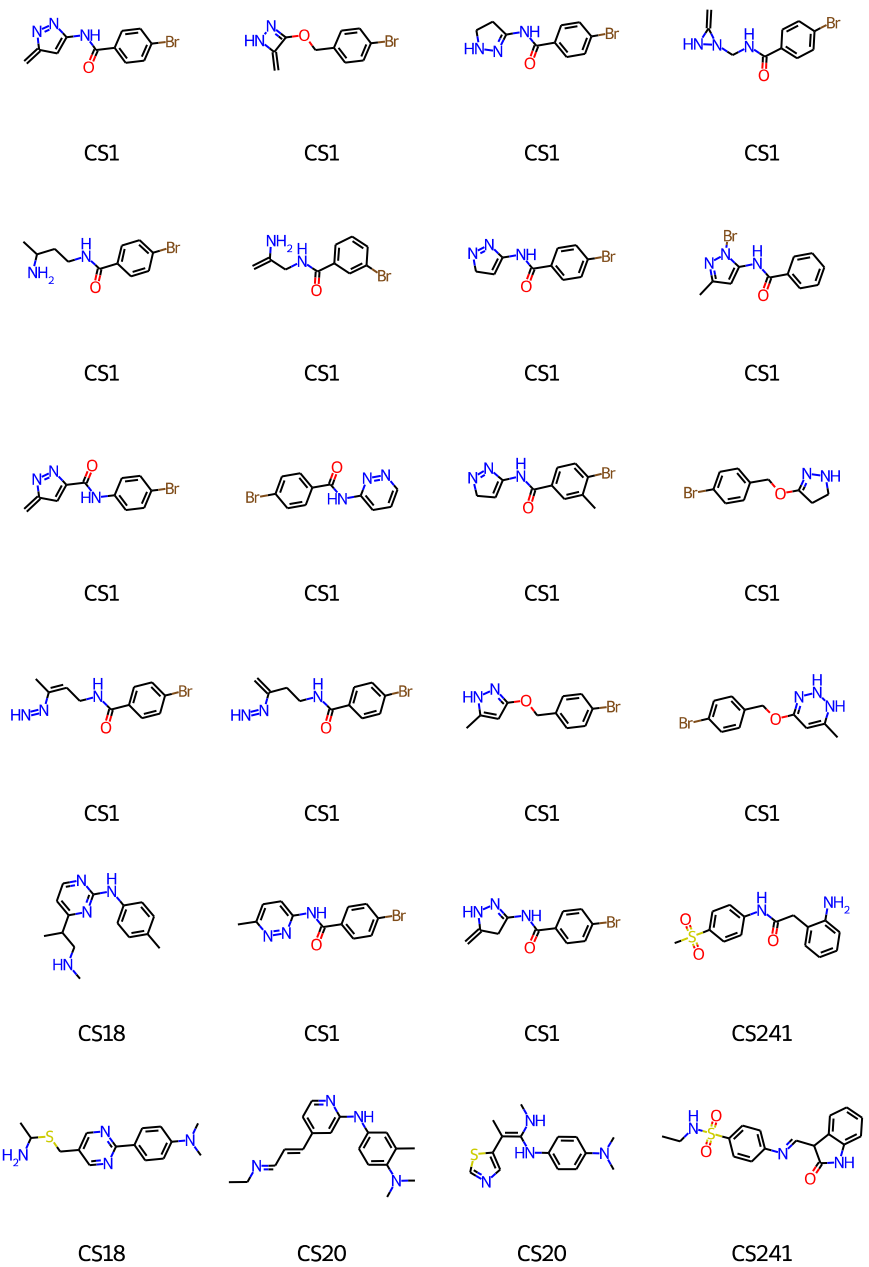


Figure 7.8: Top 24 filtered molecules—according to GNINA's CNNaffinity—generated with the receptor-conditional VAE sampling from the prior distribution conditioned by CDK2. The CDK2 inhibitors of Fig. 7.4 are used as seeds and sampling is performed with a variability factor of $\eta = 1.0$.

distributions but disaggregated by seed. We observe that most generated poses, which are generated within the binding site, have reasonable RMSD changes upon optimisation with UFF first and the AutoDock Vina SF second. However, the poses produced are mostly considered poor by the CNN SF, especially for BRD4. This is in part attributed to the large number of unphysical molecules produced by `liGAN`, which are very different from the real molecules used to train GNINA.

For CNNaffinity, it is interesting to see how it changes compared to the seed molecules. Ideally, a useful generative model would use the seed molecules as starting point to generate new molecules with better binding affinity. As shown in Fig. 7.10 this is not the case: for both CDK2 and BRD4, the generated molecules have a lower median predicted binding affinity than the seed molecules, and only some outliers show better (predicted) binding affinity.

7.5.2 Prior Sampling

Prior sampling of the receptor-constrained VAE is performed similarly to the ligand VAE by sampling latent space vectors from a normal distribution $\mathcal{N}(0, \eta^2)$ and decoding them into generated densities. However, in contrast with the ligand VAE, the latent space is conditioned by an input receptor. The input receptor is encoded by a non-variational encoder into a latent representation, which is concatenated to the sampled latent representation of the ligand. The concatenated vector is used to decode a generated density for the ligand, which depends on the receptor used as input via the latent representation as well as the skip connections.

Fig. H.15 shows the distributions of some molecular properties for latent space sampling with a variability factor $\eta = 1$, conditioned by both BRD4 and CDK2. Interestingly, despite the sampling being conditioned by the receptor, the distribution of molecular properties presented in Fig. H.15 does not seem to differ considerably between the two receptors, suggesting that receptor conditioning is rather weak, and it might be playing a minor role in the generation of new molecules. The number of molecules with ring systems is higher (see Fig. H.16), although as shown in Fig. 7.11 many rings are small three- or four-member rings. Additionally, there are molecules with an unusually high number of rings which again outline the problem with fused rings.

Fig. 7.11 shows some random samples from the prior distribution conditioned by both BRD4 and CDK2. The generated molecules are not drug-like and have no resemblance whatsoever with real inhibitors, despite the receptor conditioning.

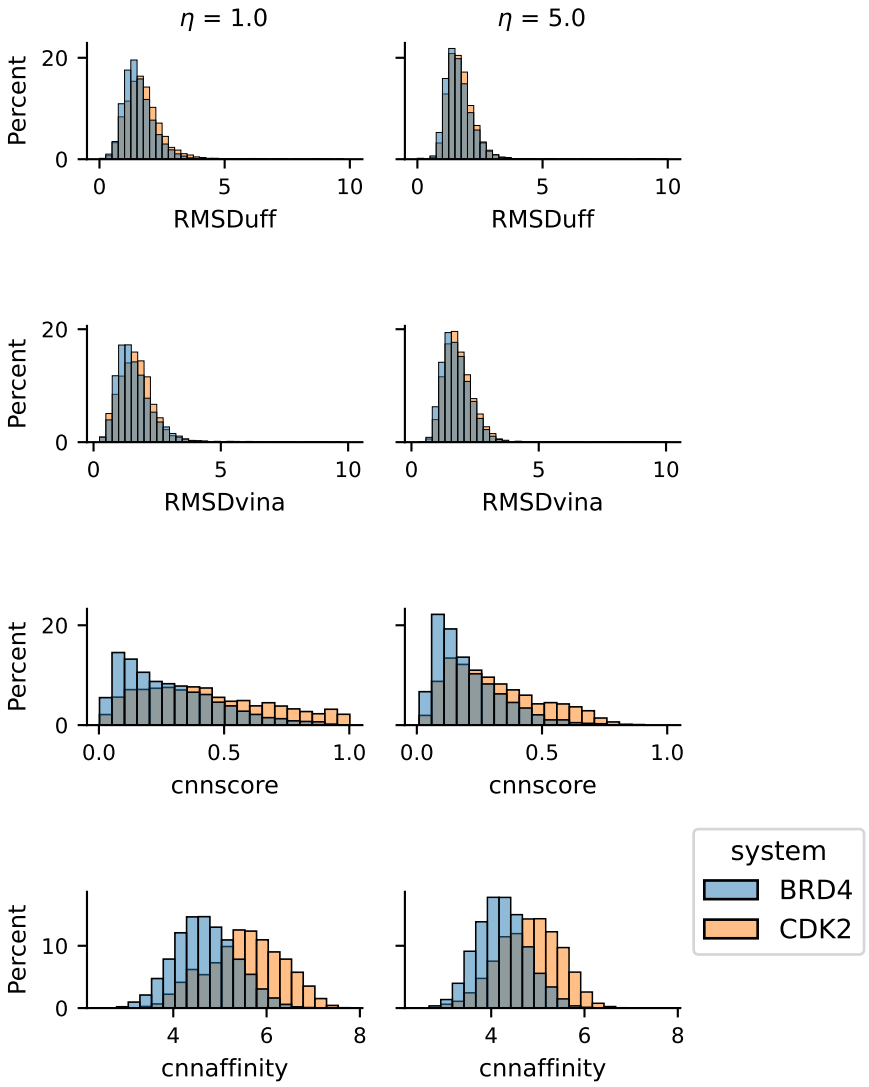
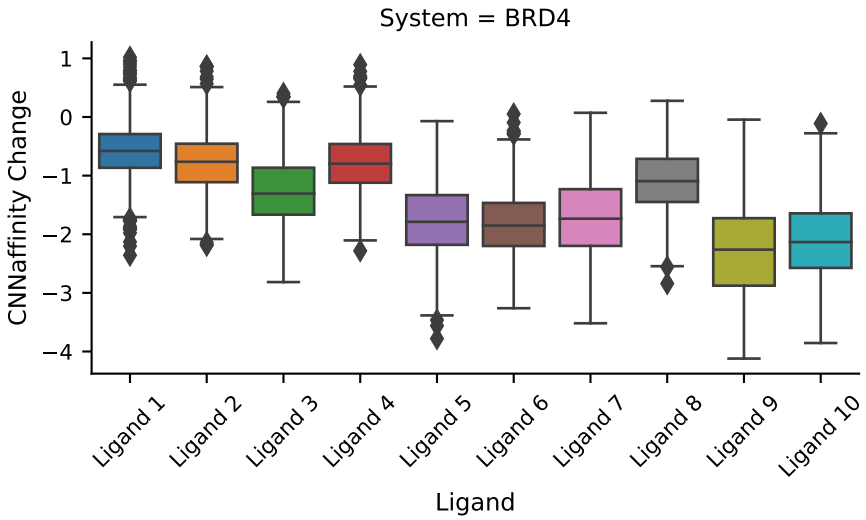
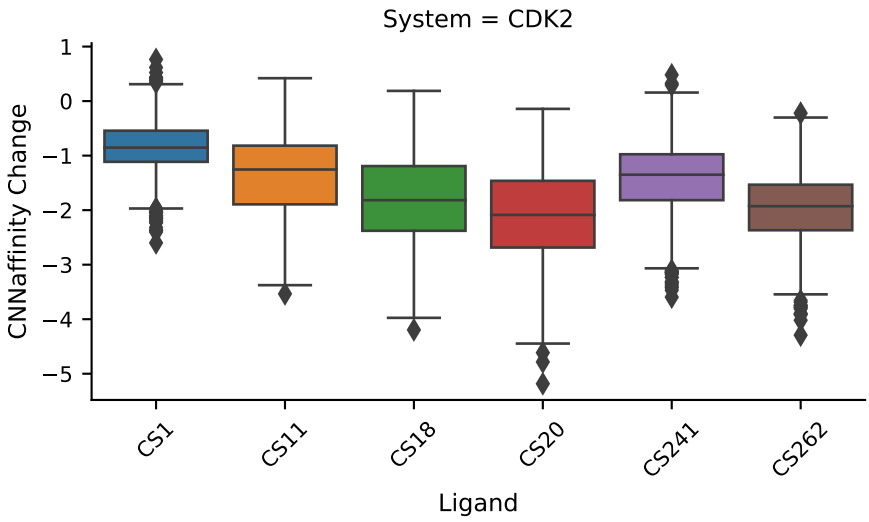


Figure 7.9: Distributions of RMSD, CNNscore, and CNNaffinity for generated molecules. Molecules are generated with the receptor-constrained VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds.

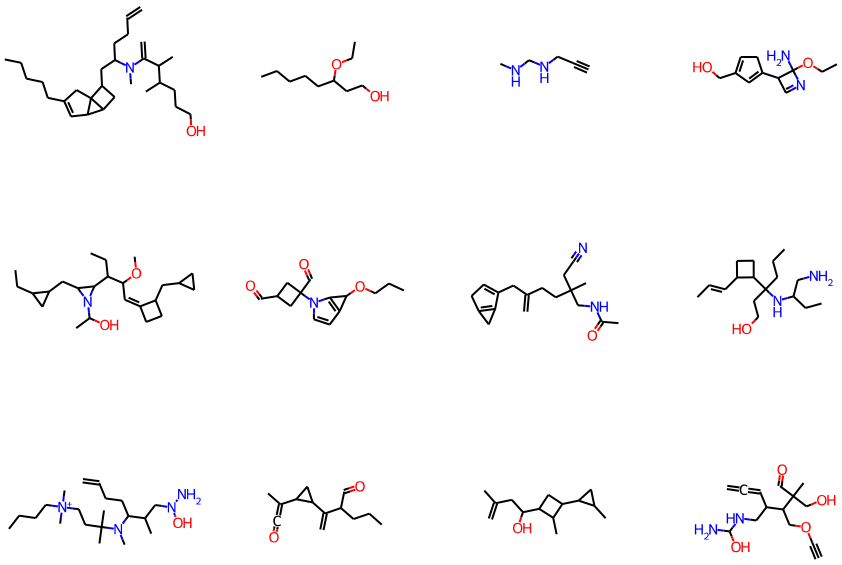


(a) BRD4

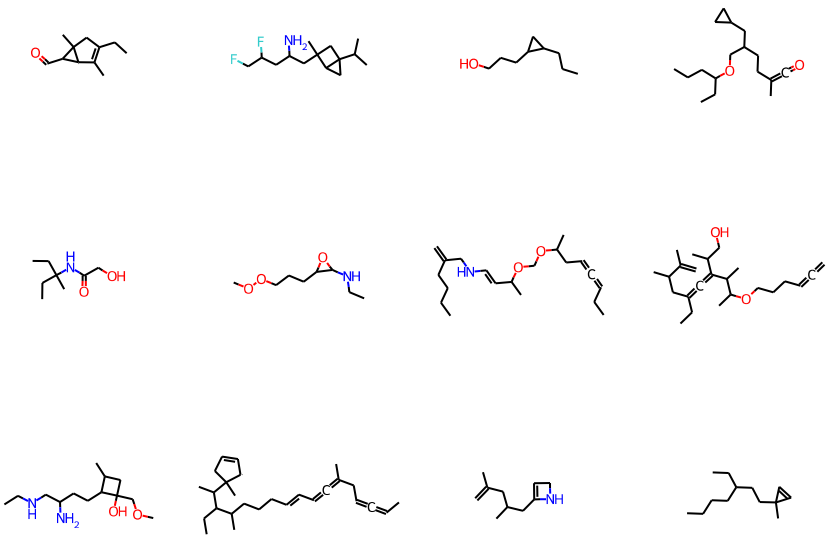


(b) CDK2

Figure 7.10: Change in predicted binding affinity between generated molecules and their corresponding seed molecule for BRD4 and CDK2 inhibitors.



(a) BRD4



(b) CDK2

Figure 7.11: Random sample of molecules generated from the prior distribution of the receptor-conditional VAE, conditioned by either BRD4 or CDK2.

7.6 Reconstruction Problems

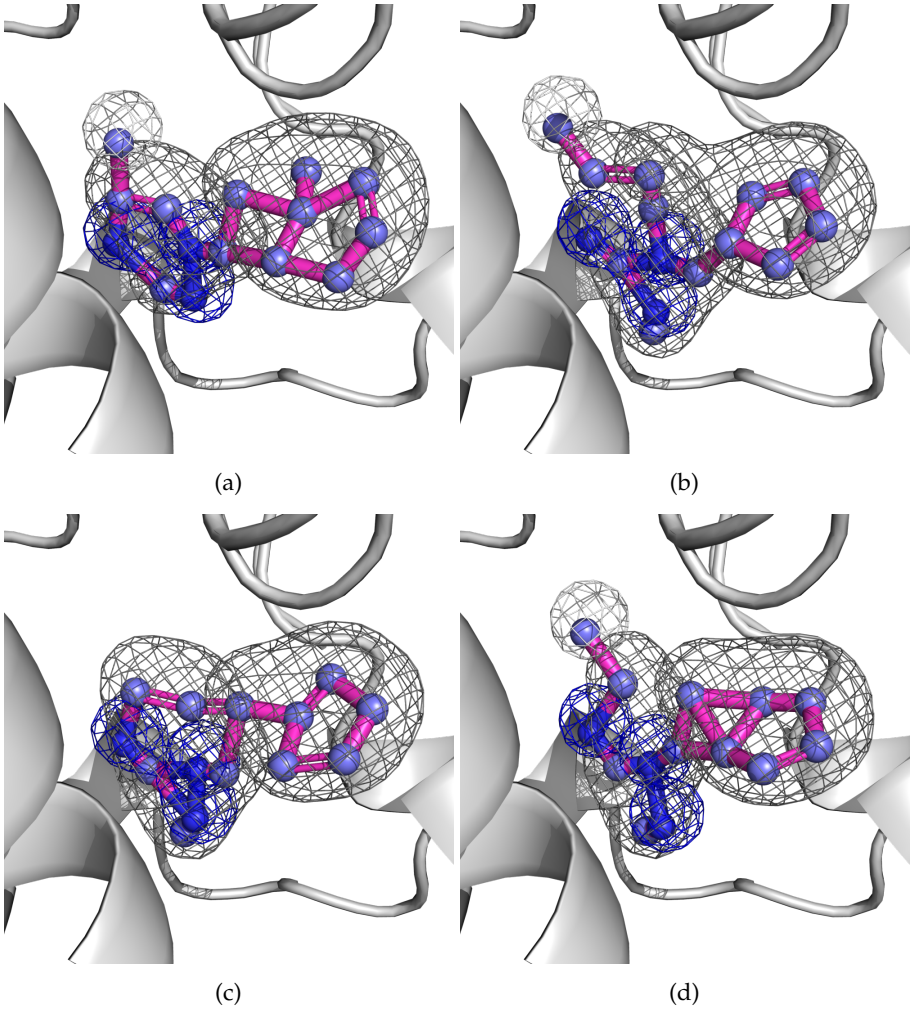
While densities generated from the posterior distribution look reasonable and rather similar to the real densities of the seed molecules—as expected from the minimisation of the reconstruction loss, and as shown in Fig. H.17—, the reconstructed molecules are considerably different to the input molecule. As already outlined in the previous sections, this is mostly due to the poor reconstruction performed by the reconstruction algorithm, shortly described in section 7.2.

Fig. 7.12 highlights the fitting problems, using BRD4 ligand 1 as an example and densities generated with the receptor-conditional VAE (see Fig. H.17 for the generated densities around the seed ligand). In all four examples, we can see a blob of aromatic carbon density corresponding to the aryl ring in the seed molecule. The automatic rule-based reconstruction procedure recognises correctly that such density can accommodate multiple carbon atoms. Unfortunately, in many cases the gradient-based optimisation of atomic positions pushes one of the atoms near the centre of the density, to increase the density overlap. Once the gradient-based optimisation of atomic position finds a local minimum, bonds are added. Given the erroneous position of the central atom, the aryl ring is usually not reconstructed correctly, and strained multi-cyclic systems are obtained instead. Double bonds are finally added—because the atoms are being fitted to an aromatic density—so that valency rules are fulfilled, and the resulting molecule is "valid" as far as RDKit is concerned.

A problem related to the structures obtained by the fitting procedure is that the final UFF/Vina-optimised structures do not fit the generated density any more. Therefore, it is very difficult—or impossible—to evaluate if and how protein-ligand interactions play an important role in the generation since the generative model produces a density but the optimised generated molecule might not have the same interactions. This problem is shown in Fig. 7.13, where the final UFF- and Vina-optimised poses corresponding to the generated molecules of Fig. 7.12 are displayed.

The observations from Fig. 7.12—which represents a common case when aromatic rings or fused rings are present in the seed, as is the case for the BRD4 inhibitors considered here—, and the in-depth study of the previous sections show that the main problem of `liGAN` is the conversion of generated densities into molecules. Because of this problem, the final re-constructed molecules are unlikely to correspond to the point in latent space from which a generated density is decoded.[5]

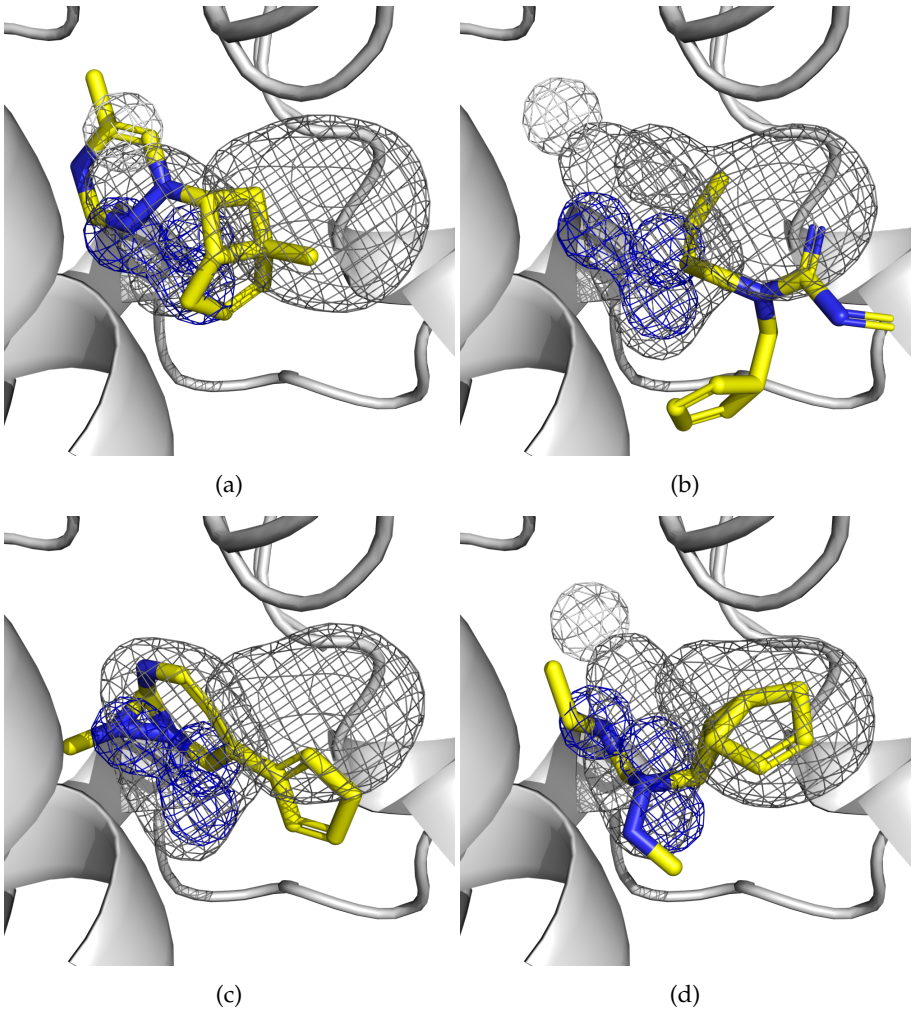
[5]This disconnect between the latent space and the generated molecules prevents efficient latent space exploration and optimisation. We implemented an interface between the molecular swarm



(a) (b)

(c) (d)

Figure 7.12: Random examples of erroneous reconstruction caused by the placement of atoms within widespread generated densities for BRD4 (ligand 1). Densities are generated using the receptor-conditional VAE.



(a) (b)

(c) (d)

Figure 7.13: Random examples of poor fitting of the generated density after optimisation with UFF and AutoDock Vina for BRD4 (ligand 1). Densities are generated using the receptor-conditional VAE.

Our exploration suggests that the density representation used in `liGAN`—and inherited from Gnina—might not be ideal for molecular reconstruction from generated densities. While re-training `liGAN` with different density representations is outside the scope of this project, we postulate that a more localised density combined with more expressive atom types—better representing aromaticity information or ring membership for heteroatoms—as well as a different aggregation function—maximum instead of sum, as in Jiménez, Doerr, et al. (2017)—might facilitate the tricky reconstruction of a molecule from a generated density.

Given the problems with the original fitting method outlined so far, in the next sections we explore some strategies aiming to circumvent such issues.

7.7 Fit Densities Around Murcko Scaffold

Given the severe problems with `liGAN` reconstruction outlined in the previous sections, especially related to a poor reconstruction of ring systems, we now try to devise some strategies to palliate such problems. To reduce the computational costs associated with our experiments we focus only on the receptor-conditional VAE, which is far more interesting from a drug-discovery perspective than the ligand VAE. Additionally, we limit our experiments to a variability factor $\eta = 1.0$, since this is the variability factor that was used to train the model and higher variability factors exacerbate the reconstruction problems, as we have seen in previous sections.

The first strategy we apply to improve the reconstruction of molecules from generated densities is to retain the Murcko scaffold of the seed molecule and only fit the density corresponding to the functional groups attached to said scaffold. Our strategy proceeds as follows:

1. compute the density of the seed molecule,
2. generate a new density from the seed density using the receptor-conditional VAE,
3. obtain the Murcko scaffold of the seed molecule,
4. compute the density of the Murcko scaffold,
5. subtract the density of the Murcko scaffold from the generated density,
6. fit atoms to the density difference,

optimiser of Winter et al. (2019) and `liGAN` but sampled molecules resulted to be similar to the random samples obtained from the prior distribution.

	Samples	Valid (%)	Minimised (%)	Matching Seed (%)
BRD4	10 000	97.06	97.06	16.43
CDK2	6000	99.78	99.78	12.80

Table 7.3: Percentages of valid and minimised molecules obtained by sampling the posterior distribution of the receptor-conditional VAE seeded with BRD4 and CDK2 inhibitors, and using the scaffold trick. Valid molecules are molecules that can be parsed by RDKit, while minimised molecules are molecules obtained with optimisation using UFF.

7. remove fitted atoms overlapping with the Murcko scaffold (if any),

8. add bonds between fitted atoms and the Murcko scaffold.

The removal of fitted atoms overlapping with atoms in the Murcko scaffold is necessary because the atoms of the Murcko scaffold might be typed differently—since they have different substituents—from the atoms in the original molecule. Given that different atom types are associated with different density channels, the density difference (generated density minus density of the scaffold) will not affect such atoms. Therefore, there might be remaining density that can be fitted coming from atoms of the scaffold. This suggests that a different atom typing scheme—which is conserved when going from a molecule to its Murcko scaffold—might be more appropriate. Fig. 7.14 shows a schematic representation of our *scaffold trick*.

Tab. 7.3 reports some statistics of sampling 1000 times per seed from the posterior distribution using the scaffold trick described above. We can see that the percentage of valid molecules is significantly higher than the one obtained with the original method. Additionally, all valid molecules obtained with the scaffold trick can be optimised using UFF, in contrast with the original method (see Tab. 7.2 for comparison). However, the expected drawback of the scaffold trick is also immediately apparent in the percentage of molecules matching the seed. While only a handful of molecules was matching the seed with the original method, the scaffold trick produces the same molecule as the seed around 15% of times. This is of course undesired since it results in wasted computational time.

Fig. H.18 shows the distributions of molecular properties for both BRD4 and CDK2 inhibitors sampled from the posterior distribution using the scaffold trick. As expected, we see a clear shift towards better values for the QED and SA scores. However, it is clear that the generated molecules are much more similar to the known inhibitors used as input, and as already mentioned above there is a significant percentage of generated molecules matching the seed. Fig. H.19 and Fig. H.20 directly compare the distributions of molecular properties obtained

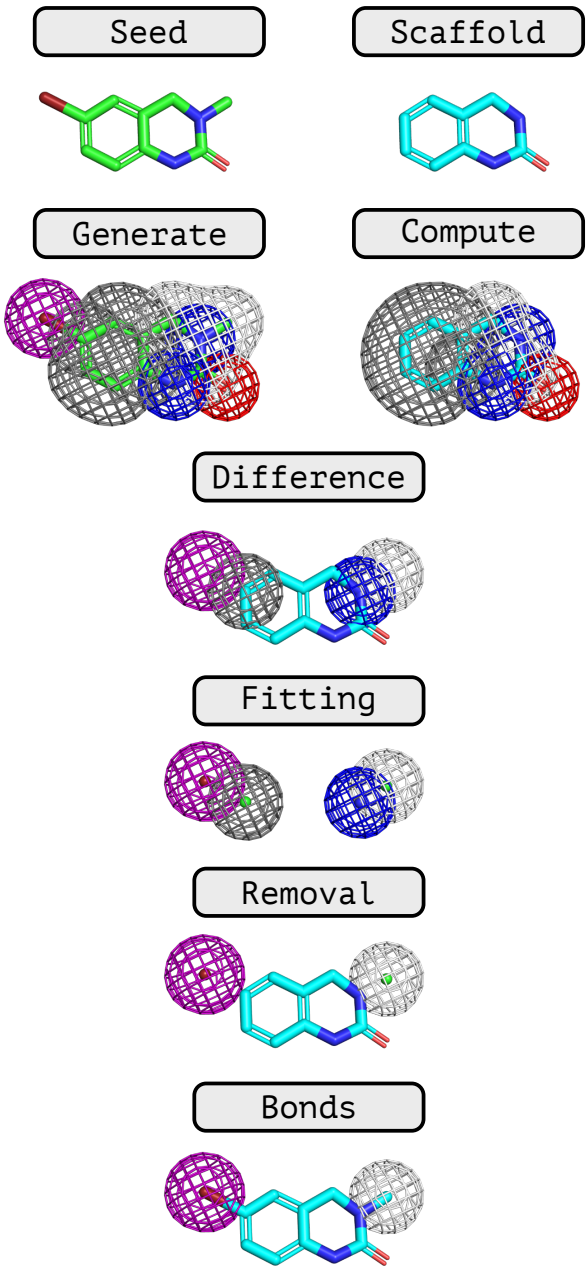


Figure 7.14: Schematic representation of the scaffold trick. A density representation is obtained for both the seed molecule (generated) and its Murcko scaffold (computed). The density of the Murcko scaffold is then removed from the generated density and atoms are fitted to the remaining density. Finally, atoms overlapping with the scaffold are removed, and bonds between the scaffold and the fitted atoms are added.

with and without the scaffold trick, where the differences between the two methods becomes very clear.

Fig. H.21 and Fig. H.22 show the top (unfiltered) inhibitors ranked by GNINA's CNNaffinity for BRD4 and CDK2, respectively. For CDK2 we notice that while sulphur atoms are fitted around the scaffold, the sulfonyl group present in the same seed molecules is never reconstructed—at least for the ligands with the best CNNaffinity. We can see, however, that the generated molecule are drug-like, and similar to the real inhibitors used as seeds (as expected).

Fig. 7.15 shows the distribution of RMSDs as well as the CNNscore and CNNaffinity obtained with GNINA. We can see that with the scaffold trick, the CNNscore—which indicates how good a ligand pose is—is now skewed towards higher values. Compared to the distributions of Fig. 7.9 (see Fig. H.23 and Fig. H.24 for a direct comparison) this is likely because molecules generated with the scaffold trick are drug-like molecules similar to the ones contained in GNINA's training set, unlike the molecules obtained by reconstructing directly from the density. The CNNaffinity is also shifted towards higher values (higher predicted binding affinity).

However, if we look at the difference in binding affinity between the generated molecules and the original inhibitors used as seed reported in Fig. 7.16, it is clear that the generated molecules often have a lower predicted binding affinity (GNINA's CNNaffinity) than the seed molecules. We can see that the change in binding affinities is better when using the scaffold trick than when re-constructing molecules directly from the density. However, the median predicted binding affinity change is still smaller than zero, which indicates that most generated molecules are predicted to be worse binders than the corresponding seed molecules.

Finally, another major drawback of the scaffold trick is that it can only be applied to sampling the posterior distribution, where a seed molecule—usually a known inhibitor—is available. Therefore, sampling from the prior distribution—which is the most interesting application when trying to generate molecules against a novel target for which there are no known inhibitors—is not possible with the scaffold trick.

7.8 Fit Fragments to Generated Densities

The use of Murcko scaffolds presented in the previous section works well to avoid connectivity problems in the molecule since only simple side chains need to be fitted within the generated density. However, this simple methodology has the drawback of reducing the diversity of generated molecules. This is

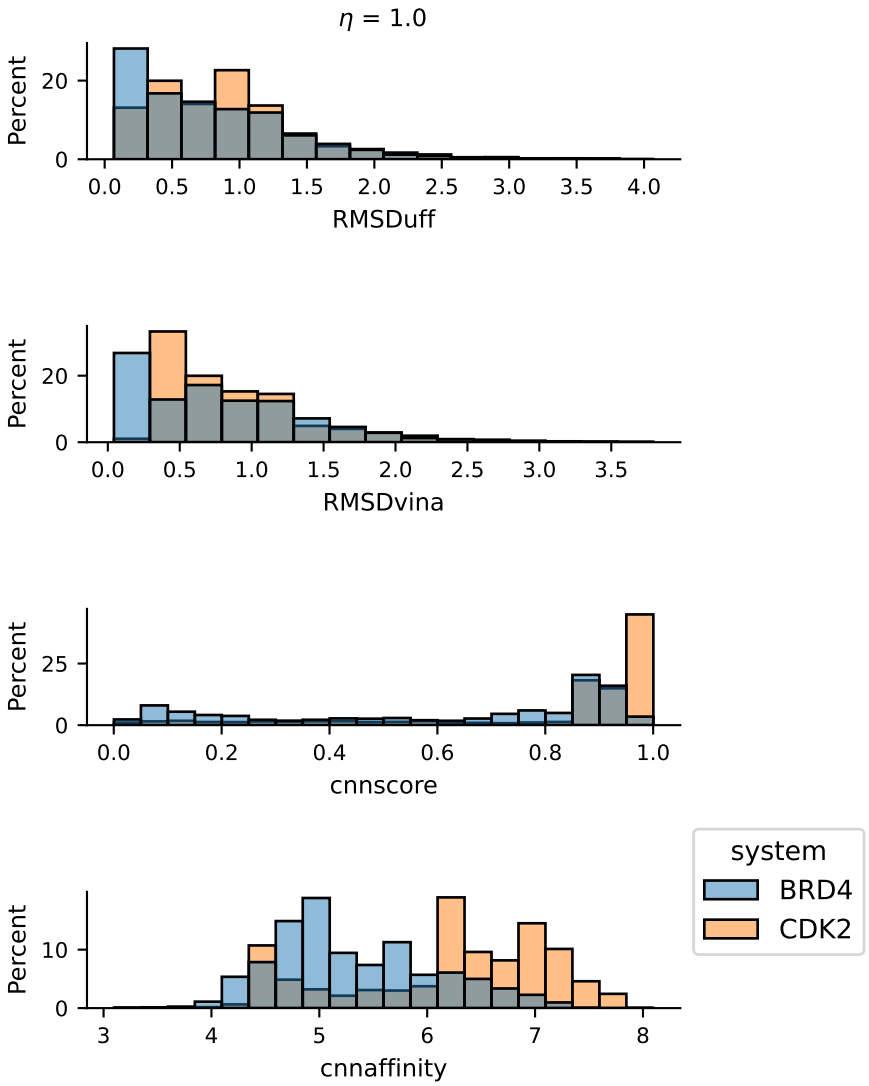
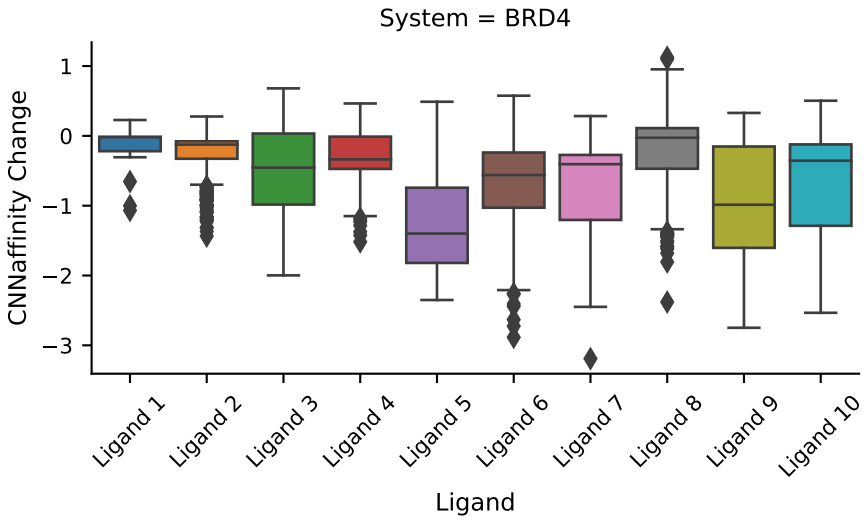
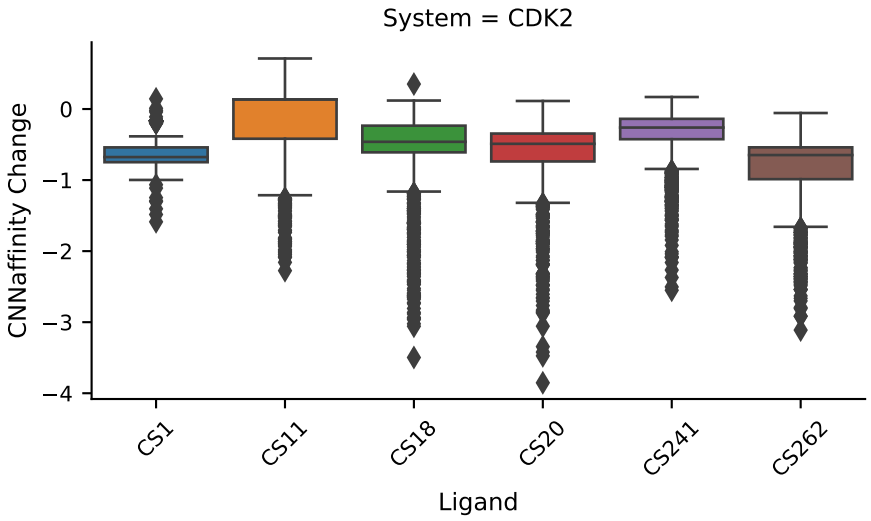


Figure 7.15: Distributions of RMSD, CNNscore, and CNNaffinity for generated molecules. Molecules are generated with the receptor-constrained VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds and applying the scaffold trick.



(a) BRD4



(b) CDK2

Figure 7.16: Change in predicted binding affinity between generated molecules and their corresponding seed molecule for BRD4 and CDK2. Molecules are generated with the scaffold trick.

problematic for two reasons: the low diversity limits the exploration of new regions of chemical space, and it results in wasted computational time.

In this section we try to fit small fragments to the generated density, to avoid the problems of reconstructing ring systems while trying to maintain good diversity in the generated molecules. Given that we only have generated densities, and therefore we lack atomic coordinates, positioning fragments within the density is no simple task. Shape-based alignment methods—such as ROCS (Grant, Gallardo, et al. 1996), ShaEP (Vainio et al. 2009), SHAFTS (Liu, Jiang, et al. 2011), Phase Shape (Sastry et al. 2011), and VAMS (Koes and Camacho 2014)—make use of atomic coordinates, while techniques from structural biology often assume that the correct answer is known and simply need to be aligned properly to the experimental electron density. In this case, however, we need to find the molecule—or a collection of fragments—that best fits the density without any prior knowledge of atomic coordinated since only the generated density is provided by `liGAN`'s VAE.

One possible approach to fit molecules to the generated density is to use a library of known fragments, align all of them to the generated density, and select the best alignments based on the density fit—which ideally considers both shape and chemical information.

7.8.1 SENSAAS

To fit fragments to the generated density we need a method that can work directly with densities since that is the only information directly generated by `liGAN`. One such method is SENSAAS, a recently developed shape-based alignment method based on the registration of coloured point clouds (Douguet and Payan 2020).

SENSAAS performs shape-based alignment of molecules or molecular fragments represented as point clouds which consists of two steps: the generation of 3D point clouds representing the surface of the molecules or molecular fragments to be aligned, and the alignment of said point clouds. In SENSAAS, point clouds are generated by computing van der Waals surfaces with the `nsc` software (Eisenhaber and Argos 1993; Eisenhaber, Lijnzaad, et al. 1995). The point clouds obtained from `nsc` are then subsampled and aligned using established computer vision techniques for the registration of point clouds—the process of aligning two point clouds—implemented in the Open3D library (Zhou, Park, et al. 2018), a modern C++ library for 3D data processing with convenient Python bindings.

Point cloud registration is performed by SENSAAS in two steps: a global geometry-based (shape-based) alignment—which only considers the global shape of the molecular surface—, and a refinement of the shape-based alignment

that considers both the shape and physicochemical properties (represented as colours) of the two molecular surfaces. A short description of the global and coloured registration methods is reported in section H.1.

To work more efficiently with SENSAAS, we forked the original repository, and we included several improvements in our own version.[6] The improvements include exposing different internal variables from the command line for fast experimentation, removing stale code using an earlier version of the Open3D library, as well as removing repeated calculations thus speeding up the code compared to the original implementation. Finally, we implemented a stripped-down version of the method, to simplify the code and speed-up development.

Given that `libmolgrid` does not play nicely with conda-installed Open Babel (see libmolgrid/issue#62, last accessed October 1, 2022) a Singularity container with Open Babel and `libmolgrid` installed from source is needed to run most experiments performed in the following sections. The Singularity container definition is available at RMeli/containers (last accessed October 1, 2022).

The data and scripts related to this section are available at RMeli/density-match (last accessed October 1, 2022).

7.8.2 Point Clouds from `libmolgrid` Densities

SENSAAS uses the third-party `nsc` code to generate coloured point clouds representing molecular surfaces (Eisenhaber and Argos 1993; Eisenhaber, Lijnzaad, et al. 1995). In our case, however, we need to use the densities generated by `liGAN` and convert them into point cloud-based surfaces. Therefore, we need to assess the use of `libmolgrid`-generated point clouds within the SENSAAS methodology to determine if this is a viable approach.

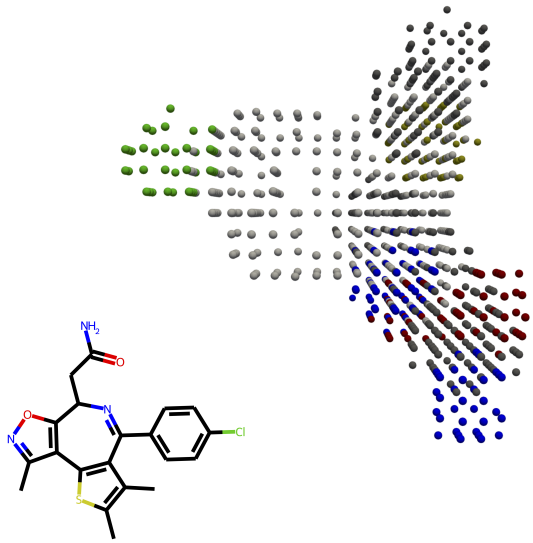
From visual inspection using the Open3D library, a reasonable point-cloud representation of the molecular surface based on `libmolgrid`-generated densities is obtained by taking the isosurface corresponding to the density values in the interval $[0.5, 0.6]$. Two examples of said isosurfaces are reported in 7.17.

To evaluate the use of point clouds obtained as isosurfaces of `libmolgrid`-generated densities—instead of the point clouds obtained with `nsc` as in the original implementation—we randomly rotated and translated the BRD4 inhibitors of Fig. 7.3 and the CDK2 inhibitors of Fig. 7.4 and used our custom version SENSAAS to compute the roto-translation between the point cloud of the rotated and translated molecule and the point cloud of the original molecule.

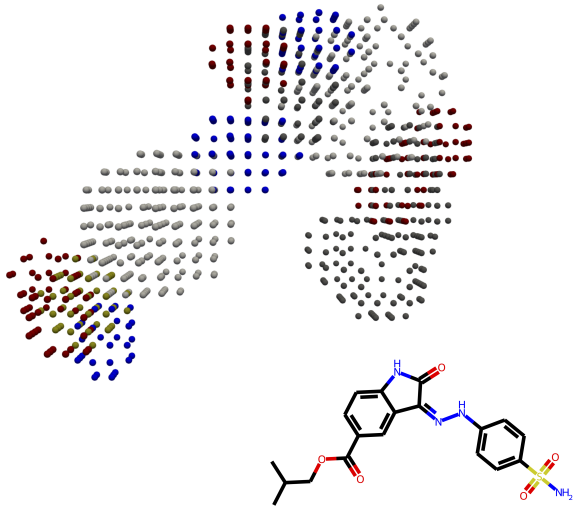
To colour the point clouds we used two strategies:

- colour using the four classes of SENSAAS (see Tab. H.1)

[6] We tried to contribute back to the original SENSAAS implementation but most pull requests were ignored.



(a) BRD4, Ligand 10



(b) CDK2, Ligand CS262

Figure 7.17: Examples of `libmolgrid`-generated point clouds.

- colour using `libmolgrid` atom types (see Tab. E.1).

When colouring the point clouds using `libmolgrid` atom types, the hydrophobic and non-hydrophobic labels for carbon atoms have been combined, as well as the different donor and acceptor types. This is necessary to minimise the mismatches between the density of the molecule under consideration and one of the fragments, that might have different properties due to different substituents. This issue was already encountered when developing the scaffold trick, as discussed above, and would benefit from the design of more suited atom types.

Fig. 7.18 (`libmolgrid` colour scheme) and Fig. H.25 (SENSAAS colour scheme) show the RMSD distributions between the reference molecule and its randomly rotated and translated counterpart after alignment using our custom version of SENSAAS, over 25 repeats. Repeating the alignment multiple times is needed because the registration of coloured point clouds has a stochastic element,[7] therefore results might differ in different runs.

We see that the alignment performed by our custom version of SENSAAS using `libmolgrid`-generated densities is near-perfect for most systems. In Fig. 7.18 (`libmolgrid` colour scheme) we have only one ligand—CDK2 inhibitor CS18—where the median RMSD is not close to 0.0 Å and another ligand—BRD4 inhibitor 7—shows a broad distribution. Three other ligands have some outliers, showing that the alignment can sometimes fail due to its stochastic nature. In Fig. H.25 we have a similar situation: only one ligand—still CDK2 inhibitor CS18—has a median RMSD larger than 0.0 Å, one ligand—CDK2 inhibitor CS20—has a broad distribution, and three ligands present some outliers.

Fig. H.26 shows the three-dimensional structures of some failed alignments run with the SENSAAS colouring scheme. Both failed alignments belong to the same chemical series of compounds; one is the CDK2 inhibitor CS18—which fails consistently as shown in Fig. H.25—while the other one is CS20—which has a broad distribution as shown in Fig. H.25. From the figures, it is clear that there is still a reasonable volume overlap between the molecules being aligned, but the orientation is wrong. This is a consequence of the two-step nature of coloured point cloud registration, which requires a first coarse alignment with uncoloured point clouds followed by a refinement based on colours.

The results of this section show that, in general, isosurfaces of `libmolgrid`-generated densities are a reasonable substitute for the point clouds representing the molecular surface obtained with `nsc`. This allows us to use the SENSAAS alignment procedure with `liGAN`-generated densities.[8]

[7] Until recently it was not possible to set the seed of the random number generator (RGN) in Open3D (see Open3D/issues#1263, last accessed October 1, 2022)

[8] `nsc` comes only with an academic licence. Our custom version of SENSAAS using `libmolgrid`-

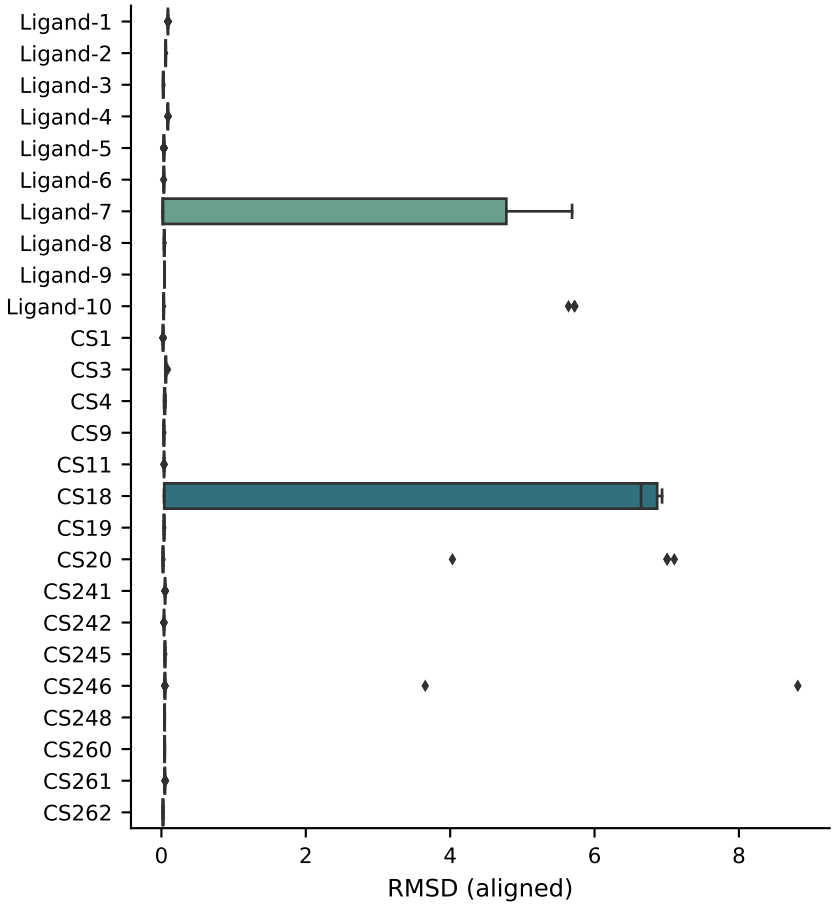


Figure 7.18: Distributions of final RMSDs—over 25 repeats—for the alignment of a molecule and its randomly rotated and translated counterpart, obtained using `libmolgrid`-generated densities and the `libmolgrid` colour scheme.

7.8.3 Assessment of Murcko Scaffold Fit

To further evaluate the SENSAAS pipeline towards our end goal of fitting (cyclic) fragments to generated densities, we assess the performance of the method on fitting Murcko scaffolds (Bemis and Murcko 1996) to their source molecule. Given the similar performance of the two colouring methods tested above, and the need for a fine-grained alignment in our scenario, we only use the `libmolgrid` colouring scheme hereafter.

Murcko scaffolds are obtained with RDKit, and randomly rotated and translated as described above. Given the stochastic nature of the alignment algorithm, we repeated the same alignment 25 times for each ligand—starting from the same initial position. The distribution of RMSDs between the original molecule and the aligned Murcko scaffold for each ligand are reported in Fig. 7.19.

We can see that, for many systems, the alignment always leads to the correct result. For CDK2 ligands CS18 and CS19 there are clear outliers. For BRD4 ligand 3 and for CDK2 ligands CS20 and CS261 we have a narrow distribution with a median far from 0 Å, indicating that these systems fail consistently, although the few outliers at 0 Å show that the correct answer can be found at times. BRD4 ligand 6 always fails to be aligned properly; the RMSD distribution is quite wide, but there are no outliers at 0 Å. Finally, for BRD4 ligands 4, 7, and 10, and for CDK2 ligands CS246 we have a broad distribution but with the median close to 0 Å or at 0 Å, meaning that the correct alignment is still found more than half of the time. These results confirm that the stochastic alignment procedure—while being far from perfect—seems to work reasonably well for most systems when aligning Murcko scaffolds to their source molecule. However, the smaller size of the Murcko scaffold compared to the whole molecule is likely exacerbating the problem of good volume alignments with poor physicochemical matches already described in Fig. H.26.

7.8.4 Fitting Cyclic Fragments

Since the main issue of `liGAN`'s reconstruction is the suboptimal reconstruction of carbocycles and heterocycles, we want to use our `libmolgrid`-based version of SENSAAS to fit mono- and bi-cyclic fragments to the generated densities. Once fragments are correctly placed, we can remove their density from the generated density and fit the reminder as already performed in section 7.7, where we introduced the Murcko scaffold trick.

Ring systems—especially flat aromatic rings—are common in FBDD and

generated densities does not have this limitation and could therefore be more attractive in industrial settings.

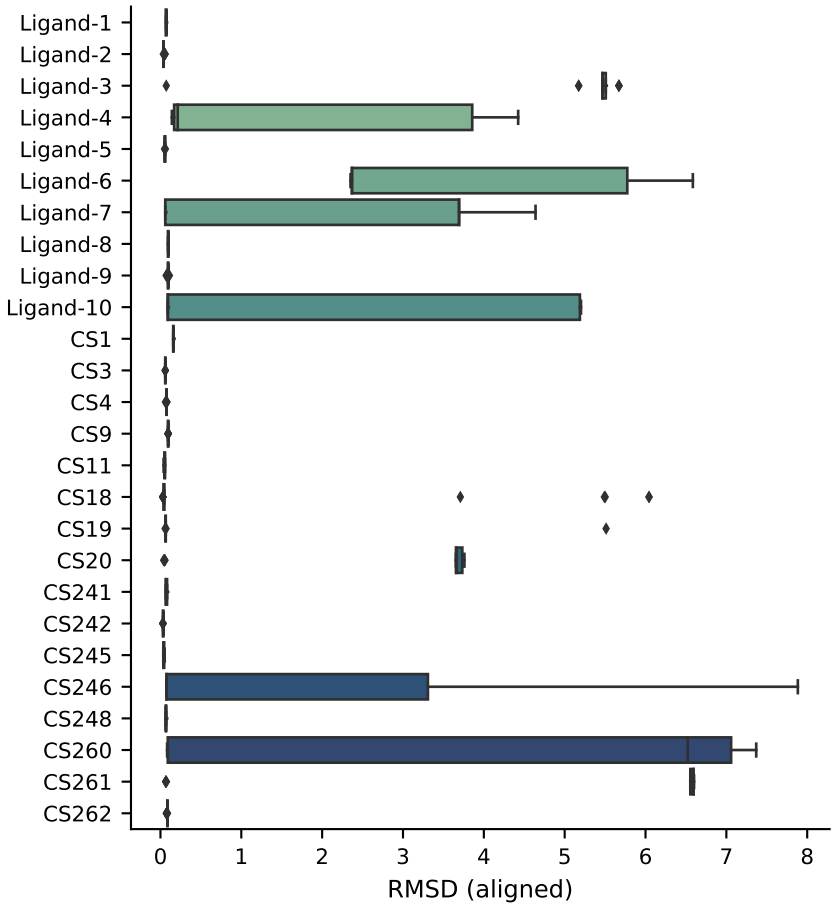


Figure 7.19: RMSD distributions between the original molecule and its aligned Murcko scaffold, obtained over 25 repeats.

lead optimisation (Aldeghi, Malhotra, et al. 2014). The virtual exploratory heterocyclic library (VEHICLe) is an enumeration of 24 847 small aromatic ring systems, 1701 of which were already synthesized at the time of publication (Pitt et al. 2009).

The fragments in the VEHICLe library are scored for synthetic tractability according to a RF model trained on data from published ring systems. The score is the probability of a given fragment being synthetically tractable—according to the model. For reference, benzene has a score $p_{good} = 0.96$, and none of the known ring systems is predicted to be synthetically intractable while 9% of the unknown ring systems are predicted to be tractable (Pitt et al. 2009).

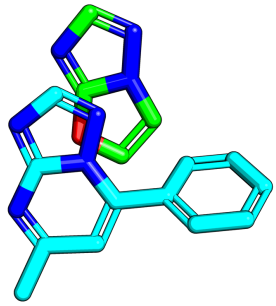
Since the SENSAAS-based alignment is computationally expensive compared with other shape-based methods, we restrict the use of the VEHICLe library to fragments predicted to be synthetically tractable ($p_{good} > 0.5$). This results in a total of 5429 fragments to be aligned with generated densities. The distribution of scores for all compounds in the VEHICLe library is reported in Fig. H.28, where it is clear that the choice of considering only predicted synthetically tractable compounds considerably reduces the computational burden. However, with an alignment time of about 0.5 s per fragment (using GPU acceleration in Open3D), aligning all fragments to a single generated molecule remains computationally expensive. This is particularly true when many generated densities per seed are produced, and when the seed contains multiple cyclic fragments (for which an iterative procedure with multiple alignments or clustering might be necessary), resulting in a combinatorial explosion.

To get 3D coordinates for the fragments to be aligned to real or generated densities, a single conformer for each fragment was generated using the ETKDGv3 conformer generator (Riniker and Landrum 2015; Wang, Witek, et al. 2020) available in RDKit and the point cloud representation of each fragment was pre-computed using the `libmolgrid` colouring scheme. The densities of the inhibitors of Fig. 7.3 and Fig. 7.4 were also pre-computed, to avoid repeated calculations.

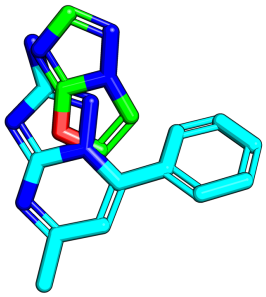
Fig. 7.20 and Fig. 7.21 show the top alignment for each BRD4 ligand (for ligand 1 to 6 and for ligand 7 to 10, respectively), while Fig. 7.22 shows the top alignment for each CDK2 ligand.

We can immediately see that despite the reasonable performance when aligning whole molecules or the Murcko scaffolds, the performance of fragment alignment is rather poor. If we look at the top 3 poses, the situation is similar, indicating that it is not possible to reliably find a good alignment amongst the top poses for small fragments of the VEHICLe library.

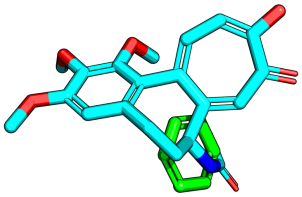
It is not clear, however, if this is a deficiency of the alignment method or of the density representation. To elucidate this, we performed the alignment of



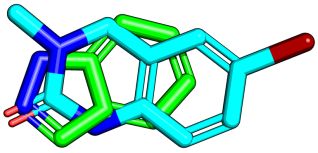
(a) Ligand 1



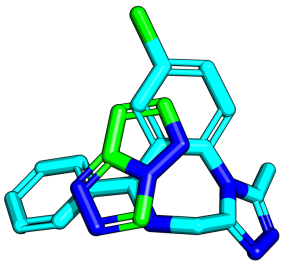
(b) Ligand 2



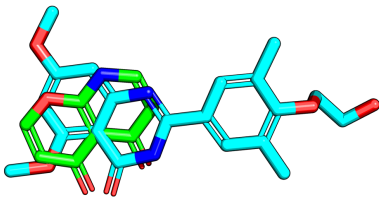
(c) Ligand 3



(d) Ligand 4

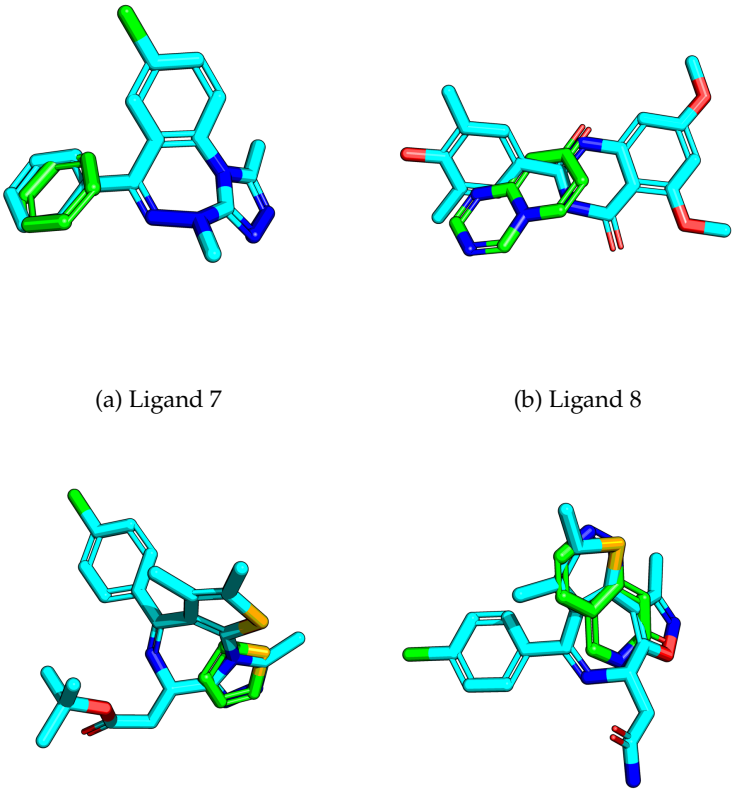


(e) Ligand 5



(f) Ligand 6

Figure 7.20: Best alignments of VEHICLe fragments to BRD4 inhibitors (ligands 1 to 6) using `libmolgrid` densities.



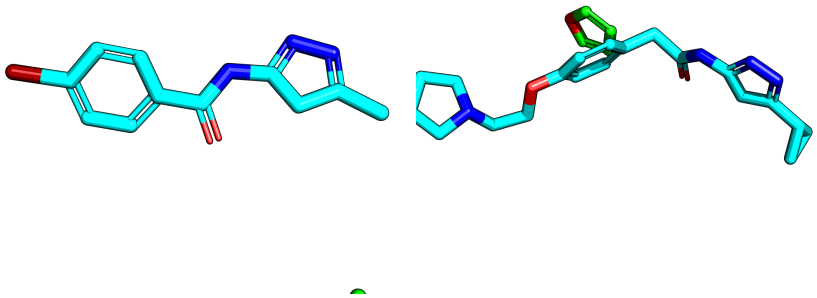
(a) Ligand 7

(b) Ligand 8

(c) Ligand 9

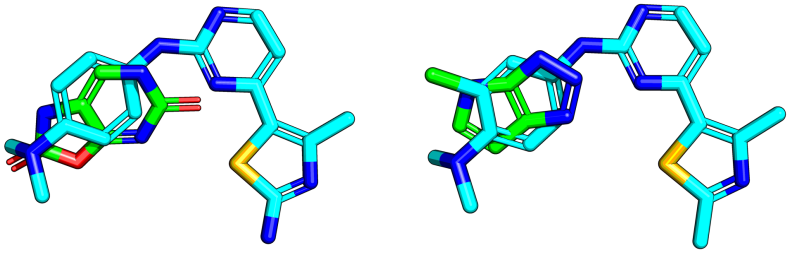
(d) Ligand 10

Figure 7.21: Best alignments of VEHICLe fragments to BRD4 inhibitors (ligands 7 to 10) using `libmolgrid` densities.



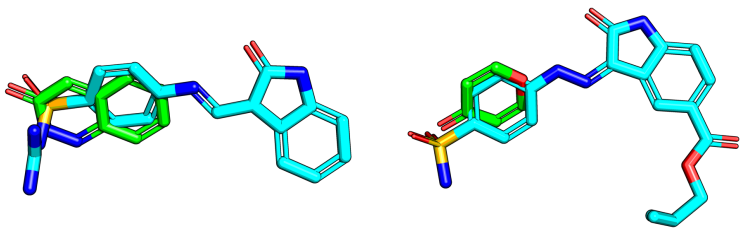
(a) Ligand CS1

(b) Ligand CS11



(c) Ligand CS18

(d) Ligand CS20



(e) Ligand CS241

(f) Ligand CS262

Figure 7.22: Best alignments of VEHICLe fragments to CDK2 inhibitors using `libmolgrid` densities.

fragments in the VEHICLe library using the original SENSAAS implementation based on `nsc` point clouds. Given the multiscale nature of SENSAAS, which downsamples the `nsc` point clouds at different resolutions and picks the best alignment, this process is computationally more intensive than the alignment to `libmolgrid` densities, for which the resolution is fixed at 0.5 Å.[9]

Our results show that while the original SENSAAS implementation provides tighter volume alignments—thanks to the multiscale nature of the downsampling of `nsc`-generated point clouds—, the final alignments are no better than the ones obtained much more quickly with `libmolgrid` generated densities (see Fig. H.29, Fig. H.30, and Fig. H.31). This suggests that the problem is caused by the size mismatch between the whole molecule point cloud and the fragment point cloud.

Unfortunately, without a near-perfect fitting of fragments to the generated densities, the added fragment would not correspond to the physicochemical properties of the density. Therefore, while the molecules built by linking fragments would solve connectivity issues, they would exacerbate the non-correspondence between the generated density and the final molecule.

7.9 Discussion

In this chapter we explored the density-based generative models developed by Ragoza, Masuda, et al. (2020) and Masuda et al. (2020), which build on the density representation used in CNN SFs (Ragoza, Hochuli, et al. 2017) discussed and exploited in Chapter 4. By extensively evaluating the methods for two specific targets of pharmaceutical interest—BRD4 and CDK2—, we highlighted the severe shortcomings of the reconstruction of molecules from generated densities. While the distributions of molecular properties look reasonable, the generated structures—while being "valid"—are not drug-like and contain many fused three- and four-member rings. By applying stringent filters on the QED and SA, only a few of the generated molecules remain. Visual inspection of generated densities from which the molecules are reconstructed indicates that most of the issues come from the reconstruction process. While generated densities look reasonable, the atom fitting procedure positions atoms in a way that makes it difficult to reconstruct drug-like molecules, and very strained and unphysical structures are often reconstructed instead. This mismatch between generated densities and the corresponding reconstructed molecules prevents

[9]The way SENSAAS is implemented also contributes to the high computational cost. The virtual screening script `meta-sensaas.py` executes `sensaas.py` using `os.system`—instead of importing and using the different Python modules—which in turn executes the external `ncs` code, also using `os.system`. This approach has a considerable overhead and involves a lot of I/O operations.

a fair evaluation of the generative model itself, and it remains unclear if the generated densities are truly complementary to the receptor binding site.

Similar work based on shape-based 3D convolutional autoencoders has been proposed for both ligand-based (Skalic, Jiménez, et al. 2019) and structure-based (Skalic, Sabbadin, et al. 2019) drug-discovery applications. However, in the work of Skalic, Jiménez, et al. (2019) and Skalic, Sabbadin, et al. (2019), the hard problem of reconstructing a molecule from the generated density is circumvented completely: the generated 3D molecular densities are fed to a shape captioning network—combining convolutions and long short-term memory (LSTM) units—that produces a SMILES string as output. While using a NN to further decode the generated density into a molecule is a very interesting concept, producing a SMILES string completely loses the information about the 3D molecular conformation. It would be interesting to develop a "captioning" network that builds a three-dimensional structure given a generated density—or directly from the latent space—, but this is likely to be challenging and was outside the scope of this study.

During our evaluation, we tried different strategies to fix or palliate the issues with molecular reconstruction in the original `liGAN` implementation of Masuda et al. (2020). First, we exploited the Murcko scaffold of the seed molecule to keep the core fixed and only perturb the functional groups attached to the scaffold. This method works well in generating drug-like and synthetically accessible molecules since the scaffold of real inhibitors is retained. However, this method presents major drawbacks: a considerable number of generated molecules match the seed molecule—resulting in wasted computations—, it can't be applied to prior sampling—arguably the most interesting application of structure-based generative models—, and does not allow the generation of novel scaffolds thus preventing *scaffold hopping* (Böhm, Flohr, et al. 2004). Additionally, while the method produces molecules with a better CNNscore and a better CNNaffinity than the original method, the predicted binding affinities remain lower, on average, than the ones of the seed molecules. This indicates that the modifications to the side chains are not particularly interesting from a medicinal chemistry perspective since the model might have failed to learn important protein-ligand interactions. However, it is worth bearing in mind that this problem is exacerbated by the fact that the seed molecules are real inhibitors, and therefore represent a challenging baseline.

To increase the novelty of generated molecules and circumvent the limitations of the scaffold trick, we tried to fit a fragment library to the generated density. This is a very challenging task since there are no atomic positions available and therefore common shape-based approaches cannot be applied in these circumstances. Therefore, to align fragments to generated densities we exploited

the registration of coloured point clouds, closely following SENSAAS (Douguet and Payan 2020). While we demonstrated that the SENSAAS methodology works well with `libmolgrid`-generated densities to align whole molecules or Murcko scaffolds to their source, the method does not perform very well with small cyclic fragments from the VEHICLe library. Small cyclic fragments were selected since most of the reconstruction problems outlined above stem from difficulties in building cyclic structures, while substituents attached to the scaffold are easier to fit—as we demonstrated with the scaffold trick. Unfortunately, the two-step nature of the point cloud registration—a coarse shape-based alignment followed by shape- and colour-based refinement—prevents a good alignment based on atom types: the coarse shape-based alignment places the fragment within the generated density, but the refinement procedure is unable to produce very good poses by correcting the good shape overlap to better match pharmacophoric features.

While the one-shot generation of novel molecules conditioned by a receptor structure—and, optionally, a known inhibitor—proposed by Masuda et al. (2020) is a very interesting concept, the outlined issues prevent the method to be useful in current industrial drug discovery pipelines. The reconstruction of molecules from generated densities remains a challenging—and unsolved—problem. Our study indicates that a different typing scheme that explicitly considers aromaticity or ring membership could make the reconstruction easier to perform. The legacy atom types inherited from Gnina, while good for docking, might not be ideal for *de novo* design. Additionally, a more localised density representation and a different aggregation function—such as the maximum, instead of the sum—for overlapping atomic densities should facilitate the fitting of atoms to the generated density, which is arguably the major issue with the current approach.

During this study a new version of `liGAN` was released (Ragoza, Masuda, et al. 2022). This new version uses a different density representation which is indeed more localised and decouples the atom types in elements and other atomic properties—aromatic, HBD, and HBA. From preliminary tests, the new version shows clear shifts towards better values in the distributions of SA and QED for generated molecules, although problems with ring systems remain.

In future work, it would be interesting to see if the point-cloud-based fitting of fragments developed here works better with the new density representation, or if a density representation that can make the fitting easier and more robust can be developed. For the method to become useful in real drug discovery campaigns, the way a molecule is constructed from a generated density needs to be radically improved.

7.10 Conclusions

The automatic generation of novel molecular entities or the transformation of small fragments (hits) into potent inhibitors with desirable pharmacodynamic and pharmacokinetic properties against a target of interest is the ultimate goal of AI applications in 3D *de novo* design (Hadfield and Deane 2022). A lot of effort has been devoted to 2D generative models (Bilodeau et al. 2022; Meyers et al. 2021), and more recently the focus has shifted to the arguably more interesting and more challenging problem of generation of small molecules within a given binding site.

Here we focussed on the work of Masuda et al. (2020), outlining the shortcomings of the density-based one-shot generation which requires a very challenging post-processing step. A new iteration of the model has since been developed (Ragoza, Masuda, et al. 2022). The new iteration of `liGAN` is significantly different from the original version evaluated here. Interestingly, several changes to the density representation—different atom typing and more localised Gaussian atomic densities—have been introduced. However, while a quick evaluation of the new model shows clear shifts towards better values in the QED and SA distributions, significant problems with the connectivity of reconstructed molecules remain. As expected from our study, the hyperparameters of the density representation have a significant impact on the quality of the reconstructed molecules with the current reconstruction scheme.

Other approaches have focussed on a step-wise reconstruction of the ligand within the binding site. Some approaches exploit the successful research in ligand-based generative model and add structure-based information via docking and reinforcement learning (Thomas et al. 2021). Li, Pei, et al. (2021b) combined a 3D ligand-based NN called L-Net (Li, Pei, et al. 2021a) with Monte Carlo tree search (MCTS) to steer the 3D ligand-based molecular generation towards high-affinity binders for a given binding site. Similarly, Xu et al. (2021) developed a sample-and-dock procedure where novel 2D molecular graphs are generated with a junction-tree variational autoencoder (JTVAE) (Jin et al. 2018) from a given molecule, docked to the target of interest, and the best candidate is selected as the new input for the JTVAE so that the procedure is performed iteratively.

The most recent approaches using step-wise molecular reconstruction work directly in 3D. Luo et al. (2021) developed a model that given a binding site estimates the probability density of atoms' positions in 3D space and builds a whole molecule using an autoregressive sampling scheme—atoms are sampled one at a time—, while Powers et al. (2022) developed a method for lead optimisation that expands a small fragment-like molecule using $\mathrm{E}(3)$ equivariant NNs that learn to select fragments from a library and attach them to the current molecule

within the binding site. While fragment-based approaches have the advantage of creating sensible structures (since ring systems are part of the fragments), they are limited by the fragment library they use.

Given the shortcomings that we identified with the `liGAN` method and the growing number of 3D structure-based generative methods being developed, new benchmarks for 3D structure-based generative models are badly needed. In particular, the community needs to focus on easy-to-use and high-quality FOSS implementations (Walters 2020) that can be easily evaluated and reproduced, and on challenging benchmarks—similar to the MOSES (Polykovskiy et al. 2020) and GuacaMol (Brown, Fiscato, et al. 2019) benchmarks for 2D generative models—tailored to 3D structure-based methods. These benchmarks should focus on the evaluation of the 3D molecular structure and connectivity—which needs to be physically sensible—as well as the evaluation of the pose and binding affinity of the generated molecule against targets of pharmaceutical interest. As others have noticed, we should aim to generate molecules that dock well (Cieplinski et al. 2020), and that are easily synthesizable (Gao and Coley 2020).

While the field of 3D structure-based *de novo* design has exciting perspectives, and it is progressing fast, the actual usefulness of the latest models in real drug discovery applications remains to be proven (Walters and Murcko 2020b). Additionally, since the direct generation of good binders with the right pharmacodynamic and pharmacokinetic properties given a target of interest will remain challenging for the foreseeable future, it is important to focus on methods that can be interpreted and used by trained medicinal chemists to have some impact in current drug discovery campaigns.

8

Conclusions

DL, thanks to continuous advances in computer hardware and software, is quickly permeating all branches of drug discovery, and many other scientific disciplines (Pandey et al. 2022). In this work we studied and developed different DL methods with applications in the early stages of SBDD.

First, we extended SOTA CNN-based SFs to docking with flexible residues. Initially, we identified several issues with the code inherited from the standard docking software Smina that were fixed in Gnina—a DL-based docking software—and backported to Smina. Once docking with flexible residues could be performed correctly with classical SFs, we modified Gnina so that the internal SOTA CNN SF would work correctly when docking with flexible residues. **These contributions enabled docking with flexible residues in two widely used molecular docking software: Smina (Koes, Baumgartner, et al. 2013) and Gnina (McNutt et al. 2021).**

With the new capabilities of Gnina at hand, we performed a large-scale comparison of cross-docking with rigid and flexible residues, using the default CNN model carefully selected in McNutt et al. (2021). **This large docking study showed that the advantages of docking with flexible residues are system-dependent and that docking to a rigid receptor is better, on average, when the difference between the cognate and target receptors is small (as expected).** However, by focussing on a smaller data set for which receptor

flexibility has been deemed essential to obtain accurate docking results, we demonstrated the importance of carefully choosing the residues to be treated as flexible with the methodology implemented in Smina and Gnina. **When residues to be treated as flexible are automatically selected with distance-based criteria, the performance does not significantly improve compared to rigid docking. However, when residues to be treated as flexible are carefully selected, it is clear that the performance is much higher and comparable with multistep methods taking into account the global receptor flexibility as well.** This observation calls for simple and effective heuristic (or ML-based) rules to automatically select the most important residues to be treated as flexible, which can be an exciting area of future research.

Following the groundwork done to enable docking with flexible residues with Gnina, we then turned out efforts to the implementation of a CNN-based SF trained explicitly for flexible docking. Therefore, we prepared a suitable data set for training such SFs based on the cross-docking data set of Wierbowski et al. (2019). **By training and evaluating several CNN-based SFs, we demonstrated that combining the annotation for the ligand pose and the pose of the flexible residues is too noisy, and that Gnina performance in cross-docking is not significantly better than the classical SF AutoDock Vina when optimised crystal poses are removed from the data set.** However, when optimised crystal poses are included in the test set, the performance of Gnina is significantly better. This observation explains the good performance of Gnina's default model on cross-docking—when used in the docking pipeline—because it can correctly identify near-native ligand poses.

To experiment with different ideas for CNN-based SFs for docking with flexible residues, we re-implemented the Gnina SF from the ground up in a Python package based on PyTorch and PyTorch-Ignite. **Our `gnina-torch` implementation—which strictly follows software engineering best practices—allows the reproduction of previous results obtained with Gnina, while providing a flexible and extensible implementation based on a modern DL framework—as opposed to the discontinued Caffe framework.** Additionally, thanks to Andrew McNutt who provided the original Caffe weights converted to PyTorch, **`gnina-torch` provides easily accessible (`pip`-installable) pre-trained Gnina models, thus facilitating the use of CNN-based SFs within new or existing software and pipelines**—such as `liGAN`, the FEgrow workflow (Bieniek et al. 2022), or web-based applications. Thanks to the latter feature, `gnina-torch` is as close as possible to an official PyTorch implementation of the Gnina SF, and we hope our software will get widespread adoption.

Exploiting the flexibility of the `gnina-torch` implementation, we built a multi-task model capable of working with separate annotations for the ligand pose and

the pose of the flexible residues. Using these models, we performed an extensive hyperparameter optimisation. We observed that while the annotation for the ligand pose is learnable, the RMSD-based annotation for the flexible residues leads to near-random performance. **Therefore, we concluded that while a RMSD-based annotation is good for the ligand pose, it is too noisy to annotate the pose of flexible residues.** Future research—enabled by `gnina-torch`'s flexibility and extensibility—should focus on alternative annotations based for example on the number of conserved protein-ligand contacts, or interaction FPs.

While CNN SFs significantly improve the results of docking compared to classical SFs (McNutt et al. 2021), they lack rotational invariance and therefore require extensive data augmentation during training, and possibly require averaging over several predictions during inference. To palliate this problem, we developed and evaluated a different architecture that respects rotational and translational invariance by construction. Inspired by the impressive work in the field of NN potentials, we built a SF based on AEVs, which respects such physical constraints. **We demonstrated that our architecture can predict protein-ligand binding affinities as well as or better than other SOTA DL SFs, while being rotationally and translationally invariant, and faster to train.** Concurrent work to ours explored similar ideas for ligand-based models (McCorkindale et al. 2020), confirming that the use of DL architectures and methodologies used in quantum chemistry might be useful for binding affinity predictions as well. Therefore, the early stages of the drug discovery process can benefit from developments in other fields of chemistry and physics, and further exploration in this direction is warranted.

Unfortunately, while our binding affinity prediction method performs comparably with other SOTA DL SFs, it suffers from an over-reliance on ligand features. This is a common problem of DL SFs and currently there is a lot of ongoing research effort on how to reduce this problem with data augmentation techniques (Scantlebury et al. 2020) and how to represent and learn only non-covalent interactions (Volkov et al. 2022). How to better represent interactions with architectures that respect the physical constraints of the problem at hand is an exciting direction for future work. This is an ongoing research stream, aiming to combine AEVs with GNN architectures—similarly to the work of Karlov et al. (2020)—, and with simple protein-ligand interaction features developed by Moesser et al. (2022), which we hope will rely less on ligand features thus leading to more transferrable models.

Finally, in collaboration with Evotec, we explored structure-based generative models. Generative models have the potential to significantly transform drug discovery. While ligand-based generative models have been extensively explored in the past years and are still a very active area of research (Palazzesi and Pozzan

2022), the structure-based generative model developed by Masuda et al. (2020) explored here was one of the first of its kind, generating molecular entities directly within a given binding site. From an industrial perspective, we wanted to understand if such structure-based generative model could be effectively integrated into Evotec's drug discovery pipeline. **Unfortunately, after extensive tests we concluded that most of the molecules generated using the model proposed by Masuda et al. (2020) are not drug-like, and mostly uninteresting from medicinal chemistry perspective.** We linked these problems to the hard step of fitting atoms within the generated densities, and we, therefore, focused on possible fixes for such generative model—without much success. While a new version of the generative model has been developed in the meantime (Ragoza, Masuda, et al. 2022), some preliminary tests indicate that several of the issues identified with the first iteration of the model remain. Evotec will therefore explore alternative structure-based generative models in the future.

A lot of work remains to be done before structure-based generative models can be effectively applied in real drug discovery projects. We identified several issues with the current literature, where only simple and uninformative metrics are often reported. Going forward, it will be extremely important to build challenging benchmarks for structure-based generative models. In particular, it will be important to focus on the synthesizability and the stability of the generated compounds. As we demonstrated here, a validity criterion based on satisfied valency rules as well as a uniqueness criterion based on the molecular graph are rather useless, and they do not allow filtering out completely unreasonable molecules from a physical and a medicinal chemistry perspective. Unbiased and stringent benchmarks especially tailored for structure-based generative models are needed, and building such benchmarks is an interesting research challenge for the future.

In addition to the scientific output of this work, briefly recapitulated above, an impactful by-product is the amount of FOSS software produced, as well as the contributions to widely used FOSS packages.

Contributions to Gnina now allow docking with flexible residues using CNN SFs, which opens up new and interesting uses of Gnina as well as new research directions. Several bugfixes introduced in Gnina have been backported to Smina, thus fixing outstanding issues with a widely used classical docking software.

While one of the greatest features of Gnina is the possibility to use the CNN SF within the docking pipeline—while most other ML/DL SF can only be applied as post-processing—, this work resulted in a fully featured and easily extensible Python package implementing the CNN SF. The PyTorch package implements most of the features of the original Gnina SF—including pre-trained models—while providing the extensibility needed for fast prototyping toward

new research directions.

Out of frustration for the lack of a simple and lightweight package for symmetry-corrected RMSD calculation, we developed `spyrmsd`. Our package is currently being integrated within MDAnalysis, it was very handy when working with ProDy—to compute symmetry corrected RMSD for protein side chains—, and it is easily installable and easy to integrate into other software and existing pipelines.

Given that most of the work presented here is built on top of FOSS, several bugs were identified in existing software packages and subsequently fixed. Additionally, new functionality was also introduced in existing codebases, to facilitate our work. **A non-exhaustive list of the main contributions to FOSS steaming from this work, included in software packages with a large user base, is provided in Appendix A.** While often under-appreciated in academic settings, we believe that FOSS contributions related to this work are as important as the research performed herein, and benefit the large community of molecular modellers.

This page intentionally contains only this sentence.

A

Code Contributions

Contents

FOSS is an essential ingredient of computational sciences and molecular modelling (Pirhadi et al. 2016). Although FOSS comes with some disadvantages in terms of sustainability (Krylov et al. 2015), it often provides researchers with SOTA algorithms and software that can be used, dissected, and improved. This is particularly important for reproducibility, and validation by the scientific community. The foundations of this work lay on FOSS and many contributions steaming from this thesis—both bugfixes and enhancements—were incorporated into different packages and libraries.

This appendix is a collection of all open-source contributions related to this work.

A.1 Open Babel

Open Babel is an open-source library—written in C and C++—allowing to convert, analyse, and store data produced by common molecular modelling software (O'Boyle, Banck, et al. 2011). Open Babel is released under the GNU General Public Licence (GPL).

Contributions to Open Babel are listed below, in inverted chronological order:

6. Make `OBMol::NumHvyAtoms` `const`-qualified (PR #2300)
5. Add `GetNumHvyAtoms` to `OBResidue` (PR #2299)
4. Fix GitHub "Issue" template (PR #2082)
3. Ignore in-source installation (PR #2027)
2. Fix PDB and PDBQT insertion codes handling (PR #1998)
1. Fix CMake problem with FindRapidJSON (PR #1988)

A.2 `libmolgrid`

`libmolgrid` is a GPU-accelerated library to compute atomic densities on a grid from molecular data (Sunseri and Koes 2020). `libmolgrid` is written in C++/CUDA and provides Python bindings that allow flawless integration with SOTA DL libraries such as PyTorch (Paszke et al. 2019), Caffe (Jia et al. 2014) or TensorFlow (Abadi et al. 2015; Chollet et al. 2015).

Contributions to `libmolgrid` are listed below, in inverted chronological order:

3. Add PyTest `conftest.py` file (PR #49)
2. Add support for Open Babel 3 (PR #17)
1. Fix supported GPU architectures according to CUDA version (PR #5)

A.3 Gnina

Gnina is an open-source DL framework for molecular docking (Ragoza, Hochuli, et al. 2017). Gnina is based on Smina (Koes, Baumgartner, et al. 2013) which is itself a fork of AutoDock Vina (Trott and Olson 2009). Gnina combines docking capabilities together with the molecular gridding library `libmolgrid` (Sunseri and Koes 2020) and the Caffe DL framework (Jia et al. 2014) to provide novel DL SFs based on CNNs.

Contributions to Gnina are listed below, in inverted chronological order:

19. Check number of models in original receptor (PR #128)
18. Fix Caffe's `draw.py` (PR #121)
17. Fix compilation warnings (PR #109)
16. Write test output locally (PR #108)
15. Remove output of flexible docking test (PR #106)
14. Check number of atoms in flexible residues (PR #105)
13. Correctly append `PYTHONPATH` for `Caffe` test (PR #101)
12. Add `flex_max` keyword to set a soft limit on the number of flexible residues (PR #95)
11. Add `flex_limit` keyword to set a hard limit on the number of flexible residues (PR #94)
10. Remove CMake warning for Open Babel 2 (PR #92)
9. Add support for Open Babel 3 (PR #82)
8. Add missing serialisation for `struct residue` (PR #74)
7. Enable optimisation of flexible side chains (PR #73)
6. Fix gradients visualisation in `gninavis` (PR #72)
5. Add PDB insertion codes support for `makeflex.py` (PR #65)
4. Support PDB files without "elements" column in `makeflex.py` with automatic element perception (PR #64)
3. Fix `gninacheck` test invocation for new versions of Boost C++ (PR #62)
2. README description of `makeflex.py` script (PR #61)
1. Fixes for `makeflex.py` (PR #60)

All contributions to GNINA's scripts are listed below, in inverted chronological order:

4. Add lower threshold for PyMol arrows (PR #31)
3. Low memory `combine_rows.py` (PR #30)
2. Open files in context (PR #29)
1. Fixes for clustering pipeline (#26)

All contributions to the analysis pipeline for the GNINA 1.0 publication (McNutt et al. 2021) are listed below, in inverted chronological order:

5. Explain flexible docking pipeline (PR #5)
4. Flexible docking analysis (PR #4)
3. Flexible docking: methods (PR #3)
2. Specify `usecol` type in RMSD plotting script (PR #2)
1. `coalescer.py` with empty values (PR #1)

A.4 MDAnalysis

MDAnalysis is an open-source library written in Python for the analysis of MD simulations stored in many popular file formats (Gowers et al. 2016; Michaud-Agrawal et al. 2011).

Contributions to MDAnalysis are listed below, in inverted chronological order:

29. Remove OpenMP private clauses by narrowing variables' scope (PR #3706)
28. Use `Results` class for `WaterBridge` analysis (PR #3287)
27. MOL2 parser populates elements attribute (PR #3063)
26. Write CONECT record only once in multi-MODEL PDB files (PR #3020)
25. Fix documentation and examples for `rotateby` (PR #2903)
24. Ignore `duecredit` output (PR #2697)
23. Use user-provided REMARK argument in `class XYZWriter` (PR #2694)
22. Fix issue preventing to write PDB files if cell dimensions were unset (PR #2685)
21. Remove spurious `icode` variable and fix PEP8 for `class PDBParser` (PR #2677)
20. Fix similarity and connectivity selections by using residue indices instead of `resids` (PR #2673)
19. Improve `auxiliary` coverage (PR #2570)
18. Ignore local HTML coverage report (PR #2568)

17. Use `import seaborn` instead of `import seaborn.apionly` (PR #2556)
16. Use temporary directory when testing `align.AlignTraj` (PR #2534)
15. Standardise `select` keyword (PR #2528 and PR #2531)
14. Ignore `TypeError` when `class PDBWriter` is trying to remove a `class StringIO` object with invalid coordinates, so that the correct exception can be raised instead (PR #2518)
13. Added a `min_mass` parameter (set to 0.9 by default) to `guess_hydrogens` (PR #2516)
12. Cap `seaborn` version for Python 2 support (PR #2499)
11. Fix upstream deprecation in `matplotlib` (PR #2498)
10. Remove old module `MDAnalysis.migration` (PR #2496)
9. Remove explicit use of `Bio.Alphabet`'s `self.atoms.names` (PR #2457)
8. Add `self.elements` attribute to `class XYZParser` and use `self.elements` in `class XYZWriter` with fallback to `self.atoms.names` (PR #2456)
7. Fix indentation problems (PR #2454)
6. Added citation to `__init__.py` docstring (PR #2451)
5. Fixed wrong variable in `auxiliary` module (#2411)
4. Fix variable name typo (PR #2409)
3. Try upper case atom names when guessing mass (PR #2331)
2. Fix `class PDBWriter` and `class PDBReader` newlines for PDB header (PR #2325)
1. Ignore `.mol2` status bit strings, if present (PR #2319)

Additional contribution to the MDAnalysis User Guide are listed below, in inverted chronological order:

4. Improve development environment installation instructions (PR #203)
3. Add more details on dependencies (PR #64)
2. Improve quick start instructions (PR #58)
1. Improve installation instructions (PR #57)

A.5 TorchANI

TorchANI is a PyTorch implementation of the ANI family of NNPs (Gao, Ramezanghorbani, et al. 2020; Smith, Isayev, et al. 2017).

Contributions to TorchANI are listed below, in inverted chronological order:

3. Use `torch.div` instead of `//` (PR #615)
2. Swap custom `class AdamW(torch.optim.optimizer)` for the equivalent class in PyTorch (PR #464)
1. Fix device for `class SpeciesConverter`'s forward pass (PR #462)

B

Symmetry-Corrected RMSD Calculations in Python

Contents

R. Meli and P. C. Biggin, "spyrmsd: symmetry-corrected RMSD calculations in Python", *J. Cheminform.* 12, 49 (2020). DOI: 10.1186/s13321-020-00455-2 (Meli and Biggin 2020)

RMSD calculations play a fundamental role in the comparison of different conformers of the same ligand. This is particularly important in the evaluation of protein-ligand docking, where different ligand poses are generated by docking software and their quality is often assessed by RMSD calculations. Unfortunately, many tools for RMSD calculations do not take into account the symmetry of

the molecule, remain difficult to integrate flawlessly in cheminformatics and ML pipelines—which are often written in Python—or are shipped within large code bases. Here we present a new FOSS tools for RMSD calculations written in Python, designed to be extremely lightweight and easy to integrate into existing software.

B.1 Introduction

Computational SBDD has steadily gained traction partially thanks to the constant improvements in available software, now often free and open source—especially when developed in academia. Protein-ligand docking in particular is now a standard tool employed in the early stages of drug discovery pipelines to screen possible drugs acting on a known target of interest. See Chapter 2 for a brief introduction of protein-ligand docking.

The performance of docking programs is often assessed by their ability to reproduce the crystallographic pose of the bound ligand. A common metric to evaluate the difference between the predicted binding pose and the crystallographic pose is the heavy-atoms RMSD (Mukherjee et al. 2010), although other metrics have been suggested (Leung et al. 2019). RMSD calculations are also used in other contexts, for example for the evaluation of diversity in generated conformers (O'Boyle, Banck, et al. 2011).

Many simple scripts to compute RMSDs are based on the assumption of a direct one-to-one mapping between atoms of different conformers of the same ligand. In different words, atoms are often assumed to be labelled according to their position in a coordinate file (or data structure) and they are paired according to such label. This assumption breaks down when such labels are not conserved—i.e. the order of atoms is different in the two structures being compared—and/or for symmetric molecules. In the case of symmetric molecules, different binding poses can be chemically identical but different in terms of atom-atom mapping. Since molecular connectivity is naturally represented by graphs—atoms as vertices and bonds as edges—, tools from graph theory can be used to obtain the correct atom-atom mapping for two different conformers of the same molecule, thus avoiding the problems outlined above.

Here we present a new Python tool, called `spyrmsd`, for the calculation of symmetry-corrected RMSDs based on graph isomorphisms.

B.2 Implementation

`spyrmsd` is implemented in pure Python. Therefore, it is easy to integrate in current Python libraries and Python pipelines, particularly common in cheminformatics and ML projects. In this section we describe the implementation of the different types of RMSD calculations available in `spyrmsd`, their use, and their shortcomings.

B.2.1 Standard RMSD

Let us call $\mathbf{A}$ and $\mathbf{B}$ the $N \times 3$ matrices of atomic coordinates of two conformers A and B of the same molecule. The standard RMSD is simply defined as

$$\mathrm{RMSD}_{\mathrm{standard}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{3}(A_{ij} - B_{ij})^2}. \tag{B.1}$$

If we define the displacement $\mathbf{r}_i = \mathbf{a}_i - \mathbf{b}_i$—where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$ and $\mathbf{b}_i$ is the i-th row of $\mathbf{B}$—the standard RMSD can be written more compactly as

$$\mathrm{RMSD}_{\mathrm{standard}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbf{r}_i^2}. \tag{B.2}$$

This simple formula, which assumes the atomic coordinates to be provided in the same order for both conformers, is easy to compute. In `spyrmsd` the calculation of $\mathrm{RMSD}_{\mathrm{standard}}$ is vectorised using `numpy` (Harris et al. 2020).

A serious drawback of standard RMSD calculations is that they do not take into account molecular symmetry. This is problematic since atoms are intrinsically indistinguishable and therefore symmetry operations conserve molecular properties. Fig. B.1 shows the atom-atom mapping for benzene with and without symmetry correction after a mirror operation; it is clear that a simple positional atom-atom mapping will lead to artificially inflated results. Symmetry corrections would lead to the correct result expected with indistinguishable atoms.

B.2.2 Quaternion Characteristic Polynomial Method

Standard RMSD calculations take into account the possible translations between the two conformers. To measure conformational similarity—and neglect translations—the RMSD can be computed on optimally superimposed structures. This minimised RMSD for a pair of molecules can be computed efficiently using the quaternion characteristic polynomial (QCP) method (Theobald 2005), which

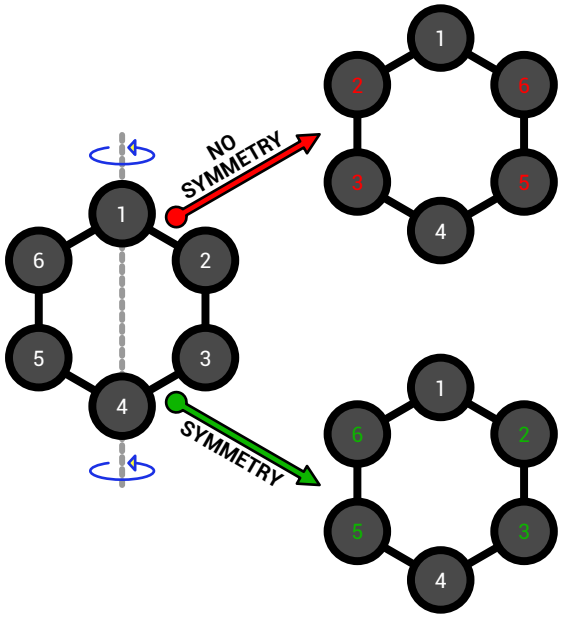


Figure B.1: Atom-atom mapping for the benzene molecule after a mirror operation with and without symmetry correction.

circumvent the need of finding orthogonal (rigid-body) rotations and special considerations for edge cases.

The QCP methods is based on the calculation a 4×4 symmetric key matrix (Theobald 2005)

$$\mathbf{K} = \begin{pmatrix} S_{00}+S_{11}+S_{22} & S_{12}-S_{21} & S_{20}-S_{02} & S_{01}-S_{10} \\ & S_{00}-S_{11}-S_{22} & S_{01}+S_{10} & S_{20}+S_{02} \\ & & -S_{00}+S_{11}-S_{22} & S_{12}+S_{21} \\ & & & -S_{00}-S_{11}+S_{22} \end{pmatrix}, \tag{B.3}$$

where

$$S_{ij} = \sum_{k=1}^{N} B_{ki} A_{kj}. \tag{B.4}$$

The minimum RMSD is then given by (Theobald 2005)

$$\mathrm{RMSD}_{\min} = \sqrt{\frac{G_A + G_B - 2\lambda_{\max}}{N}}, \tag{B.5}$$

where $G_A = \mathrm{Tr}\,\mathbf{A}^T\mathbf{A}$, $G_B = \mathrm{Tr}\,\mathbf{B}^T\mathbf{B}$ and $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{K}$. The eigenvalues of $\mathbf{K}$ can be obtained by finding the roots of the characteristic polynomial $P(\lambda) = \det(\mathbf{K} - \lambda\mathbf{I})$, where $\mathbf{I}$ is the identity matrix. For the matrix $\mathbf{K}$ the characteristic polynomial is given by (Theobald 2005)

$$P(\lambda) = \lambda^4 + C_2\lambda^2 + C_1\lambda + C_0, \tag{B.6}$$

where $C_2 = -2 \operatorname{Tr} \mathbf{M}^T \mathbf{M}$ (with $\mathbf{M} = \mathbf{B}^T \mathbf{A}$), $C_1 = -8 \det \mathbf{M}$ and $C_0 = \det \mathbf{K}$. The largest characteristic polynomial root $P(\lambda_{max}) = 0$ can be efficiently computed using the Newton-Raphson method (Quarteroni et al. 2007) starting from the initial guess $(G_A + G_B)/2$ (Theobald 2005).

Care should be taken when the two molecules A and B overlap perfectly. In such case, $\lambda_{max} = (G_A + G_B)/2$ and therefore the term $G_A + G_B - 2\lambda_{max}$ in Eq. (B.5) can become negative due to numerical errors.

In `spyrmsd` the solution of the characteristic polynomial equation $P(\lambda_{max}) = 0$ is based on the Newton-Raphson method implemented in `scipy` (Virtanen et al. 2020) while other vector and matrix operations are vectorised using `numpy`.

B.2.3 Hungarian Algorithm for Symmetry Correction

The Hungarian algorithm (Kuhn 1955; Munkres 1957) is an algorithm to solve the linear weight assignment problem (Ignazio and Cavalier 1993)—also known as minimum weight matching in bipartite graphs—and has been previously proposed as a method to introduce symmetry corrections in RMSD calculations (Allen and Rizzo 2014). If $\mathbf{D}$ is the $N \times N$ matrix of squared pairwise distances between all atoms of the conformer A to all atoms of the conformer B, the linear weight assignment problem consists in finding the assignment matrix $\mathbf{X}$ that minimises the assignment cost $\sum_{ij} D_{ij} X_{ij}$, where $X_{ij} = 1$ if and only if atom i of conformer A is assigned to atom j of conformer B. The RMSD computed using the Hungarian algorithm is therefore given by

$$\sqrt{\frac{1}{N} \min_{\mathbf{X}} \sum_{ij} D_{ij} X_{ij}}, \tag{B.7}$$

under the constraint that each row is assigned to exactly one column and each column to exactly one row. This definition is however problematic, since the solution of the assignment problem could end up pairing atoms of different elements. To avoid this drawback, the assignment problem is solved for every element separately (Allen and Rizzo 2014). If $\mathbf{D}^e$ is the $N_e \times N_e$ matrix of squared pairwise distances between atoms of element e of conformer A to atoms of element e of the conformer B, the RMSD computed using the Hungarian algorithm is given by

$$\mathrm{RMSD}_{\mathrm{Hungarian}} = \sqrt{\frac{1}{N} \sum_{e} \min_{\mathbf{X}^e} \sum_{ij} D^e_{ij} X^e_{ij}}, \tag{B.8}$$

where $X^e_{ij} = 1$ if and only if atom i of element e in conformer A is assigned to atom j of element e in conformer B and were the sum on e runs over all elements

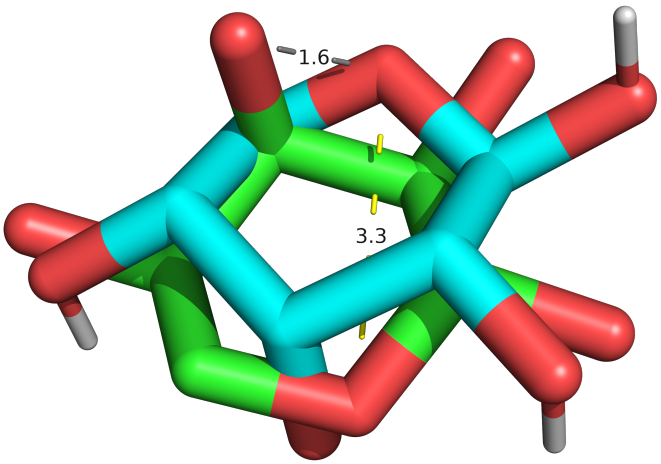


Figure B.2: Crystal pose (green) and second-best docking pose (cyan) for the ligand of the protein-ligand complex 1DRJ. The Hungarian method assigns the ring oxygen to an oxygen atom nearby ($d = 1.6$ Å, grey) while the graph isomorphism method correctly maps one ring oxygen to the other ($d = 3.3$ Å, yellow). The Hungarian method results in an artificially low RMSD of 1.00 Å, compared to the correct RMSD of 2.46 Å obtained with the graph isomorphisms method.

of the molecule.

Even when the Hungarian algorithm is used to assign atoms of the same element, problems can arise from the fact that the algorithm is not aware of the overall molecular structure. This could result in unphysical assignments which break the molecular graph and result in artificially low RMSD values (Bell and Zhang 2019). Fig. B.2 shows a simple situation where unphysical assignments arise and lead to a RMSD value lower than the correct one.

B.2.4 Graph Isomorphisms for Symmetry Correction

To overcome the problems of the Hungarian algorithm, tools from graph theory can be borrowed to perform an optimal atom-atom assignment based on graph isomorphisms. A molecule can be represented as a graph $\mathcal{G}(V, E)$—hereafter referred to molecular graph—where the vertices V are associated to atoms and the edges E are associated to bonds. If two conformers A and B are represented

by graphs $\mathcal{G}_A$ and $\mathcal{G}_B$, respectively, the mapping of atoms of molecule A to atoms of molecule B becomes a graph isomorphism problem. An isomorphism between graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ is a bijective mapping of the vertices of graph A to vertices of graph B that preserves the edge structure of the graphs (molecular connectivity in the case of molecular graphs). With the bijective mapping connecting vertices (atoms) of $\mathcal{G}_A$ to vertices (atoms) of $\mathcal{G}_B$ the RMSD between the two molecules can be computed using the standard RMSD formulation of Eq. (B.2).

`spyrmsd` can leverage `networkx` (Hagberg et al. 2008) or `graph-tool` (Peixoto 2014) for graph representation and graph matching. All possible graph isomorphisms are computed using the VF2 algorithm (Cordella et al. 2004) and the lowest RMSD among all isomorphisms is retained.

The graph isomorphism problem is a non-polynomial (NP) problem (Shamir and Tsur 1999) and therefore symmetry-corrected RMSD calculations are only suited for small to medium-sized molecules. To improve speed, graph isomorphisms are cached by default when computing the RMSD between multiple conformations of the same molecule.

B.2.5 The VF2 Algorithm for Graph Isomorphism

There are several algorithms to find if two graphs are isomorphic and obtain all possible isomorphisms (Lee et al. 2012). Arguably the most common one is the VF2 algorithm (Cordella et al. 2004; Cordella et al. 2001).

Matching two graphs $\mathcal{G}_A(V_A, E_A)$ and $\mathcal{G}_B(V_B, E_B)$ consists in finding a mapping M between the nodes of $\mathcal{G}_A$ and the nodes of $\mathcal{G}_B$—usually expressed as the set of pairs (a, b) with $a \in \mathcal{G}_A$ and $b \in \mathcal{G}_B$. The mapping M is called an *isomorphism* if and only if the mapping is bijective and preserves the edge structure of the two graphs.

Finding a graph isomorphism can be expressed in terms of a partial mapping (partial solution) $M(s)$, where s is the current state of the matching process. The successor state s' consists in the addition of a pair of matched nodes (a, b) to the partial graphs associated with the state s. Nodes are added if and only if the new state is consistent with the graph isomorphism, that is, if $\mathcal{G}_A(s')$ and $\mathcal{G}_B(s')$ associated to $M(s')$ are isomorphic.

The VF2 algorithm uses a set of rules to verify the consistency of the successor state, combined with k-look-ahead rules that allow to check in advance if a consistent state s' has no consistent successors after k steps. This look-ahead allows to quickly prune steps that would lead to inconsistency. The feasibility rules are encoded as a feasibility function $F(a, b, s)$ accounting for both *syntactic feasibility*—which depends only on the structure of the graphs—and *semantic*

feasibility—which takes into account node and edge attributes.[1]

The VF2 algorithm starts from an initial state s_0 corresponding to an empty mapping $M(s_0) = \emptyset$ and for each state s the set $P(s)$ of possible matched node pairs candidates to be added to the current state s to obtain the successor state s' is computed. For each pair $(a, b) \in P(s)$, $F(a, b, s)$ is evaluated and if the addition is feasible—also considering the look-ahead—, then the successor state is obtained as $s' = s \cup (a, b)$. The procedure is repeated for each new state and terminated when $M(s)$ covers all nodes of $\mathcal{G}_A$ and $\mathcal{G}_B$.

B.2.6 Application Programming Interface

The main module of `spyrmsd` is the `rmsd` module, where all the high-level RMSD functions are implemented. The following functions are available to the user:

- `rmsd` for the computation of the standard RMSD,
- `hrmsd` for the computation of RMSD using the Hungarian algorithm,
- `symmrmsd` for the computation of symmetry-corrected RMSD,

`symmrmsd` should always be used for small molecules, to get the right symmetry-corrected RMSD. `rmsd` is provided to compute the standard RMSD when symmetry does not play a role (or when the molecular graph is too large to efficiently apply symmetry-corrections) and atoms are listed in the same order. `hrmsd` is provided for comparison with existing implementations and should not be used otherwise, because of the problems outlined above.

The minimum RMSD—computed using the QCP method (Theobald 2005)—can be obtained with the keyword `minimize=True`, with and without symmetry-corrections.

`spyrmsd` is designed to be easily integrated in existing Python libraries or pipelines. For this reason, the application programming interface (API) is minimalistic: only atomic coordinates and atomic numbers (`rmsd` and `hrmsd`) and molecular adjacency matrices (`symmrmsd`) have to be passed to RMSD functions in the form of `numpy` arrays. This simple API make `spyrmsd` completely agnostic of the way molecules are stored in different software, as long as they can provide the minimal information required.

B.2.7 Standalone RMSD Tool

`spyrmsd` also offers a standalone RMSD tool as a CLI exposing the functionality of the `rmsd` and `symmrmsd` functions. The `hrmsd` function is not exposed in the CLI,

[1] In the case of molecular graphs, node attributes correspond to elements or, more generally, atom types.

to avoid erroneous calculations.

Molecular input is handled by Open Babel (O'Boyle, Banck, et al. 2011)—via its Python interface `pybel` (O'Boyle, Morley, et al. 2008)—, or RDKit (Landrum 2022). Such packages are also responsible for building the adjacency matrix representing the molecular graph.

Open Babel's own RMSD calculation tool, `obrms`, is expected to be faster than `spyrmsd` as a standalone tool since it does not have any Python overhead.

B.3 Results

B.3.1 Testing

To test the correctness of `spyrmsd` against Open Babel's `obrms` we re-docked the PDBbind refined set (Liu, Li, et al. 2014; Wang, Fang, Lu, and Wang 2004) with Smina (Koes, Baumgartner, et al. 2013) to generate different ligand conformations. Smina was run using the default settings with protein PDB files stripped of water molecules. The top ten binding poses were retained, resulting in a total of 40431 different conformations. The RMSD of each docked pose with respect to the crystal pose was computed using symmetry-corrected RMSD with and without minimisation (using the QCP method).

To elucidate the relationship between RMSDs obtained using `spyrmsd` and `obrms` with and without minimisation we computed the Pearson's correlation coefficient and the maximum absolute error. The RMSDs computed with the two software correlates perfectly (Pearson's correlation coefficient of 1.00) and present a maximum absolute error of 5×10^{-5} Å. This gives us great confidence that the two independent implementations are equivalent.[2]

A comparison between the Hungarian method and symmetry-corrected RMSD obtained via graph isomorphisms is presented in Fig. B.3. As previously pointed out, the Hungarian method can result in assignments incompatible with the molecular connectivity and therefore lead to artificially low RMSD values (Bell and Zhang 2019). Therefore, the `hrmsd` function is provided only for comparison with existing software and should not be used otherwise.

B.3.2 Speed

By design, `spyrmsd` is written fully in Python. While it leverages fast libraries that are easy to install (using the `pip` or `conda` package managers), there is some

[2]The RMSD of the ligand poses obtained in Chapter 4 and Chapter 5 were also computed with both `spyrmsd` and `obrms`, where we obtained again a perfect correlation and very small numerical errors.

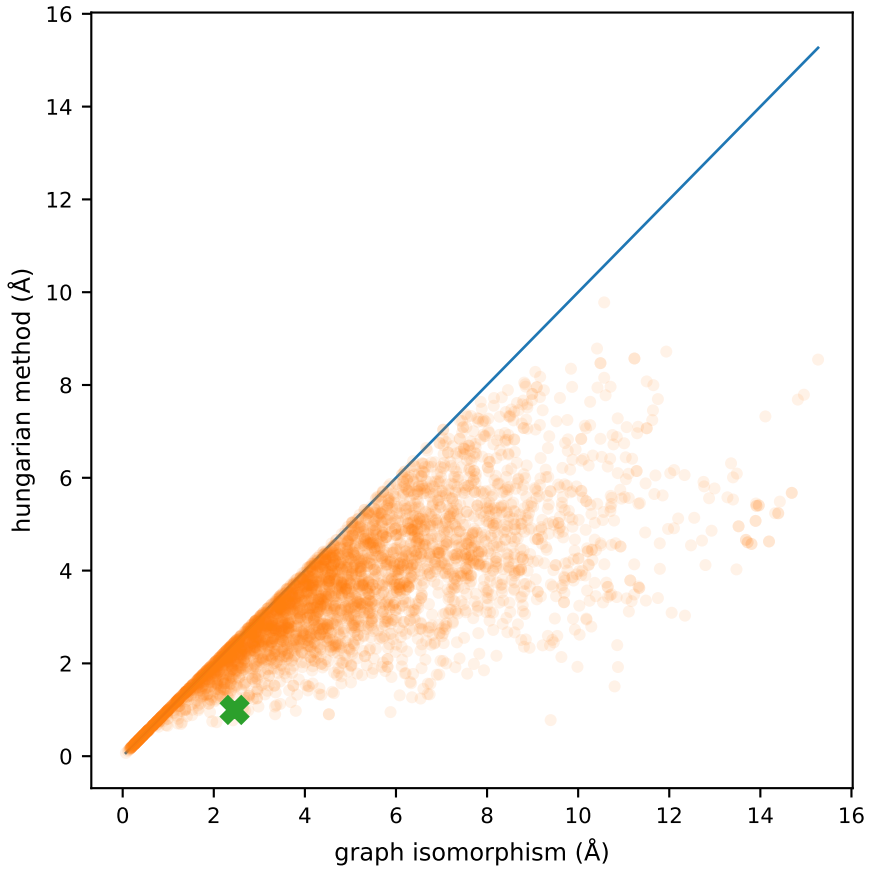


Figure B.3: Comparison between symmetry corrections performed with the Hungarian method or leveraging graph isomorphisms. The Hungarian algorithm often results in artificially low RMSDs because of atom-atom assignments breaking the molecular graph. The green cross corresponds to the protein-ligand complex 1DRJ analysed in Fig. B.2.

overhead compared to the most efficient implementations in other compiled libraries.

Fig. B.4a shows a speed comparison between `spyrmsd` and `obrms` for 100 randomly selected systems. Error bars are obtained by repeating the measurements 25 times. `spyrmsd` is usually comparable or an order of magnitude slower than `obrms`. This is expected since Python comes with some overhead compared to compiled code. The difference between `graph-tool` and `networkx` backends is more difficult to elucidate: `graph-tool` seems to be generally slightly faster, but `networkx` has clearly more variation from system to system (see Fig. B.4b).[3]

B.4 Discussion

Despite being somewhat slower than other SOTA tools for RMSD calculation, we believe that `spyrmsd` could be extremely useful to the community: it is a lightweight tool with focussed functionality, it is easy to use and integrate in existing Python codebases and pipelines, and it is easy to install via popular package managers.

B.4.1 Easy Installation

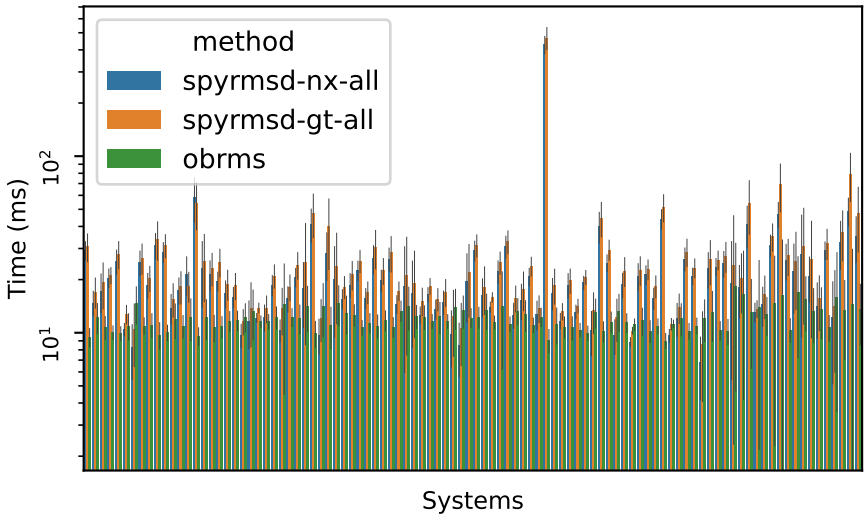
`spyrmsd` is available on the Python Package Index (PyPI) and via the `conda` package manager on the conda-forge channel. This provides easy cross-platform installation of `spyrmsd` and all its dependencies to work as a library (with `networkx`). On macOS and Linux, users can get some speed improvement by installing `graph-tool`, which is also available via the `conda` package manager.

To use `spyrmsd` as a standalone tool, users will have to install either Open Babel or RDKit with their preferred installation method.

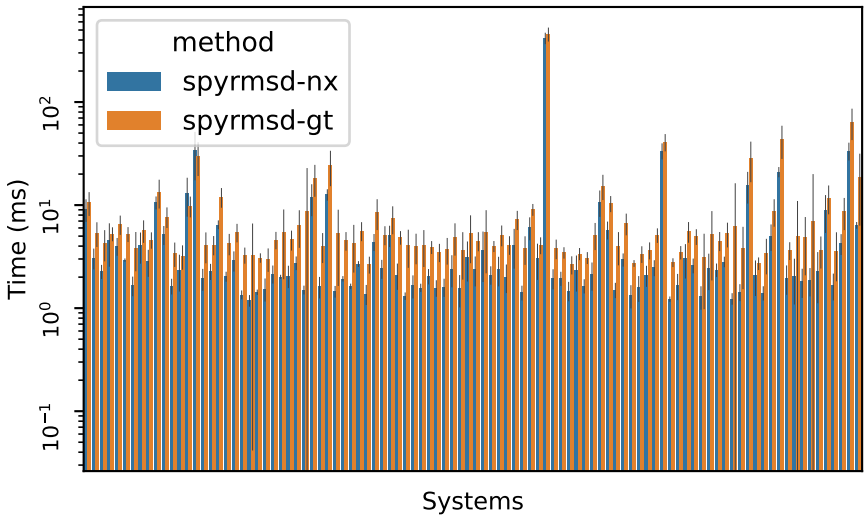
B.4.2 Easy Integration in Existing Libraries

`spyrmsd` is easy to integrate to existing pipelines thanks to its clean and simple API. Standard RMSD calculations require atomic coordinates and atomic numbers only, while symmetry-corrected RMSD calculations also require adjacency matrices to compute graph isomorphisms. Atomic coordinates and atomic numbers are usually readily available in most Python libraries dealing with molecular file formats, while the adjacency matrix of a molecule is easy to build from bond connectivity.

[3] Benchmarking was performed on an Apple MacBook Pro (macOS 10.15) with a 2.6 GHz 6-Core Intel Core i7 processor and 32 GB of 2400 MHz DDR4 memory.



(a) `spyrmsd` versus `obrms`



(b) `networkx` versus `graph-tool`

Figure B.4: RMSD calculation time for 100 randomly selected systems. Error bars indicate the standard deviation over 25 repeats. Comparisons between `spyrmsd` and `obabel` (a) include input time, while comparisons between `networkx` versus `graph-tool` (b) do not include input time. `spyrmsd` is comparable or an order of magnitude slower than `obrms`. `networkx` shows a large variability between systems, while `graph-tool` is more consistent.

We believe that the simple API will favour the integration of `spyrmsd` in many existing libraries, bringing symmetry-corrected RMSD calculations to widely used packages.

B.4.3 Software Engineering Best Practices

The development of `spyrmsd` is based on modern software engineering best practices. The code is version-controlled using `git` (Chacon and Straub 2014), and it is freely available on GitHub (github.com/RMeli/spyrmsd, last accessed October 1, 2022), released under the open-source and permissive MIT licence.

The code is extensively tested using `pytest` (Krekel et al. 2014). Tests are run automatically every time a new version of the code is pushed to GitHub thanks to Travis-CI bindings for continuous integration. The code coverage of the test suite is reported on Codecov, which provides easy-to-read reports. A code coverage of 100% is targeted, so that all lines of code are executed at least once during tests.

The code is compatible with Python 3.6 or above. Static analysis tools are constantly applied to the code to catch errors that would be otherwise missed or discovered only during execution. We use `mypy` to perform static checks and `flake8` to detect style and formatting issues. Such tools help maintain correctness and stability for future developments as well as a clean codebase.

Finally, the code is documented using Python docstrings and the documentation is built automatically using `sphinx`. This will likely make it easier to fully understand the codebase thus facilitating the adoption of `spyrmsd` by other libraries.

B.5 Conclusions

`spyrmsd` provides robust symmetry-corrected RMSD calculations with a clean and simple API that is easy to integrate in existing Python libraries and pipelines. We believe that such a tool could be useful to the wider community of molecular modellers and cheminformaticians.

Future development of the software will focus on improved automatic bond perception—to automatically build molecular adjacency matrices—, and speed.

This page intentionally contains only this sentence.

C

Supplementary Information for Chapter 2

Contents

C.1 AutoDock Vina Scoring Function

AutoDock Vina is a FOSS program for molecular docking (Trott and Olson 2009).

All interaction functions $f_{t_i t_j}(r_{ij})$ are cut off at $r_{ij} = 8\,\text{Å}$. The SF h_{t_i,t_j} is a weighted sum of steric interactions, hydrophobic interactions between hydrophobic atoms, and hydrogen bonding.

C.1.1 Steric Interactions

There are three terms for steric interactions:

$$h_{\text{gauss},1}(d_{ij}) = e^{-\left(\frac{d_{ij}}{0.5\,\text{Å}}\right)^2}, \tag{C.1}$$

$$h_{\text{gauss},2}(d_{ij}) = e^{-\left(\frac{d_{ij}-3\,\text{Å}}{2\,\text{Å}}\right)^2}, \tag{C.2}$$

$$h_{\text{repulsion}}(d_{ij}) = \begin{cases} d_{ij}^2, & d_{ij} < 0\,\text{Å} \\ 0, & d_{ij} \geqslant 0\,\text{Å} \end{cases} \tag{C.3}$$

C.1.2 Hydrophobic Interactions

Hydrophobic interactions are represented by the following term:

$$h_{\text{hydrophobic}}(d_{ij}) = \begin{cases} 1, & d_{ij} < 0.5\,\text{Å} \\ -\frac{d_{ij} - 1.5\,\text{Å}}{1.0\,\text{Å}}, & 0.5\,\text{Å} < d_{ij} \leqslant 1.5\,\text{Å} \\ 0, & d_{ij} > 1.5\,\text{Å} \end{cases} \tag{C.4}$$

C.1.3 Hydrogen Bonding

Hydrogen bond interactions are represented by the following term:

$$h_{\text{H-bond}}(d_{ij}) = \begin{cases} 1, & d_{ij} < -0.7\,\text{Å} \\ -\frac{d_{ij}}{0.7\,\text{Å}}, & 0.5\,\text{Å} < d_{ij} \geqslant 1.5\,\text{Å} \\ 0, & d_{ij} > 0\,\text{Å} \end{cases} \tag{C.5}$$

Metal atoms are considered as hydrogen bond donors.

C.1.4 Free Energy Estimation

The conformation-independent function used to estimate the free energy for the optimal binding mode and the score for other binding modes is

$$g(c_{\text{inter}}) = \frac{c_{\text{inter}}}{1 + wN_{\text{rot}}} \tag{C.6}$$

where N_{rot} is the number of the rotatable bonds for the ligand and w is an empirical weight.

D

Supplementary Information for Chapter 3

Contents

D.1 Multivariate Gaussian Distribution

The multivariate Gaussian distribution of mean μ and covariance Σ is given by

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k \det \Sigma}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{D.1}$$

where $\det \Sigma$ is the *generalised variance*.

D.2 Kullback-Leibler Divergence

The KL divergence—also known as relative entropy—is a measure of how a probability distribution $q(\mathbf{x})$ differs from a probability distribution $p(\mathbf{x})$—over

the same random variable $\mathbf{x}$—, and it is defined as

$$\mathcal{D}_{\mathrm{KL}}(p\|q) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})}\right]. \tag{D.2}$$

For a continuous random variable, the KL divergence is therefore

$$\mathcal{D}_{\mathrm{KL}}(p\|q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}\, d\mathbf{x}. \tag{D.3}$$

The KL divergence is non-negative, and it is 0 if and only if $q(\mathbf{x})$ and $p(\mathbf{x})$ are the same distribution. However, it does not represent a distance between the two distributions since it is not symmetric:

$$\mathcal{D}_{\mathrm{KL}}(p\|q) \neq \mathcal{D}_{\mathrm{KL}}(q\|p). \tag{D.4}$$

E

Supplementary Information for Chapter 4

Ligand channels	Receptor channels
AliphaticCarbonHydrophobe	AliphaticCarbonHydrophobe
AliphaticCarbonNonHydrophobe	AliphaticCarbonNonHydrophobe
AromaticCarbonHydrophobe	AromaticCarbonHydrophobe
AromaticCarbonNonHydrophobe	AromaticCarbonNonHydrophobe
Bromine Iodine	Bromine Iodine Chlorine Fluorine
Chlorine	—
Fluorine	—
Nitrogen NitrogenA	Nitrogen NitrogenA
NitrogenD NitrogenDA	NitrogenD NitrogenDA
Oxygen OxygenA	Oxygen OxygenA
OxygenDA OxygenD	OxygenDA OxygenD
Sulfur SulfurAcceptor	Sulfur SulfurAcceptor
Phosphorus	Phosphorus
—	Calcium
—	Zinc
GenericMetal B Mn Mg Zn Ca Fe	GenericMetal B Mn Mg Fe

Table E.1: Ligand and receptor channels for GNINA. "A" indicates acceptor types, "D" indicates donor types, and "DA" indicates donor/acceptor types. There are 14 ligand channels and 14 receptor channels, for a total of 28 channels.

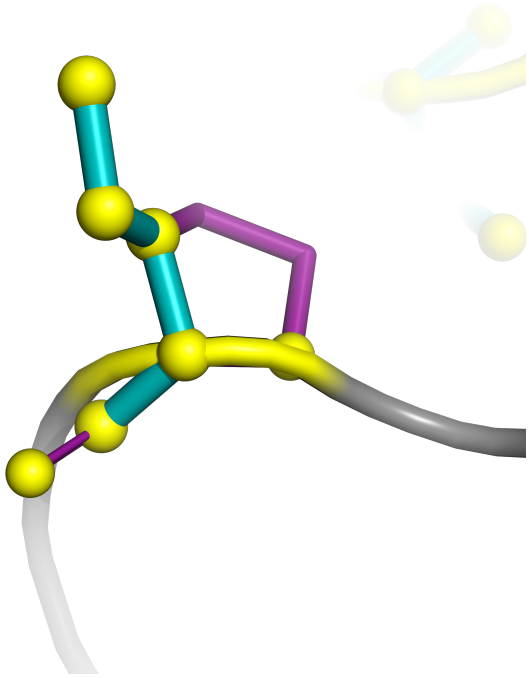


Figure E.1: Bug with proline residue treated as flexible. The proline residue ring is broken during the Monte Carlo search of optimal docking poses. Teal and yellow represent the docked pose (with broken ring) while violet represents the original crystal structure.

Pocket	Receptor	Ligand	Reason
ACES	1JJB	1ACJ	No flexible residues
ACES	1JJB	1ZGB	No flexible residues
ACES	1JJB	2CMF	No flexible residues
CDK2	3QQJ	3IG7	No flexible residues
IGF1R	5FXR	1JQH	No flexible residues
IGF1R	5FXR	2OJ9	No flexible residues
IGF1R	5FXR	2ZM3	No flexible residues
IGF1R	5FXR	3LVP	No flexible residues
IGF1R	5FXR	3NW6	No flexible residues
IGF1R	5FXR	3NW7	No flexible residues
JAK2	4F08	4D0W	No flexible residues
JAK2	4F08	4E4M	No flexible residues
JAK2	4F08	5CF6	No flexible residues
MK01	4GSB	4FV2	No flexible residues
MK01	4GSB	4ZZM	No flexible residues
MK01	4GSB	5LCJ	No flexible residues
MK01	4GSB	5NHV	No flexible residues
MK10	1UKI	2G01	No flexible residues
MK10	1UKI	3ELJ	No flexible residues
MK10	1UKI	3RTP	No flexible residues
MK10	4HYS	4L7F	No flexible residues
SRC	3UQG	3DQX	No flexible residues
SRC	3UQG	5D10	No flexible residues
SRC	3UQG	5J5S	No flexible residues
CP2C9	1R9O	5W0C	Added spurious bond
FA10	2XBV	2FZZ	Broken disulfide bond
FA10	1IQE	2RA0	Broken disulfide bond
FA10	2XBV	2Y82	Broken disulfide bond
FA10	1IQE	3KQB	Broken disulfide bond
KIF11	4BXN	1X88	Broken spurious bond
KIF11	4BXN	2IEH	Broken spurious bond
KIF11	4BXN	2X7D	Broken spurious bond
KIF11	4BXN	3K3B	Broken spurious bond

Table E.2: Pocket, receptor and ligand identifiers for the systems of the cross-docking data set excluded from the analysis of flexible docking, together with the reason for exclusion. All systems where docked with GNINA producing ten poses. 24 systems are discarded since no flexible residues were identified. Nine systems are discarded because bonding information is different between input and output files, eight of which because of broken disulfide bonds.

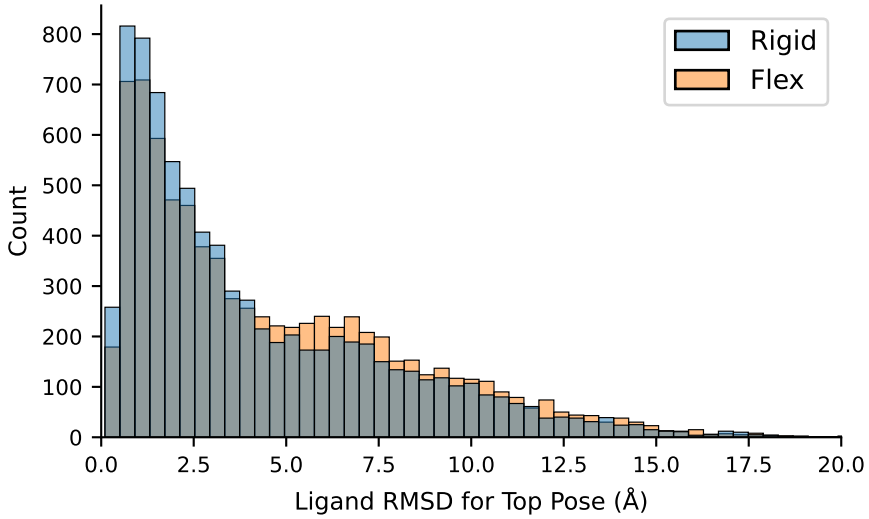


Figure E.2: Ligand pose RMSD distributions for the top pose obtained with standard (rigid) or flexible (flex) docking on the cross-docking data set using Gnina's default model.

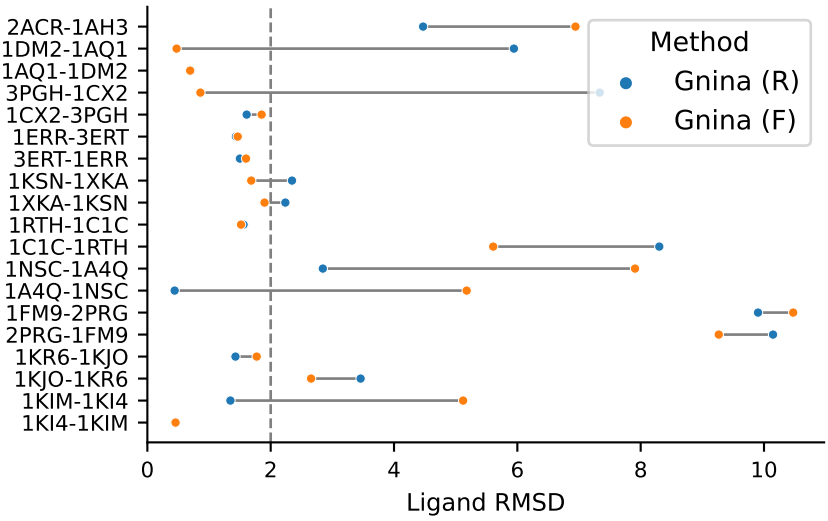


Figure E.3: Comparison between rigid docking (R) and docking with flexible residues (F) with Gnina on the data set used to evaluate the IFD protocol. Flexible residues are automatically selected based on the cognate ligand.

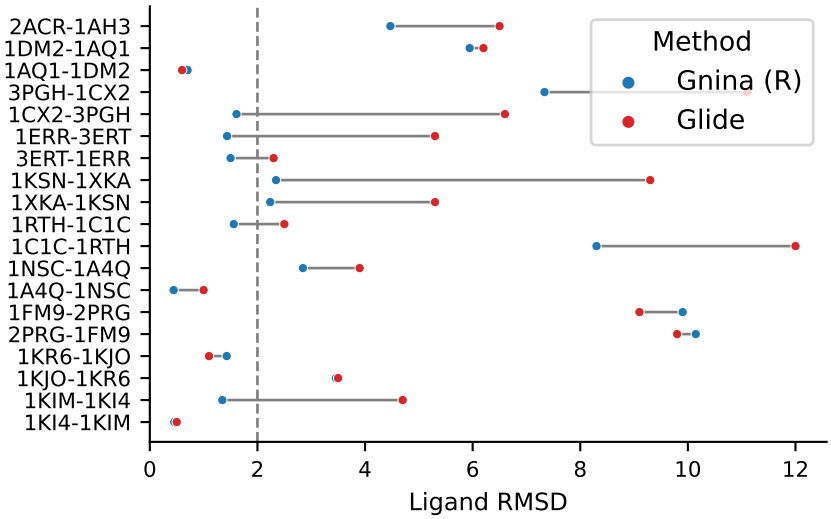


Figure E.4: Comparison between Gnina and Glide on the data set used to evaluate the IFD protocol. Gnina is able to provide low-RMSD poses for 9 systems, compared to only 4 for Glide. Therefore, Gnina provides a much stronger baseline than Glide for this data set.

Receptor	Ligand	Residues
2ACR	1AH3	A:111,A:122,A:300
1DM2	1AQ1	A:10,A:13,A:33,A:83,A:84
1AQ1	1DM2	—
3PGH	1CX2	A:355,A:523
1CX2	3PGH	A:120
1ERR	3ERT	A:404,A:421,A:524,A:525
3ERT	1ERR	A:346,A:420,A:421,A:424
1KSN	1XKA	A:99
1XKA	1KSN	C:174,C:192
1RTH	1C1C	A:229,A:236
1C1C	1RTH	A:95,A:100,A:101,A:227,B:138
1NSC	1A4Q	A:220,A:244,A:274
1A4Q	1NSC	—
1FM9	2PRG	D:284,D:289
2PRG	1FM9	A:282,A:286,A:360,A:363,A:473
1KR6	1KJO	—
1KJO	1KR6	A:112
1KIM	1KI4	A:132,A:168
1KI4	1KIM	—

Table E.3: PDB IDs for the systems—receptor and non-cognate ligand—used to test the IFD protocol, together with the residues clashing with the non-cognate ligand.

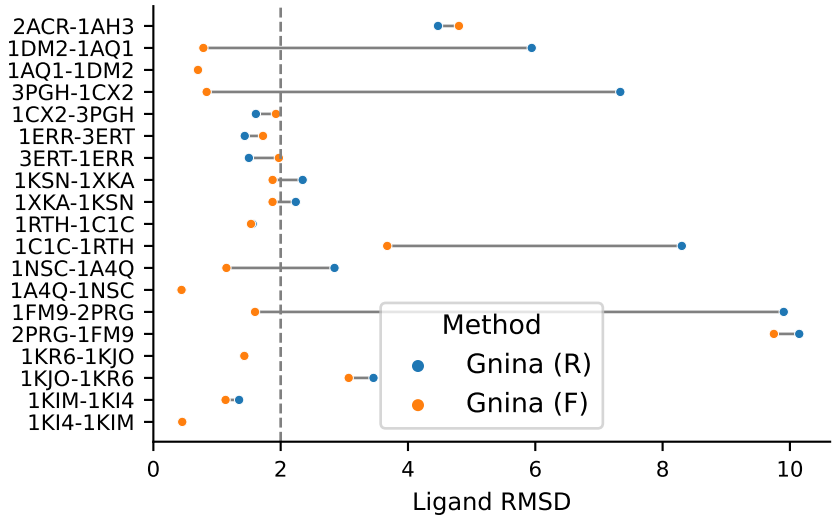


Figure E.5: Comparison between rigid docking (R) and docking with flexible residues (F) with Gnina on the data set used to evaluate the IFD protocol. Flexible residues are manually selected.

F

Supplementary Information for Chapter 5

Contents

F.1 ProBiS Algorithm

ProBiS is an algorithm for detection of structurally similar binding sites by local structural alignment. Structurally similar binding sites are defined as the collection of functional groups sharing similar geometrical conformations in the binding site (Konc and Janežič 2010). Since minimising the RMSD between residues in a binding site does not ensure that the protein backbones are aligned as well, a different algorithm to determine the degree of similarity between binding sites is required. Here we give a brief summary of the algorithm introduces by Konc and Janežič (2010).

In the ProBiS algorithm, surface residues are initially identified by computing the solvent accessible surface area (SASA), and such residues are then encoded as a graph, where the vertices represent different functional groups and edges

are added between atoms closer than 15 Å. Once the full protein graph with n vertices is obtained, n overlapping sub-graphs are constructed—one for each vertex—by retaining all vertices within 15 Å from the central vertex. This representation allows a fast pre-screening of similar sub-graphs between different proteins: the difference between the distance matrices of the sub-graphs of different proteins is used to compute a similarity score between the sub-graphs (Konc and Janezic 2007). Only the sub-graph pairs—one for each protein—with high similarity score are retained and further processed.

From the pair of sub-graphs considered similar, a product graph—a graph with vertex features combined, and edges if the original vertices where connected in both graphs—is constructed. The algorithm then proceeds to find the maximum clique—the subset of vertices of the original graph so that every pair of vertices are adjacent—of the product graph. Every maximum clique for the different product graphs corresponding to different sub-graph pairing corresponds to a (local) structural alignment and superposition of the two proteins being compared. The two proteins are superimposed with all the possible local structural alignment and a set of scores—surface vector angles, RMSD, . . . —are computed to determine the best alignments. The high-scoring residues are considered part of structurally similar binding sites.

In this work we used ProBiS to superimpose a pair of binding sites, as described in Francoeur et al. (2020). The binding site is automatically defined by the residues within 7.0 Å from the cognate ligand.

Pocket	Receptor	Ligand	Reason
ACES	1JJB	1ACJ	No flexible residues
ACES	1JJB	1ZGB	No flexible residues
ACES	1JJB	2CMF	No flexible residues
CDK2	3QQJ	3IG7	No flexible residues
IGF1R	5FXR	1JQH	No flexible residues
IGF1R	5FXR	2OJ9	No flexible residues
IGF1R	5FXR	2ZM3	No flexible residues
IGF1R	5FXR	3LVP	No flexible residues
IGF1R	5FXR	3NW6	No flexible residues
IGF1R	5FXR	3NW7	No flexible residues
JAK2	4F08	4D0W	No flexible residues
JAK2	4F08	4E4M	No flexible residues
JAK2	4F08	5CF6	No flexible residues
MK01	4GSB	4FV2	No flexible residues
MK01	4GSB	4ZZM	No flexible residues
MK01	4GSB	5LCJ	No flexible residues
MK01	4GSB	5NHV	No flexible residues
MK10	1UKI	2G01	No flexible residues
MK10	1UKI	3ELJ	No flexible residues
MK10	1UKI	3RTP	No flexible residues
MK10	4HYS	4L7F	No flexible residues
SRC	3UQG	3DQX	No flexible residues
SRC	3UQG	5D10	No flexible residues
SRC	3UQG	5J5S	No flexible residues
HXK4	4DCH	3S41	No matching flexible residues
HXK4	4DCH	4NO7	No matching flexible residues
MET	3CCN	5HOA	No matching flexible residues
FA10	2XBV	2FZZ	Broken disulfide bond
FA10	1IQE	2RA0	Broken disulfide bond
FA10	2XBV	2Y82	Broken disulfide bond
FA10	1IQE	3KQB	Broken disulfide bond
GCR	3MNP	3CLD	Added spurious bond
KIF11	4BXN	1X88	Broken spurious bond
KIF11	4BXN	2IEH	Broken spurious bond
KIF11	4BXN	2X7D	Broken spurious bond
KIF11	4BXN	3K3B	Broken spurious bond
LKHA4	4RSY	3FTZ	Added spurious bond
LKHA4	3FHE	3FUJ	Added spurious bond

Table F.1: Pocket, receptor and ligand identifiers for the systems of the cross-docking data set excluded from the analysis of flexible docking, together with the reason for exclusion. All systems where docked with Gnina using the AutoDock Vina scoring function, and producing 20 poses. 24 systems are discarded since no flexible residues were identified. Three systems are discarded because there are no matching residues within the ones treated as flexible. Eleven systems were discarded because bonding information was different between input and output files, eight of which because of broken disulfide bonds.

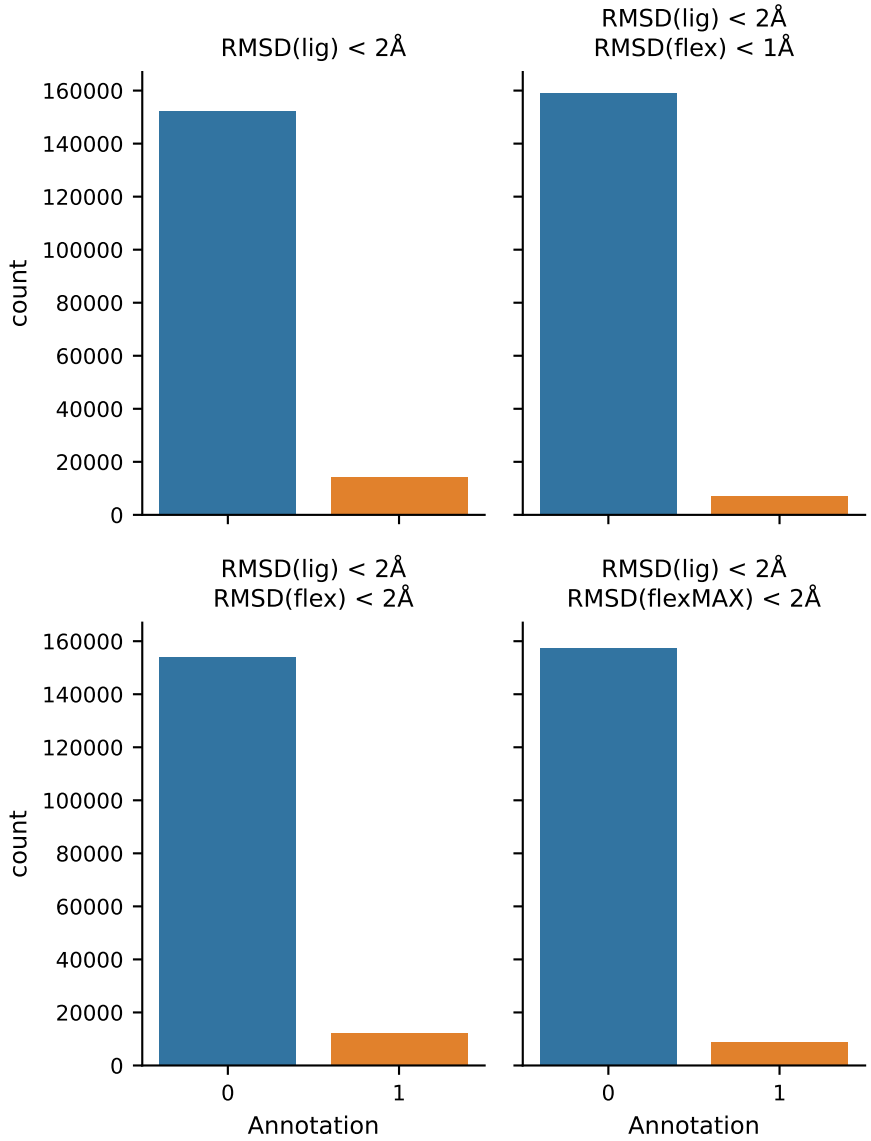


Figure F.1: Data imbalance for the cross-docking data set with flexible side chains, for different ligand- and receptor-based annotations into low- and high-RMSD classes. The different annotations are reported at the top of each graph, where RMSD(lig) denotes the RMSD for the ligand, RMSD(flex) the RMSD of flexible residues, and RMSD(flexMAX) the maximum per-residue RMSD.

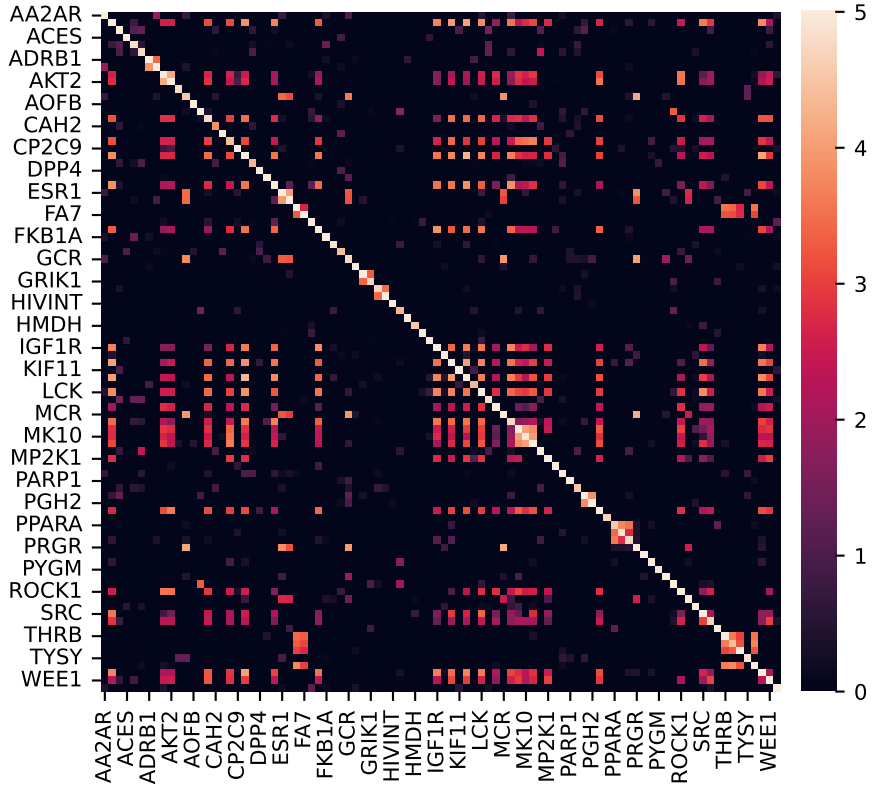


Figure F.2: ProBiS z-scores for pairwise alignments between pockets of the cross-docking benchmark data set, clipped on the interval $[0, 5]$. A z-score of 0 is arbitrarily assigned to all pairs for which no alignment could be found.

Pocket A	Pocket B	z-score (A $\mapsto$ B)	z-score (B $\mapsto$ A)
ABL1	MET	3.53	3.44
ADRB2	ADRB1	3.92	3.42
EGFR	IGF1R	3.53	3.37
FA10	THRB	3.52	3.42
JAK2	IGF1R	3.53	3.39
JAK2	MET	3.53	3.43
JAK2	VGFR2	3.56	3.47
LCK	SRC	3.80	3.37
PPARA	PPARD	3.74	3.40
VGFR2	FAK1	3.50	3.06

Table F.2: Pocket pairwise alignments for which the z-score of forward A $\mapsto$ B and backward B $\mapsto$ A alignments fall on either sides of the threshold of 3.5.

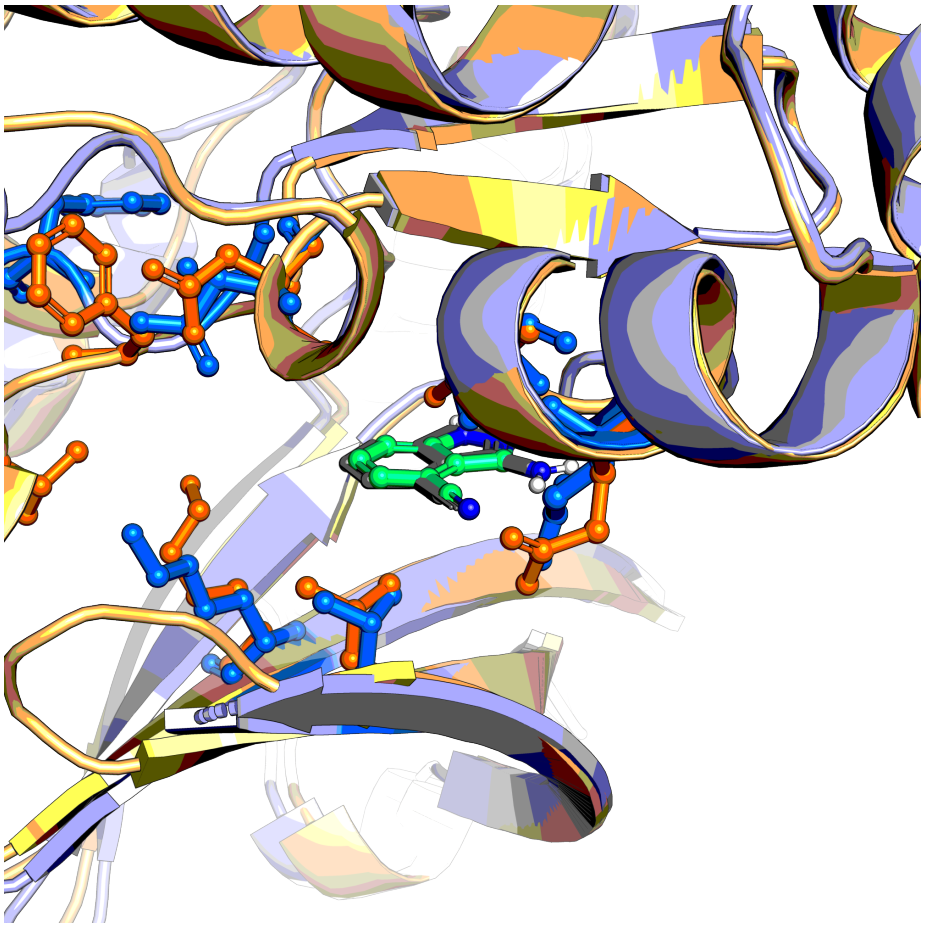
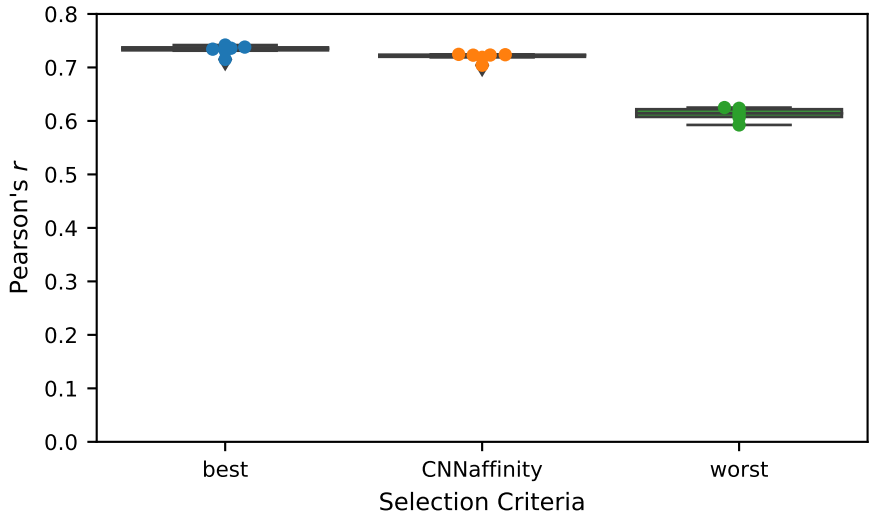
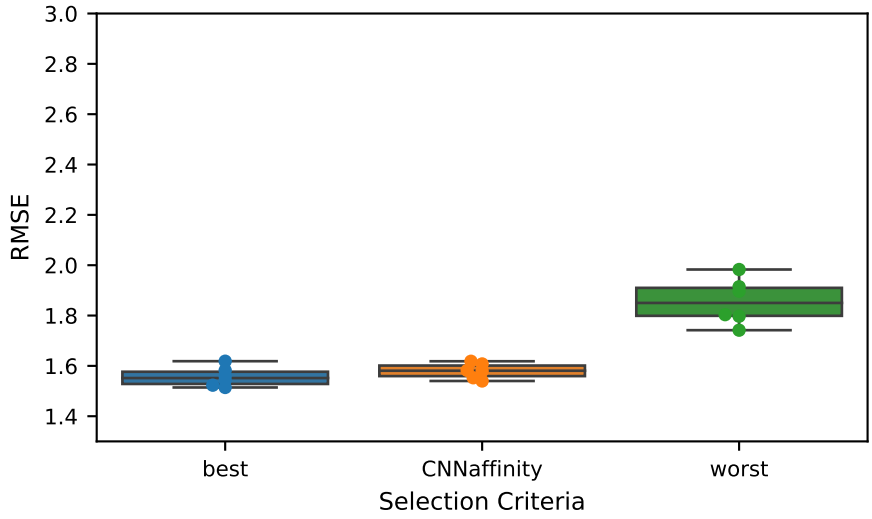


Figure F.3: Cross-docked pose (rank 11) for ligand 3ZLY docked into target 3W8Q for the MP2K1 pocket of the cross-docking data set.

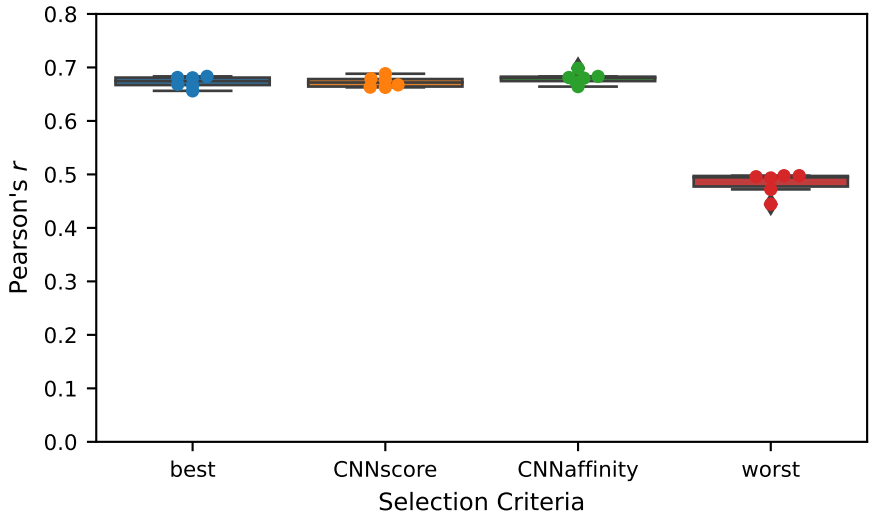


(a)

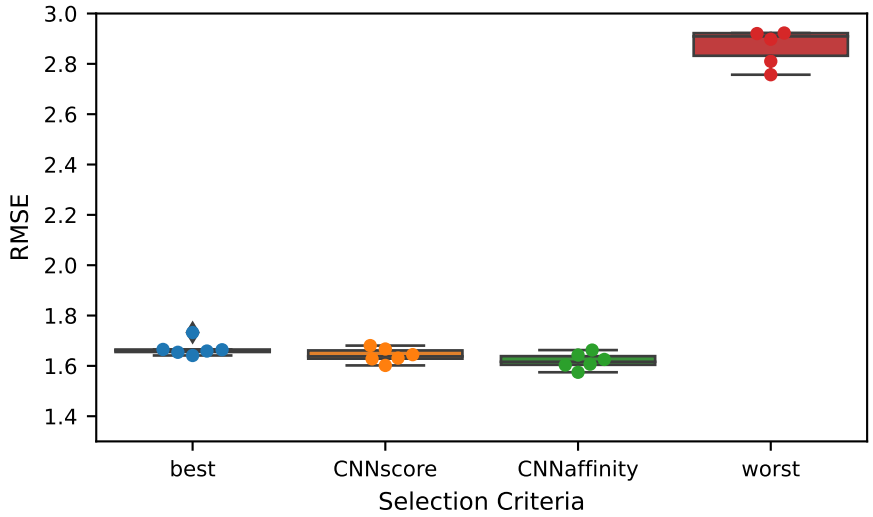


(b)

Figure F.4: Pearson's correlation coefficient and RMSE for binding affinity prediction on CASF-2016 obtained with `gnina-torch` while training on crystal poses.



(a)



(b)

Figure F.5: Pearson's correlation coefficient and RMSE for binding affinity prediction on CASF-2016 obtained with `gnina-torch` while training on docked poses.

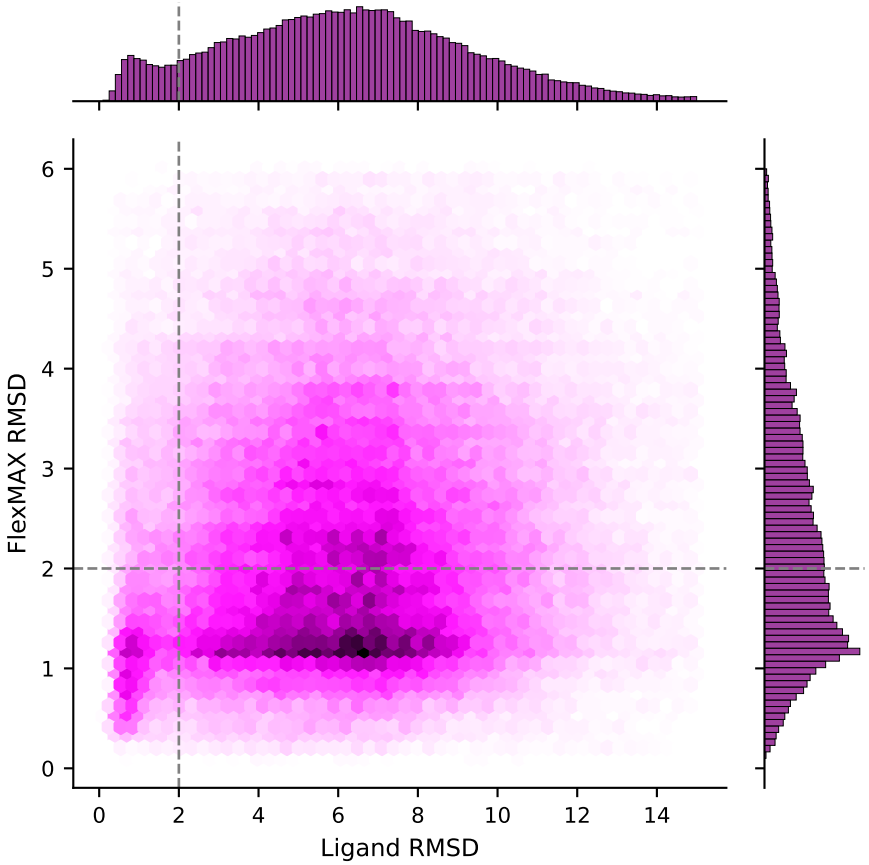


Figure F.6: RMSD distribution for the ligand poses versus (pose-cognate) maximum pre-residue RMSD distribution of the flexible side chains. Dashed lines represent the boundaries for the different annotation.

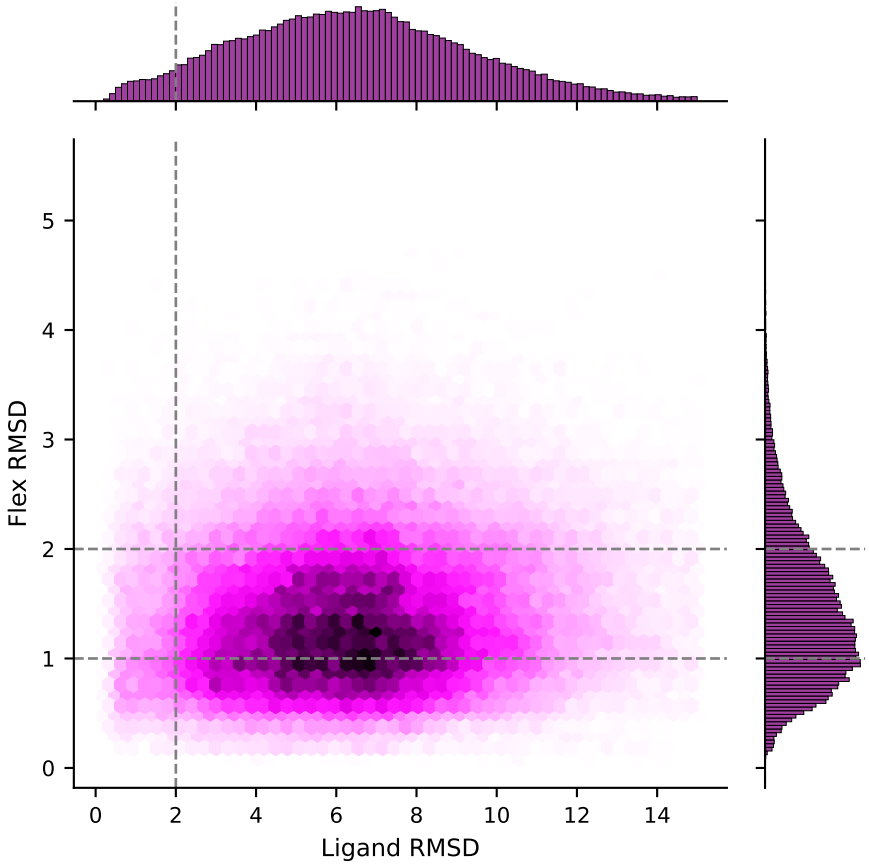


Figure F.7: RMSD distribution for the ligand poses versus (pose-cognate) RMSD distribution of the flexible side chains. Dashed lines represent the boundaries for the different annotation. Minimised crystal poses have been removed.

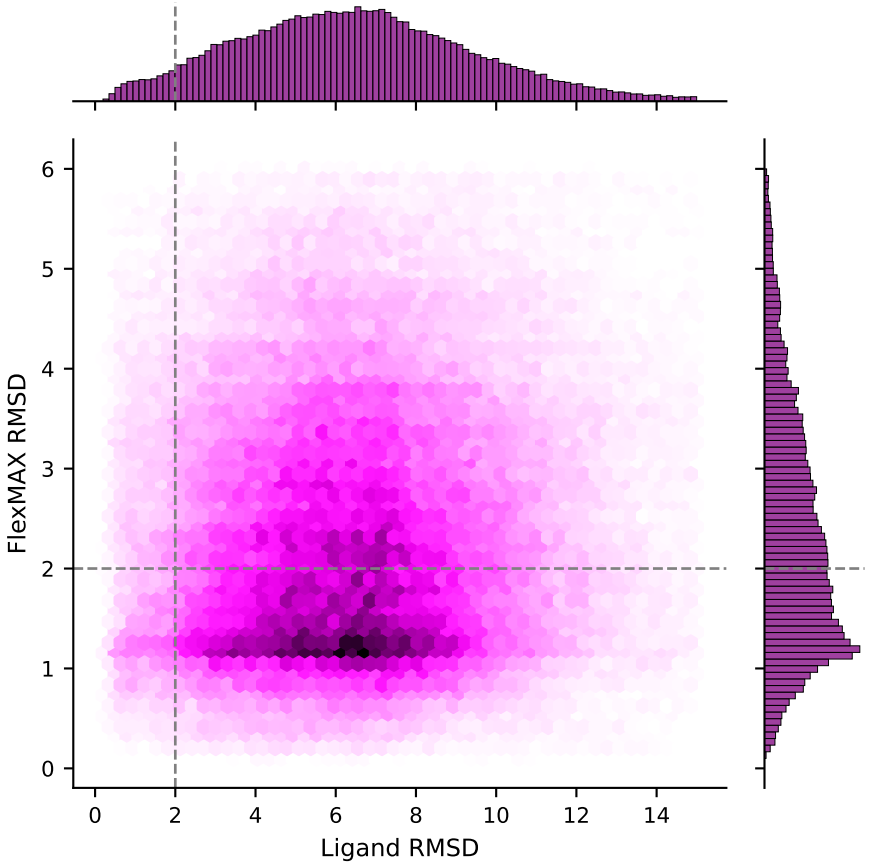


Figure F.8: RMSD distribution for the ligand poses versus (pose-cognate) maximum pre-residue RMSD distribution of the flexible side chains. Dashed lines represent the boundaries for the different annotation. Minimised crystal poses have been removed.

This page intentionally contains only this sentence.

G

Supplementary Information for Chapter 6

Contents

G.1 Examples of AEV Computation

G.1.1 Water Molecule

As a simple example of how ACSFs are used to construct AEVs, let us consider a simple water molecule with the fictitious coordinates of Tab. G.1. If this is the only system we want to describe, we have only two elements ($N_e = 2$) and we need to compute three AEVs, one for each atom.

	index	x	y	z
H	1	1	0	0
H	2	0	1	0
O	3	0	0	0

Table G.1: Fictitious coordinates for a water molecule.

Radial symmetry functions are parametrised by η_R and R_s. For simplicity, we only consider here $\eta_R = 1$ and $R_s = \{0, 1\}$ (two radial functions per pair). Angular symmetry functions are parametrised by η_A, R_s, θ_s and ζ. For simplicity, we only consider here $\eta_A = 1$, $R_s = 0$, $\theta_s = 0$ and $\zeta = 1$ (one angular function per triplet).

The AEV for the oxygen atom (for $n_e = 2$) has the following form:

$$\begin{aligned}&\mathbf{G}_3^O = \\ &[G^R_{3;H,R_s=0}, G^R_{3;H,R_s=1}, G^R_{3,O,R_s=0}, G^R_{3,O,R_s=1}; G^A_{3;H,H}, G^A_{3;H,O}, G^A_{3;O,O}].\end{aligned} \tag{G.1}$$

Since there are no other oxygen atoms in the system, we have $G^R_{3,O,R_s=0} = 0$, $G^A_{3,H,O} = 0$, and $G^A_{3,O,O} = 0$ since such ACSF depend on one or two neighbouring oxygen atoms within the cutoff distance R_c (which we consider here large enough to include all atoms of the system). The AEV for the oxygen atom therefore reduces to

$$\mathbf{G}_3^O = [G^R_{3;H,R_s=0}, G^R_{3;H,R_s=1}, 0, 0; G^A_{3;H,H}, 0, 0]. \tag{G.2}$$

Explicitly, the non-zero ACSF composing the AEV for the oxygen atom are

$$G^R_{3;H,R_s=0} = \sum_{\substack{j\neq 3\\ j\in H}} e^{-R^2_{3,j}} f_c(R_{3,j}) = e^{-R^2_{3,1}} f_c(R_{3,1}) + e^{-R^2_{3,2}} f_c(R_{3,2}), \tag{G.3}$$

$$G^R_{3;H,R_s=1} = \sum_{\substack{j\neq 3\\ j\in H}} e^{-(R_{3,j}-1)^2} f_c(R_{3,j}) \tag{G.4}$$

$$= e^{-(R_{3,1}-1)^2} f_c(R_{3,1}) + e^{-(R_{3,2}-1)^2} f_c(R_{3,2}), \tag{G.5}$$

$$G^A_{3;H,H} = \sum_{\substack{j,k\neq 3\\ j\in H, k\in H}} [1+\cos(\theta_{3,j,k})] e^{-\left(\frac{R_{3,j}+R_{3,k}}{2}\right)^2} f_c(R_{3,j}) f_c(R_{3,k}) \tag{G.6}$$

$$= [1+\cos(\theta_{3,1,2})] e^{-\left(\frac{R_{3,1}+R_{3,2}}{2}\right)^2} f_c(R_{3,1}) f_c(R_{3,2}). \tag{G.7}$$

If we consider R_c to be large enough so that $f_c(R) \approx 1$, and we use the geometry defined in Tab. G.1, we can perform an explicit calculation for the particular configuration considered here (where $R_{3,1} = R_{3,2} = 1$ and $\theta_{3,1,2} = \pi/2$):

$$\mathbf{G}_3^O = [2e^{-1}, 2, 0, 0; e^{-1}, 0, 0] \tag{G.8}$$

The AEV for oxygen encodes its atomic environment, and it is rotationally and translationally invariant by construction.

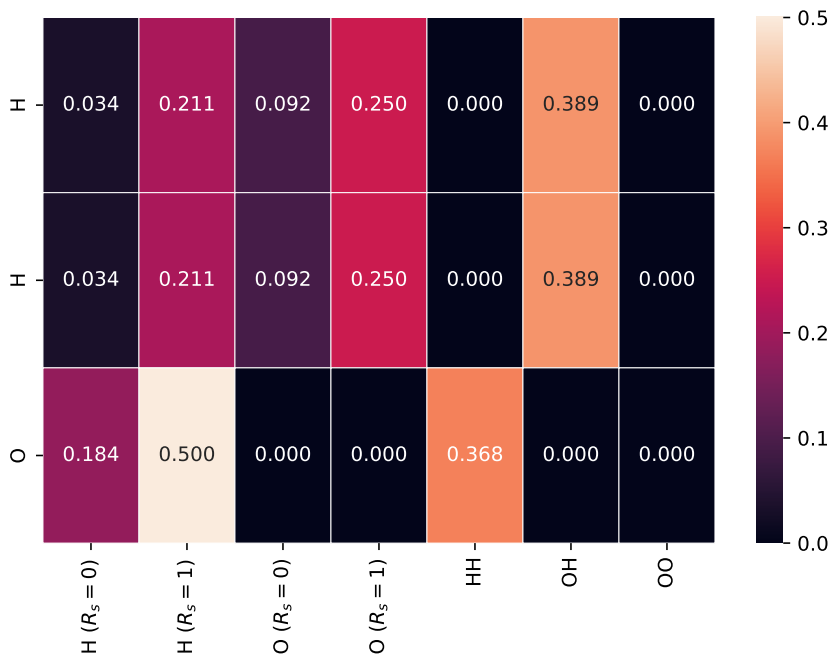


Figure G.1: AEVs for the atoms of the water molecule defined in Tab G.1. The two hydrogen atoms have the same AEVs because of symmetry.

The same procedure can be repeated for every atom of the system, so that all atoms are described by their own AEV. We can therefore describe the whole system with a matrix of AEVs of dimension $N_{atoms} \times N_{AEVs}$, where N_{AEVs}—the size of an AEV—depends on the number of elements n_e as well as the number of different values for the parameters $R_s, \eta_R, \ldots$.

Fig. G.1 shows the AEVs for the atoms of the water of Tab. G.1, computed with TorchANI (Gao, Ramezanghorbani, et al. 2020). Discrepancies with the analytical calculation above come from the fact that the radial part is multiplied by a factor of $1/4$ in the TorchANI implementation (see TorchANI code, last accessed October 1, 2022).

G.1.2 Ammonia

Let us consider another simple example: ammonia. Again, we only have two elements ($n_e = 2$) and we need to compute four AEVs, one for each atom. If we consider the same parametrisation for radial and angular symmetry functions

described above, we have the following AEV for nitrogen:

$$\mathbf{G}^{N} = [G^{R}_{N;H,R_s=0}, G^{R}_{N;H,R_s=1}, G^{R}_{N;N,R_s=0}, G^{R}_{N;N,R_s=1}; G^{A}_{N;H,H}, G^{A}_{N;H,N}, G^{A}_{N;N,N}]. \tag{G.9}$$

Since there are no other nitrogen atoms in the system, the AEV for the only nitrogen atom reduces to

$$\mathbf{G}^{N} = [G^{R}_{N;H,R_s=0}, G^{R}_{N;H,R_s=1}, 0, 0; G^{A}_{N;H,H}, 0, 0]. \tag{G.10}$$

Explicitly, denoting d_{NH} the nitrogen-hydrogen distance, we have

$$G^{R}_{N;H,R_s=0} = \sum_{\substack{j\neq N \\ j\in H}} e^{-d^2_{NH}} f_c(d_{NH}) = 3e^{-d^2_{NH}} f_c(d_{NH}), \tag{G.11}$$

$$G^{R}_{N;H,R_s=1} = \sum_{\substack{j\neq N \\ j\in H}} e^{-(d_{NH}-1)^2} f_c(d_{NH}) = 3e^{-(d_{NH}-1)^2} f_c(d_{NH}), \tag{G.12}$$

$$\begin{aligned} G^{A}_{N;H,H} &= \sum_{\substack{j,k\neq N \\ j\in H, k\in H}} [1+\cos(\theta_{N;HH})] e^{-\left(\frac{d_{NH}+d_{NH}}{2}\right)^2} f_c(d_{NH}) f_c(d_{NH}) \\ &= 3[1+\cos(\theta_{N;HH})] e^{-(d_{NH})^2} f_c^2(d_{NH}). \end{aligned} \tag{G.13}$$

Using $d_{NH} = 1$ and $\theta_{N;HH} = 109.5$ we have the following AEV for nitrogen

$$\mathbf{G}^{N} = \left[3e^{-1}, 3, 0, 0; 3\left(1+\cos(109.5)\right)e^{-1}, 0, 0\right]. \tag{G.14}$$

Fig. G.2 shows the AEVs for ammonia computed with TorchANI (Gao, Ramezanghorbani, et al. 2020). Discrepancies with the analytical calculation above come from the fact that the radial part is multiplied by a factor of 1/4 in the TorchANI implementation (see TorchANI code, last accessed October 1, 2022) and that the angle between two vectors is computed as $\theta = \arccos(0.95 * c)$ where $c = \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{|\boldsymbol{v}_1||\boldsymbol{v}_2|}$ instead of $\theta = \arccos(c)$ (see TorchANI code, last accessed October 1, 2022).

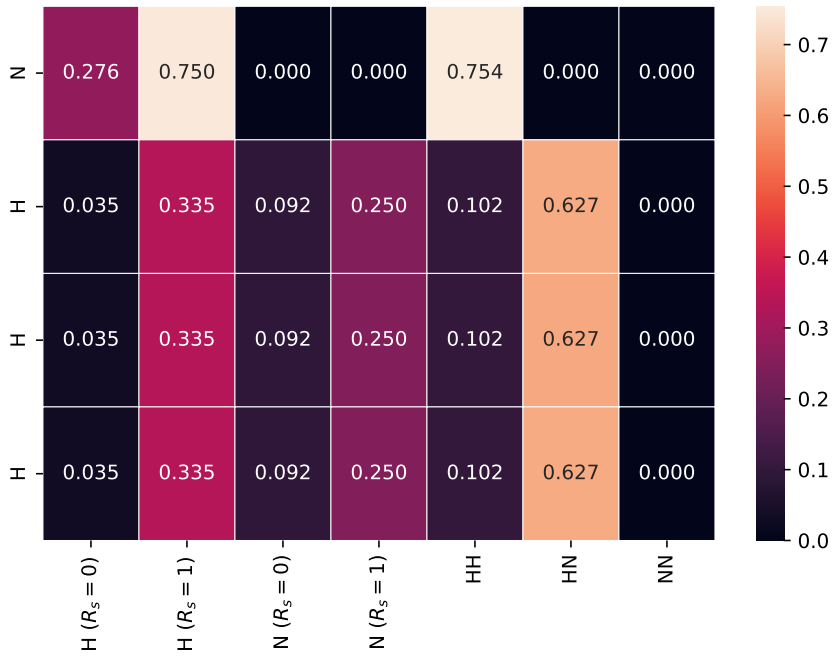


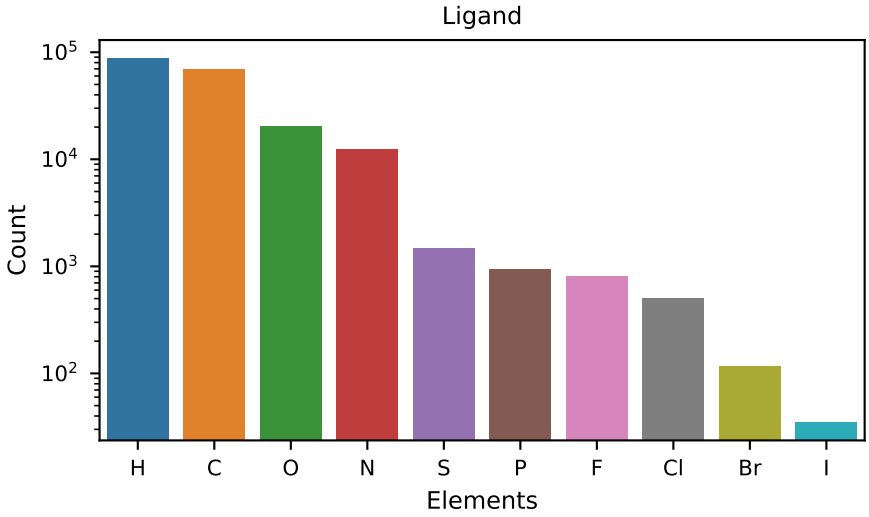
Figure G.2: AEVs for the atoms in the ammonia molecule with $d_{NH} = 1$ and $\theta_{N;HH} = 109.5$. The three hydrogen atoms have the same AEVs because of symmetry.

Distance (Å)	RMSE	Pearson's r	Time (s/epoch)
0.0	1.52	0.72	1.6
2.5	1.51	0.74	3.3
3.0	1.42	0.76	5.0
3.5	1.37	0.78	5.7
4.0	1.35	0.78	6.3

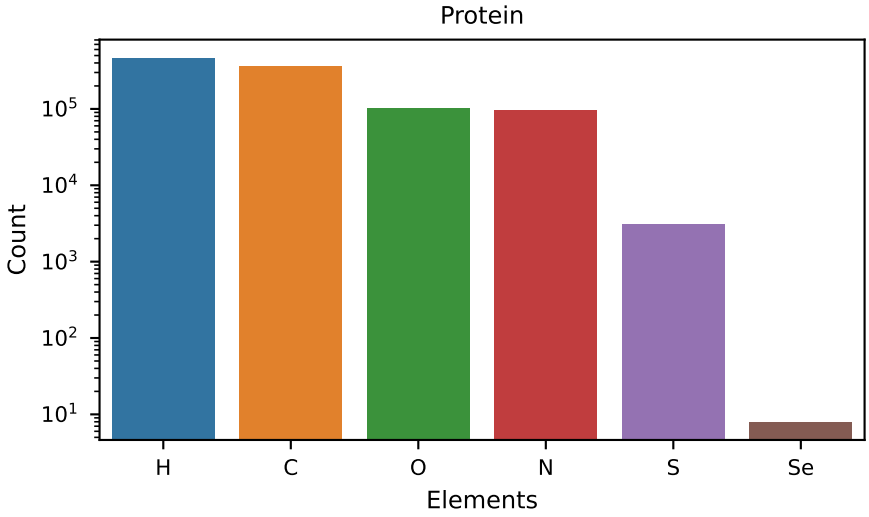
Table G.2: Model performance—with consensus scoring—on the validation set for different values of d, all else being fixed to optimal values (256-128-64-1 feed-forward atomic NNs, batch size of 64 and dropout probability of 25%). The approximate training time per epoch is also reported. Training is performed on an NVIDIA GeForce GTX 1080 Ti GPU.

Model	AEV	Test Set	RMSE	Pearson's r
AEScore	P + L	CASF-2013	<u>1.46</u>	<u>0.76</u>
AEScore	L	CASF-2013	1.65	0.70
AEScore (no H)	P + L	CASF-2013	1.48	0.75
AEScore (no H)	L	CASF-2013	1.69	0.67
Δ-AEScore	P + L	CASF-2013	1.53	0.74
Δ-AEScore	L	CASF-2013	1.65	0.68
Δ-AEScore (no H)	P + L	CASF-2013	1.52	0.74
Δ-AEScore (no H)	L	CASF-2013	1.61	0.70
Vina (optim)	—	CASF-2013	1.83	0.60
AEScore	P + L	CASF-2016	1.30	0.80
AEScore	L	CASF-2016	1.49	0.74
AEScore (no H)	P + L	CASF-2016	1.28	0.81
AEScore (no H)	L	CASF-2016	1.59	0.69
AEScore_{95} (no H)	P + L	CASF-2016	<u>1.22</u>	<u>0.83</u>
AEScore_{95} (no H)	L	CASF-2016	1.50	0.72
Δ-AEScore	P + L	CASF-2016	1.34	0.79
Δ-AEScore	L	CASF-2016	1.41	0.76
Δ-AEScore (no H)	P + L	CASF-2016	1.32	0.80
Δ-AEScore (no H)	L	CASF-2016	1.40	0.77
Vina (optim)	—	CASF-2016	1.75	0.60

Table G.3: Comparison between models incorporating atoms from the protein and the ligand (P + L, $d = 3.5$ Å) or atoms of the ligand only (L). The best performance for each test set is underlined. RMSE values are given in pK units.



(a) Ligand



(b) Protein

Figure G.3: Number of atoms—including protein atoms of residues within d = 3.5 Å from the ligand—for each element in the PDBbind 2016 refined set when only protein residues are considered. The following PDB IDs correspond to selenoproteins: 1UJ6, 1NU3, 3GPO, 2WQP, 3M89, 3HX3, 2QRY.

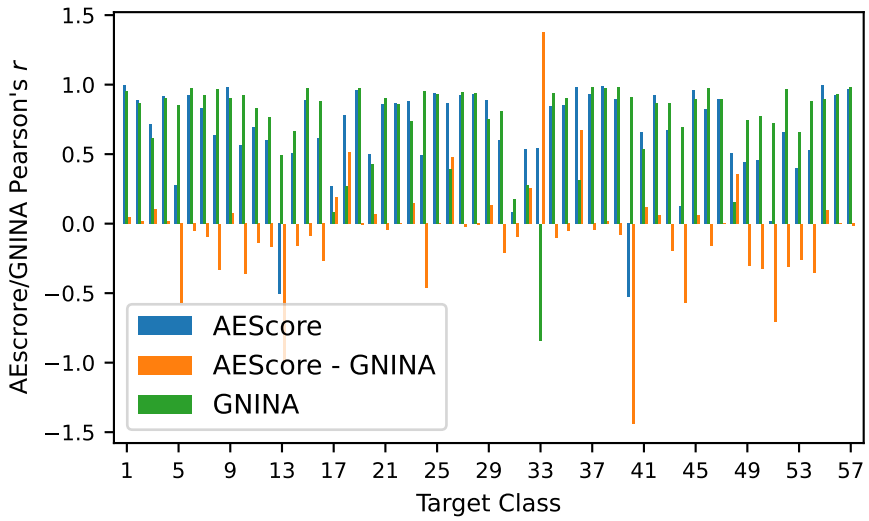


Figure G.4: Comparison of per-class Pearson's correlation coefficients between AEScore and Gnina. The difference in correlation coefficients between the two methods (AEScore − Gnina) is also reported.

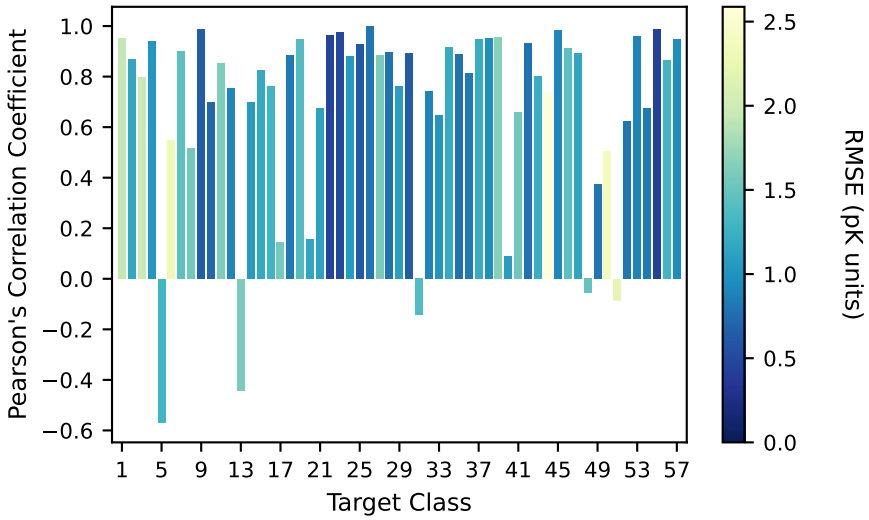


Figure G.5: Per-class Pearson's correlation coefficients colour-coded by RMSE in pK units, for the 57 classes of the CASF-2016 data set. The model is trained on systems without hydrogen atoms.

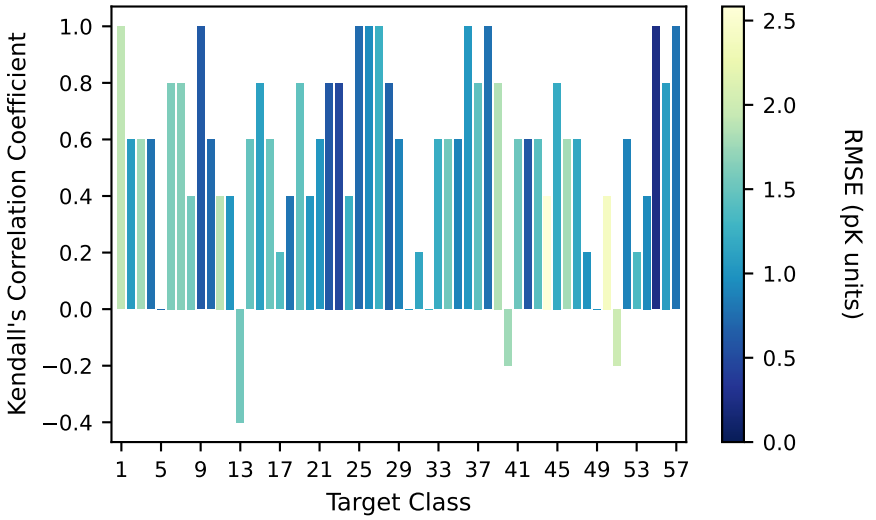


Figure G.6: Per-class Kendall's correlation coefficients colour-coded by RMSE in pK units, for the 57 classes of the CASF-2016 data set.

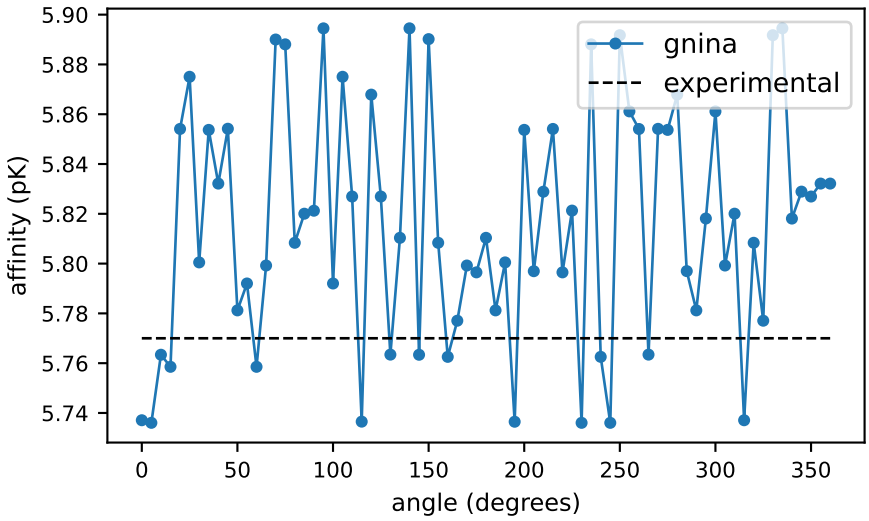


Figure G.7: Predicted binding affinity as a function of the angle of rotation for the CNN-based scoring function Gnina on the 1O5B complex—the one with smaller absolute prediction error—of the CASF 2016 data set. Predictions are obtained using the pre-trained `default2018` model. The complex is rotated around the protein centre of mass along the z axis.

Model	Type	Training	Test	RMSE	r
AEScore†	NN	R 2016	CASF-2013	1.46	0.76
AEScore†(no H)	NN	R 2016	CASF-2013	1.48	0.75
RosENet	CNN	R 2016	CASF-2013	1.43	0.80
AGL-Score	ML	R 2016	CASF-2013	1.46	0.79
1D2D CNN	CNN	R 2013	CASF-2013	1.47	0.78
Res4HTMD	CNN	R 2016	CASF-2013	1.48	0.77
NNScore 2.0	ML	G 2018	CASF-2013	—	0.75
RF Score	ML	G 2018	CASF-2013	—	0.75
Pafnucy	CNN	G 2016	CASF-2013	1.62	0.70
Vina (optim)	—	—	CASF-2013	1.82	0.61
AEScore†	NN	R 2016	CASF-2016	1.30	0.80
AEScore†(no H)	NN	R 2016	CASF-2016	1.28	0.81
$AEScore_{95}$†(no H)	NN	R 2016	CASF-2016	1.22	0.83
AGL-Score	ML	R 2016	CASF-2016	1.28	0.83
NNScore 2.0	NN	G 2018	CASF-2016	—	0.82
RF Score	ML	G 2016	CASF-2016	—	0.81
pair	NN	R 2018	CASF-2016	1.45	0.78
Vina (optim)	—	—	CASF-2016	1.75	0.59
AEScore†	NN	R 2016	C 2016	1.32	0.80
AEScore†(no H)	NN	R 2016	C 2016	1.32	0.81
$AEScore_{95}$†(no H)	NN	R 2016	C 2016	1.22	0.83
1D2D CNN	CNN	R 2016	C 2016	1.21	0.85
Res4HTMD	CNN	R 2016	C 2016	1.25	0.83
RosENet	CNN	R 2016	C 2016	1.24	0.82
KDeep	CNN	R 2016	C 2016	1.27	0.82
AutoDock & RF-Score	ML	R 2016	C 2016	1.36	0.82
DeepAtom	CNN	R 2016	C 2016	1.32	0.81
AK-score (ensemble)	CNN	R 2018	C 2016	—	0.81
gnina	CNN	G 2016	C 2016	1.37	0.80
RF Score	ML	R 2016	C 2016	1.39	0.80
Pafnucy	CNN	G 2016	C 2016	1.42	0.78
gnina	CNN	R 2016	C 2016	1.50	0.73
AK-score single	CNN	R 2018	C 2016	—	0.76

† This work.

Table G.4: Performance of different machine learning and deep learning models for affinity prediction on the CASF-2013 and CASF-2016 benchmarks (Afifi and Al-Sadek 2018; Ballester and Mitchell 2010b; Boyles et al. 2019; Cang et al. 2018; Durrant and McCammon 2011b; Francoeur et al. 2020; Hassan-Harrirou et al. 2020; Jiménez, Škalič, et al. 2018; Kwon et al. 2020; Li, Fu, et al. 2019; Nguyen and Wei 2019; Stepniewska-Dziubinska et al. 2018; Zhu et al. 2020). NN denotes feed-forward neural networks, CNN denotes convolutional neural networks, and ML denotes "classical" machine learning methods (random forests, gradient boosting trees, ...). "Refined" (R), "general" (G) and "core" (C) all refer to the PDBbind data set.

H

Supplementary Information for Chapter 7

H.1 Global and Coloured Point Cloud Registration

SENSAAS performs shape-based alignment of molecules using point cloud registration. In this section we briefly outline how the global and coloured registration of point clouds work. Point cloud registration is a common technique from computer vision, and the algorithms discussed below are readily accessible in the Open3D library (Zhou, Park, et al. 2018).

H.1.1 Global Point Cloud Registration

The global registration of two point clouds—which does not require any alignment for initialisation—is performed using the random sample consensus (RANSAC) algorithm (Fischler and Bolles 1981). The point clouds are pre-processed by downsampling the number of points in each point cloud and by computing fast point feature histograms (FPHFs), a set of pose-invariant 33-dimensional features describing the local geometry around a given point in a point cloud (Rusu, Blodow, Marton, et al. 2008; Rusu, Blodow, and Beetz 2009). The RANSAC algorithm then randomly sample n_{RANSAC} points from

the source point cloud and detects the corresponding points in the target point cloud by finding the nearest-neighbours in the 33-dimensional FPHF space. Candidate matches are pruned by a fast pruning algorithm allowing to remove false matches. The remaining candidate matches are used to compute the transformation between the two point clouds, that is validated using the whole point clouds. The best transformation is finally retained.

H.1.2 Coloured Point Cloud Registration

Once a global alignment is determined, the transformation between the source and target point clouds can be used to initialise the local geometry- and colour-based refinement (Park et al. 2017). The coloured point cloud registration is based on an extended variant of the iterative closest point (ICP) algorithm which combines the point-to-plane ICP geometric objective $E_G(\mathbf{T})$ (Chen and Medioni 1992; Rusinkiewicz and Levoy 2001) and a photometric (colour-based) objective $E_C(T)$ into a combined objective

$$E(\mathbf{T}) = (1-\lambda)E_C(\mathbf{T}) + \lambda E_G(\mathbf{T}), \tag{H.1}$$

where $\mathbf{T}$ is the transformation matrix between the two point clouds to be determined. The geometric term is given by

$$E_G(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} ((\mathbf{p}-\mathbf{T}\mathbf{q})\cdot\mathbf{n}_\mathbf{p})^2 \tag{H.2}$$

where $\mathcal{K}$ is the correspondence set between the two point clouds at the current iteration, and $\mathbf{n}_\mathbf{p}$ is the normal of point $\mathbf{p}$—which can be estimated. Similarly, the photometric objective is defined as

$$E_C(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} ((C_\mathbf{p}(\mathbf{f}_\mathbf{p}(\mathbf{T}\mathbf{q})) - C(\mathbf{q}))^2 \tag{H.3}$$

where $C_\mathbf{p}(\cdot)$ is a continuous generalisation of $C(\cdot)$—which returns the colour of its argument—giving the colour of $\mathbf{q}$ on the tangent plane of $\mathbf{p}$ (defined by $\mathbf{n}_\mathbf{p}$), while $\mathbf{f}_\mathbf{p}(\mathbf{s}) = \mathbf{s} - \langle\mathbf{n}_\mathbf{p}, \mathbf{s}-\mathbf{p}\rangle\mathbf{n}_\mathbf{p}$ projects a point to the tangent plane. Starting with an initial approximated alignment—often obtained by global registration—the local geometry- and colour-based alignment given by the transformation $\mathbf{T}$ is iteratively refined by finding the current correspondence set $\mathcal{K}$ and subsequently update $\mathbf{T}$ to minimise $E(\mathbf{T})$ until convergence criteria are met.

	Atoms	Reason
Class I	Non-polar H, Cl, Br, I	Non-polar surface
Class II	Polar H, N, O, S, F	Polar surface
Class III	C, P, B	"Skeleton atoms"
Class IV	Everything else	—

Table H.1: SENSAAS colour classes identifying typical pharmacophoric features.

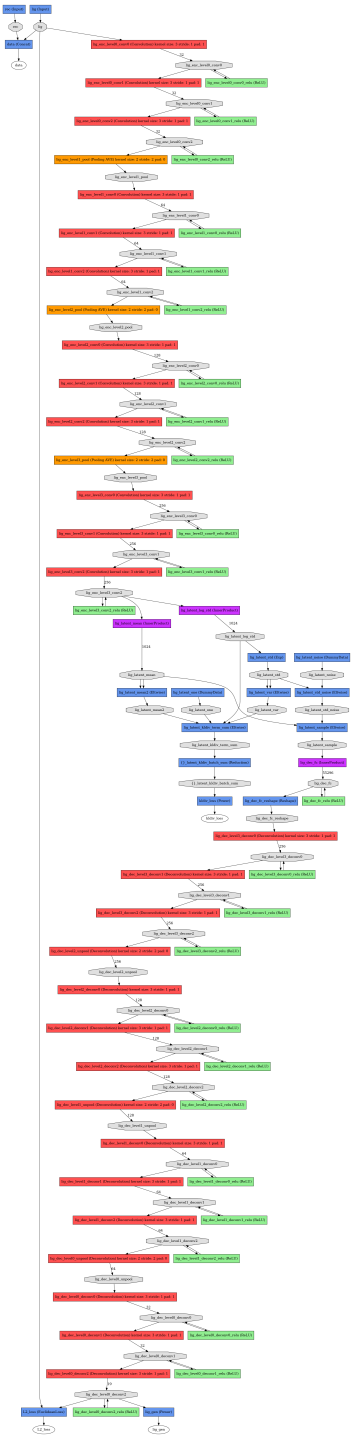


Figure H.1: Architecture of the ligand VAE as implemented in `liGAN`, obtained with Caffe's `draw_net.py` script.

Figure H.2: Architecture of the receptor-constrained VAE architecture as implemented in `liGAN`, obtained with Caffe's `draw_net.py` script.

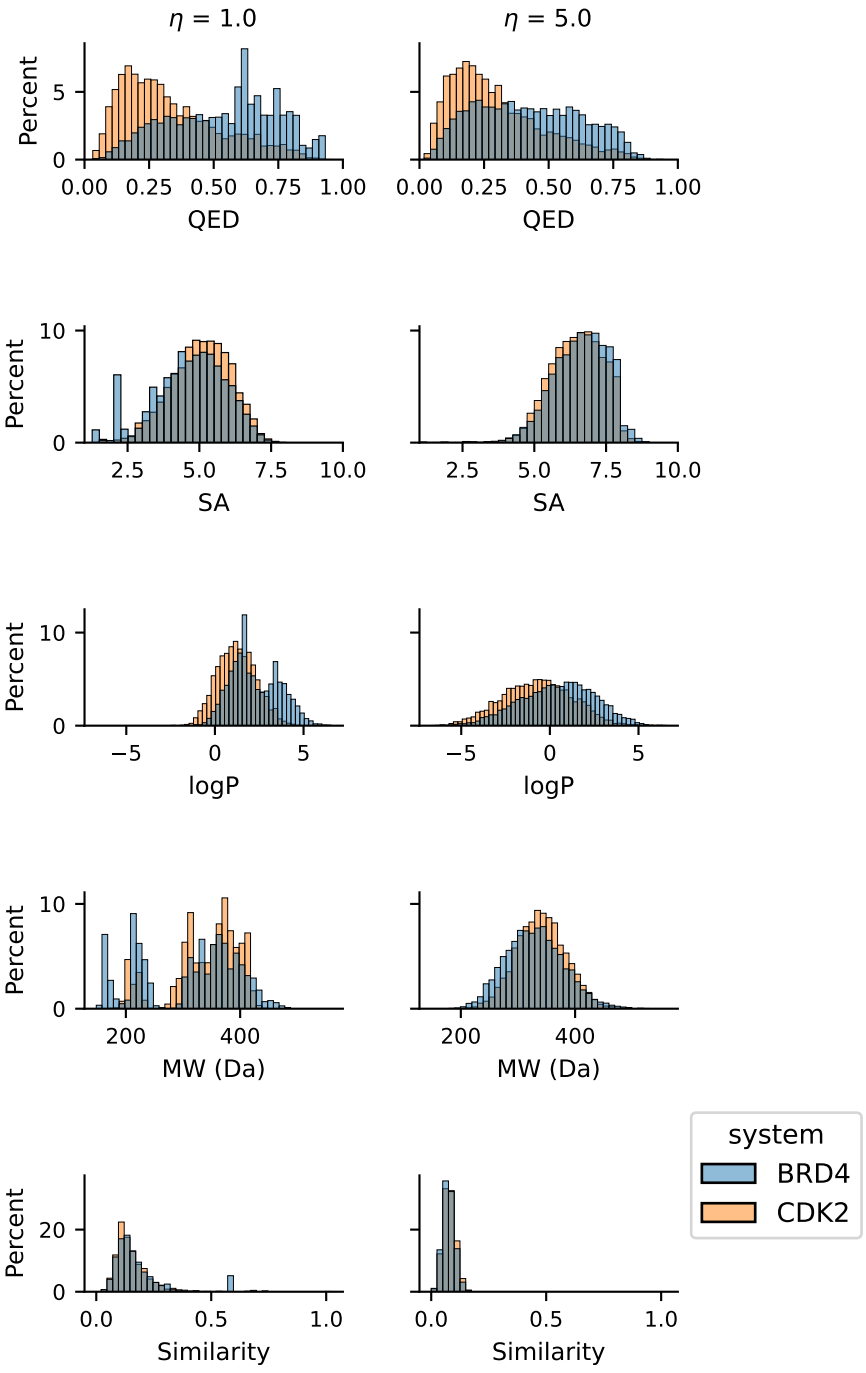


Figure H.3: Distributions of QED, SA, log p, MW, and similarity with the seed for molecules generated with the ligand VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds.

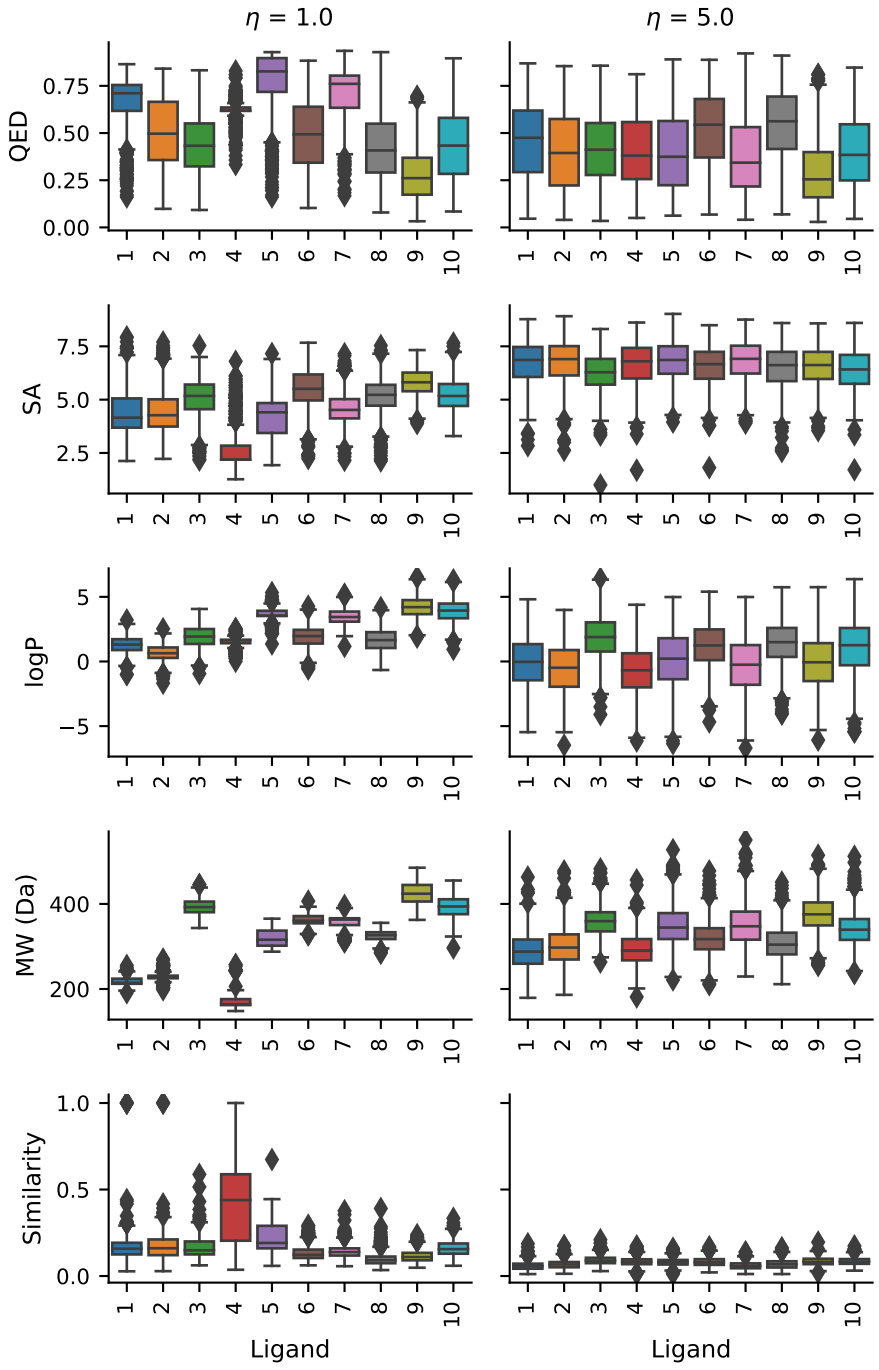


Figure H.4: Distributions of QED, SA, log p, MW, and similarity with the seed for molecules generated with the ligand VAE using BRD4 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta = 1.0$ and $\eta = 5.0$).

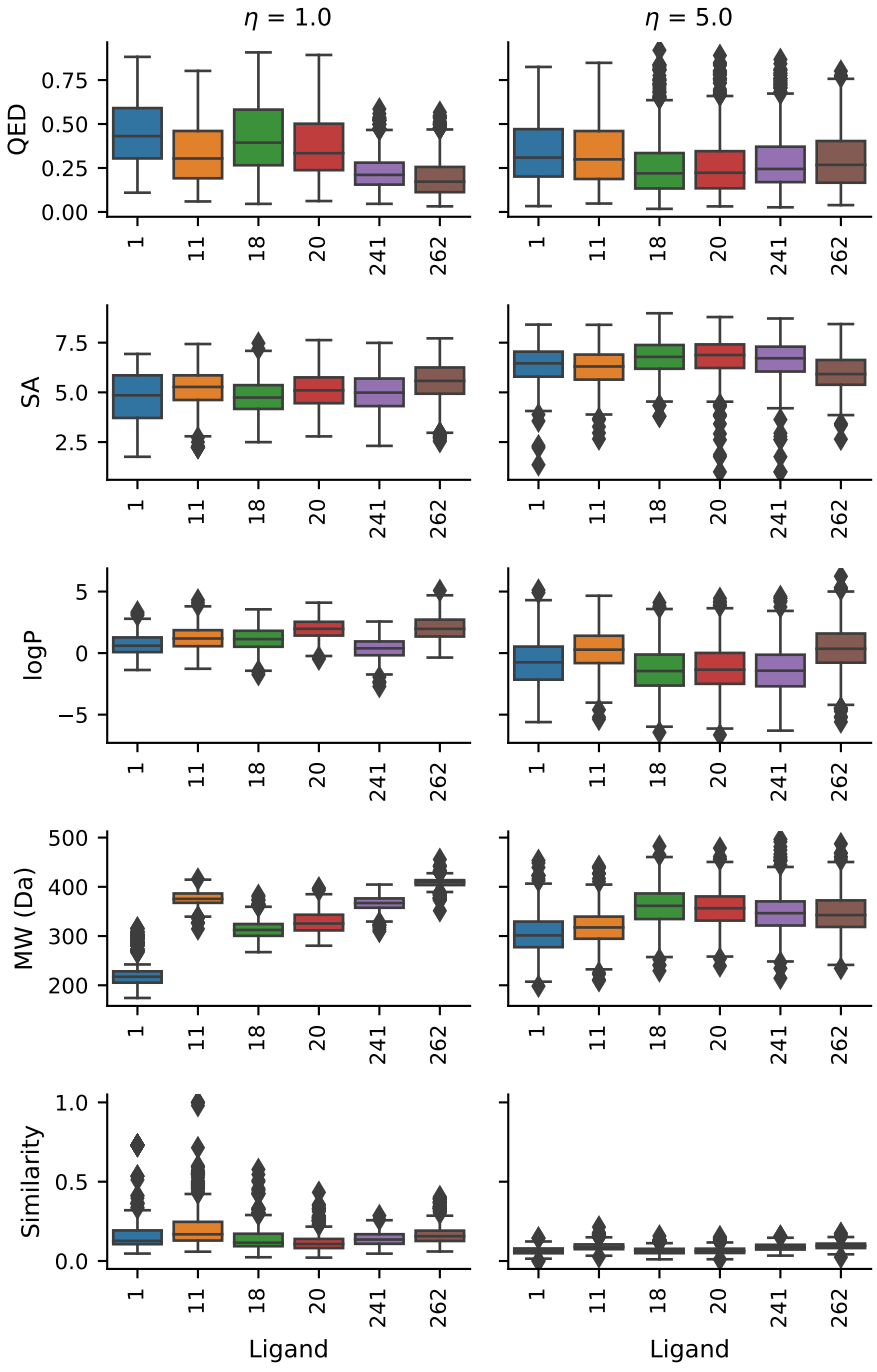
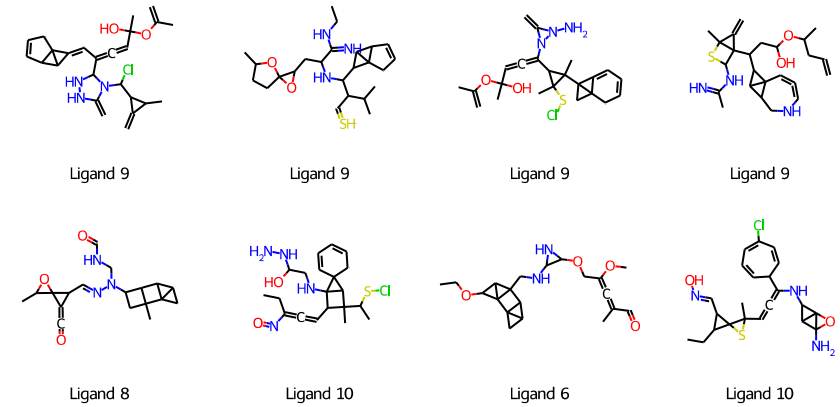
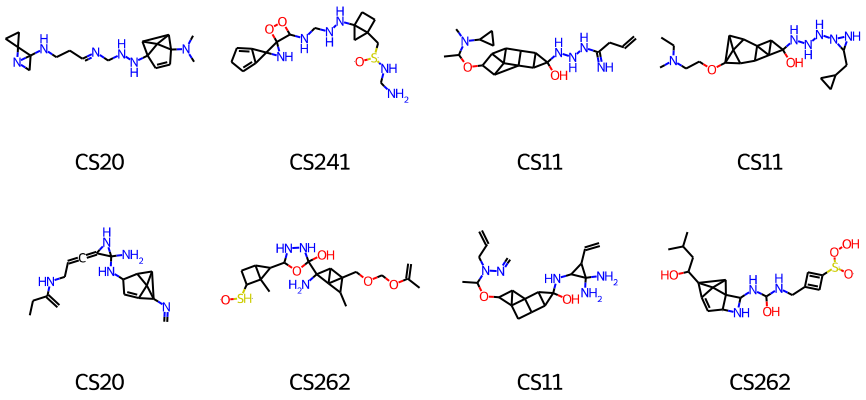


Figure H.5: Distributions of QED, SA, log p, MW and similarity with the seed for molecules generated with the ligand VAE using CDK2 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta = 1.0$ and $\eta = 5.0$).



(a) BRD4



(b) CDK2

Figure H.6: Molecules generated from BRD4 and CDK2 seeds using the ligand VAE with a variability factor $\eta = 1.0$. Generated molecules are ordered from lowest (top left) to highest (bottom right) RMSD_{UFF}, and they are filtered according to the following criteria: QED $\in [0.0, 0.2]$ and SA $\in [7, 10]$.

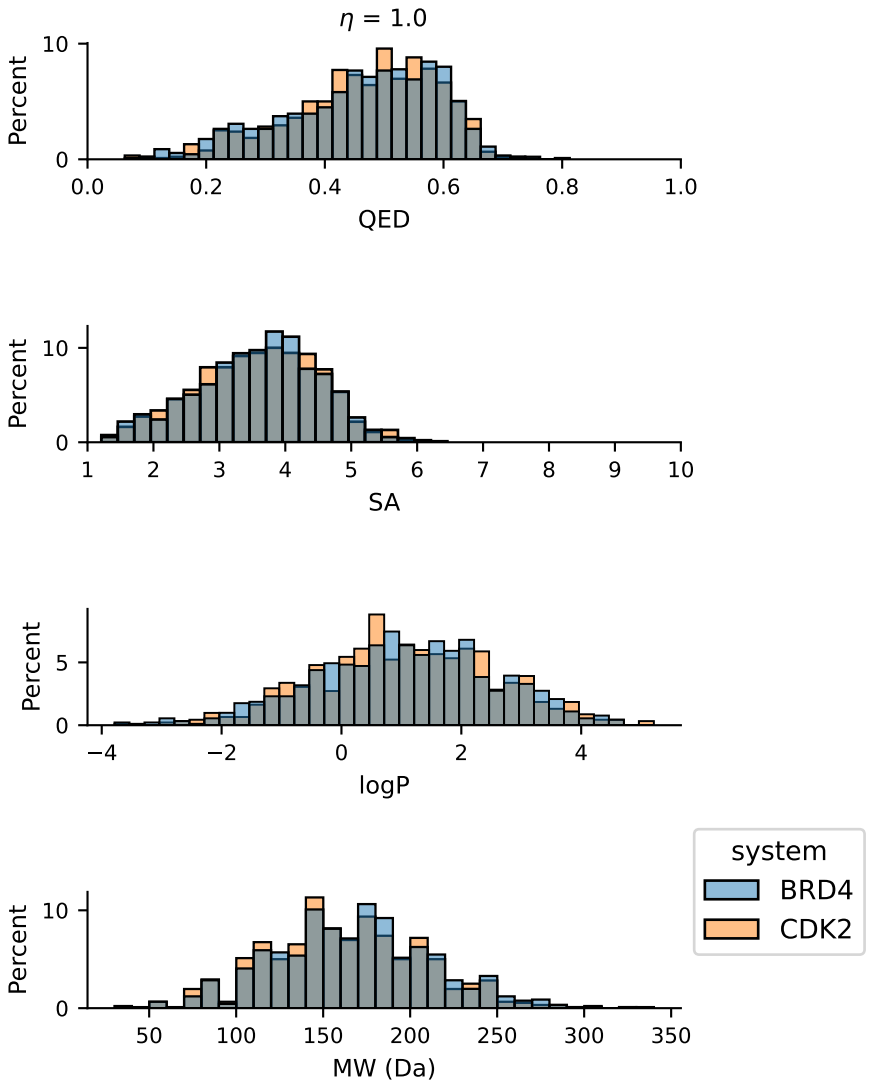


Figure H.7: Distributions of QED, SA, log p, and MW for generated molecules sampled from the prior distribution. Molecules are generated with the ligand VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds. The seed is ignored when sampling the prior distribution (but required as input), as reflected in the similarity between the distributions.

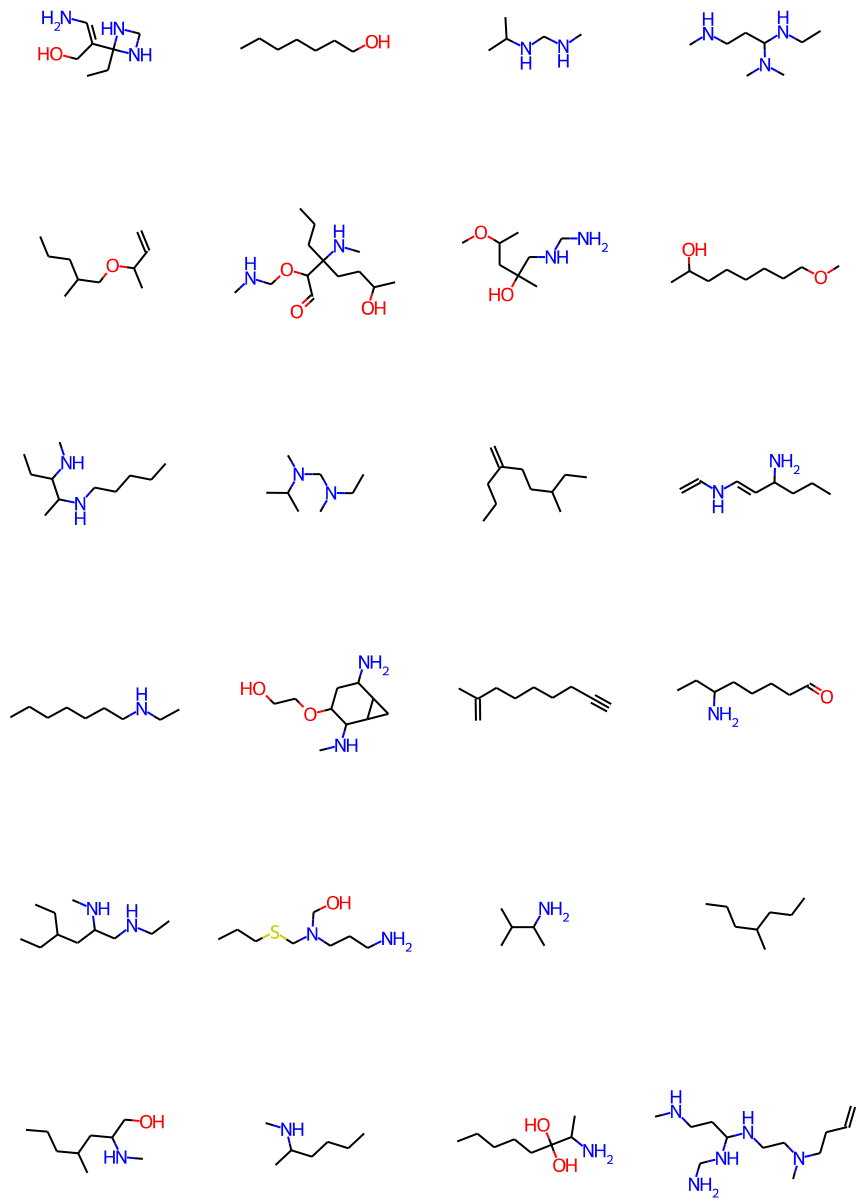


Figure H.8: Random sample of molecules generated from the prior distribution of the ligand VAE.

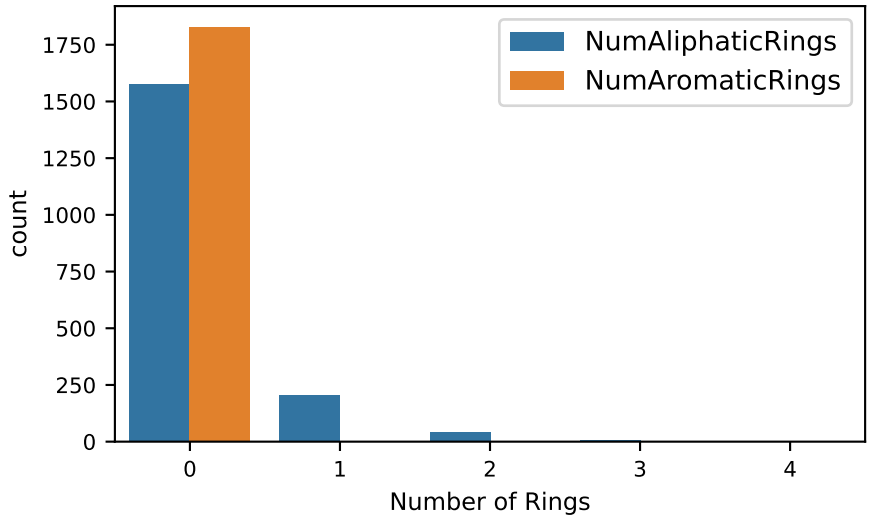


Figure H.9: Number of aromatic and aliphatic rings in molecules generated by sampling 2000 times from the prior distribution of the ligand VAE.

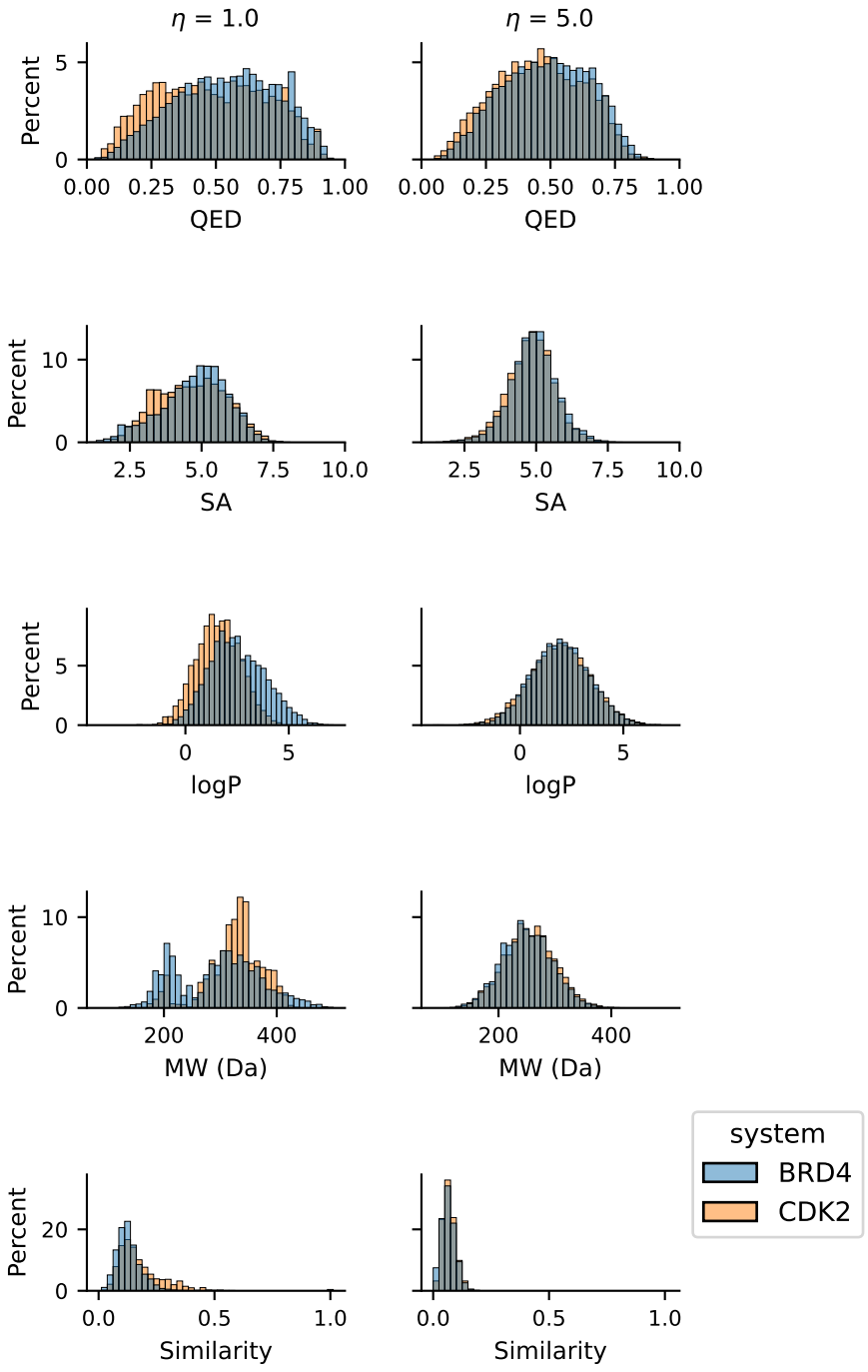


Figure H.10: Distributions of QED, SA, log p, MW, and similarity with the seed for generated molecules. Molecules are generated with the receptor-constrained VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds.

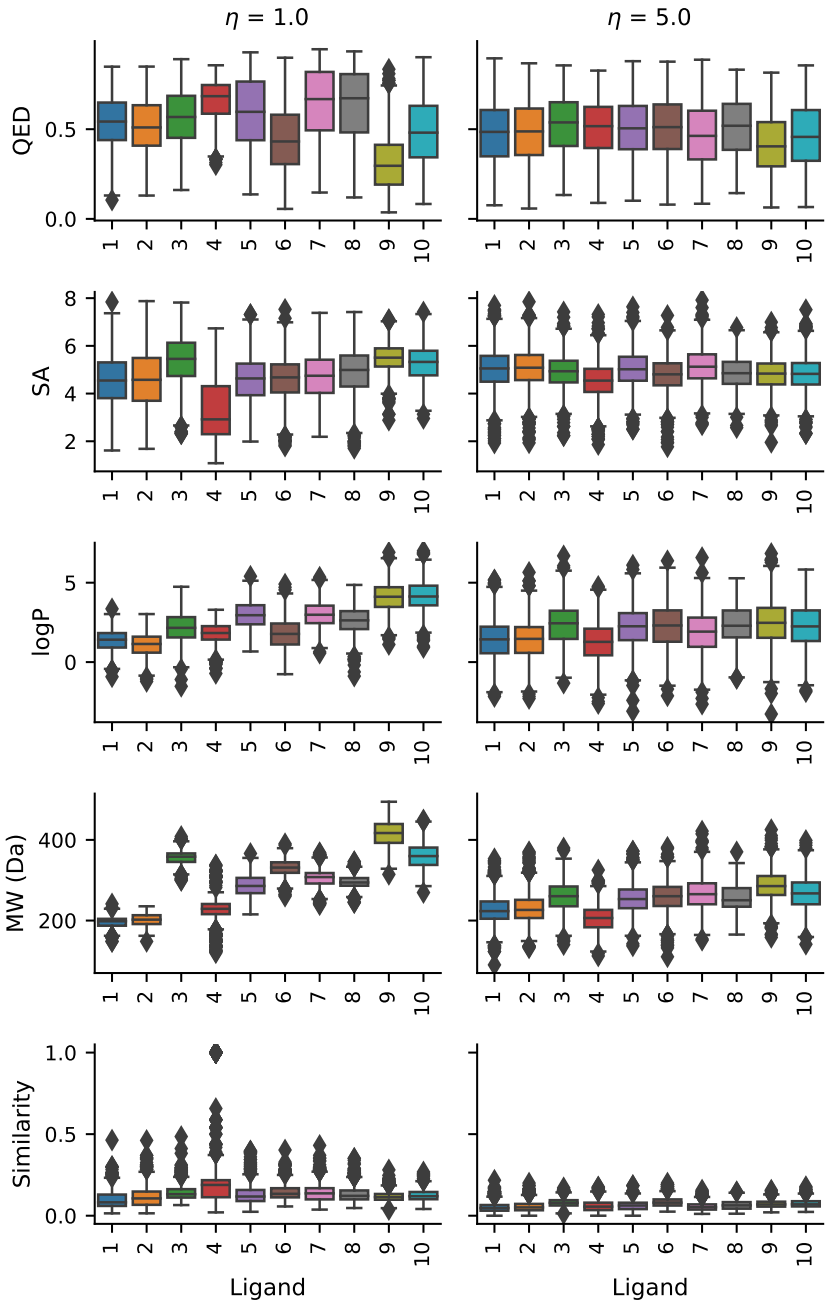


Figure H.11: Distributions of QED, SA, $\log p$, MW, and similarity with the seed for molecules generated with the receptor-constrained VAE using BRD4 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta \in \{1.0, 5.0\}$).

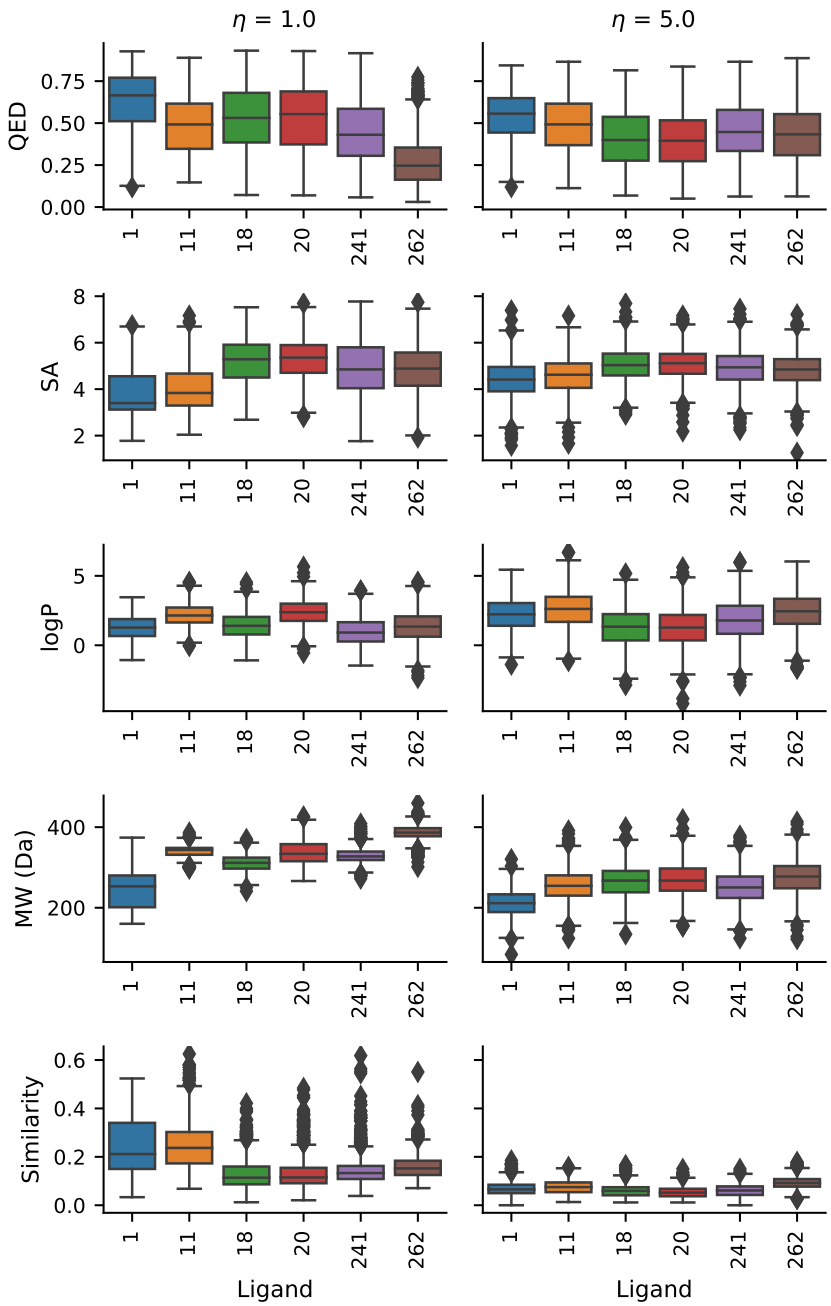


Figure H.12: Distributions of QED, SA, log p, MW, and similarity with the seed for molecules generated with the receptor-constrained VAE using CDK2 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta \in \{1.0, 5.0\}$).

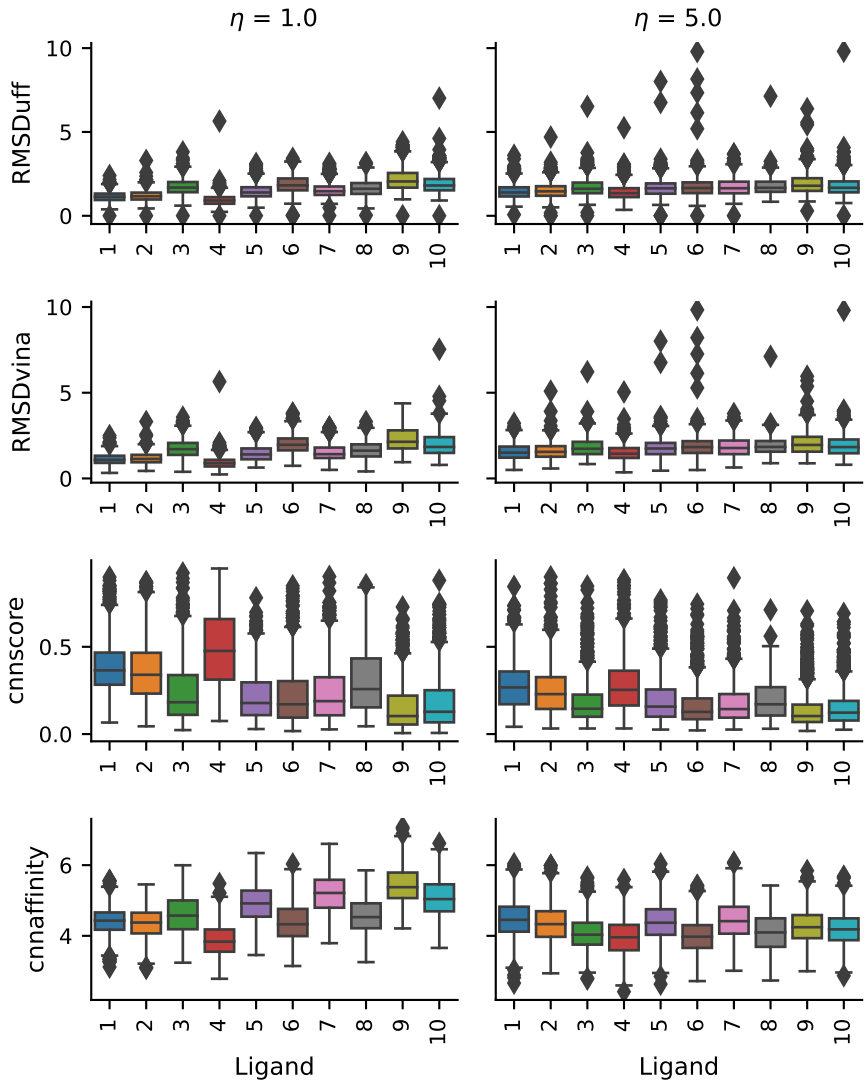


Figure H.13: Distributions of RMSD, CNNscore, and CNNaffinity for molecules generated with the receptor-constrained VAE using BRD4 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta = 1.0$ and $\eta = 5.0$).

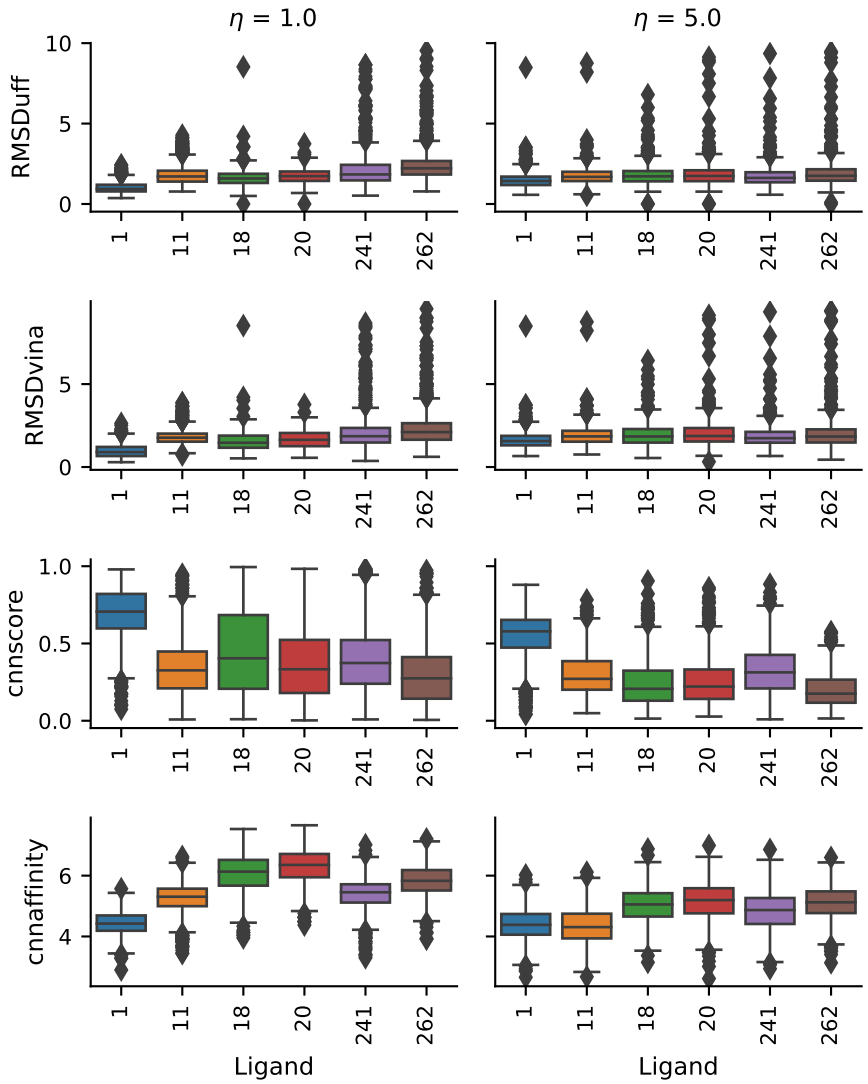


Figure H.14: Distributions of RMSD, CNNscore, and CNNaffinity for molecules generated with the receptor-constrained VAE using CDK2 inhibitors as seed. Distributions are shown for each seed separately and for different variability factors ($\eta = 1.0$ and $\eta = 5.0$).

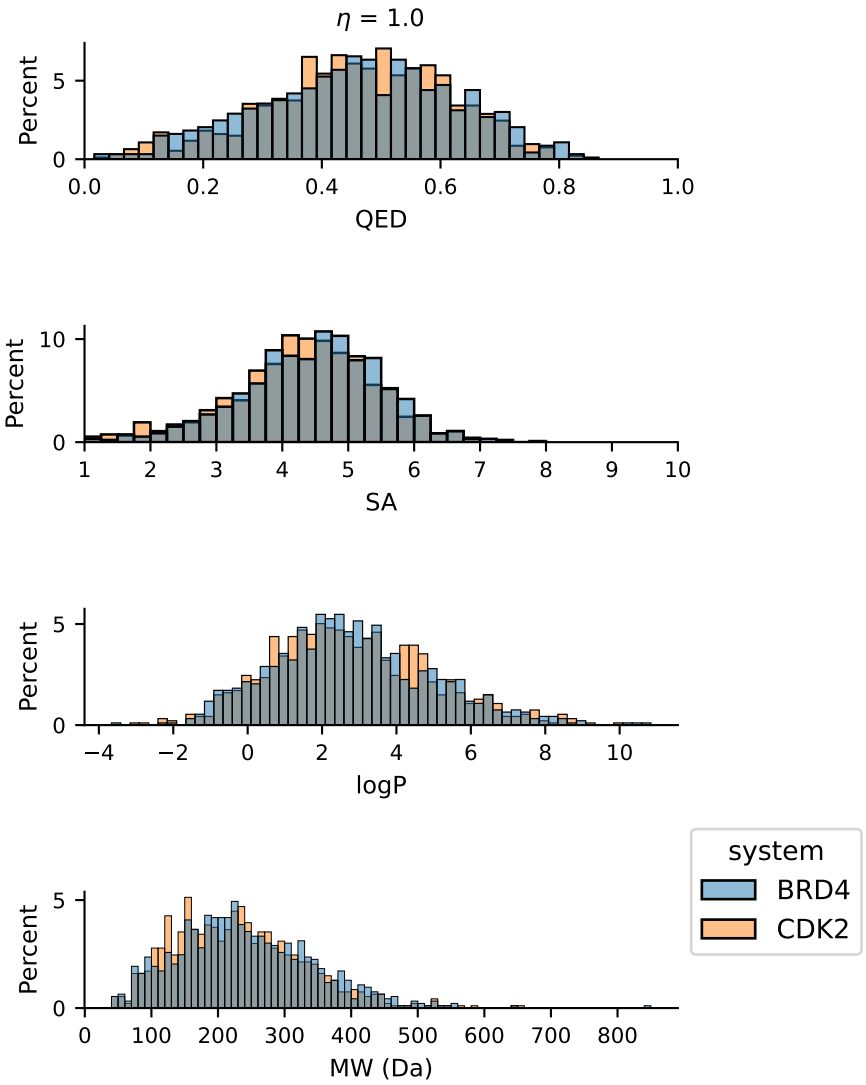


Figure H.15: Distributions of QED, SA, log p, and MW for generated molecules sampled from the prior distribution. Molecules are generated with the receptor-conditional VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds.

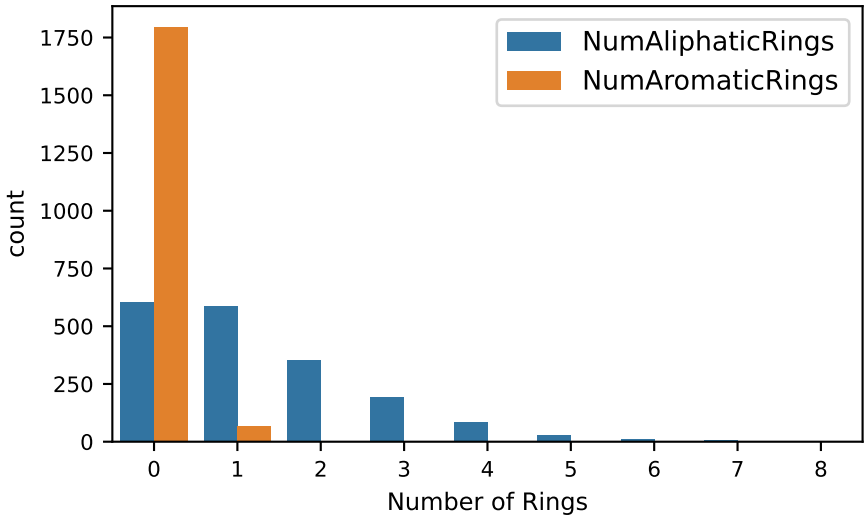
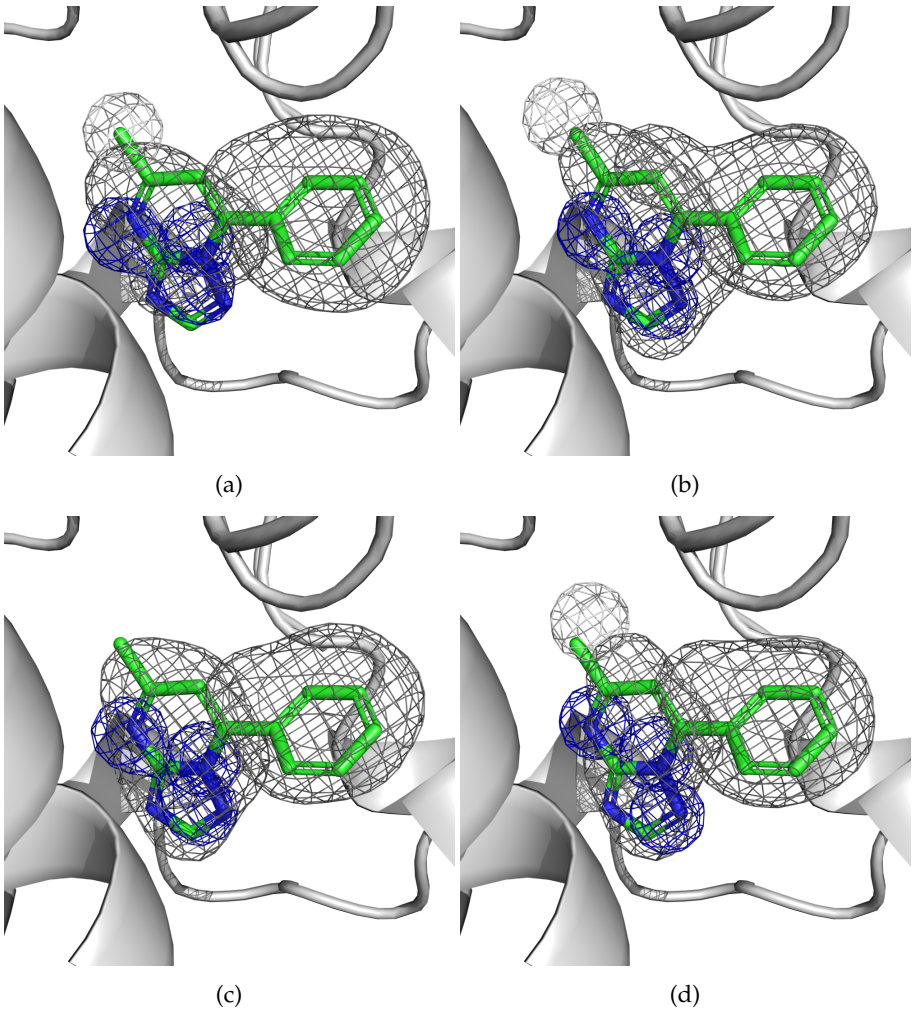


Figure H.16: Number of aromatic and aliphatic rings in molecules generated by sampling 1000 times from the prior distribution of the receptor conditional VAE, for both BRD4 and CDK2.



(a) (b)

(c) (d)

Figure H.17: Random examples of generated densities for BRD4, displayed around the corresponding seed ligand (ligand 1). Densities are generated using the receptor-conditional VAE.

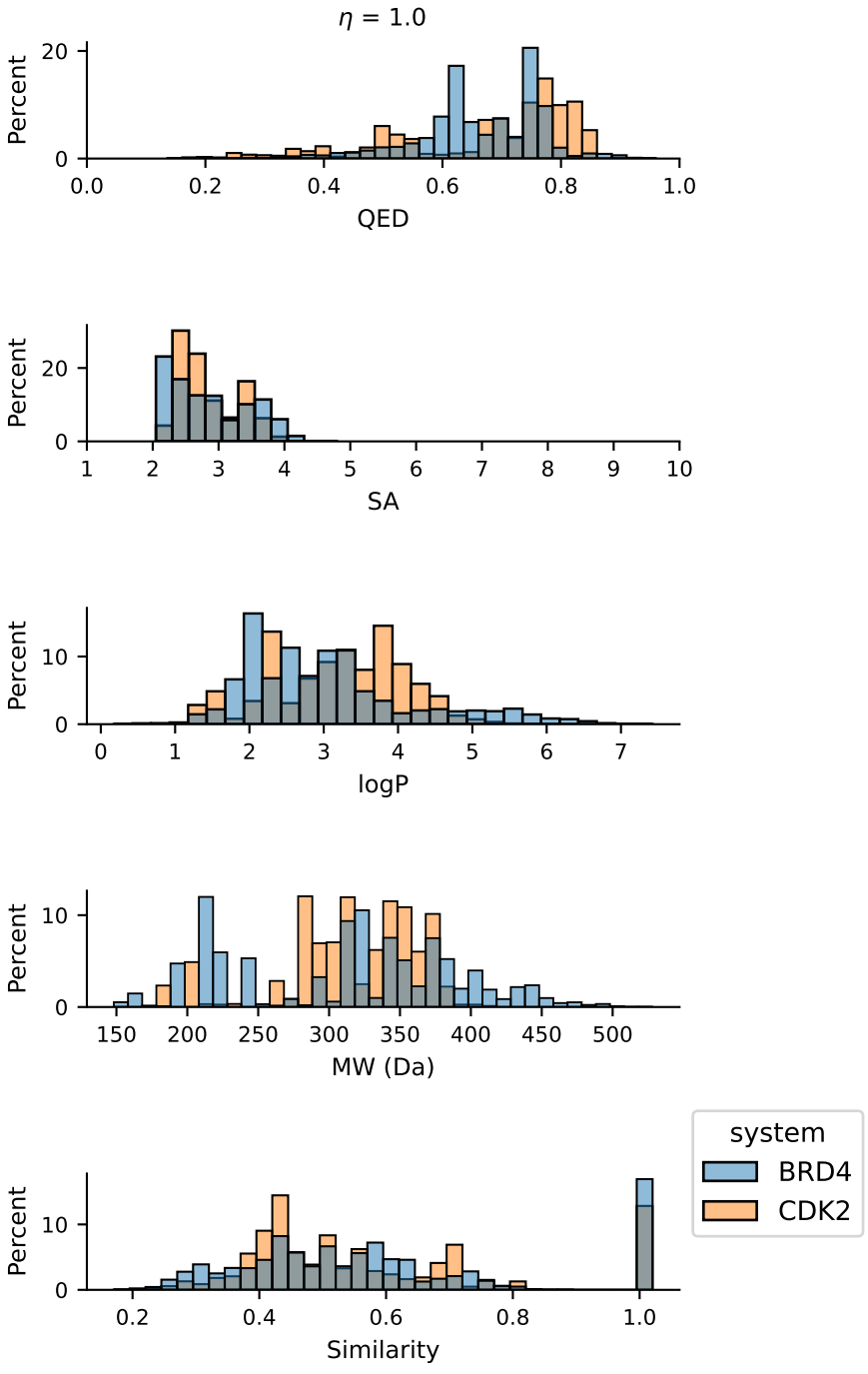


Figure H.18: Distributions of selected molecular properties for generated molecules. Molecules are generated with the receptor-constrained VAE using BRD4 (blue) and CDK2 (orange) inhibitors as seeds, and applying the scaffold trick.

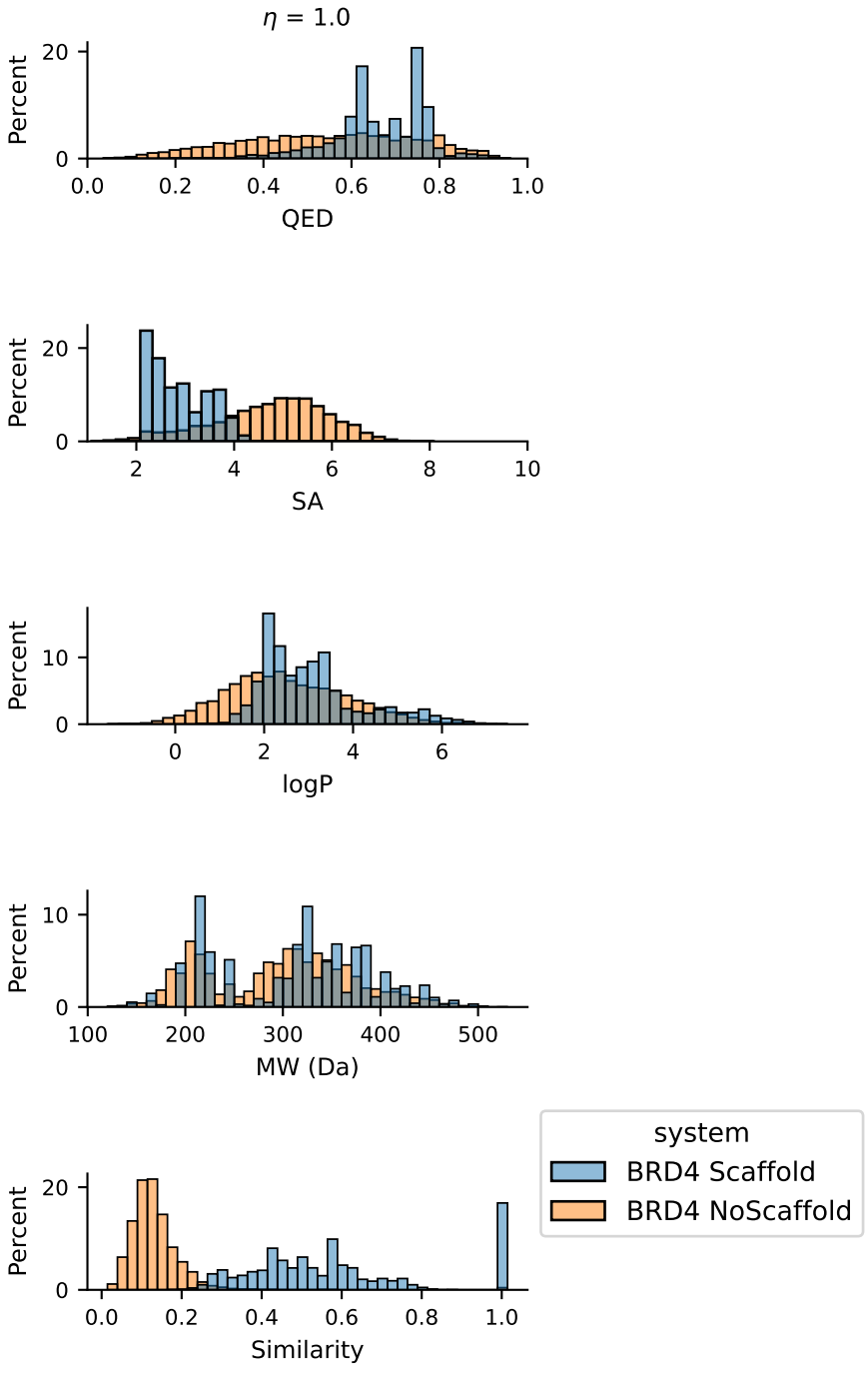


Figure H.19: Comparison of distributions of selected molecular properties for generated molecules starting from BRD4 inhibitors. Molecules are generated with the receptor-constrained VAE with (blue) and without (orange) the scaffold trick.

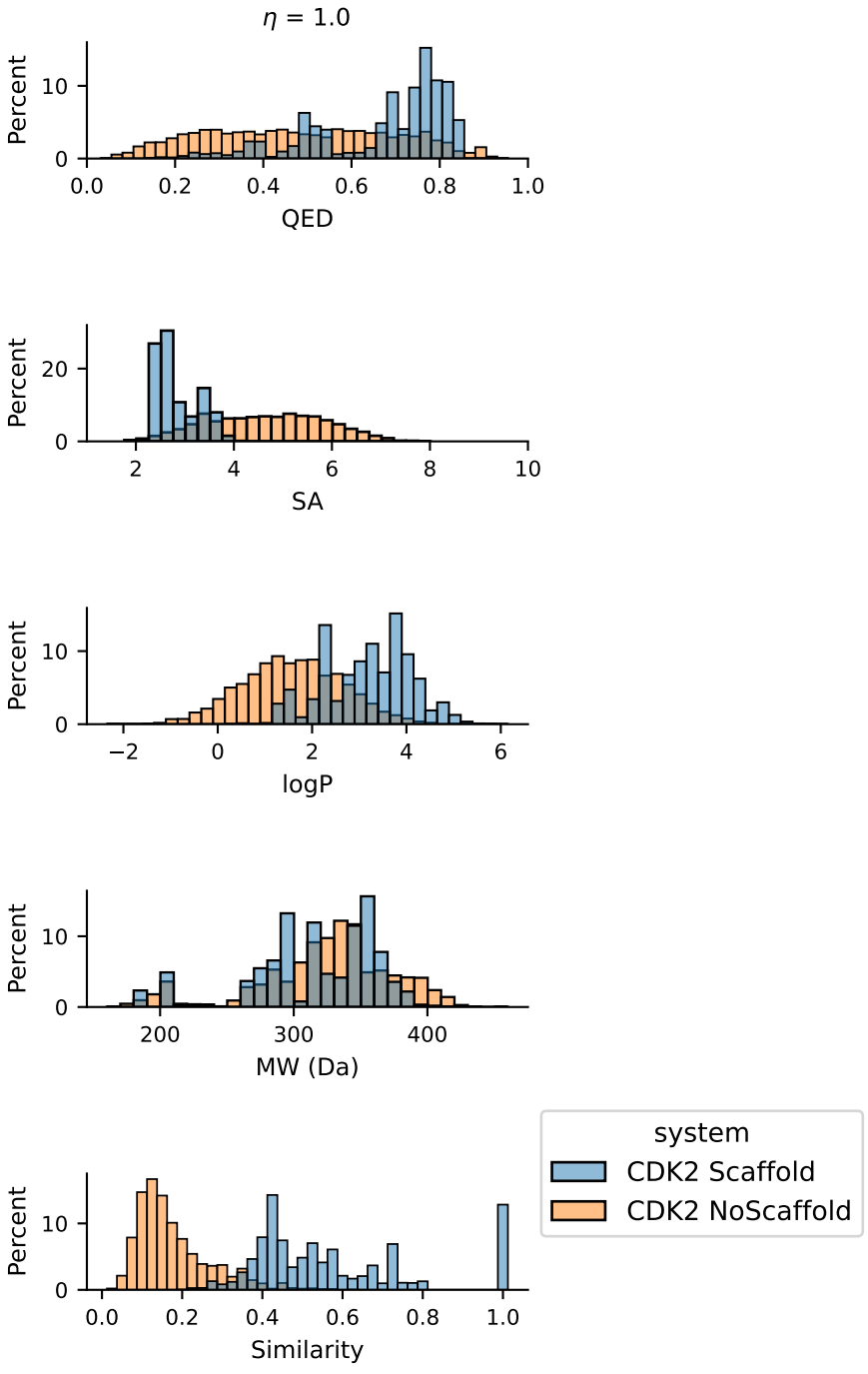


Figure H.20: Comparison of distributions of selected molecular properties for generated molecules starting from CDK2 inhibitors. Molecules are generated with the receptor-constrained VAE with (blue) and without (orange) the scaffold trick.

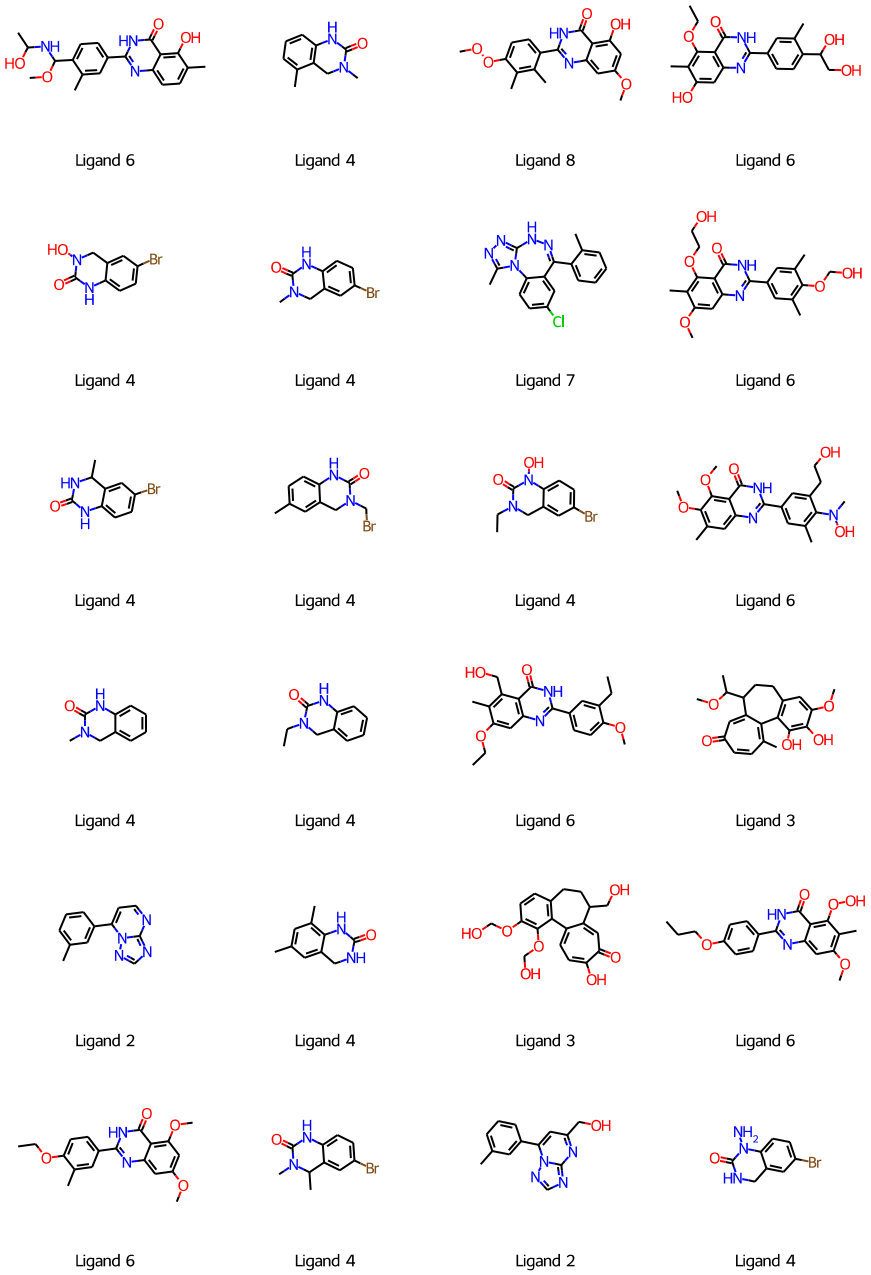


Figure H.21: Top 24 BRD4 inhibitors—according to Gnina's CNNaffinity—generated using the scaffold trick.

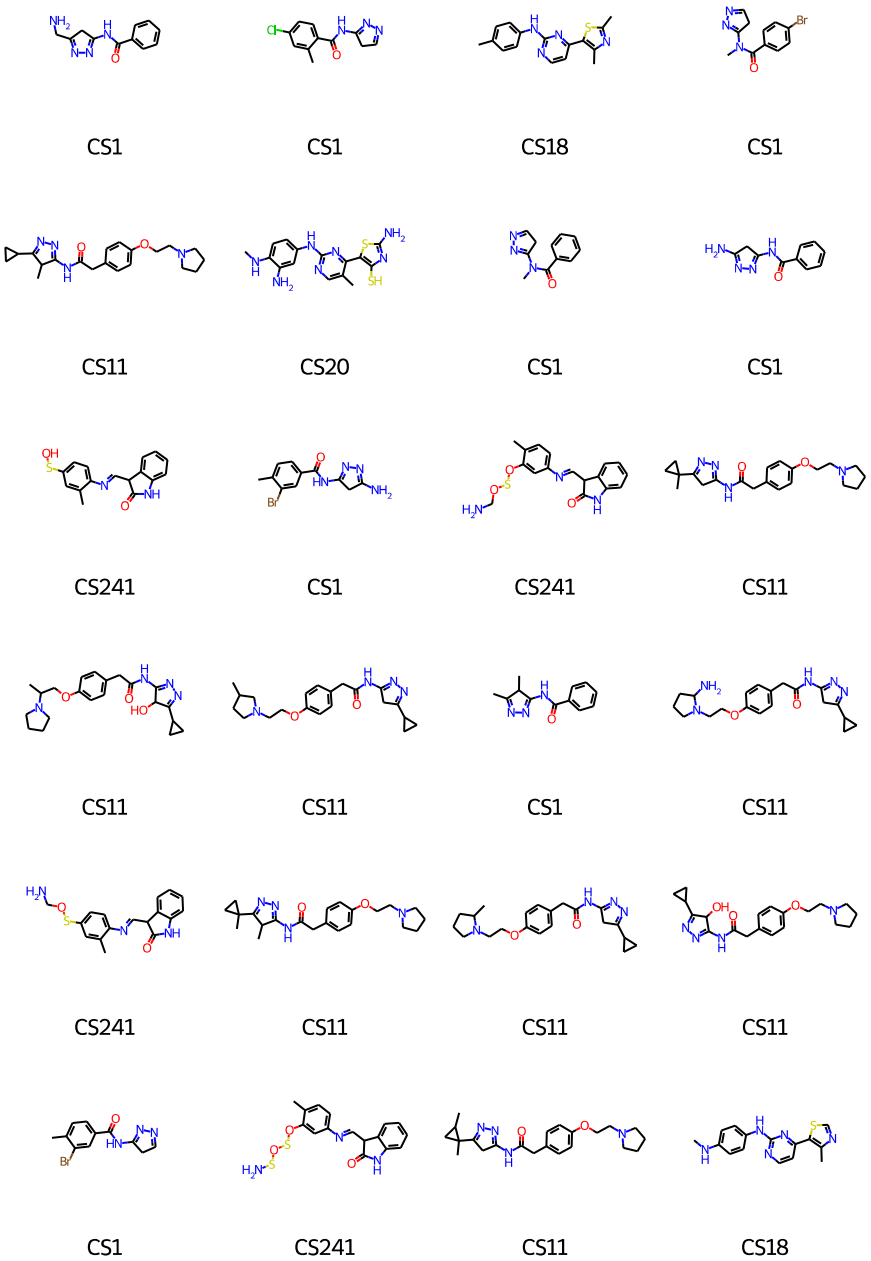


Figure H.22: Top 24 CDK2 inhibitors—according to GNINA's CNNaffinity—generated using the scaffold trick.

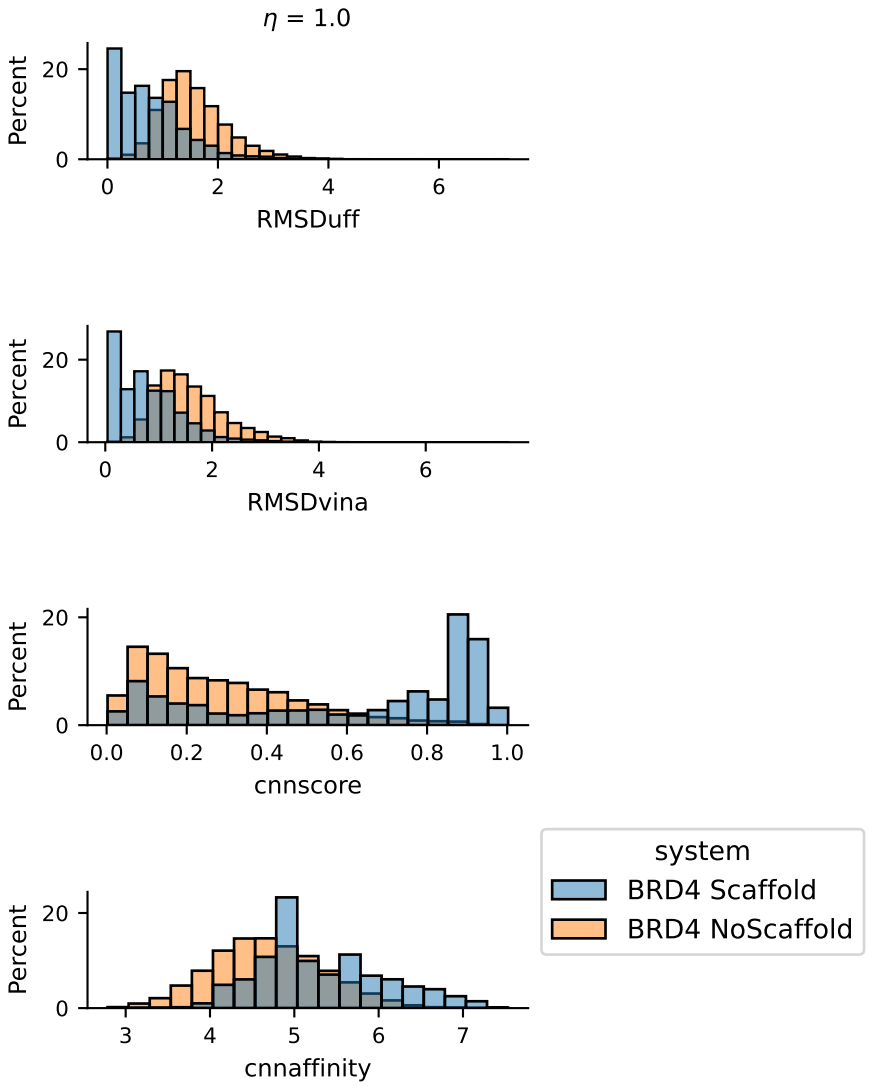


Figure H.23: Comparison of distributions of RMSD, CNNscore, and CNNaffinity for generated molecules starting from BRD4 inhibitors. Molecules are generated with the receptor-constrained VAE with (blue) and without (orange) the scaffold trick.

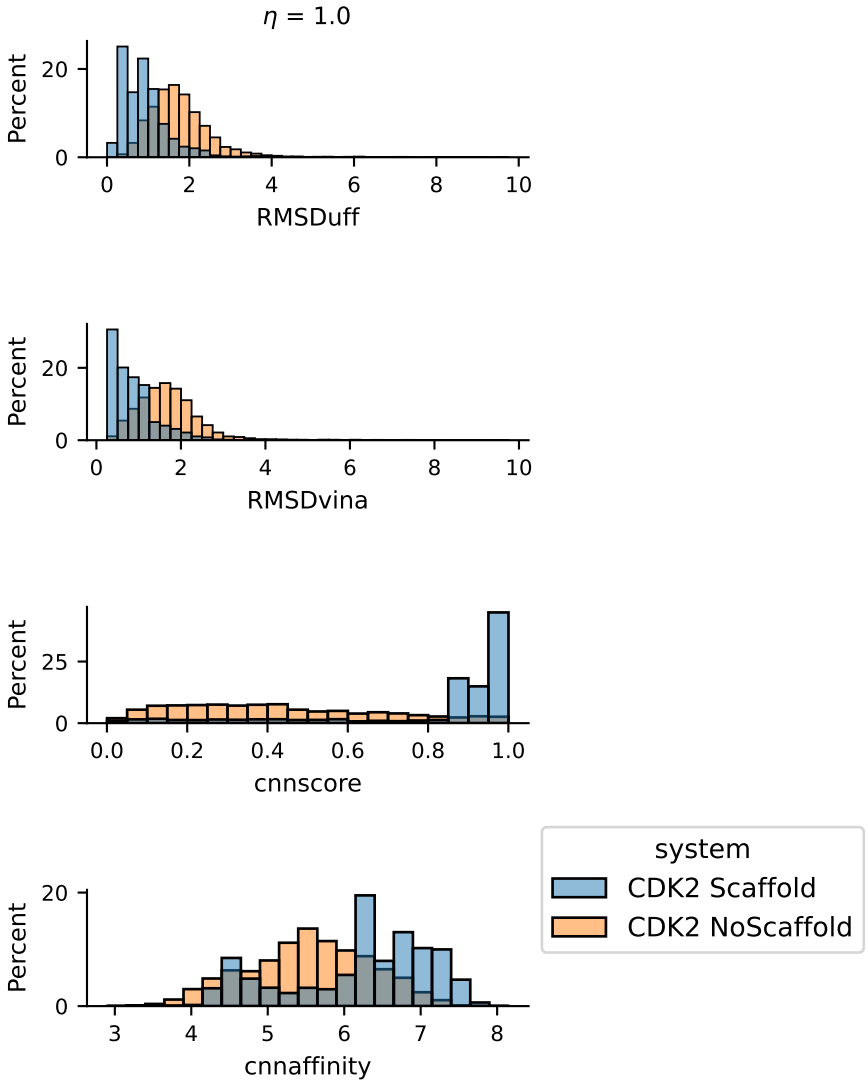


Figure H.24: Comparison of distributions of RMSD, CNNscore, and CNNaffinity for generated molecules starting from CDK2 inhibitors. Molecules are generated with the receptor-constrained VAE with (blue) and without (orange) the scaffold trick.

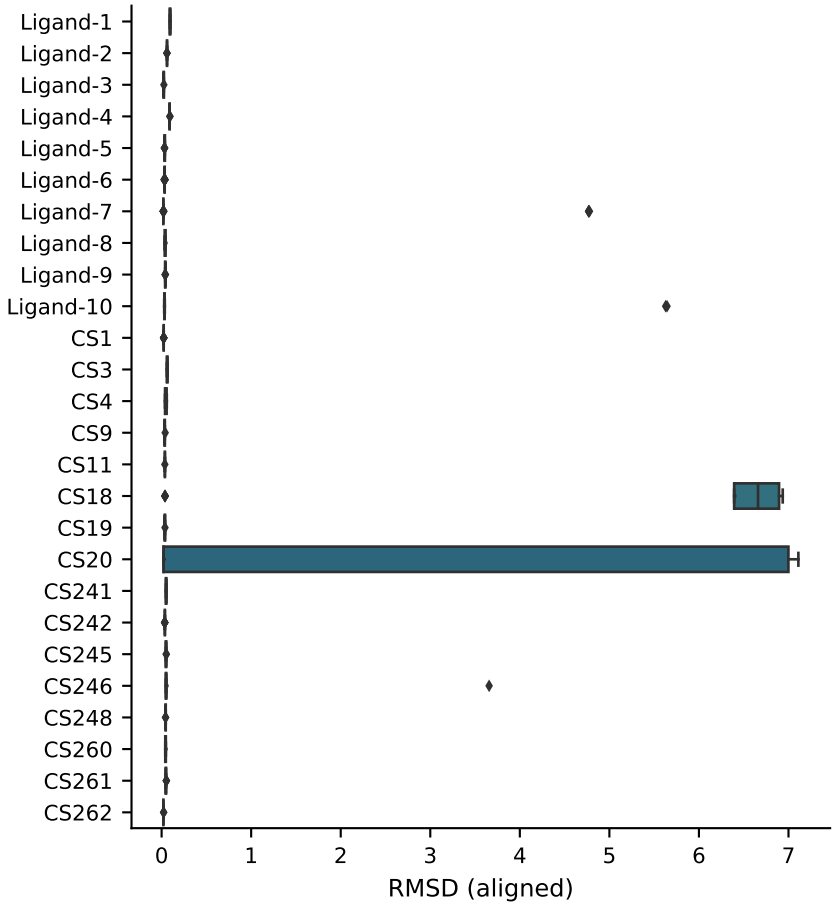
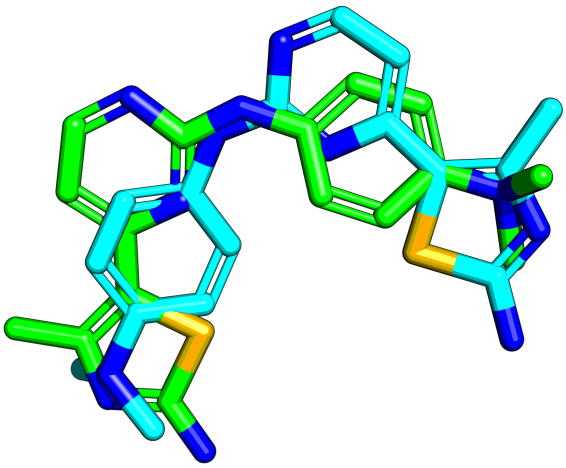
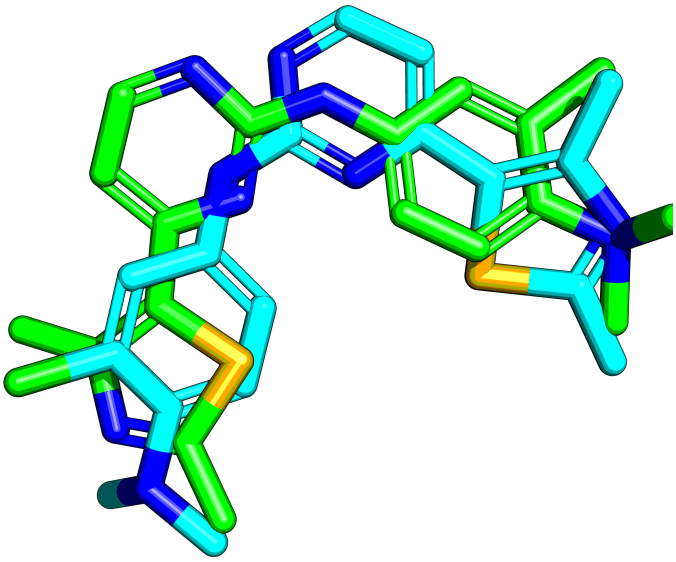


Figure H.25: Distributions of final RMSDs—over 25 repeats—for the alignment of a molecule and its randomly rotated and translated counterpart, obtained using `libmolgrid`-generated densities and the SENSAAS colour scheme.

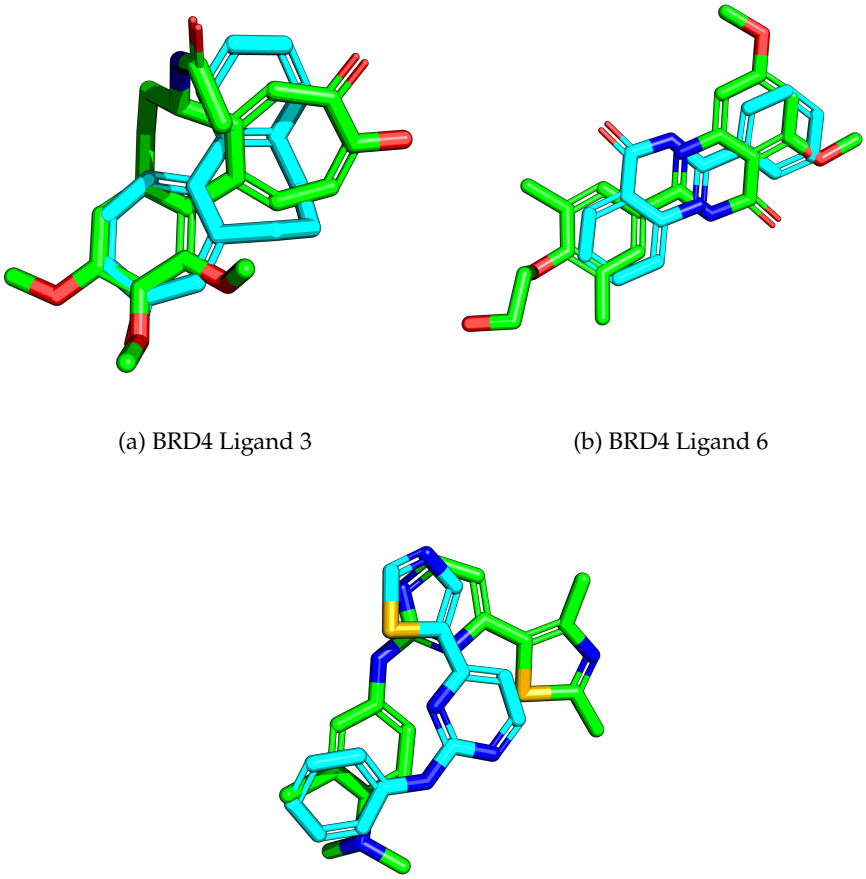


(a) CS18



(b) CS20

Figure H.26: Three dimensional visualisation of alignment failures (first alignment run) using SENSAAS' colouring scheme.



(a) BRD4 Ligand 3

(b) BRD4 Ligand 6

(c) CDK2 CS20

Figure H.27: Three dimensional visualisation of alignment failures (first alignment run) for Murcko scaffolds, using `libmolgrid`'s colouring scheme.

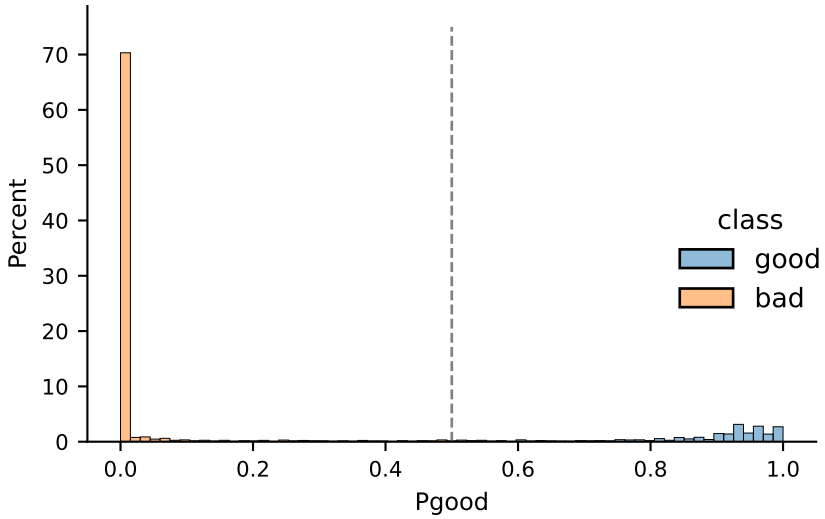
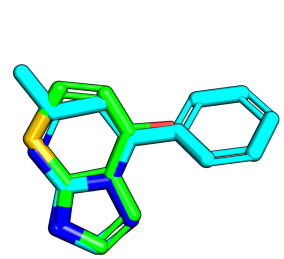
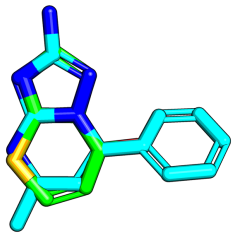


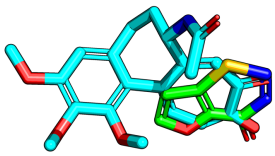
Figure H.28: Probability P_{good} of synthetically tractable molecules for the VEHICLe library. Here we consider a threshold $P_{good} = 0.5$ to distinguish synthetically tractable (good) from synthetically intractable (bad) fragments.



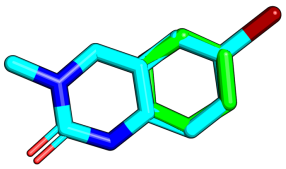
(a) Ligand 1



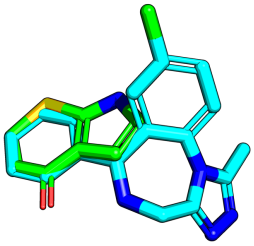
(b) Ligand 2



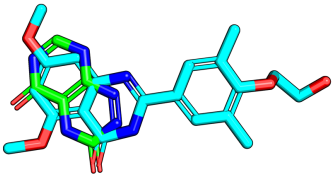
(c) Ligand 3



(d) Ligand 4

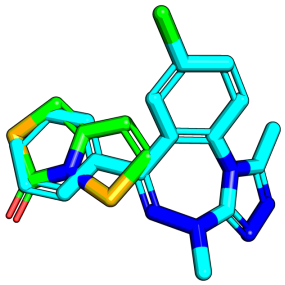


(e) Ligand 5

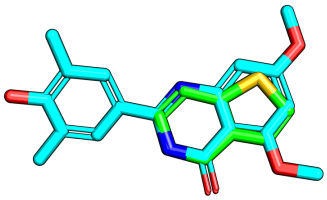


(f) Ligand 6

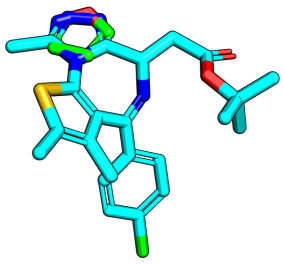
Figure H.29: Best alignments of VEHICLe fragments to BRD4 inhibitors (ligands 1 to 6) using SENSAAS.



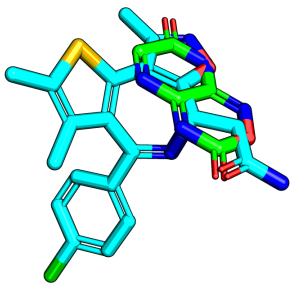
(a) Ligand 7



(b) Ligand 8

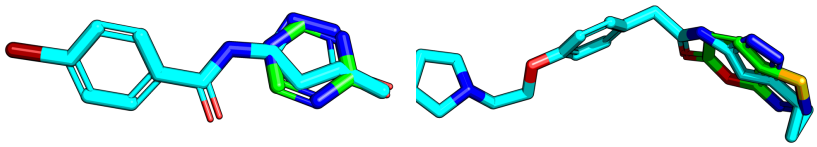


(c) Ligand 9



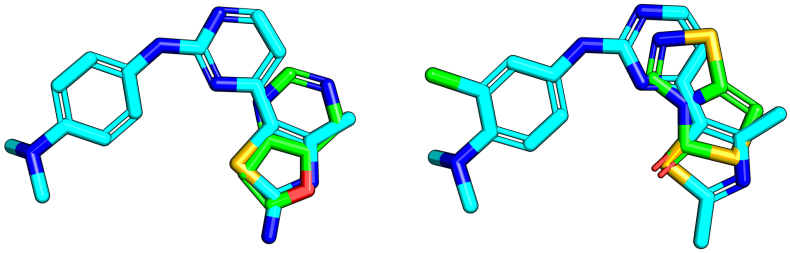
(d) Ligand 10

Figure H.30: Best alignments of VEHICLe fragments to BRD4 inhibitors (ligands 7 to 10) using SENSAAS.



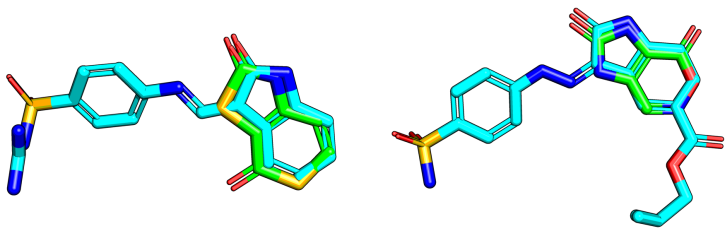
(a) Ligand CS1

(b) Ligand CS11



(c) Ligand CS18

(d) Ligand CS20



(e) Ligand CS241

(f) Ligand CS262

Figure H.31: Best alignments of VEHICLe fragments to CDK2 inhibitors using SENSAAS.

Software

The work presented here was made possible also thanks to a large amount of FOSS and other software. Since some software and libraries are not directly cited in the main text, most of the relevant citations are reported here.

Software

Bioinformatics software:

- ProBiS (Konc, Depolli, et al. 2012; Konc and Janežič 2010)

Docking software:

- AutoDock Vina (Trott and Olson 2009)
- GNINA (McNutt et al. 2021; Ragoza, Hochuli, et al. 2017)
- smina (Koes, Baumgartner, et al. 2013)

Visualisation software:

- ChimeraX (Goddard et al. 2017; Pettersen et al. 2020)
- PyMol (Schrödinger 2015)

Utilities:

- GNU Parallel (Bhatt et al. 1998)

Python

Python libraries for numerical computation:

- NetworkX (Hagberg et al. 2008)
- NumPy (Harris et al. 2020)

- pandas (McKinney 2010)
- SciPy (Virtanen et al. 2020)

Python libraries for visualisation:

- Matplotlib (Hunter 2007)
- PyVista (Sullivan and Kaszynski 2019)
- seaborn (Waskom 2021)

Domain-specific Python libraries:

- libmolgrid (Sunseri and Koes 2020)
- MDAnalysis (Gowers et al. 2016; Michaud-Agrawal et al. 2011)
- Open3D (Zhou, Park, et al. 2018)
- ProDy (Bakan, Dutta, et al. 2014; Bakan, Meireles, et al. 2011; Zhang et al. 2021)
- TorchANI (Gao, Ramezanghorbani, et al. 2020)

ML and DL frameworks:

- Caffe (Jia et al. 2014)
- PyTorch (Paszke et al. 2019)
- scikit-learn (Abraham et al. 2014)

Other Python libraries:

- Snakemake (Köster and Rahmann 2012; Mölder et al. 2021)

Bibliography

Journal Articles

Abraham, Alexandre, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux

2014 "Machine learning for neuroimaging with scikit-learn", *Front. Neuroinf.*, 8, 85, pp. 2825-2830, DOI: 10.3389/fninf.2014.00014.

Ackloo, Suzanne, Rima Al-awar, Rommie E Amaro, Cheryl H Arrowsmith, Hatylas Azevedo, Robert A Batey, Yoshua Bengio, Ulrich AK Betz, Cristian G Bologa, John D Chodera, Wendy D. Cornell, Ian Dunham, Gerhard F. Ecker, Kristina Edfeldt, Aled M. Edwards, Michael K. Gilson, Claudia R. Gordijo, Gerhard Hessler, Alexander Hillisch, Anders Hogner, John J. Irwin, Johanna M. Jansen, Daniel Kuhn, Andrew R. Leach, Alpha A. Lee, Uta Lessel, Maxwell R. Morgan, John Moult, Ingo Muegge, Tudor I. Oprea, Benjamin G. Perry, Patrick Riley, Sophie A. L. Rousseaux, Kumar Singh Saikatendu, Vijayaratnam Santhakumar, Matthieu Schapira, Cora Scholten, Matthew H. Todd, Masoud Vedadi, Andrea Volkamer, and Timothy M. Willson

2022 "CACHE (Critical Assessment of Computational Hit-finding Experiments): A public–private partnership benchmarking initiative to enable the development of computational methods for hit-finding", *Nat. Rev. Chem.*, 6, 4, pp. 287-295, DOI: 10.1038/s41570-022-00363-z.

Adcock, Stewart A. and J. Andrew McCammon

2006 "Molecular Dynamics: Survey of Methods for Simulating the Activity of Proteins", *Chem. Rev.*, 106, 5, pp. 1589-1615, DOI: 10.1021/cr040426m.

Afifi, Karim and Ahmed Farouk Al-Sadek

2018 "Improving classical scoring functions using random forest: The non-additivity of free energy terms' contributions in binding", *Chem. Biol. Drug Des.*, 92, 2, pp. 1429-1434, DOI: 10.1111/cbdd.13206.

Aggarwal, Rishal, Akash Gupta, Vineeth Chelur, C. V. Jawahar, and U. Deva Priyakumar

2021 "DeepPocket: Ligand Binding Site Detection and Segmentation using 3D Convolutional Neural Networks", *J. Chem. Inf. Model.*, DOI: 10.1021/acs.jcim.1c00799.

Ahmed, Aqeel, Richard D. Smith, Jordan J. Clark, James B. Dunbar, and Heather A. Carlson

2014 "Recent improvements to Binding MOAD: A resource for protein–ligand binding affinities and structures", *Nucleic Acids Res.*, 43, D1, pp. D465-D469, DOI: 10.1093/nar/gku1088.

Ain, Qurrat Ul, Antoniya Aleksandrova, Florian D. Roessler, and Pedro J. Ballester

2015 "Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening", *WIREs Comput. Mol. Sci.*, 5, 6, pp. 405-424, DOI: 10.1002/wcms.1225.

Albanese, Steven K., John D. Chodera, Andrea Volkamer, Simon Keng, Robert Abel, and Lingle Wang

2020 "Is Structure-Based Drug Design Ready for Selectivity Optimization?", *J. Chem. Inf. Model.*, 60, 12, pp. 6211-6227, DOI: 10.1021/acs.jcim.0c00815.

Aldeghi, Matteo, Alexander Heifetz, Michael J. Bodkin, Stefan Knapp, and Philip C. Biggin

2016 "Accurate calculation of the absolute free energy of binding for drug molecules", *Chem. Sci.*, 7, 1, pp. 207-218, DOI: 10.1039/c5sc02678d.

2017 "Predictions of Ligand Selectivity from Absolute Binding Free Energy Calculations", *J. Am. Chem. Soc.*, 139, 2, pp. 946-957, DOI: 10.1021/jacs.6b11467.

Aldeghi, Matteo, Shipra Malhotra, David L. Selwood, and Ah Wing Edith Chan

2014 "Two- and Three-dimensional Rings in Drugs", *Chem. Biol. Drug Des.*, 83, 4, pp. 450-461, DOI: 10.1111/cbdd.12260.

Alibay, Irfan, Aniket Magarkar, Daniel Seeliger, and Philip C. Biggin

2022 "Evaluating the use of absolute binding free energy in the fragment optimization process." *ChemRxiv*, DOI: 10.26434/chemrxiv-2022-cw2kq-v3.

Allen, William J. and Robert C. Rizzo

2014 "Implementation of the Hungarian Algorithm to Account for Ligand Symmetry and Similarity in Structure-Based Design", *J. Chem. Inf. Model.*, 54, 2, pp. 518-529, DOI: 10.1021/ci400534h.

Altae-Tran, Han, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande

2017 "Low data drug discovery with one-shot learning", *ACS Cent. Sci.*, 3, 4, pp. 283-293, DOI: 10.1021/acscentsci.6b00367.

Altschul, Stephen F., Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman

1990 "Basic local alignment search tool", *J. Mol. Biol.*, 215, 3, pp. 403-410, DOI: 10.1016/s0022-2836(05)80360-2.

Amaro, Rommie E., Riccardo Baron, and J. Andrew McCammon

2008 "An improved relaxed complex scheme for receptor flexibility in computer-aided drug design", *J. Comput. Aided Mol. Des.*, 22, 9, pp. 693-705, DOI: 10.1007/s10822-007-9159-2.

Amaro, Rommie E., Jerome Baudry, John Chodera, Özlem Demir, J. Andrew McCammon, Yinglong Miao, and Jeremy C. Smith

2018 "Ensemble Docking in Drug Discovery", *Biophys. J.*, 114, 10, pp. 2271-2278, DOI: 10.1016/j.bpj.2018.02.038.

Anand, Ashish, Ganesan Pugalenthi, Gary B. Fogel, and P. N. Suganthan

2010 "An approach for classification of highly imbalanced data using weighting and undersampling", *Amino Acids*, 39, 5, pp. 1385-1391, DOI: 10.1007/s00726-010-0595-2.

Anderson, Amy C, Robert H O'Neil, Toral S Surti, and Robert M Stroud

2001 "Approaches to solving the rigid receptor problem by identifying a minimal set of flexible residues during ligand docking", *Chem. Biol.*, 8, 5, pp. 445-457, DOI: 10.1016/S1074-5521(01)00023-0.

Bajorath, Jürgen

2002 "Integration of virtual and high-throughput screening", *Nat. Rev. Drug Discov.*, 1, 11, pp. 882-894, DOI: 10.1038/nrd941.

Bajusz, Dávid, Anita Rácz, and Károly Héberger

2015 "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?", *J. Cheminf.*, 7, 1, pp. 1-13, DOI: 10.1186/s13321-015-0069-3.

2019 "Comparison of Data Fusion Methods as Consensus Scores for Ensemble Docking", *Molecules*, 24, 15, p. 2690, DOI: 10.3390/molecules24152690.

Bakan, A., A. Dutta, W. Mao, Y. Liu, C. Chennubhotla, T. R. Lezon, and I. Bahar

2014 "Evol and ProDy for bridging protein sequence evolution and structural dynamics", *Method. Biochem. Anal.*, 30, 18, pp. 2681-2683, DOI: 10.1093/bioinformatics/btu336.

Bakan, A., L. M. Meireles, and I. Bahar

2011 "ProDy: Protein Dynamics Inferred from Theory and Experiments", *Method. Biochem. Anal.*, 27, 11, pp. 1575-1577, DOI: 10.1093/bioinformatics/btr168.

Baldi, Pierre and Kurt Hornik

1989 "Neural networks and principal component analysis: Learning from examples without local minima", *Neural Networks*, 2, 1, pp. 53-58, DOI: 10.1016/0893-6080(89)90014-2.

Ballester, Pedro J. and John B. O. Mitchell

2010a "A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking", *Method. Biochem. Anal.*, 26, 9, pp. 1169-1175, DOI: 10.1093/bioinformatics/btq112.

Ballester, Pedro J. and John B.O. Mitchell

2010b "A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking", *Bioinformatics*, 26, 9, pp. 1169-1175, DOI: 10.1093/bioinformatics/btq112.

Bartók, Albert P. and Gábor Csányi

2015 "Gaussian approximation potentials: A brief tutorial introduction", *Int. J. Quantum Chem.*, 115, 16, pp. 1051-1057, DOI: 10.1002/qua.24927.

Bartók, Albert P., Sandip De, Carl Poelking, Noam Bernstein, James R. Kermode, Gábor Csányi, and Michele Ceriotti

2017 "Machine learning unifies the modeling of materials and molecules", *Sci. Adv.*, 3, 12, e1701816, DOI: 10.1126/sciadv.1701816.

Bartók, Albert P., Risi Kondor, and Gábor Csányi

2013 "On representing chemical environments", *Phys. Rev. B*, 87, 18, p. 184115, DOI: 10.1103/PhysRevB.87.184115.

Bartók, Albert P., Mike C. Payne, Risi Kondor, and Gábor Csányi

2010 "Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons", *Phys. Rev. Lett.*, 104, 13, p. 136403, DOI: 10.1103/physrevlett.104.136403.

Baskin, Igor I

2020 "The power of deep learning to ligand-based novel drug discovery", *Expert Opin. Drug Discovery*, 15, 7, pp. 755-764, DOI: 10.1080/17460441.2020.1745183.

Beal, Jacob, Traci Haddock-Angelli, Markus Gershater, Kim de Mora, Meagan Lizarazo, Jim Hollenhorst, Randy Rettberg, and iGEM Interlab Study Contributors

2016 "Reproducibility of Fluorescent Expression from Engineered Biological Constructs in E. coli", *PLoS ONE*, 6, 11, DOI: 10.1371/journal.pone.0150182.

Behler, Jörg

2011 "Neural network potential-energy surfaces in chemistry: A tool for large-scale simulations", *Phys. Chem. Chem. Phys.*, 13, 40, p. 17930, DOI: 10.1039/c1cp21668f.

Behler, Jörg and Michele Parrinello

2007 "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces", *Phys. Rev. Lett.*, 98, 14, p. 146401, DOI: 10.1103/physrevlett.98.146401.

Bell, Eric W. and Yang Zhang

2019 "DockRMSD: An open-source tool for atom mapping and RMSD calculation of symmetric molecules through graph isomorphism", *J. Cheminf.*, 11, 1, DOI: 10.1186/s13321-019-0362-7.

Bemis, Guy W. and Mark A. Murcko
1996 "The Properties of Known Drugs. 1. Molecular Frameworks", *J. Med. Chem.*, 39, 15, pp. 2887-2893, DOI: 10.1021/jm9602928.

Bhatt, S., R. Fujimoto, A. Ogielski, and K. Perumalla
1998 "Parallel simulation techniques for large-scale networks", *IEEE Commun. Mag.*, 36, 8, pp. 42-47, DOI: 10.1109/35.707816.

Bickerton, G. Richard, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins
2012 "Quantifying the chemical beauty of drugs", *Nat. Chem.*, 4, 2, pp. 90-98, DOI: 10.1038/nchem.1243.

Bieniek, Mateusz, Ben Cree, Rachael Pirie, Joshua Horton, Natalie Tatum, and Daniel Cole
2022 "An Open-Source Molecular Builder and Free Energy Preparation Workflow", *ChemRxiv*, DOI: 10.26434/chemrxiv-2022-hr5q4.

Bilodeau, Camille, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F. Jensen
2022 "Generative models for molecular discovery: Recent advances and challenges", *WIREs Comput. Mol. Sci.*, e1608, DOI: 10.1002/wcms.1608.

Blundell, Tom L and S Patel
2004 "High-throughput X-ray crystallography for drug discovery", *Curr. Opin. Pharmacol.*, 4, 5, pp. 490-496, DOI: 10.1016/j.coph.2004.04.007.

Böhm, Hans-Joachim
1994 "The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure", *J. Comput. Aided Mol. Des.*, 8, 3, pp. 243-256, DOI: 10.1007/bf00126743.

Böhm, Hans-Joachim, Alexander Flohr, and Martin Stahl
2004 "Scaffold hopping", *Drug Discovery Today: Technol.*, 1, 3, pp. 217-224, DOI: 10.1016/j.ddtec.2004.10.009.

Bourlard, H. and Y. Kamp
1988 "Auto-association by multilayer perceptrons and singular value decomposition", *Biol. Cybern.*, 59, 4-5, pp. 291-294, DOI: 10.1007/bf00332918.

Boyles, Fergus, Charlotte M Deane, and Garrett M Morris
2019 "Learning from the ligand: Using ligand-based features to improve binding affinity prediction", *Method. Biochem. Anal.*, 36, 3, pp. 758-764, DOI: 10.1093/bioinformatics/btz665.

Brenk, Ruth, Alessandro Schipani, Daniel James, Agata Krasowski, Ian Hugh Gilbert, Julie Frearson, and Paul Graham Wyatt
2008 "Lessons Learnt from Assembling Screening Libraries for Drug Discovery for Neglected Diseases", *ChemMedChem*, 3, 3, pp. 435-444, DOI: 10.1002/cmdc.200700139.

Brown, Dean G. and Heike J. Wobst
2021 "A Decade of FDA-Approved Drugs (2010–2019): Trends and Future Directions", *J. Med. Chem.*, 64, 5, pp. 2312-2338, DOI: 10.1021/acs.jmedchem.0c01516.

Brown, Nathan, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher
2019 "GuacaMol: benchmarking models for de novo molecular design", *J. Chem. Inf. Model.*, 59, 3, pp. 1096-1108, DOI: 10.1021/acs.jcim.8b00839.

Broyden, C. G.
1970 "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations", *IMA J. Appl. Math.*, 6, 1, pp. 76-90, DOI: 10.1093/imamat/6.1.76.

Bursulaya, Badry D, Maxim Totrov, Ruben Abagyan, and Charles L Brooks
2003 "Comparative study of several algorithms for flexible ligand docking", *J. Comput. Aided Mol. Des*, 17, 11, pp. 755-763.

Cai, Chenjing, Shiwei Wang, Youjun Xu, Weilin Zhang, Ke Tang, Qi Ouyang, Luhua Lai, and Jianfeng Pei
2020 "Transfer learning for drug discovery", *J. Med. Chem.*, 63, 16, pp. 8683-8694, DOI: 10.1021/acs.jmedchem.9b02147.

Cang, Zixuan, Lin Mu, and Guo-Wei Wei
2018 "Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening", *PLoS Comput. Biol.*, 14, 1, e1005929, DOI: 10.1371/journal.pcbi.1005929.

Cao, Hong, Xiao-Li Li, David Yew-Kwong Woon, and See-Kiong Ng
2013 "Integrated Oversampling for Imbalanced Time Series Classification", *IEEE Trans. Knowl. Data Eng.*, 25, 12, pp. 2809-2822, DOI: 10.1109/tkde.2013.37.

Capecchi, Alice, Daniel Probst, and Jean-Louis Reymond
2020 "One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome", *J. Cheminf.*, 12, 1, pp. 1-15, DOI: 10.1186/s13321-020-00445-4.

Chang, Chia-en A., Wei Chen, and Michael K. Gilson
2007 "Ligand configurational entropy and protein binding", *Proc. Natl. Acad. Sci.*, 104, 5, pp. 1534-1539, DOI: 10.1073/pnas.0610494104.

Chen, Hongming, Thierry Kogej, and Ola Engkvist
2018 "Cheminformatics in drug discovery, an industrial perspective", *Mol. Inf.*, 37, 9-10, p. 1800041, DOI: 10.1002/minf.201800041.

Chen, Lieyang, Anthony Cruz, Steven Ramsey, Callum J. Dickson, Jose S. Duca, Viktor Hornak, David R. Koes, and Tom Kurtzman
2019 "Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening", *PLoS ONE*, 14, 8, e0220113, DOI: 10.1371/journal.pone.0220113.

Chen, X., Y. Lin, M. Liu, and M. K. Gilson

2002 "The Binding Database: Data management and interface design", *Method. Biochem. Anal.*, 18, 1, pp. 130-139, DOI: 10.1093/bioinformatics/18.1.130.

Chen, Xi, Ming Liu, and Michael Gilson

2001 "BindingDB: A Web-Accessible Molecular Recognition Database", *Comb. Chem. High Throughput Screening*, 4, 8, pp. 719-725, DOI: 10.2174/1386207013330670.

Chen, Yang and Gérard Medioni

1992 "Object modelling by registration of multiple range images", *Image Vision Comput.*, 10, 3, pp. 145-155, DOI: 10.1016/0262-8856(92)90066-c.

Cheng, Tiejun, Xun Li, Yan Li, Zhihai Liu, and Renxiao Wang

2009 "Comparative Assessment of Scoring Functions on a Diverse Test Set", *J. Chem. Inf. Model.*, 49, 4, pp. 1079-1093, DOI: 10.1021/ci9000053.

Cheng, Yung-Chi and William H Prusoff

1973 "Relationship between the inhibition constant (KI) and the concentration of inhibitor which causes 50 per cent inhibition (I50) of an enzymatic reaction", *Biochem. Pharmacol.*, 22, 23, pp. 3099-3108, DOI: 10.1016/0006-2952(73)90196-2.

Cieplinski, Tobiasz, Tomasz Danel, Sabina Podlewska, and Stanislaw Jastrzebski

2020 "We should at least be able to design molecules that dock well", *arXiv*, DOI: 10.48550/arXiv.2006.16955.

Cohen, Philip, Darren Cross, and Pasi A. Jänne

2021 "Kinase drug discovery 20 years after imatinib: Progress and future directions", *Nat. Rev. Drug Discov.*, 20, 7, pp. 551-569, DOI: 10.1038/s41573-021-00195-4.

Cole, Daniel J., Letif Mones, and Gábor Csányi

2020 "A machine learning based intramolecular potential for a flexible organic molecule", *Faraday Discuss.*, 224, pp. 247-264, DOI: 10.1039/d0fd00028k.

Congreve, Miles, Robin Carr, Chris Murray, and Harren Jhoti

2003 "A 'rule of three' for fragment-based lead discovery?", *Drug Discovery Today*, 8, 19, pp. 876-877, DOI: 10.1038/nrd3926-c1.

Cordella, Luigi P, Pasquale Foggia, Carlo Sansone, and Mario Vento

2004 "A (sub) graph isomorphism algorithm for matching large graphs", *IEEE T. Pattern Anal.*, 26, 10, pp. 1367-1372, DOI: 10.1109/TPAMI.2004.75.

Croston, Glenn E

2017 "The utility of target-based discovery", *Expert Opin. Drug Discovery*, 12, 5, pp. 427-429, DOI: 10.1080/17460441.2017.1308351.

Darby, John F., Adam P. Hopkins, Seishi Shimizu, Shirley M. Roberts, James A. Brannigan, Johan P. Turkenburg, Gavin H. Thomas, Roderick E. Hubbard, and Marcus Fischer
2019 "Water Networks Can Determine the Affinity of Ligand Binding to Proteins", *J. Am. Chem. Soc.*, 141, 40, pp. 15818-15826, DOI: 10.1021/jacs.9b06275.

De Cao, Nicola and Thomas Kipf
2018 "MolGAN: An implicit generative model for small molecular graphs", *arXiv*, DOI: 10.48550/arXiv.1805.11973.

Degiacomi, Matteo T
2019 "Coupling molecular dynamics and deep learning to mine protein conformational space", *Structure*, 27, 6, pp. 1034-1040, DOI: 10.1016/j.str.2019.03.018.

Derringer, George and Ronald Suich
1980 "Simultaneous Optimization of Several Response Variables", *J. Qual. Technol.*, 12, 4, pp. 214-219, DOI: 10.1080/00224065.1980.11980968.

Dhalluin, Christophe, Justin E. Carlson, Lei Zeng, Cheng He, Aneel K. Aggarwal, Ming-Ming Zhou, and Ming-Ming Zhou
1999 "Structure and ligand of a histone acetyltransferase bromodomain", *Nature*, 399, 6735, pp. 491-496, DOI: 10.1038/20974.

Dickson, Michael and Jean Paul Gagnon
2004 "Key factors in the rising cost of new drug discovery and development", *Nat. Rev. Drug Discov.*, 3, 5, pp. 417-429, DOI: 10.1038/nrd1382.

DiMasi, Joseph A., Henry G. Grabowski, and Ronald W. Hansen
2016 "Innovation in the pharmaceutical industry: New estimates of R&D costs", *J. Health Econ.*, 47, pp. 20-33, DOI: 10.1016/j.jhealeco.2016.01.012.

Doersch, Carl
2021 "Tutorial on Variational Autoencoders", *arXiv*, DOI: 10.48550/arXiv.1606.05908.

Douguet, Dominique and Frédéric Payan
2020 "SENSAAS: Shape-based Alignment by Registration of Colored Point-based Surfaces", *Mol. Inform.*, 39, 8, p. 2000081, DOI: 10.1002/minf.202000081.

Dral, Pavlo O
2020 "Quantum chemistry in the age of machine learning", *J. Phys. Chem. Lett.*, 11, 6, pp. 2336-2347, DOI: 10.1021/acs.jpclett.9b03664.

Drew, Kurt L M, Hakim Baiman, Prashanna Khwaounjoo, Bo Yu, and Jóhannes Reynisson
2011 "Size estimation of chemical space: How big is it?", *J. Pharm. Pharmacol.*, 64, 4, pp. 490-495, DOI: 10.1111/j.2042-7158.2011.01424.x.

Drews, Jürgen

2000 "Drug Discovery: A Historical Perspective", *Science*, 287, 5460, pp. 1960-1964, DOI: 10.1126/science.287.5460.1960.

Duan, Yingchao, Yuanyuan Guan, Wenping Qin, Xiaoyu Zhai, Bin Yu, and Hongmin Liu

2018 "Targeting BRD4 for cancer therapy: Inhibitors and degraders", *Med. Chem. Comm.*, 9, 11, pp. 1779-1802, DOI: 10.1039/c8md00198g.

Dumoulin, Vincent and Francesco Visin

2016 "A guide to convolution arithmetic for deep learning", *arXiv*, DOI: 10.48550/arXiv.1603.07285.

Dunbar, James B., Richard D. Smith, Kelly L. Damm-Ganamet, Aqeel Ahmed, Emilio Xavier Esposito, James Delproposto, Krishnapriya Chinnaswamy, You-Na Kang, Ginger Kubish, Jason E. Gestwicki, Jeanne A. Stuckey, and Heather A. Carlson

2013 "CSAR Data Set Release 2012: Ligands, Affinities, Complexes, and Docking Decoys", *J. Chem. Inf. Model.*, 53, 8, pp. 1842-1852, DOI: 10.1021/ci4000486.

Dunbrack Jr, Roland L

2002 "Rotamer libraries in the 21st century", *Curr. Opin. Struct. Biol.*, 12, 4, pp. 431-440, DOI: 10.1016/S0959-440X(02)00344-5.

Durant, Joseph L., Burton A. Leland, Douglas R. Henry, and James G. Nourse

2002 "Reoptimization of MDL Keys for Use in Drug Discovery", *J. Chem. Inf. Comp. Sci.*, 42, 6, pp. 1273-1280, DOI: 10.1021/ci010132r.

Durrant, Jacob D and J Andrew McCammon

2010 "Computer-aided drug-discovery techniques that account for receptor flexibility", *Curr. Opin. Pharmacol.*, 10, 6, pp. 770-774, DOI: 10.1016/j.coph.2010.09.001.

2011a "Molecular dynamics simulations and drug discovery", *BMC Biol.*, 9, 71, pp. 1-9, DOI: 10.1186/1741-7007-9-71.

2011b "NNScore 2.0: A Neural-Network Receptor–Ligand Scoring Function", *J. Chem. Inf. Model.*, 51, 11, pp. 2897-2903, DOI: 10.1021/ci2003889.

Eisenhaber, Frank and Patrick Argos

1993 "Improved strategy in analytic surface calculation for molecular systems: Handling of singularities and computational efficiency", *J. Comput. Chem.*, 14, 11, pp. 1272-1280, DOI: 10.1002/jcc.540141103.

Eisenhaber, Frank, Philip Lijnzaad, Patrick Argos, Chris Sander, and Michael Scharf

1995 "The double cubic lattice method: Efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies", *J. Comput. Chem.*, 16, 3, pp. 273-284, DOI: 10.1002/jcc.540160303.

Eldridge, Matthew D., Christopher W. Murray, Timothy R. Auton, Gaia V. Paolini, and Roger P. Mee
1997 "Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes", *J. Comput. Aided Mol. Des.*, 11, 5, pp. 425-445, DOI: 10.1023/a:1007996124545.

Ericksen, Spencer S., Haozhen Wu, Huikun Zhang, Lauren A. Michael, Michael A. Newton, F. Michael Hoffmann, and Scott A. Wildman
2017 "Machine Learning Consensus Scoring Improves Performance Across Targets in Structure-Based Virtual Screening", *J. Chem. Inf. Model.*, 57, 7, pp. 1579-1590, DOI: 10.1021/acs.jcim.7b00153.

Erlanson, Daniel A, Stephen W Fesik, Roderick E Hubbard, Wolfgang Jahnke, and Harren Jhoti
2016 "Twenty years on: the impact of fragments on drug discovery", *Nat. Rev. Drug Discovery*, 15, 9, pp. 605-619, DOI: 10.1038/nrd.2016.109.

Ertl, Peter
2021 "Magic Rings: Navigation in the Ring Chemical Space Guided by the Bioactive Rings", *J. Chem. Inf. Model.*, 62, 9, pp. 2164-2170, DOI: 10.1021/acs.jcim.1c00761.

Ertl, Peter and Ansgar Schuffenhauer
2009 "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions", *J. Cheminf.*, 1, 1, p. 8, DOI: 10.1186/1758-2946-1-8.

Esposito, Carmen, Gregory A. Landrum, Nadine Schneider, Nikolaus Stiefl, and Sereina Riniker
2021 "GHOST: Adjusting the Decision Threshold to Handle Imbalanced Data in Machine Learning", *J. Chem. Inf. Model.*, 61, 6, pp. 2623-2640, DOI: 10.1021/acs.jcim.1c00160.

Feher, Victoria A, Jacob D Durrant, Adam T Van Wart, and Rommie E Amaro
2014 "Computational approaches to mapping allosteric pathways", *Curr. Opin. Struct. Biol.*, 25, pp. 98-103, DOI: 10.1016/j.sbi.2014.02.004.

Feixas, Ferran, Steffen Lindert, William Sinko, and J. Andrew McCammon
2014 "Exploring the role of receptor flexibility in structure-based drug discovery", *Biophys. Chem.*, 186, pp. 31-45, DOI: 10.1016/j.bpc.2013.10.007.

Filippakopoulos, Panagis, Sarah Picaud, Oleg Fedorov, Marco Keller, Matthias Wrobel, Olaf Morgenstern, Franz Bracher, and Stefan Knapp
2012 "Benzodiazepines and benzotriazepines as protein interaction inhibitors targeting bromodomains of the BET family", *Bioorg. Med. Chem.*, 20, 6, pp. 1878-1886, DOI: 10.1016/j.bmc.2011.10.080.

Filippakopoulos, Panagis, Jun Qi, Sarah Picaud, Yao Shen, William B. Smith, Oleg Fedorov, Elizabeth M. Morse, Tracey Keates, Tyler T. Hickman, Ildiko Felletar, Martin Philpott, Shonagh Munro, Michael R. McKeown, Yuchuan Wang, Amanda L. Christie, Nathan West, Michael J. Cameron, Brian Schwartz, Tom D. Heightman, Nicholas La Thangue, Christopher A. French, Olaf Wiest, Andrew L. Kung, Stefan Knapp, and James E. Bradner

2010 "Selective inhibition of BET bromodomains", *Nature*, 468, 7327, pp. 1067-1073, DOI: 10.1038/nature09504.

Fischler, Martin A. and Robert C. Bolles

1981 "Random sample consensus", *Commun. ACM*, 24, 6, pp. 381-395, DOI: 10.1145/358669.358692.

Fish, Paul V., Panagis Filippakopoulos, Gerwyn Bish, Paul E. Brennan, Mark E. Bunnage, Andrew S. Cook, Oleg Federov, Brian S. Gerstenberger, Hannah Jones, Stefan Knapp, Brian Marsden, Karl Nocka, Dafydd R. Owen, Martin Philpott, Sarah Picaud, Michael J. Primiano, Michael J. Ralph, Nunzio Sciammetta, and John D. Trzupek

2012 "Identification of a Chemical Probe for Bromo and Extra C-Terminal Bromodomain Inhibition through Optimization of a Fragment-Derived Hit", *J. Med. Chem.*, 55, 22, pp. 9831-9837, DOI: 10.1021/jm3010515.

Fletcher, R.

1970 "A new approach to variable metric algorithms", *Comput. J.*, 13, 3, pp. 317-322, DOI: 10.1093/comjnl/13.3.317.

Francoeur, Paul G., Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B. Iovanisci, Ian Snyder, and David R. Koes

2020 "Three-Dimensional Convolutional Neural Networks and a Cross-Docked Data Set for Structure-Based Drug Design", *J. Chem. Inf. Model.*, 60, 9, pp. 4200-4215, DOI: 10.1021/acs.jcim.0c00411.

Friesner, Richard A., Jay L. Banks, Robert B. Murphy, Thomas A. Halgren, Jasna J. Klicic, Daniel T. Mainz, Matthew P. Repasky, Eric H. Knoll, Mee Shelley, Jason K. Perry, David E. Shaw, Perry Francis, and Peter S. Shenkin

2004 "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy", *J. Med. Chem.*, 47, 7, pp. 1739-1749, DOI: 10.1021/jm0306430.

Friesner, Richard A., Robert B. Murphy, Matthew P. Repasky, Leah L. Frye, Jeremy R. Greenwood, Thomas A. Halgren, Paul C. Sanschagrin, and Daniel T. Mainz

2006 "Extra Precision Glide: Docking and Scoring Incorporating a Model of Hydrophobic Enclosure for Protein-Ligand Complexes", *J. Med. Chem.*, 49, 21, pp. 6177-6196, DOI: 10.1021/jm051256o.

Fukushima, Kunihiko

1980 "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biol. Cybern.*, 36, 4, pp. 193-202, DOI: 10.1007/bf00344251.

Gao, Wenhao and Connor W. Coley
2020 "The Synthesizability of Molecules Proposed by Generative Models", *J. Chem. Inf. Model.*, 60, 12, pp. 5714-5723, DOI: 10.1021/acs.jcim.0c00174.

Gao, Xiang, Farhad Ramezanghorbani, Olexandr Isayev, Justin S. Smith, and Adrian E. Roitberg
2020 "TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials", *J. Chem. Inf. Model.*, 60, 7, pp. 3408-3415, DOI: 10.1021/acs.jcim.0c00451.

Garcia, V., J.S. Sánchez, and R.A. Mollineda
2012 "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance", *Knowl.-Based. Syst.*, 25, 1, pp. 13-21, DOI: 10.1016/j.knosys.2011.06.013.

Gaudelet, Thomas, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B R Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L Blundell, Michael M Bronstein, and Jake P Taylor-King
2021 "Utilizing graph machine learning within drug discovery and development", *Brief. Bioinform.*, 6, DOI: 10.1093/bib/bbab159.

Gawehn, Erik, Jan A Hiss, and Gisbert Schneider
2016 "Deep learning in drug discovery", *Mol. Inf.*, 35, 1, pp. 3-14, DOI: 10.1002/minf.201501008.

Gebauer, Niklas W. A., Michael Gastegger, Stefaan S. P. Hessmann, Klaus-Robert Müller, and Kristof T. Schütt
2022 "Inverse design of 3D molecular structures with conditional generative neural networks", *Nat. Commun.*, 13, 1, DOI: 10.1038/s41467-022-28526-y.

Gehling, Victor S., Michael C. Hewitt, Rishi G. Vaswani, Yves Leblanc, Alexandre Côté, Christopher G. Nasveschuk, Alexander M. Taylor, Jean-Christophe Harmange, James E. Audia, Eneida Pardo, Shivangi Joshi, Peter Sandy, Jennifer A. Mertz, Robert J. Sims, Louise Bergeron, Barbara M. Bryant, Steve Bellon, Florence Poy, Hariharan Jayaram, Ravichandran Sankaranarayanan, Sreegouri Yellapantula, Nandana Bangalore Srinivasamurthy, Swarnakumari Birudukota, and Brian K. Albrecht
2013 "Discovery, Design, and Optimization of Isoxazole Azepine BET Inhibitors", *ACS Med. Chem. Lett.*, 4, 9, pp. 835-840, DOI: 10.1021/ml4001485.

Genheden, Samuel and Ulf Ryde
2015 "The MM/PBSA and MM/GBSA methods to estimate ligand-binding affinities", *Expert Opin. Drug Discovery*, 10, 5, pp. 449-461, DOI: 10.1517/17460441.2015.1032936.

Ghose, Arup K. and Gordon M. Crippen
1986 "Atomic Physicochemical Parameters for Three Dimensional Structure-Directed Quantitative Structure-Activity Relationships I. Partition Coefficients as a Measure of Hydrophobicity", *J. Comput. Chem.*, 7, 4, pp. 565-577, DOI: 10.1002/jcc.540070419.

Gilson, Michael K. and Barry H. Honig
1986 "The dielectric constant of a folded protein", *Biopolymers*, 25, 11, pp. 2097-2119, DOI: 10.1002/bip.360251106.

Gilson, Michael K., Tiqing Liu, Michael Baitaluk, George Nicola, Linda Hwang, and Jenny Chong
2015 "BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology", *Nucleic Acids Res.*, 44, D1, pp. D1045-D1053, DOI: 10.1093/nar/gkv1072.

Goddard, Thomas D., Conrad C. Huang, Elaine C. Meng, Eric F. Pettersen, Gregory S. Couch, John H. Morris, and Thomas E. Ferrin
2017 "UCSF ChimeraX: Meeting modern challenges in visualization and analysis", *Protein Sci.*, 27, 1, pp. 14-25, DOI: 10.1002/pro.3235.

Goldfarb, Donald
1970 "A family of variable-metric methods derived by variational means", *Math. Comput.*, 24, 109, pp. 23-26, DOI: 10.1090/s0025-5718-1970-0258249-6.

Gómez-Bombarelli, Rafael, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamin Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik
2018 "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules", *ACS Cent. Sci.*, 4, 2, pp. 268-276, DOI: 10.1021/acscentsci.7b00572.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio
2014 "Generative Adversarial Networks", DOI: 10.48550/arxiv.1406.2661.

Grant, Barry J, Alemayehu A Gorfe, and J Andrew McCammon
2010 "Large conformational changes in proteins: Signaling and other functions", *Curr. Opin. Struc. Biol.*, 20, 2, pp. 142-147, DOI: 10.1016/j.sbi.2009.12.004.

Grant, J. A., M. A. Gallardo, and B. T. Pickup
1996 "A fast method of molecular shape comparison: A simple application of a Gaussian description of molecular shape", *J. Comput. Chem.*, 17, 14, pp. 1653-1666, DOI: 10.1002/(sici)1096-987x(19961115)17:14<1653::aid-jcc7>3.0.co;2-k.

Grosjean, Harold, Mehtap Işık, Anthony Aimon, David Mobley, John Chodera, Frank von Delft, and Philip C. Biggin
2022 "SAMPL7 protein-ligand challenge: A community-wide evaluation of computational methods against fragment screening and pose-prediction", *J. Comput.-Aided Mol. Des.*, pp. 1-21, DOI: 10.1007/s10822-022-00452-7.

Guedes, Isabella A., André M. S. Barreto, Diogo Marinho, Eduardo Krempser, Mélaine A. Kuenemann, Olivier Sperandio, Laurent E. Dardenne, and Maria A. Miteva
2021 "New machine learning and physics-based scoring functions for drug discovery", *Sci. Rep.*, 11, 1, pp. 1-19, DOI: 10.1038/s41598-021-82410-1.

Guedes, Isabella A., Felipe S. S. Pereira, and Laurent E. Dardenne
2018 "Empirical Scoring Functions for Structure-Based Virtual Screening: Applications, Critical Aspects, and Challenges", *Front. Pharmacol.*, 9, p. 1089, DOI: 10.3389/fphar.2018.01089.

Gupta, Anvita, Alex T Müller, Berend JH Huisman, Jens A Fuchs, Petra Schneider, and Gisbert Schneider
2018 "Generative recurrent networks for de novo drug design", *Mol. Inf.*, 37, 1-2, p. 1700111, DOI: 10.1002/minf.201700111.

Guvench, Olgun and Alexander D. MacKerell
2009 "Computational Fragment-Based Binding Site Identification by Ligand Competitive Saturation", *PLoS Comput. Biol.*, 5, 7, e1000435, DOI: 10.1371/journal.pcbi.1000435.

Hadfield, Thomas E. and Charlotte M. Deane
2022 "AI in 3D compound design", *Curr. Opin. Struc. Biol.*, 73, p. 102326, DOI: 10.1016/j.sbi.2021.102326.

Hadfield, Thomas E., Fergus Imrie, Andy Merritt, Kristian Birchall, and Charlotte M. Deane
2022 "Incorporating Target-Specific Pharmacophoric Information into Deep Generative Models for Fragment Elaboration", *J. Chem. Inf. Model.*, 62, 10, pp. 2280-2292, DOI: 10.1021/acs.jcim.1c01311.

Hahn, David F, Christopher I Bayly, Hannah E Bruce Macdonald, John D Chodera, Antonia SJS Mey, David L Mobley, Laura Perez Benito, Christina EM Schindler, Gary Tresadern, and Gregory L Warren
2021 "Best practices for constructing, preparing, and evaluating protein-ligand binding affinity benchmarks", *arXiv*, DOI: 10.48550/arXiv.2105.06222.

Halgren, Thomas A., Robert B. Murphy, Richard A. Friesner, Hege S. Beard, Leah L. Frye, W. Thomas Pollard, and Jay L. Banks
2004 "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening", *J. Med. Chem.*, 47, 7, pp. 1750-1759, DOI: 10.1021/jm030644s.

Harrington, E. C. Jr
1965 "The desirability function", *Ind. Qual. Control.*, 21, pp. 494-498.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Rio, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant
2020 "Array programming with NumPy", *Nature*, 585, 7825, pp. 357-362, DOI: 10.1038/s41586-020-2649-2.

Hassan-Harrirou, Hussein, Ce Zhang, and Thomas Lemmin
2020 "RosENet: Improving Binding Affinity Prediction by Leveraging Molecular Mechanics Energies with an Ensemble of 3D Convolutional Neural Networks", *J. Chem. Inf. Model.*, 60, 6, pp. 2791-2802, DOI: 10.1021/acs.jcim.0c00075.

Heinzelmann, Germano, Niel M. Henriksen, and Michael K. Gilson
2017 "Attach-Pull-Release Calculations of Ligand Binding and Conformational Changes on the First BRD4 Bromodomain", *J. Chem. Theory Comput.*, 13, 7, pp. 3260-3275, DOI: 10.1021/acs.jctc.7b00275.

Hingerty, B. E., R. H. Ritchie, T. L. Ferrell, and J. E. Turner
1985 "Dielectric effects in biopolymers: The theory of ionic saturation revisited", *Biopolymers*, 24, 3, pp. 427-439, DOI: 10.1002/bip.360240302.

Hinton, Geoffrey, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury
2012 "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups", *IEEE Signal Process Mag.*, 29, 6, pp. 82-97, DOI: 10.1109/msp.2012.2205597.

Hopkins, Andrew L, György M Keserü, Paul D Leeson, David C Rees, and Charles H Reynolds
2014 "The role of ligand efficiency metrics in drug discovery", *Nat. Rev. Drug Discovery*, 13, 2, pp. 105-121, DOI: 10.1038/nrd4163.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White
1989 "Multilayer feedforward networks are universal approximators", *Neural Networks*, 2, 5, pp. 359-366, DOI: 10.1016/0893-6080(89)90020-8.

Hu, Liegi, Mark L. Benson, Richard D. Smith, Michael G. Lerner, and Heather A. Carlson
2005 "Binding MOAD (Mother Of All Databases)", *Proteins: Struct., Funct., Bioinf.*, 60, 3, pp. 333-340, DOI: 10.1002/prot.20512.

Huang, David Z, J. Christian Baber, and Sogole Sami Bahmanyar
2021 "The challenges of generalizability in artificial intelligence for ADME/Tox endpoint and activity prediction", *Expert Opin. Drug Discovery*, 16, 9, pp. 1045-1056, DOI: 10.1080/17460441.2021.1901685.

Huang, Gao, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger
2019 "Convolutional Networks with Dense Connectivity", *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1-1, DOI: 10.1109/tpami.2019.2918284.

Huang, Niu, Brian K. Shoichet, and John J. Irwin
2006 "Benchmarking Sets for Molecular Docking", *J. Med. Chem.*, 49, 23, pp. 6789-6801, DOI: 10.1021/jm0608356.

Huang, Sheng-You, Sam Z. Grinter, and Xiaoqin Zou
2010 "Scoring functions and their evaluation methods for protein–ligand docking: Recent advances and future directions", *Phys. Chem. Chem. Phys.*, 12, 40, p. 12899, DOI: 10.1039/c0cp00151a.

Huang, Sheng-You and Xiaoqin Zou
2010 "Inclusion of Solvation and Entropy in the Knowledge-Based Scoring Function for Protein-Ligand Interactions", *J. Chem. Inf. Model.*, 50, 2, pp. 262-273, DOI: 10.1021/ci9002987.

Hubel, D. H.
1959 "Single unit activity in striate cortex of unrestrained cats", *J. Physiol.*, 147, 2, pp. 226-238, DOI: 10.1113/jphysiol.1959.sp006238.

Hubel, D. H. and T. N. Wiesel
1959 "Receptive fields of single neurones in the cat's striate cortex", *J. Physiol.*, 148, 3, pp. 574-591, DOI: 10.1113/jphysiol.1959.sp006308.

Huey, Ruth, Garrett M. Morris, Arthur J. Olson, and David S. Goodsell
2007 "A semiempirical free energy force field with charge-based desolvation", *J. Comput. Chem.*, 28, 6, pp. 1145-1152, DOI: 10.1002/jcc.20634.

Hughes, James P, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott
2011 "Principles of early drug discovery", *Br. J. Pharmacol.*, 162, 6, pp. 1239-1249, DOI: 10.1111/j.1476-5381.2010.01127.x.

Hunter, John D.
2007 "Matplotlib: A 2D Graphics Environment", *Comput. Sci. Eng.*, 9, 3, pp. 90-95, DOI: 10.1109/mcse.2007.55.

Imrie, Fergus, Anthony R Bradley, Mihaela van der Schaar, and Charlotte M Deane
2020 "Deep generative models for 3D linker design", *J. Chem. Inf. Model.*, 60, 4, pp. 1983-1995, DOI: 10.1021/acs.jcim.9b01120.

Imrie, Fergus, Anthony R. Bradley, and Charlotte M. Deane
2021 "Generating property-matched decoy molecules using deep learning", *Method. Biochem. Anal.*, 37, 15, pp. 2134-2141, DOI: 10.1093/bioinformatics/btab080.

Imrie, Fergus, Anthony R. Bradley, Mihaela van der Schaar, and Charlotte M. Deane
2018 "Protein Family-Specific Models Using Deep Neural Networks and Transfer Learning Improve Virtual Screening and Highlight the Need for More Data", *J. Chem. Inf. Model.*, 58, 11, pp. 2319-2330, DOI: 10.1021/acs.jcim.8b00350.

Imrie, Fergus, Thomas E. Hadfield, Anthony R. Bradley, and Charlotte M. Deane
2021 "Deep generative design with 3D pharmacophoric constraints", *Chem. Sci.*, 12, 43, pp. 14577-14589, DOI: 10.1039/d1sc02436a.

Jaccard, Paul
1912 "The Distribution Of The Flora In The Alpine Zone", *New Phytol.*, 11, 2, pp. 37-50, DOI: 10.1111/j.1469-8137.1912.tb05611.x.

Jeffrey, Philip D., Alicia A. Russo, Kornelia Polyak, Emma Gibbs, Jerard Hurwitz, Joan Massagué, and Nikola P. Pavletich
1995 "Mechanism of CDK activation revealed by the structure of a cyclinA-CDK2 complex", *Nature*, 376, 6538, pp. 313-320, DOI: 10.1038/376313a0.

Jhoti, Harren, Glyn Williams, David C Rees, and Christopher W Murray
2013 "The 'rule of three' for fragment-based drug discovery: where are we now?", *Nat. Rev. Drug Discovery*, 12, 8, pp. 644-644, DOI: 10.1038/nrd3926-c1.

Jiménez, J, S Doerr, G Martinez-Rosell, A S Rose, and G De Fabritiis
2017 "DeepSite: Protein-binding site predictor using 3D-convolutional neural networks", *Bioinformatics*, 33, 19, pp. 3036-3042, DOI: 10.1093/bioinformatics/btx350.

Jiménez, José, Miha Škalič, Gerard Martinez-Rosell, and Gianni De Fabritiis
2018 "KDEEP: Protein–ligand Absolute Binding Affinity Prediction via 3D-Convolutional Neural Networks", *J. Chem. Inf. Model.*, 58, 2, pp. 287-296, DOI: 10.1021/acs.jcim.7b00650.

Jiménez-Luna, José, Francesca Grisoni, Nils Weskamp, and Gisbert Schneider
2021 "Artificial intelligence in drug discovery: Recent advances and future perspectives", *Expert Opin. Drug Discovery*, 16, 9, pp. 949-959, DOI: 10.1080/17460441.2021.1909567.

Jing, Yankang, Yuemin Bian, Ziheng Hu, Lirong Wang, and Xiang-Qun Sean Xie
2018 "Deep Learning for Drug Design: An AI Paradigm for Drug Discovery in the Big Data Era", *AAPS J*, 20, 3, pp. 1-10, DOI: 10.1208/s12248-018-0210-0.

Johansson, Simon, Amol Thakkar, Thierry Kogej, Esben Bjerrum, Samuel Genheden, Tomas Bastys, Christos Kannas, Alexander Schliep, Hongming Chen, and Ola Engkvist
2019 "AI-assisted synthesis prediction", *Drug Discovery Today: Technol.*, 32, pp. 65-72, DOI: 10.1016/j.ddtec.2020.06.002.

Jones, T. A., J. Y. Zou, S. W. Cowan, and M. Kjeldgaard
1991 "Improved methods for building protein models in electron density maps and the location of errors in these models", *Acta Crystallogr., Sect. A: Found. Crystallogr.*, 47, 2, pp. 110-119, DOI: 10.1107/s0108767390010224.

Kanev, Georgi K, Chris de Graaf, Bart A Westerman, Iwan J P de Esch, and Albert J Kooistra
2020 "KLIFS: An overhaul after the first 5 years of supporting kinase research", *Nucleic Acids Res.*, 49, D1, pp. D562-D569, DOI: 10.1093/nar/gkaa895.

Karlov, Dmitry S, Sergey Sosnin, Maxim V Fedorov, and Petr Popov
2020 "graphDelta: MPNN scoring function for the affinity prediction of protein–ligand complexes", *ACS Omega*, 5, 10, pp. 5150-5159.

Kendall, M. G.
1938 "A New Measure Of Rank Correlation", *Biometrika*, 30, 1-2, pp. 81-93, DOI: 10.1093/biomet/30.1-2.81.

Kim, Sunghwan, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al.
2021 "PubChem in 2021: new data content and improved web interfaces", *Nucleic Acids Res.*, 49, D1, pp. D1388-D1395.

Kingma, Diederik P. and Jimmy Ba
2014 "ADAM: A Method for Stochastic Optimization", *arXiv*, DOI: 10.48550/arXiv.1412.6980.

Kirsch, Philine, Alwin M Hartman, Anna KH Hirsch, and Martin Empting
2019 "Concepts and core principles of fragment-based drug design", *Molecules*, 24, 23, p. 4309, DOI: 10.3390/molecules24234309.

Koes, David Ryan, Matthew P. Baumgartner, and Carlos J. Camacho
2013 "Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise", *J. Chem. Inf. Model.*, 53, 8, pp. 1893-1904, DOI: 10.1021/ci300604z.

Koes, David Ryan and Carlos J Camacho
2014 "Shape-based virtual screening with volumetric aligned molecular shapes", *J. Comput. Chem.*, 35, 25, pp. 1824-1834, DOI: 10.1002/jcc.23690.

Konc, Janez, Matjaž Depolli, Roman Trobec, Kati Rozman, and Dušanka Janežič
2012 "Parallel-ProBiS: Fast parallel algorithm for local structural comparison of protein structures and binding sites", *J. Comput. Chem.*, 33, 27, pp. 2199-2203, DOI: 10.1002/jcc.23048.

Konc, Janez and Dušanka Janezic
2007 "An improved branch and bound algorithm for the maximum clique problem", *MATCH Commun. Math. Comput. Chem.*, 58, 5, pp. 569-590.

Konc, Janez and Dušanka Janežič
2010 "ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment", *Method. Biochem. Anal.*, 26, 9, pp. 1160-1168, DOI: 10.1093/bioinformatics/btq100.

Kooistra, Albert J., Georgi K. Kanev, Oscar P.J. van Linden, Rob Leurs, Iwan J.P. de Esch, and Chris de Graaf
2015 "KLIFS: A structural kinase-ligand interaction database", *Nucleic Acids Res.*, 44, D1, pp. D365-D371, DOI: 10.1093/nar/gkv1082.

Koshland, D. E.
1958 "Application of a Theory of Enzyme Specificity to Protein Synthesis", *Proc. Natl. Acad. Sci.*, 44, 2, pp. 98-104, DOI: 10.1073/pnas.44.2.98.

Köster, Johannes and Sven Rahmann
2012 "Snakemake—a scalable bioinformatics workflow engine", *Bioinformatics*, 28, 19, pp. 2520-2522, DOI: 10.1093/bioinformatics/bts480.

Kouzarides, T.
2000 "Acetylation: A regulatory modification to rival phosphorylation?", *EMBO J.*, 19, 6, pp. 1176-1179, DOI: 10.1093/emboj/19.6.1176.

Kramer, Christian, Tuomo Kalliokoski, Peter Gedeck, and Anna Vulpetti
2012 "The experimental uncertainty of heterogeneous public K i data", *J. Med. Chem.*, 55, 11, pp. 5165-5173.

Krenn, Mario, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik
2020 "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation", *Mach. Learn.: Sci. Technol.*, 1, 4, p. 045024, DOI: 10.1088/2632-2153/aba947.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton
2017 "ImageNet classification with deep convolutional neural networks", *Commun. ACM*, 60, 6, pp. 84-90, DOI: 10.1145/3065386.

Krylov, Anna I., John M. Herbert, Filipp Furche, Martin Head-Gordon, Peter J. Knowles, Roland Lindh, Frederick R. Manby, Peter Pulay, Chris-Kriton Skylaris, and Hans-Joachim Werner
2015 "What Is the Price of Open-Source Software?", *J. Phys. Chem. Lett.*, 6, 14, pp. 2751-2754, DOI: 10.1021/acs.jpclett.5b01258.

Kuhn, H. W.
1955 "The Hungarian method for the assignment problem", *Nav. Res. Logist. Q.*, 2, 1-2, pp. 83-97, DOI: 10.1002/nav.3800020109.

Kumar, N. Santhosh, K. Nageswara Rao, A. Govardhan, K. Sudheer Reddy, and Ali Mirza Mahmood
2014 "Undersampled K-means approach for handling imbalanced distributed data", *Lect. Notes. Artif. Int.*, 3, 1, pp. 29-38, DOI: 10.1007/s13748-014-0045-6.

Kumar, Sandeep, Buyong Ma, Chung-Jung Tsai, Neeti Sinha, and Ruth Nussinov
2008 "Folding and binding cascades: Dynamic landscapes and population shifts", *Protein Sci.*, 9, 1, pp. 10-19, DOI: 10.1110/ps.9.1.10.

Kwon, Yongbeom, Woong-Hee Shin, Junsu Ko, and Juyong Lee
2020 "AK-Score: Accurate Protein-Ligand Binding Affinity Prediction Using an Ensemble of 3D-Convolutional Neural Networks", *Int. J. Mol. Sci.*, 21, 22, p. 8424, DOI: 10.3390/ijms21228424.

Ladbury, John E.
1996 "Just add water! The effect of water on the specificity of protein-ligand binding sites and its potential application to drug design", *Chem. Biol.*, 3, 12, pp. 973-980, DOI: 10.1016/s1074-5521(96)90164-7.

Lahey, Shae-Lynn J. and Christopher N. Rowley
2020 "Simulating protein–ligand binding with neural network potentials", *Chem. Sci.*, 11, 9, pp. 2362-2368, DOI: 10.1039/c9sc06017k.

Le Cun, Y., L.D. Jackel, B. Boser, J.S. Denker, H.P. Graf, I. Guyon, D. Henderson, R.E. Howard, and W. Hubbard
1989 "Handwritten digit recognition: Applications of neural network chips and automatic learning", *IEEE Commun. Mag.*, 27, 11, pp. 41-46, DOI: 10.1109/35.41400.

Leavitt, Stephanie and Ernesto Freire
2001 "Direct measurement of protein binding energetics by isothermal titration calorimetry", *Curr. Opin. Struc. Biol.*, 11, 5, pp. 560-566, DOI: 10.1016/s0959-440x(00)00248-7.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner
1998 "Gradient-based learning applied to document recognition", *Proc. IEEE*, 86, 11, pp. 2278-2324, DOI: 10.1109/5.726791.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton
2015 "Deep learning", *Nature*, 521, 7553, pp. 436-444, DOI: 10.1038/nature14539.

Lemieux, Raymond U.
1996 "How Water Provides the Impetus for Molecular Recognition in Aqueous Solution", *Accounts Chem. Res.*, 29, 8, pp. 373-380, DOI: 10.1021/ar9600087.

Leung, S., M. Bodkin., F. von Delft, P. Brennan, and G. Morris
2019 "SuCOS is Better than RMSD for Evaluating Fragment Elaboration and Docking Poses", *ChemRxiv*, DOI: 10.26434/chemrxiv.8100203.v1.

Li, Hongjian, Kwong-Sak Leung, Man-Hon Wong, and Pedro Ballester
2015 "Low-Quality Structural and Interaction Data Improves Binding Affinity Prediction via Random Forest", *Molecules*, 20, 6, pp. 10947-10962, DOI: 10.3390/molecules200610947.

Li, Hongjian, Kam-Heung Sze, Gang Lu, and Pedro J Ballester
2020 "Machine-learning scoring functions for structure-based drug lead optimization", *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 10, 5, e1465, DOI: 10.1002/wcms.1465.
2021 "Machine-learning scoring functions for structure-based virtual screening", *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 11, 1, e1478, DOI: 10.1002/wcms.1478.

Li, Jin, Ailing Fu, and Le Zhang
2019 "An Overview of Scoring Functions Used for Protein–Ligand Interactions in Molecular Docking", *Interdiscip. Sci.-Comput. Life Sci.*, 11, 2, pp. 320-328, DOI: 10.1007/s12539-019-00327-w.

Li, Yan, Li Han, Zhihai Liu, and Renxiao Wang
2014 "Comparative Assessment of Scoring Functions on an Updated Benchmark: 2. Evaluation Methods and General Results", *J. Chem. Inf. Model.*, 54, 6, pp. 1717-1736, DOI: 10.1021/ci500081m.

Li, Yan, Zhihai Liu, Jie Li, Li Han, Jie Liu, Zhixiong Zhao, and Renxiao Wang
2014 "Comparative Assessment of Scoring Functions on an Updated Benchmark: 1. Compilation of the Test Set", *J. Chem. Inf. Model.*, 54, 6, pp. 1700-1716, DOI: 10.1021/ci500080q.

Li, Yan, Minyi Su, Zhihai Liu, Jie Li, Jie Liu, Li Han, and Renxiao Wang
2018 "Assessing protein–ligand interaction scoring functions with the CASF-2013 benchmark", *Nat. Protoc.*, 13, 4, pp. 666-680, DOI: 10.1038/nprot.2017.114.

Li, Yibo, Jianfeng Pei, and Luhua Lai
2021a "Learning to design drug-like molecules in three-dimensional space using deep generative models", *arXiv*, DOI: 10.48550/arXiv.2104.08474.
2021b "Structure-based de novo drug design using 3D deep generative models", *Chem. Sci.*, 12, 41, pp. 13664-13675, DOI: 10.1039/D1SC04444C.

Li, Yujia, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia
2018 "Learning deep generative models of graphs", *arXiv*, DOI: 10.48550/arXiv.1803.03324.

Liberatore, Elisa, Rocco Meli, and Ursula Rothlisberger
2018 "A Versatile Multiple Time Step Scheme for Efficient ab Initio Molecular Dynamics Simulations", *J. Chem. Theory Comput.*, 14, 6, pp. 2834-2842, DOI: 10.1021/acs.jctc.7b01189.

Limongelli, Vittorio, Luciana Marinelli, Sandro Cosconati, Concettina La Motta, Stefania Sartini, Laura Mugnaini, Federico Da Settimo, Ettore Novellino, and Michele Parrinello

2012 "Sampling protein motion and solvent effect during ligand binding", *Proc. Natl. Acad. Sci.*, 109, 5, pp. 1467-1472, DOI: 10.1073/pnas.1112181108.

Lin, Jung-Hsin, Alexander L Perryman, Julie R Schames, and J Andrew McCammon

2002 "Computational drug design accommodating receptor flexibility: The relaxed complex scheme", *J. Am. Chem. Soc.*, 124, 20, pp. 5632-5633, DOI: https://doi.org/10.1021/ja0260162.

Liu, Jie and Renxiao Wang

2015 "Classification of Current Scoring Functions", *J. Chem. Inf. Model.*, 55, 3, pp. 475-482, DOI: 10.1021/ci500731a.

Liu, T., Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson

2007 "BindingDB: A web-accessible database of experimentally determined protein-ligand binding affinities", *Nucleic Acids Res.*, 35, Database, pp. D198-D201, DOI: 10.1093/nar/gkl999.

Liu, Xiaofeng, Hualiang Jiang, and Honglin Li

2011 "SHAFTS: a hybrid approach for 3D molecular similarity calculation. 1. Method and assessment of virtual screening", *J. Chem. Inf. Model.*, 51, 9, pp. 2372-2385, DOI: 10.1021/ci200060s.

Liu, Zhihai, Yan Li, Li Han, Jie Li, Jie Liu, Zhixiong Zhao, Wei Nie, Yuchen Liu, and Renxiao Wang

2014 "PDB-wide collection of binding data: Current status of the PDBbind database", *Method. Biochem. Anal.*, 31, 3, pp. 405-412, DOI: 10.1093/bioinformatics/btu626.

Liu, Zhihai, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang

2017 "Forging the Basis for Developing Protein–Ligand Interaction Scoring Functions", *Accounts Chem. Res.*, 50, 2, pp. 302-309, DOI: 10.1021/acs.accounts.6b00491.

Lu, Jianing, Xuben Hou, Cheng Wang, and Yingkai Zhang

2019 "Incorporating Explicit Water Molecules and Ligand Conformation Stability in Machine-Learning Scoring Functions", *J. Chem. Inf. Model.*, 59, 11, pp. 4540-4549, DOI: 10.1021/acs.jcim.9b00645.

Lucas, Xavier, Daniel Wohlwend, Martin Hügle, Karin Schmidtkunz, Stefan Gerhardt, Roland Schüle, Manfred Jung, Oliver Einsle, and Stefan Günther

2013 "4-Acyl Pyrroles: Mimicking Acetylated Lysines in Histone Code Reading", *Angew. Chem. Int. Ed.*, 52, 52, pp. 14055-14059, DOI: 10.1002/anie.201307652.

Ma, Buyong, Sandeep Kumar, Chung-Jung Tsai, and Ruth Nussinov

1999 "Folding funnels and binding mechanisms", *Protein Eng. Des. Sel.*, 12, 9, pp. 713-720, DOI: 10.1093/protein/12.9.713.

Macarron, Ricardo, Martyn N. Banks, Dejan Bojanic, David J. Burns, Dragan A. Cirovic, Tina Garyantes, Darren V. S. Green, Robert P. Hertzberg, William P. Janzen, Jeff W. Paslay, Ulrich Schopfer, and G. Sitta Sittampalam

2011 "Impact of high-throughput screening in biomedical research", *Nat. Rev. Drug Discov.*, 10, 3, pp. 188-195, DOI: 10.1038/nrd3368.

Manby, Frederick, Thomas Miller, Peter Bygrave, Feizhi Ding, Thomas Dresselhaus, Fidel Batista-Romero, Alexander Buccheri, Callum Bungey, Sebastian Lee, Rocco Meli, Kaito Miyamoto, Casper Strinmann, Takashi Tsuchiya, Matthew Wellborn, Timothy Wiles, and Zack Williams

2019 "entos: A Quantum Molecular Simulation Package", *ChemRxiv*, DOI: 10.26434/chemrxiv.7762646.v2.

Masuda, Tomohide, Matthew Ragoza, and David Ryan Koes

2020 "Generating 3D molecular structures conditional on a receptor binding site with deep generative models", *arXiv*, DOI: 10.48550/arXiv.2010.14442.

Mater, Adam C and Michelle L Coote

2019 "Deep learning in chemistry", *J. Chem. Inf. Model.*, 59, 6, pp. 2545-2559, DOI: 10.1021/acs.jcim.9b00266.

Mayr, Lorenz M and Dejan Bojanic

2009 "Novel trends in high-throughput screening", *Curr. Opin. Pharmacol.*, 9, 5, pp. 580-588, DOI: 10.1016/j.coph.2009.08.004.

McCammon, J Andrew, Bruce R Gelin, and Martin Karplus

1977 "Dynamics of folded proteins", *Nature*, 267, 5612, pp. 585-590, DOI: 10.1038/267585a0.

McCulloch, Warren S. and Walter Pitts

1943 "A logical calculus of the ideas immanent in nervous activity", *Bull. Math. Biophys.*, 5, 4, pp. 115-133, DOI: 10.1007/bf02478259.

McNutt, Andrew T., Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes

2021 "GNINA 1.0: Molecular docking with deep learning", *J. Cheminf.*, 13, 1, p. 43, DOI: 10.1186/s13321-021-00522-2.

Meli, Rocco, Aandrew Anighoro, Michael J. Bodkin, Garrett M. Morris, and P. C. Biggin

2021 "Learning protein-ligand binding affinity with atomic environment vectors", *J. Cheminf.*, 13, 1, DOI: 10.1186/s13321-021-00536-w.

Meli, Rocco and Philip C. Biggin

2020 "spyrmsd: Symmetry-corrected RMSD calculations in Python", *J. Cheminf.*, 12, 1, p. 49, DOI: 10.1186/s13321-020-00455-2.

Meli, Rocco, Giacomo Miceli, and Alfredo Pasquarello

2017 "Oxygen DX center in $In_{0.17}Al_{0.83}N$: Nonradiative recombination and persistent photoconductivity", *Appl. Phys. Lett.*, 110, 7, p. 072101, DOI: 10.1063/1.4975934.

Meli, Rocco, Garrett M. Morris, and Philip C. Biggin

2022 "Scoring Functions for Protein-Ligand Binding Affinity Prediction Using Structure-based Deep Learning: A Review", *Front. Bioinform.*, 2, DOI: 10.3389/fbinf.2022.885983.

Mey, Antonia SJS, Bryce K Allen, Hannah E Bruce Macdonald, John D Chodera, David F Hahn, Maximilian Kuhn, Julien Michel, David L Mobley, Levi N Naden, Samarjeet Prasad, et al.

2020 "Best practices for alchemical free energy calculations", *Living J. Comp. Mol. Sci.*, 2, 1, DOI: 10.33011/livecoms.2.1.18378.

Meyers, Joshua, Benedek Fabian, and Nathan Brown

2021 "De novo molecular design and generative models", *Drug Discov. Today*, 26, 11, pp. 2707-2715, DOI: 10.1016/j.drudis.2021.05.019.

Michaud-Agrawal, Naveen, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein

2011 "MDAnalysis: A toolkit for the analysis of molecular dynamics simulations", *J. Comput. Chem.*, 32, 10, pp. 2319-2327, DOI: 10.1002/jcc.21787.

Miller, Edward B., Robert B. Murphy, Daniel Sindhikara, Kenneth W. Borrelli, Matthew J. Grisewood, Fabio Ranalli, Steven L. Dixon, Steven Jerome, Nicholas A. Boyles, Tyler Day, Phani Ghanakota, Sayan Mondal, Salma B. Rafi, Dawn M. Troast, Robert Abel, and Richard A. Friesner

2021 "Reliable and Accurate Solution to the Induced Fit Docking Problem for Protein–Ligand Binding", *J. Chem. Theory Comput.*, 17, 4, pp. 2630-2639, DOI: 10.1021/acs.jctc.1c00136.

Mobley, David L. and Michael K. Gilson

2017 "Predicting Binding Free Energies: Frontiers and Benchmarks", *Annu. Rev. Biophys.*, 46, 1, pp. 531-558, DOI: 10.1146/annurev-biophys-070816-033654.

Moesser, Marc A, Dominik Klein, Fergus Boyles, Charlotte M Deane, Andrew Baxter, and Garrett M Morris

2022 "Protein-Ligand Interaction Graphs: Learning from Ligand-Shaped 3D Interaction Graphs to Improve Binding Affinity Prediction", *bioRxiv*, DOI: 10.1101/2022.03.04.483012.

Mohammadi, Sara, Zahra Narimani, Mitra Ashouri, Rohoullah Firouzi, and Mohammad Hossein Karimi-Jafari

2022 "Ensemble learning from ensemble docking: Revisiting the optimum ensemble size problem", *Sci. Rep.*, 12, 1, pp. 1-15, DOI: 10.1038/s41598-021-04448-5.

Mölder, Felix, Kim Philipp Jablonski, Brice Letcher, Michael B Hall, Christopher H Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O Twardziok, Alexander Kanitz, Andreas Wilm, Manuel Holtgrewe, Sven Rahmann, Sven Nahnsen, and Johannes Köster

2021 "Sustainable data analysis with Snakemake", *F1000Research*, 10, DOI: 10.12688/f1000research.29032.2.

Morgan, H. L.

1965 "The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service." *J. Chem. Doc.*, 5, 2, pp. 107-113, DOI: 10.1021/c160017a018.

Morris, Garrett M., David S. Goodsell, Robert S. Halliday, Ruth Huey, William E. Hart, Richard K. Belew, and Arthur J. Olson

1998 "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function", *J. Comput. Chem.*, 19, 14, pp. 1639-1662, DOI: 10.1002/(sici)1096-987x(19981115)19:14<1639::aid-jcc10>3.0.co;2-b.

Morris, Garrett M., Ruth Huey, William Lindstrom, Michel F. Sanner, Richard K. Belew, David S. Goodsell, and Arthur J. Olson

2009 "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility", *J. Comput. Chem.*, 30, 16, pp. 2785-2791, DOI: 10.1002/jcc.21256.

Mouchlis, Varnavas D., Antreas Afantitis, Angela Serra, Michele Fratello, Anastasios G. Papadiamantis, Vassilis Aidinis, Iseult Lynch, Dario Greco, and Georgia Melagraki

2021 "Advances in De Novo Drug Design: From Conventional to Machine Learning Methods", *Int. J. Mol. Sci.*, 22, 4, p. 1676, DOI: 10.3390/ijms22041676.

Muchmore, Steven W, Jeremy J Edmunds, Kent D Stewart, and Philip J Hajduk

2010 "Cheminformatic tools for medicinal chemists", *J. Med. Chem.*, 53, 13, pp. 4830-4841, DOI: 10.1021/jm100164z.

Muegge, Ingo and Yvonne C. Martin

1999 "A General and Fast Scoring Function for Protein-Ligand Interactions: A Simplified Potential Approach", *J. Med. Chem.*, 42, 5, pp. 791-804, DOI: 10.1021/jm980536j.

Mukherjee, Sudipto, Trent E. Balius, and Robert C. Rizzo

2010 "Docking Validation Resources: Protein Family and Ligand Flexibility Experiments", *J. Chem. Inf. Model.*, 50, 11, pp. 1986-2000, DOI: 10.1021/ci1001982.

Muller, Susanne, Panagis Filippakopoulos, and Stefan Knapp

2011 "Bromodomains as therapeutic targets", *Expert Rev. Mol. Med.*, 13, e29, DOI: 10.1017/s1462399411001992.

Munkres, James

1957 "Algorithms for the Assignment and Transportation Problems", *J. Soc. Ind. Appl. Math.*, 5, 1, pp. 32-38, DOI: 10.1137/0105003.

Muratov, Eugene N., Jürgen Bajorath, Robert P. Sheridan, Igor V. Tetko, Dmitry Filimonov, Vladimir Poroikov, Tudor I. Oprea, Igor I. Baskin, Alexandre Varnek, Adrian Roitberg, Olexandr Isayev, Stefano Curtalolo, Denis Fourches, Yoram Cohen, Alan Aspuru-Guzik, David A. Winkler, Dimitris Agrafiotis, Artem Cherkasov, and Alexander Tropsha

2020 "QSAR without borders", *Chem. Soc. Rev.*, 49, 11, pp. 3525-3564, DOI: 10.1039/d0cs00098a.

Murray, Christopher W and David C Rees

2009 "The rise of fragment-based drug discovery", *Nature Chem.*, 1, 3, pp. 187-192, DOI: 10.1038/nchem.217.

Musil, Felix, Andrea Grisafi, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti

2021 "Physics-Inspired Structural Representations for Molecules and Materials", *Chem. Rev.*, 121, 16, pp. 9759-9815, DOI: 10.1021/acs.chemrev.1c00021.

Mysinger, Michael M., Michael Carchia, John. J. Irwin, and Brian K. Shoichet

2012 "Directory of Useful Decoys, Enhanced DUD-E: Better Ligands and Decoys for Better Benchmarking", *J. Med. Chem.*, 55, 14, pp. 6582-6594, DOI: 10.1021/jm300687e.

Nannenga, Brent L, Dan Shi, Andrew G W Leslie, and Tamir Gonen

2014 "High-resolution structure determination by continuous-rotation data collection in MicroED", *Nat. Methods*, 11, 9, pp. 927-930, DOI: 10.1038/nmeth.3043.

Nekooeimehr, Iman and Susana K. Lai-Yuen

2016 "Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets", *Expert Syst. Appl.*, 46, pp. 405-416, DOI: 10.1016/j.eswa.2015.10.031.

Nguyen, Duc Duy and Guo-Wei Wei

2019 "AGL-Score: Algebraic Graph Learning Score for Protein-Ligand Binding Scoring, Ranking, Docking, and Screening", *J. Chem. Inf. Model.*, 59, 7, pp. 3291-3304, DOI: 10.1021/acs.jcim.9b00334.

Noé, Frank, Gianni De Fabritiis, and Cecilia Clementi

2020 "Machine learning for protein folding and dynamics", *Curr. Opin. Struct. Biol.*, 60, pp. 77-84, DOI: 10.1016/j.sbi.2019.12.005.

O'Boyle, Noel and Andrew Dalke

2018 "DeepSMILES: An adaptation of SMILES for use in machine-learning of chemical structures", *ChemRxiv*, DOI: 10.26434/chemrxiv.7097960.v1.

O'Boyle, Noel M, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison

2011 "Open Babel: An open chemical toolbox", *J. Cheminf.*, 3, 1, DOI: 10.1186/1758-2946-3-33.

O'Boyle, Noel M, Chris Morley, and Geoffrey R Hutchison
2008 "Pybel: A Python wrapper for the OpenBabel cheminformatics toolkit", *Chem. Cent. J.*, 2, 1, DOI: 10.1186/1752-153x-2-5.

Oda, Akifumi, Keiichi Tsuchida, Tadakazu Takakura, Noriyuki Yamaotsu, and Shuichi Hirono
2006 "Comparison of consensus scoring strategies for evaluating computational models of protein- ligand complexes", *J. Chem. Inf. Model.*, 46, 1, pp. 380-391, DOI: 10.1021/ci050283k.

Palazzesi, Ferruccio and Alfonso Pozzan
2022 "Deep Learning Applied to Ligand-Based De Novo Drug Design", *Artificial Intelligence in Drug Design*, pp. 273-299, DOI: 10.1007/978-1-0716-1787-8_12.

Pandey, Mohit, Michael Fernandez, Francesco Gentile, Olexandr Isayev, Alexander Tropsha, Abraham C Stern, and Artem Cherkasov
2022 "The transformational role of GPU computing and deep learning in drug discovery", *Nat. Mach. Intell.*, 4, 3, pp. 211-221, DOI: 10.1038/s42256-022-00463-x.

Pearlman, David A. and Paul S. Charifson
2001 "Are Free Energy Calculations Useful in Practice? A Comparison with Rapid Scoring Functions for the p38 MAP Kinase Protein System", *J. Med. Chem.*, 44, 21, pp. 3417-3423, DOI: 10.1021/jm0100279.

Pettersen, Eric F., Thomas D. Goddard, Conrad C. Huang, Elaine C. Meng, Gregory S. Couch, Tristan I. Croll, John H. Morris, and Thomas E. Ferrin
2020 "UCSF ChimeraX : Structure visualization for researchers, educators, and developers", *Protein Sci.*, 30, 1, pp. 70-82, DOI: 10.1002/pro.3943.

Picaud, Sarah, Christopher Wells, Ildiko Felletar, Deborah Brotherton, Sarah Martin, Pavel Savitsky, Beatriz Diez-Dacal, Martin Philpott, Chas Bountra, Hannah Lingard, Oleg Fedorov, Susanne Müller, Paul E. Brennan, Stefan Knapp, and Panagis Filippakopoulos
2013 "RVX-208, an inhibitor of BET transcriptional regulators with selectivity for the second bromodomain", *Proc. Natl. Acad. Sci.*, 110, 49, pp. 19754-19759, DOI: 10.1073/pnas.1310658110.

Pirhadi, Somayeh, Jocelyn Sunseri, and David Ryan Koes
2016 "Open source molecular modeling", *J. Mol. Graph. Model.*, 69, pp. 127-143, DOI: 10.1016/j.jmgm.2016.07.008.

Pitt, William R., David M. Parry, Benjamin G. Perry, and Colin R. Groom
2009 "Heteroaromatic Rings of the Future", *J. Med. Chem.*, 52, 9, pp. 2952-2963, DOI: 10.1021/jm801513z.

Plaut, Elad
2018 "From Principal Subspaces to Principal Components with Linear Autoencoders", *arXiv*, DOI: 10.48550/arXiv.1804.10253.

Plowright, Alleyn T, Craig Johnstone, Jan Kihlberg, Jonas Pettersson, Graeme Robb, and Richard A Thompson

2012 "Hypothesis driven drug design: improving quality and effectiveness of the design-make-test-analyse cycle", *Drug Discovery Today*, 17, 1-2, pp. 56-62, DOI: 10.1016/j.drudis.2011.09.012.

Poltavsky, Igor and Alexandre Tkatchenko

2021 "Machine learning force fields: Recent advances and remaining challenges", *J. Phys. Chem. Lett.*, 12, 28, pp. 6551-6564, DOI: 10.1021/acs.chemrev.0c01111.

Polyak, Boris T

1964 "Some methods of speeding up the convergence of iteration methods", *USSR Comput. Math. Math. Phys.*, 4, 5, pp. 1-17, DOI: 10.1016/0041-5553(64)90137-5.

Polykovskiy, Daniil, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al.

2020 "Molecular sets (MOSES): a benchmarking platform for molecular generation models", *Front. Pharmacol.*, 11, p. 1931, DOI: doi.org/10.3389/fphar.2020.565.

Popova, Mariya, Olexandr Isayev, and Alexander Tropsha

2018 "Deep reinforcement learning for de novo drug design", *Sci. Adv.*, 4, 7, eaap7885, DOI: 10.1126/sciadv.aap7885.

Powers, Alexander, Helen Yu, Patricia Suriana, and Ron Dror

2022 "Fragment-Based Ligand Generation Guided by Geometric Deep Learning on Protein-Ligand Structure", *bioRxiv*, DOI: 10.1101/2022.03.17.484653.

Pretorius, Sy

2016 "Phase III trial failures: costly, but preventable", *Applied Clinical Trials*, 25, 8/9, p. 36.

Pu, Limeng, Rajiv Gandhi Govindaraj, Jeffrey Mitchell Lemoine, Hsiao-Chun Wu, and Michal Brylinski

2019 "DeepDrug3D: Classification of ligand-binding pockets in proteins with a convolutional neural network", *PLoS Comput. Biol.*, 15, 2, e1006718, DOI: 10.1371/journal.pcbi.1006718.

Quiroga, Rodrigo and Marcos A Villarreal

2016 "Vinardo: A scoring function based on AutoDock Vina improves scoring, docking, and virtual screening", *PLoS ONE*, 11, 5, e0155183, DOI: 10.1371/journal.pone.0155183.

Ragoza, Matthew, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes

2017 "Protein-Ligand Scoring with Convolutional Neural Networks", *J. Chem. Inf. Model.*, 57, 4, pp. 942-957, DOI: 10.1021/acs.jcim.6b00740.

Ragoza, Matthew, Tomohide Masuda, and David Ryan Koes
2020 "Learning a Continuous Representation of 3D Molecular Structures with Deep Generative Models", *arXiv*, DOI: 10.48550/arXiv.2010.08687.
2022 "Generating 3D molecules conditional on receptor binding sites with deep generative models", *Chem. Sci.*, 13, 9, pp. 2701-2713, DOI: 10.1039/d1sc05976a.

Ragoza, Matthew, Lillian Turner, and David Ryan Koes
2017 "Ligand pose optimization with atomic grid-based convolutional neural networks", *arXiv*, DOI: 10.48550/arXiv.1710.07400.

Raies, Arwa B and Vladimir B Bajic
2016 "In silico toxicology: computational methods for the prediction of chemical toxicity", *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 6, 2, pp. 147-172, DOI: 10.1002/wcms.1240.

Ramakrishnan, Raghunathan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld
2015 "Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach", *J. Chem. Theory Comput.*, 11, 5, pp. 2087-2096, DOI: 10.1021/acs.jctc.5b00099.

Ramaswamy, Venkata K, Samuel C Musson, Chris G Willcocks, and Matteo T Degiacomi
2021 "Deep learning protein conformational space with convolutions and latent interpolations", *Phys. Rev. X*, 11, 1, p. 011052, DOI: 10.1103/PhysRevX.11.011052.

Rappe, A. K., C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff
1992 "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations", *J. Am. Chem. Soc.*, 114, 25, pp. 10024-10035, DOI: 10.1021/ja00051a040.

Reymond, Jean-Louis, Ruud van Deursen, Lorenz C. Blum, and Lars Ruddigkeit
2010 "Chemical space as a source for new drugs", *Med. Chem. Comm.*, 1, 1, p. 30, DOI: 10.1039/c0md00020e.

Ricci-Lopez, Joel, Sergio A. Aguila, Michael K. Gilson, and Carlos A. Brizuela
2021 "Improving Structure-Based Virtual Screening with Ensemble Docking and Machine Learning", *J. Chem. Inf. Model.*, 61, 11, pp. 5362-5376, DOI: 10.1021/acs.jcim.1c00511.

Riniker, Sereina and Gregory A. Landrum
2015 "Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation", *J. Chem. Inf. Model.*, 55, 12, pp. 2562-2574, DOI: 10.1021/acs.jcim.5b00654.

Rogers, David and Mathew Hahn
2010 "Extended-Connectivity Fingerprints", *J. Chem. Inf. Model.*, 50, 5, pp. 742-754, DOI: 10.1021/ci100050t.

Ross, Gregory A., Garrett M. Morris, and Philip C. Biggin
2012 "Rapid and Accurate Prediction and Scoring of Water Molecules in Protein Binding Sites", *PLoS ONE*, 7, 3, e32036, DOI: 10.1371/journal.pone.0032036.
2013 "One Size Does Not Fit All: The Limits of Structure-Based Models in Drug Discovery", *J. Chem. Theory Comput.*, 9, 9, pp. 4266-4274, DOI: 10.1021/ct4004228.

Roughley, Stephen D. and Allan M. Jordan
2011 "The Medicinal Chemist's Toolbox: An Analysis of Reactions Used in the Pursuit of Drug Candidates", *J. Med. Chem.*, 54, 10, pp. 3451-3479, DOI: 10.1021/jm200187y.

Rufa, Dominic A., Hannah E. Bruce Macdonald, Josh Fass, Marcus Wieder, Patrick B. Grinaway, Adrian E. Roitberg, Olexandr Isayev, and John D. Chodera
2020 "Towards chemical accuracy for alchemical free energy calculations with hybrid physics-based machine learning/molecular mechanics potentials", *bioRxiv*, DOI: 10.1101/2020.07.29.227959.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams
1986 "Learning representations by back-propagating errors", *Nature*, 323, 6088, pp. 533-536, DOI: 10.1038/323533a0.

Rupp, Matthias, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld
2012 "Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning", *Phys. Rev. Lett.*, 108, 5, p. 058301, DOI: 10.1103/physrevlett.108.058301.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei
2015 "ImageNet Large Scale Visual Recognition Challenge", *Int. J. Comput. Vision*, 115, 3, pp. 211-252, DOI: 10.1007/s11263-015-0816-y.

Salo-Ahen, Outi MH, Ida Alanko, Rajendra Bhadane, Alexandre MJJ Bonvin, Rodrigo Vargas Honorato, Shakhawath Hossain, André H Juffer, Aleksei Kabedev, Maija Lahtela-Kakkonen, Anders Støttrup Larsen, et al.
2020 "Molecular dynamics simulations in drug discovery and pharmaceutical development", *Processes*, 9, 1, p. 71, DOI: 10.3390/pr9010071.

Sanchez-Lengeling, Benjamin and Alán Aspuru-Guzik
2018 "Inverse molecular design using machine learning: Generative models for matter engineering", *Science*, 361, 6400, pp. 360-365, DOI: 10.1126/science.aat2663.

Sastry, G Madhavi, Steven L Dixon, and Woody Sherman
2011 "Rapid shape-based ligand alignment and virtual screening method based on atom/feature-pair similarities and volume overlap scoring", *J. Chem. Inf. Model.*, 51, 10, pp. 2455-2466, DOI: 10.1021/ci2002704.

Scantlebury, Jack, Nathan Brown, Frank Von Delft, and Charlotte M. Deane
2020 "Data Set Augmentation Allows Deep Learning-Based Virtual Screening to Better Generalize to Unseen Target Classes and Highlight Important Binding Interactions", *J. Chem. Inf. Model.*, 60, 8, pp. 3722-3730, DOI: 10.1021/acs.jcim.0c00263.

Schenone, Monica, Vlado Dančik, Bridget K Wagner, and Paul A Clemons
2013 "Target identification and mechanism of action in chemical biology and drug discovery", *Nat. Chem. Bio.*, 9, 4, pp. 232-240, DOI: 10.1038/nchembio.1199.

Schneider, Gisbert
2018 "Automating drug discovery", *Nat. Rev. Drug Discovery*, 17, 2, pp. 97-113, DOI: 10.1038/nrd.2017.232.

Schneider, Gisbert and David E. Clark
2019 "Automated De Novo Drug Design: Are We Nearly There Yet?", *Angew. Chem. Int. Ed.*, 58, 32, pp. 10792-10803, DOI: 10.1002/anie.201814681.

Schneider, Gisbert and Uli Fechner
2005 "Computer-based de novo design of drug-like molecules", *Nat. Rev. Drug Discovery*, 4, 8, pp. 649-663, DOI: 10.1038/nrd1799.

Schneider, Petra and Gisbert Schneider
2016 "De Novo Design at the Edge of Chaos", *J. Med. Chem.*, 59, 9, pp. 4077-4086, DOI: 10.1021/acs.jmedchem.5b01849.

Schneider, Petra, W Patrick Walters, Alleyn T Plowright, Norman Sieroka, Jennifer Listgarten, Robert A Goodnow, Jasmin Fisher, Johanna M Jansen, José S Duca, Thomas S Rush, Matthias Zentgraf, John Edward Hill, Elizabeth Krutoholow, Matthias Johler, Jeff Blaney, Kimito Funatsu, Chris Luebkemann, and Gisbert Schneider
2020 "Rethinking drug design in the artificial intelligence era", *Nat. Rev. Drug Discovery*, 19, 5, pp. 353-364, DOI: 10.1038/s41573-019-0050-3.

Schwaller, Philippe, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee
2019 "Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction", *ACS Cent. Sci.*, 5, 9, pp. 1572-1583, DOI: 10.1021/acscentsci.9b00576.

Schwaller, Philippe, Daniel Probst, Alain C Vaucher, Vishnu H Nair, David Kreutter, Teodoro Laino, and Jean-Louis Reymond
2021 "Mapping the space of chemical reactions using attention-based neural networks", *Nat. Mach. Intell.*, 3, 2, pp. 144-152, DOI: 10.1038/s42256-020-00284-w.

Scior, Thomas, Andreas Bender, Gary Tresadern, José L. Medina-Franco, Karina Martinez-Mayorga, Thierry Langer, Karina Cuanalo-Contreras, and Dimitris K. Agrafiotis
2012 "Recognizing Pitfalls in Virtual Screening: A Critical Review", *J. Chem. Inf. Model.*, 52, 4, pp. 867-881, DOI: 10.1021/ci200528d.

Shamir, Ron and Dekel Tsur
1999 "Faster subtree isomorphism", *J. Algorithms*, 33, 2, pp. 267-280, DOI: 10.1006/jagm.1999.1044.

Shanno, D. F.
1970 "Conditioning of quasi-Newton methods for function minimization", *Math. Comput.*, 24, 111, pp. 647-656, DOI: 10.1090/s0025-5718-1970-0274029-x.

Shannon, C. E.
1948 "A Mathematical Theory of Communication", *Bell Syst. Tech. J.*, 27, 3, pp. 379-423, DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Sherman, Woody, Tyler Day, Matthew P. Jacobson, Richard A. Friesner, and Ramy Farid
2005 "Novel Procedure for Modeling Ligand/Receptor Induced Fit Effects", *J. Med. Chem.*, 49, 2, pp. 534-553, DOI: 10.1021/jm050540c.

Sieg, Jochen, Florian Flachsenberg, and Matthias Rarey
2019 "In Need of Bias Control: Evaluating Chemical Data for Machine Learning in Structure-Based Virtual Screening", *J. Chem. Inf. Model.*, 59, 3, pp. 947-961, DOI: 10.1021/acs.jcim.8b00712.

Simm, Gregor NC and José Miguel Hernández-Lobato
2019 "A generative model for molecular distance geometry", *arXiv*, DOI: 10.48550/arXiv.1909.11459.

Skalic, Miha, José Jiménez, Davide Sabbadin, and Gianni De Fabritiis
2019 "Shape-Based Generative Modeling for de Novo Drug Design", *J. Chem. Inf. Model.*, 59, 3, pp. 1205-1214, DOI: 10.1021/acs.jcim.8b00706.

Skalic, Miha, Davide Sabbadin, Boris Sattarov, Simone Sciabola, and Gianni De Fabritiis
2019 "From Target to Drug: Generative Modeling for the Multimodal Structure-Based Ligand Design", *Mol. Pharmaceut.*, 16, 10, pp. 4282-4291, DOI: 10.1021/acs.molpharmaceut.9b00634.

Sliwoski, Gregory, Sandeepkumar Kothiwale, Jens Meiler, and Edward W. Lowe
2013 "Computational Methods in Drug Discovery", *Pharmacol. Rev.*, 66, 1, pp. 334-395, DOI: 10.1124/pr.112.007336.

Smith, J. S., O. Isayev, and A. E. Roitberg
2017 "ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost", *Chem. Sci.*, 8, 4, pp. 3192-3203, DOI: 10.1039/c6sc05720a.

Smith, Justin S., Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E. Roitberg
2018 "Less is more: Sampling chemical space with active learning", *J. Chem. Phys.*, 148, 24, p. 241733, DOI: 10.1063/1.5023802.

Smith, Justin S., Benjamin T. Nebgen, Roman Zubatyuk, Nicholas Lubbers, Christian Devereux, Kipton Barros, Sergei Tretiak, Olexandr Isayev, and Adrian E. Roitberg
2019 "Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning", *Nat. Commun.*, 10, 1, p. 2903, DOI: 10.1038/s41467-019-10827-4.

Spearman, C
1961 "The proof and measurement of association between two things", pp. 45-58, DOI: 10.1037/11491-005.

Sridhar, Akshay, Gregory A. Ross, and Philip C. Biggin
2017 "Waterdock 2.0: Water placement prediction for Holo-structures with a pymol plugin", *PLoS ONE*, 12, 2, e0172743, DOI: 10.1371/journal.pone.0172743.

Stepniewska-Dziubinska, Marta M, Piotr Zielenkiewicz, and Pawel Siedlecki
2018 "Development and evaluation of a deep learning model for protein-ligand binding affinity prediction", *Method. Biochem. Anal.*, 34, 21, pp. 3666-3674, DOI: 10.1093/bioinformatics/bty374.

Struble, Thomas J, Juan C Alvarez, Scott P Brown, Milan Chytil, Justin Cisar, Renee L DesJarlais, Ola Engkvist, Scott A Frank, Daniel R Greve, Daniel J Griffin, et al.
2020 "Current and future roles of artificial intelligence in medicinal chemistry synthesis", *J. Med. Chem.*, 63, 16, pp. 8667-8682, DOI: 10.1021/acs.jmedchem.9b02120.

Su, Minyi, Guoqin Feng, Zhihai Liu, Yan Li, and Renxiao Wang
2020 "Tapping on the Black Box: How Is the Scoring Power of a Machine-Learning Scoring Function Dependent on the Training Set?", *J. Chem. Inf. Model.*, 60, 3, pp. 1122-1136, DOI: 10.1021/acs.jcim.9b00714.

Su, Minyi, Qifan Yang, Yu Du, Guoqin Feng, Zhihai Liu, Yan Li, and Renxiao Wang
2018 "Comparative Assessment of Scoring Functions: The CASF-2016 Update", *J. Chem. Inf. Model.*, 59, 2, pp. 895-913, DOI: 10.1021/acs.jcim.8b00545.

Sullivan, C. Bane and Alexander Kaszynski

2019 "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)", *Journal of Open Source Software*, 4, 37, p. 1450, DOI: 10.21105/joss.01450.

Sunseri, Jocelyn and David R. Koes

2020 "libmolgrid: Graphics Processing Unit Accelerated Molecular Gridding for Deep Learning Applications", *J. Chem. Inf. Model.*, 60, 3, pp. 1079-1084, DOI: 10.1021/acs.jcim.9b01145.

Swinney, DC

2013 "Phenotypic vs. target-based drug discovery for first-in-class medicines", *Clin. Pharmacol. Ther.*, 93, 4, pp. 299-301, DOI: 10.1038/clpt.2012.236.

Sydow, Dominique, Jaime Rodriguez-Guerra, and Andrea Volkamer

2022 "OpenCADD-KIFS: A Python package to fetch kinase data from the KLIFS database", *J. Open Source Softw.*, 7, 70, p. 3951, DOI: 10.21105/joss.03951.

Tanimoto, Taffee T

1958 "Elementary mathematical theory of classification and prediction".

The COVID Moonshot Consortium, John Chodera, Alpha Lee, Nir London, and Frank von Delft

2020 "COVID moonshot: open science discovery of SARS-CoV-2 main protease inhibitors by combining crowdsourcing, high-throughput experiments, computational simulations, and machine learning", *bioRxiv*, DOI: 10.1101/2020.10.29.339317.

Theobald, Douglas L.

2005 "Rapid calculation of RMSDs using a quaternion-based characteristic polynomial", *Acta Crystallogr., Sect. A: Found. Crystallogr.*, 61, 4, pp. 478-480, DOI: 10.1107/s0108767305015266.

Thomas, Morgan, Robert T Smith, Noel M O'Boyle, Chris de Graaf, and Andreas Bender

2021 "Comparison of structure- and ligand-based scoring functions for deep generative models: a GPCR case study", *J. Cheminform.*, 13, 1, pp. 1-20, DOI: 10.1186/s13321-021-00516-0.

Tiwary, Pratyush, Vittorio Limongelli, Matteo Salvalaglio, and Michele Parrinello

2015 "Kinetics of protein–ligand unbinding: Predicting pathways, rates, and rate-limiting steps", *PNAS*, 112, 5, E386-E391, DOI: 10.1073/pnas.1424461112.

Trott, Oleg and Arthur J. Olson

2009 "AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading", *J. Comput. Chem.*, 31, 2, pp. 455-461, DOI: 10.1002/jcc.21334.

Tsai, Chung-Jung, Sandeep Kumar, Buyong Ma, and Ruth Nussinov
1999 "Folding funnels, binding funnels, and protein function", *Protein Sci.*, 8, 6, pp. 1181-1190, DOI: 10.1110/ps.8.6.1181.

Unke, Oliver T., Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller
2021 "Machine Learning Force Fields", *Chem. Rev.*, 121, 16, pp. 10142-10186, DOI: 10.1021/acs.chemrev.0c01111.

Vainio, Mikko J., J. Santeri Puranen, and Mark S. Johnson
2009 "ShaEP: Molecular Overlay Based on Shape and Electrostatic Potential", *J. Chem. Inf. Model.*, 49, 2, pp. 492-502, DOI: 10.1021/ci800315d.

Van Linden, Oscar P. J., Albert J. Kooistra, Rob Leurs, Iwan J. P. de Esch, and Chris de Graaf
2013 "KLIFS: A Knowledge-Based Structural Database To Navigate Kinase–Ligand Interaction Space", *J. Med. Chem.*, 57, 2, pp. 249-277, DOI: 10.1021/jm400378w.

Verdonk, Marcel L., Jason C. Cole, Michael J. Hartshorn, Christopher W. Murray, and Richard D. Taylor
2003 "Improved protein-ligand docking using GOLD", *Proteins: Struct., Funct., Bioinf.*, 52, 4, pp. 609-623, DOI: 10.1002/prot.10465.

Vincent, Fabien, Arsenio Nueda, Jonathan Lee, Monica Schenone, Marco Prunotto, and Mark Mercola
2022 "Phenotypic drug discovery: recent successes, lessons learned and new directions", *Nat. Rev. Drug Discovery*, pp. 1-16, DOI: 10.1038/s41573-022-00472-w.

Vincent, Pascal
2011 "A Connection Between Score Matching and Denoising Autoencoders", *Neural Comput.*, 23, 7, pp. 1661-1674, DOI: 10.1162/neco_a_00142.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R.J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors
2020 "Scipy 1.0: Fundamental algorithms for scientific computing in Python", *Nat. Methods*, 17, pp. 261-272, DOI: 10.1038/s41592-019-0686-2.

Volkov, Mikhail, Joseph-André Turk, Nicolas Drizard, Nicolas Martin, Brice Hoffmann, Yann Gaston-Mathé, and Didier Rognan
2022 "On the Frustration to Predict Binding Affinities from Protein-Ligand Structures with Deep Neural Networks", *J. Med. Chem.*, 65, 11, pp. 7946-7958, DOI: 10.1021/acs.jmedchem.2c00487.

Voulodimos, Athanasios, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis
2018 "Deep Learning for Computer Vision: A Brief Review", *Comput. Intell. Neurosci.*, 2018, pp. 1-13, DOI: 10.1155/2018/7068349.

Wagner, Jeffrey R, Christopher T Lee, Jacob D Durrant, Robert D Malmstrom, Victoria A Feher, and Rommie E Amaro
2016 "Emerging computational methods for the rational discovery of allosteric drugs", *Chem. Rev.*, 116, 11, pp. 6370-6390, DOI: 10.1021/acs.chemrev.5b00631.

Wallach, Izhar and Abraham Heifets
2018 "Most Ligand-Based Classification Benchmarks Reward Memorization Rather than Generalization", *J. Chem. Inf. Model.*, 58, 5, pp. 916-932, DOI: 10.1021/acs.jcim.7b00403.

Walters, Patrick and Mark Murcko
2020a "Assessing the impact of generative AI on medicinal chemistry", *Nat. Biotechnol.*, 38, 2, pp. 143-145, DOI: 10.1038/s41587-020-0418-2.

Walters, W Patrick
2020 "Code sharing in the open science era", *J. Chem. Inf. Model.*, 60, 10, pp. 4417-4420, DOI: 10.1021/acs.jcim.0c01000.

Walters, W Patrick and Mark Murcko
2020b "Assessing the impact of generative AI on medicinal chemistry", *Nat. Biotechol.*, 38, 2, pp. 143-145, DOI: 10.1038/s41587-020-0418-2.

Wang, Cheng and Yingkai Zhang
2016 "Improving scoring-docking-screening powers of protein-ligand scoring functions using random forest", *J. Comput. Chem.*, 38, 3, pp. 169-177, DOI: 10.1002/jcc.24667.

Wang, Renxiao, Xueliang Fang, Yipin Lu, and Shaomeng Wang
2004 "The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures", *J. Med. Chem.*, 47, 12, pp. 2977-2980, DOI: 10.1021/jm030580l.

Wang, Renxiao, Xueliang Fang, Yipin Lu, Chao-Yie Yang, and Shaomeng Wang
2005 "The PDBbind Database: Methodologies and Updates", *J. Med. Chem.*, 48, 12, pp. 4111-4119, DOI: 10.1021/jm048957q.

Wang, Shuzhe, Jagna Witek, Gregory A. Landrum, and Sereina Riniker
2020 "Improving Conformer Generation for Small Rings and Macrocycles Based on Distance Geometry and Experimental Torsional-Angle Preferences", *J. Chem. Inf. Model.*, 60, 4, pp. 2044-2058, DOI: 10.1021/acs.jcim.0c00025.

Wang, Yihang, Joao Marcelo Lamim Ribeiro, and Pratyush Tiwary
2020 "Machine learning approaches for analyzing and enhancing molecular dynamics simulations", *Curr. Opin. Struct. Biol.*, 61, pp. 139-145, DOI: 10.1016/j.sbi.2019.12.016.

Waskom, Michael
2021 "seaborn: Statistical data visualization", *J. Open Source Softw.*, 6, 60, p. 3021, DOI: 10.21105/joss.03021.

Weininger, David
1988 "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules", *J. Chem. Inf. Model.*, 28, 1, pp. 31-36, DOI: 10.1021/ci00057a005.

Weininger, David, Arthur Weininger, and Joseph L. Weininger
1989 "SMILES. 2. Algorithm for generation of unique SMILES notation", *J. Chem. Inf. Comp. Sci.*, 29, 2, pp. 97-101, DOI: 10.1021/ci00062a008.

Wierbowski, Shayne D., Bentley M. Wingert, Jim Zheng, and Carlos J. Camacho
2019 "Cross-docking benchmark for automated pose and ranking prediction of ligand binding", *Protein Sci.*, 29, 1, pp. 298-305, DOI: 10.1002/pro.3784.

Wildman, Scott A. and Gordon M. Crippen
1999 "Prediction of Physicochemical Parameters by Atomic Contributions", *J. Chem. Inf. Comp. Sci.*, 39, 5, pp. 868-873, DOI: 10.1021/ci990307l.

Winter, Robin, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert
2019 "Efficient multi-objective molecular optimization in a continuous latent space", *Chem. Sci.*, 10, 34, pp. 8016-8024, DOI: 10.1039/C9SC01928F.

Wu, Zhiyi and Philip C. Biggin
2022 "Correction Schemes for Absolute Binding Free Energies Involving Lipid Bilayers", *J. Chem. Theory Comput.*, 18, 4, pp. 2657-2672, DOI: 10.1021/acs.jctc.1c01251.

Xu, Ziqiao, Orrette R Wauchope, and Aaron T Frank
2021 "Navigating chemical space by interfacing generative artificial intelligence and molecular docking", *J. Chem. Inf. Model.*, 61, 11, pp. 5589-5600, DOI: 10.1021/acs.jcim.1c00746.

Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria
2018 "Recent Trends in Deep Learning Based Natural Language Processing", *IEEE Comput. Intell. Mag.*, 13, 3, pp. 55-75, DOI: 10.1109/mci.2018.2840738.

Zeiler, Matthew D and Rob Fergus
2013 "Visualizing and Understanding Convolutional Networks", *arXiv*, DOI: 10.48550/arxiv.1311.2901.

Zhang, She, James M Krieger, Yan Zhang, Cihan Kaya, Burak Kaynak, Karolina Mikulska-Ruminska, Pemra Doruker, Hongchun Li, and Ivet Bahar
2021 "ProDy 2.0: Increased scale and scope after 10 years of protein dynamics modelling with Python", *Method. Biochem. Anal.*, 37, 20, pp. 3657-3659, DOI: 10.1093/bioinformatics/btab187.

Zhavoronkov, Alex, Yan A. Ivanenkov, Alex Aliper, Mark S. Veselov, Vladimir A. Aladinskiy, Anastasiya V. Aladinskaya, Victor A. Terentiev, Daniil A. Polykovskiy, Maksim D. Kuznetsov, Arip Asadulaev, Yury Volkov, Artem Zholus, Rim R. Shayakhmetov, Alexander Zhebrak, Lidiya I. Minaeva, Bogdan A. Zagribelnyy, Lennart H. Lee, Richard Soll, David Madge, Li Xing, Tao Guo, and Alán Aspuru-Guzik
2019 "Deep learning enables rapid identification of potent DDR1 kinase inhibitors", *Nat. Biotechnol.*, 37, 9, pp. 1038-1040, DOI: 10.1038/s41587-019-0224-x.

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun
2018 "Open3D: A Modern Library for 3D Data Processing", *arXiv*, DOI: 10.48550/arXiv.1801.09847.

Zhou, Y.-T., R. Chellappa, A. Vaid, and B.K. Jenkins
1988 "Image restoration using a neural network", *IEEE Trans. Acoust. Speech Signal Process.*, 36, 7, pp. 1141-1151, DOI: 10.1109/29.1641.

Zhu, Fangqiang, Xiaohua Zhang, Jonathan E. Allen, Derek Jones, and Felice C. Lightstone
2020 "Binding Affinity Prediction by Pairwise Function Based on Neural Network", *J. Chem. Inf. Model.*, 60, 6, pp. 2766-2772, DOI: 10.1021/acs.jcim.0c00026.

Conference Proceedings

Baldi, Pierre
2012 "Autoencoders, Unsupervised Learning, and Deep Architectures", in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, ed. by Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, Proceedings of Machine Learning Research, PMLR, Bellevue, Washington, USA, vol. 27, pp. 37-49.

Cao, Hong, Xiao-Li Li, Yew-Kwong Woon, and See-Kiong Ng
2011 "SPO: Structure Preserving Oversampling for Imbalanced Time Series Classification", in *2011 IEEE 11th International Conference on Data Mining*, IEEE, IEEE, pp. 1008-1013, DOI: 10.1109/icdm.2011.137.

Cordella, Luigi Pietro, Pasquale Foggia, Carlo Sansone, and Mario Vento
2001 "An improved algorithm for matching large graphs", in *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pp. 149-159.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei
2009 "ImageNet: A large-scale hierarchical image database", in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, IEEE, pp. 248-255, DOI: 10.1109/cvpr.2009.5206848.

Goodfellow, Ian
2017 "NIPS 2016 Tutorial: Generative Adversarial Networks", in.

Gowers, Richard, Max Linke, Jonathan Barnoud, Tyler Reddy, Manuel Melo, Sean Seyler, Jan Domański, David Dotson, Sébastien Buchoux, Ian Kenney, and Oliver Beckstein
2016 "MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations", in *Proceedings of the Python in Science Conference*, ed. by Sebastian Benthall and Scott Rostrup, SciPy, pp. 98-105, DOI: 10.25080/majora-629e541a-00e.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton
2013 "Speech recognition with deep recurrent neural networks", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645-6649, DOI: 10.1109/ICASSP.2013.6638947.

Gurav, Vedant, Muhanned Parkar, and Parth Kharwar
2019 "Accessible and Ethical Data Annotation with the Application of Gamification", in *International Conference on Recent Developments in Science, Engineering and Technology*, Springer, pp. 68-78, DOI: 10.1007/978-981-15-5830-6_6.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart
2008 "Exploring Network Structure, Dynamics, and Function using NetworkX", in *Proceedings of the 7th Python in Science Conference*, ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman, Pasadena, CA USA, pp. 11-15.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun
2016 "Deep Residual Learning for Image Recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 770-778, DOI: 10.1109/cvpr.2016.90.

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger
2017 "Densely Connected Convolutional Networks", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 2261-2269, DOI: 10.1109/cvpr.2017.243.

Jin, Wengong, Regina Barzilay, and Tommi Jaakkola
2018 "Junction Tree Variational Autoencoder for Molecular Graph Generation", in *Proceedings of the 35th International Conference on Machine Learning*, ed. by Jennifer Dy and Andreas Krause, Proceedings of Machine Learning Research, PMLR, vol. 80, pp. 2323-2332.

Lee, Jinsoo, Wook-Shin Han, Romans Kasperovics, and Jeong-Hoon Lee
2012 "An in-depth comparison of subgraph isomorphism algorithms in graph databases", in, 2, VLDB Endowment, vol. 6, pp. 133-144, DOI: 10.14778/2535568.2448946.

Luo, Shitong, Jiaqi Guan, Jianzhu Ma, and Jian Peng
2021 "A 3D Generative Model for Structure-Based Drug Design", in *Advances in Neural Information Processing Systems*, ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Curran Associates, Inc., vol. 34, pp. 6229-6239.

Masters, Matthew, Amr H Mahmoud, Yao Wei, and Markus Alexander Lill
2022 "Deep learning model for flexible and efficient protein-ligand docking", in *ICLR 2022 Machine Learning for Drug Discovery*.

McKinney, Wes
2010 "Data Structures for Statistical Computing in Python", in *Proceedings of the Python in Science Conference*, ed. by Stefan van der Walt and Jarrod Millman, SciPy, pp. 56-61, DOI: 10.25080/majora-92bf1922-00a.

Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi
2016 "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation", in *2016 Fourth International Conference on 3D Vision (3DV)*, IEEE, IEEE, pp. 565-571, DOI: 10.1109/3dv.2016.79.

Park, Jaesik, Qian-Yi Zhou, and Vladlen Koltun
2017 "Colored Point Cloud Registration Revisited", in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, pp. 143-152, DOI: 10.1109/iccv.2017.25.

Rusinkiewicz, S. and M. Levoy
2001 "Efficient variants of the ICP algorithm", in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, IEEE, IEEE Comput. Soc, pp. 145-152, DOI: 10.1109/im.2001.924423.

Rusu, R.B., N. Blodow, Z.C. Marton, and M. Beetz
2008 "Aligning point cloud views using persistent feature histograms", in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, IEEE, pp. 3384-3391, DOI: 10.1109/iros.2008.4650967.

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz
2009 "Fast Point Feature Histograms (FPFH) for 3D registration", in *2009 IEEE International Conference on Robotics and Automation*, IEEE, IEEE, pp. 3212-3217, DOI: 10.1109/robot.2009.5152473.

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen
2016 "Improved Techniques for Training GANs", in *Advances in Neural Information Processing Systems*, ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Curran Associates, Inc., vol. 29.

Simm, Gregor, Robert Pinsler, and José Miguel Hernández-Lobato
2020 "Reinforcement learning for molecular design guided by quantum mechanics", in *International Conference on Machine Learning*, PMLR, pp. 8959-8969.

Simonovsky, Martin and Nikos Komodakis
2018 "GraphVAE: Towards generation of small graphs using variational autoencoders", in *International conference on artificial neural networks*, Springer, pp. 412-422.

Stärk, Hannes, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola
2022 "Equibind: Geometric deep learning for drug binding structure prediction", in *International Conference on Machine Learning*, PMLR, pp. 20503-20521.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton
2013 "On the importance of initialization and momentum in deep learning", in *International conference on machine learning*, PMLR, pp. 1139-1147.

Szegedy, C., Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich
2015 "Going deeper with convolutions", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 1-9, DOI: 10.1109/cvpr.2015.7298594.

Unterthiner, Thomas, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter
2014 "Deep learning as an opportunity in virtual screening", in *Proceedings of the deep learning workshop at NIPS*, vol. 27, pp. 1-9.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol
2008 "Extracting and composing robust features with denoising autoencoders", in *Proceedings of the 25th international conference on Machine learning - ICML '08*, ICML '08, ACM Press, pp. 1096-1103, DOI: 10.1145/1390156.1390294.

You, Jiaxuan, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec
2018 "Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation", in *Advances in Neural Information Processing Systems*, ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Curran Associates, Inc., vol. 31.

Books

Bailey, Dale L, Michael N Maisey, David W Townsend, and Peter E Valk
2005 *Positron Emission Tomography*, Springer-Verlag, DOI: 10.1007/b136169.

Baldi, Pierre
2021 *Deep Learning in Science*, Cambridge University Press, DOI: 10/gpxj26.

Berg, J.M., Tymoczko L. J., G.J. Gatto, and Lubert Stryer
2019 *Biochemistry*, W. H. Freeman.

Bishop, Christopher M.
2006 *Pattern Recognition and Machine Learning*, Springer, DOI: 10.1007/978-0-387-45528-0.

Brown, Nathan
2021 (ed.), *Artificial Intelligence in Drug Discovery*, Drug Discovery, The Royal Society of Chemistry, DOI: 10.1039/9781788016841.

Chacon, Scott and Ben Straub
2014 *Pro Git*, Apress, DOI: 10.1007/978-1-4842-0076-6.

Foster, David
2019 *Generative Deep Learning*, O'Reilly Media, Inc.

Géron, Aurélien
2019 *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, Inc.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville
2016 *Deep Learning*, MIT Press, deeplearningbook.org.

Ignazio, J. and T.M. Cavalier
1993 *Linear Programming*, Elsevier, DOI: 10.1016/c2009-1-28582-0.

Mitchell, T.M.
1983 *Machine Learning*, McGraw-Hill Series in Computer Science, Springer, DOI: 10.1007/978-3-662-12405-5.

Neal, Radford M.
1996 *Bayesian Learning for Neural Networks*, Springer, DOI: 10.1007/978-1-4612-0745-0.

Nelson, David L. and Michael M. Cox
2013 *Lehninger Principles of Biochemistry*, MacMillan.

Nocedal, Jorge and Stephen J. Wright
1999 *Numerical Optimization*, 2nd ed, Springer Series in Operations Research, Springer-Verlag, DOI: 10.1007/b98874.

Quarteroni, Alfio, Riccardo Sacco, and Fausto Saleri
2007 *Numerical Mathematics*, Springer, DOI: 10.1007/978-0-387-22750-4.

Rosenblatt, Frank
1962 *Perceptions and the theory of brain mechanisms*, Spartan Books.

Rumelhart, David E., James L. McClelland, and Corporate PDP Research Group
1986 (eds.), *Parallel Distributed Processing*, The MIT Press, Cambridge, MA, USA, DOI: 10.7551/mitpress/5236.001.0001.

Vlaardingerbroek, Marinus T. and Jacques A. den Boer
1996 *Magnetic Resonance Imaging*, Springer, DOI: 10.1007/978-3-662-03258-9.

Zuckerman, Daniel M.
2010 *Statistical Physics of Biomolecules*, CRC Press, DOI: 10.1201/b18849.

Software

Abadi, Martin, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafa Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
2015 *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*, tensorflow.org.

Chollet, François et al.
2015 *Keras*, keras.io.

Fomin, V., J. Anmol, S. Desroziers, J. Kriss, and A. Tejani
2020 *High-level library to help with training neural networks in PyTorch*, github.com/pytorch/ignite.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell
2014 *Caffe: Convolutional Architecture for Fast Feature Embedding*.

Krekel, H., B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laugher, and F. Bruhin
2014 *pytest*, github.com/pytest-dev/pytest.

Landrum, Greg
2022 *RDKit: Open-source Cheminformatics Software*, rdkit.org.

McCorkindale, William, Carl Poelking, and Alpha A. Lee
2020 *Investigating 3D Atomic Environments for Enhanced QSAR*.

Meli, Rocco and Andrew McNutt
2022 *gninatorch*, version 0.0.2, DOI: 10.5281/zenodo.6943066, https://github.com/RMeli/gnina-torch.

Mobley, David L. and David Slochower
2017 *MobleyLab/benchmarksets: Version 1.2*, DOI: 10.5281/zenodo.839047.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala
2019 *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett.

Peixoto, Tiago P.
2014 *The graph-tool python library*, graph-tool.skewed.de, DOI: 10.6084/m9.figshare.1164194.

Schrödinger, LLC
2015 *The PyMOL Molecular Graphics System, Version 1.8*.