Computational Approaches to Molecular Fragment Elaboration

Thomas Hadfield

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2022

Acknowledgements

First, I'd like to thank my supervisor Charlotte, who has been the best mentor I could ever have asked for. Without her subtle blend of flattery, cajoling and threats, I'm quite sure I would never have completed the projects which comprise this thesis and I've benefitted hugely from her supervision and support. On a similar note, I'm grateful for the support of Garrett Morris and my industrial supervisors Torsten Schindler (Roche), Will Pitt and Lewis Vidler (UCB) and Kris Birchall and Andy Merritt (LifeArc) for providing me with extremely useful guidance on how drug discovery works in the real world.

Next, a huge thanks to the members of the Oxford Protein Informatics Group for their friendship over the last three years. To Fergus Imrie, for tolerating endless pestering while I was getting started and for helping me get to grips with RDKit and its mysteries. Thanks to Jack for browbeating me into improving the readability of my code, to Lucy, potentially the only person who managed to use said code, and to the rest of the small molecules gang for useful feedback on my work. Most of all, thanks to Sarah, Eve, Alissa and Lewis, first-rate office mates who have always been willing to collaboratively procrastinate, offer support and share departmental gossip; I will sorely miss G&T Fridays!

Outside of the world of Protein Informatics, I've felt extremely lucky to be a member of Exeter College MCR over the last five years. Thanks to Matt, Britt and Tim, great housemates during that surreal first COVID lockdown, to Andrew for helping me to push the boundaries of how many roast potatoes I could eat in a single sitting, and to Corinna for being a source of sage advice on being a grown up.

I'd also like to thank Oxford University Badminton Club; I've greatly appreciated the competitive stimulation, the sense of escape, and have made some great friendships that I know will go well beyond Oxford. Thanks to my doubles partner, Tom, for teaching me that defence is the best form of attack, and to Ben and Jun-An for all the *Yes Minister* binges, Wetherspoons trips, and rounds off the back tees that you have selflessly acquiesced to.

I'm grateful to everyone in Oxford University Golf Club for helping me get back into Golf (and Golf Lunches!) this year. Thanks especially to Babar, and everyone else at Formby - I'll cherish those memories forever.

Finally, I'd like to thank my family, Mum, Dad and Jack. Your love has kept me going when things have been tough and your support has meant everything to me; I'm incredibly lucky to have you in my life.

Abstract

Recent advances in machine learning have led to considerable interest in their application to drug discovery, in the development of accurate predictive algorithms which can rapidly predict whether a given compound is likely to bind to a target protein, and in generative algorithms that propose novel high affinity binders *in silico*.

In this thesis, we propose a deep generative model for fragment elaboration for use in fragment-to-lead campaigns. We consider the problem of generating elaborations which are highly similar to a ground truth elaboration, first via reinforcement learning, where we propose a novel curriculum-based reward function, and also via the imposition of physically meaningful pharmacophoric constraints. We describe the development of a simple-to-use web application, allowing users to generate molecules with no prior programming experience and without installing anything.

Next, we describe a novel method for extracting useful and interpretable information from a target protein and providing it to a deep generative model. By computing a fragment hotspot map of the protein, which describes regions of the binding pocket that are likely to make a disproportionate contribution to binding affinity, we derive a set of pharmacophoric constraints which can be passed to our deep generative model, helping it generate functional groups which are likely to interact with the protein.

Finally, we consider the problem of virtual screening. In addition to being able to accurately identify high affinity binders, it is desirable that virtual screening models are able to identify the functional groups most responsible for binding. Assessing the ability of models to accurately attribute binding to specific functional groups is challenging due to the difficulty in specifying a ground truth which model attributions can be compared to. To address this, we propose a novel synthetic data generation process with a deterministic binding rule. First, we show that a recently proposed deep learning-based model is better able to identify important functional groups than a fingerprint-based model. Second, we demonstrate that training models on datasets which exhibit ligand-specific bias degrades the ability of the model to identify important functional groups, underscoring the importance of curating high quality datasets for virtual screening.

Contents

List of Figures

List of Tables

List of Abbreviations

AUC Area Under Curve

CGVAE Constrained Graph Variational Autoencoder

CNN Convolutional Neural Network

CSD Cambridge Structural Database

DUD-E Directory of Useful Decoys - Enhanced

EGNN E(n)-Equivariant Graph Neural Networks

FBDD Fragment-Based Drug Discovery

FHM Fragment Hotspot Map

GRU Gated Recurrence Unit

HBA Hydrogen Bond Acceptor

HBD Hydrogen Bond Donor

HsNMT Human N-myristoyltransferase

HTS High Throughput Screening

LBVS Ligand-Based Virtual Screening

NMT N-myristoyltransferase

PAINS Pan-Assay Interference Compounds

PCA Principal Components Analysis

PDB Protein Data Bank

PLEC Protein-Ligand Extended Connectivity

PRC Precision-Recall Curve

PvNMT Plasmodium Vivax N-myristoyltransferase

QED Quantitative Estimate of Druglikeness

RF Random Forest

RL Reinforcement Learning

ROC Receiver Operating Characteristic

SBVS Structure-Based Virtual Screening

TNF Tumour Necrosis Factor

VAE Variational Autoencoder

ZINC ZINC Is Not Commercial

Chapter 1

Introduction

1.1 Preface

This chapter contains work described in the following publication: **Thomas E. Hadfield**, and Charlotte M. Deane (2022). AI in 3D Compound Design. *Current Opinions in Structural Biology*, 73. All work described in this chapter is my own unless explicitly stated otherwise.

The vast majority of therapeutics available before the 1950's were discovered by chance with the focused development of drugs to treat a specific condition being a comparatively recent advancement (Kirsch, 2020). Whilst scientific advances have made it comparatively easier to develop new medicines, with the FDA approving a total of 50 novel therapeutics for clinical use in 2021 (Mullard, 2022), the process of developing a novel drug remains extremely expensive and time consuming, with a recent study estimating that the median cost to develop a drug was $985 million and that, on average, it took 8.3 years to conduct the research and development needed before clinical trials (Wouters et al., 2020). There is therefore a clear and urgent need to reduce the time and cost required to develop new medicines; the prohibitive cost of drug discovery can often preclude healthcare companies from attempting to develop

treatments for rare diseases (Heemstra et al., 2009), whilst a wide range of licensed therapeutics are not widely available in developing countries due to their high cost (Kremer, 2002).

A potential approach which could allow scientists to develop novel drugs more quickly and cheaply is the incorporation of computational approaches into drug discovery pipelines (Ou-Yang et al., 2012). There has been substantial interest in developing computational methods for a variety of different tasks across the drug discovery pipeline, e.g. *in silico* prediction of molecular properties (Hansch et al., 1962), *de-novo* generation of molecules (Schneider et al., 2000) and automatic synthesis planning (Funatsu and Sasaki, 1988).

Enabled by advances in computational power, deep learning algorithms have achieved remarkable predictive performance on a wide range of different tasks (e.g. (Krizhevsky et al., 2012; Silver et al., 2017)), sparking considerable interest in their application to tasks within drug discovery. The recent release of AlphaFold (Jumper et al., 2021), a deep learning model which allows highly accurate prediction of a protein's structure from its sequence, has further increased the excitement around deep learning models, as it illustrates the potential for deep learning methods to approximate highly complex natural processes.

A significant number of papers have already sought to apply deep learning methods to problems within drug discovery. Convolutional neural networks (CNNs) (LeCun et al., 1998), which leverage convolutions to capture important spatial correlations in voxelised inputs, have been used to computationally predict whether a ligand will bind to a protein (e.g. (Ragoza et al., 2017)), whilst variational autoencoders (VAEs) (Kingma and Welling, 2013) have been used to generate novel molecules which are optimal with respect to a specific property (e.g. (Gómez-Bombarelli et al., 2018)). However, whilst these methods show promise, they are yet to be fully integrated into the drug discovery toolbox; the relative scarcity of protein-ligand binding data,

combined with the high dimensionality of a protein structure makes it challenging for models to learn to use structural information in their decision making, particularly when available training sets are susceptible to ligand-specific bias e.g. (Chen et al., 2019). Drug discovery therefore appears to be some way away from its own 'AlphaFold' moment, where an algorithm could consistently generate a highly potent, synthetically accessible drug without human intervention.

This thesis focuses on the incorporation of 3D structural information into deep learning models for drug discovery. In this chapter, we discuss the concept of protein-ligand binding and the drug discovery process, in particular fragment-based drug discovery. We then describe the different machine learning approaches that we apply in this thesis and review the strengths and limitations of existing machine learning-based approaches for drug discovery. Finally, we outline the aims and contributions of this thesis.

In Chapters 2 and 3, we consider the problem of training a generative model to sample from a specific region of chemical space, so that we can control the types of molecules that it generates. In Chapter 4, we outline the development of a web application which makes our generative models more accessible to practitioners with limited programming experience. In Chapter 5, we consider the problem of incorporating target specific information into a deep generative model for fragment-based drug discovery. Finally, as the success of a generative model is dependent on being able to rapidly screen the molecules generated, we also investigate the extent to which different virtual screening models are able to accurately determine the relative importance of different functional groups to protein-ligand binding in Chapter 6.

1.2 Drug Discovery

Proteins are an essential building block of life, and carry out a wide range of functions within the cells of living systems (Whitford, 2013). Created in the ribosome using information encoded in a cell's DNA, proteins are linear chains of amino acids which adopt a complex 3-dimensional shape as they emerge from the ribosome. The 3D structure of a protein is determined by the sequence of amino acids of which it is comprised, and the protein's function within the cell is determined by its structure (Johnson et al., 2002).

Often, a biological process requires the formation of a protein complex, meaning that two or more proteins must be bound together (Jones and Thornton, 1996). In such cases, the proteins involved will typically have complementary shapes, so that they form intermolecular interactions that bind the proteins. Whilst protein-protein binding is usually highly specific, if an alternative molecule is also able to bind to the binding pocket of a protein, it may disrupt the ability of that protein carry out its biological function.

Small molecule drugs are organic compounds with a molecular weight of $< 1000Da$ that are designed to bind with high affinity to a protein which contributes to a specific biological function. Compared to alternative therapeutics, such as antibodies, their small size allows them to cross the outer plasma membrane of the cell, meaning that they can target intracellular proteins such as kinases (Chhabra, 2021). In order for a protein and ligand to bind with high affinity, they must have a high degree of both shape and charge complementarity (Eaton et al., 1995); of the approximately 10^{60} 'drug-like' molecules (Reymond, 2015), only a tiny proportion would have an appropriate shape and charge profile for a given protein target. However, in addition to requiring a small molecule to recognise the target protein, we also require that the molecule does not interrupt the function of other essential proteins within the cell by binding to them as well; in this case, an overly promiscuous drug could induce

dangerous side effects when administered. It is therefore extremely difficult to design a novel therapeutic for a novel target without knowledge of the target's 3-dimensional structure.

1.2.1 The Drug Discovery Pipeline

The process of developing a drug to treat a disease typically begins by identifying the biological process which causes it. Scientists use an array of different techniques to assess which biological components are causing the disease: If the disease of interest is well studied then a literature search might be sufficient to determine the protein which causes the disease. For lesser known targets, scientists might be able to leverage the abundance of genomic data to identify the gene which is responsible for the disease through a genome-wide association study (GWAS), to identify genetic variants associated with a disease Shu et al. (2018). An alternative strategy is to use CRISPR-Cas gene editing to activate or inhibit genes, allowing scientists to monitor whether specific proteins cause a disease (Knott and Doudna, 2018). Following the identification and validation of a target, scientists will attempt to identify 'hits', molecules which can bind to the protein of interest and facilitate a desired change in their activity.

A number of approaches for hit-identification exist. A widely-used approach is High-Throughput Screening (HTS), where a large of library of compounds is screened against the target of interest. HTS libraries are not specifically designed for a single drug discovery campaign, but are designed using the rationale that their size (up to hundreds of thousands of molecules (Follmann et al., 2019)) and diversity will give them sufficient coverage of chemical space to identify hits for the majority of targets (Villar and Hansen, 2009). A range of different assays are used to conduct HTS campaigns, including biochemical assasys, which measure the ability of a compound to inhibit enzymatic activity *in-vitro*, binding assays, which measure the binding

Figure 1.1: Illustration of the different stages of developing a new medicine.

of a compound to a protein independent of their effect on protein function, and cell-based assays, which identify hits based on their ability to induce a particular phenotype (Blay et al., 2020). Whilst HTS libraries are not typically designed with a specific target in mind, Reynolds et al. (2012) reported the development of a kinase-specific HTS library which was designed based on currently known kinase inhibitor scaffolds and known kinase binding sites and argued that the results of previous HTS campaigns should be used for the development of subsequent screens against similar targets. Computational approaches for predicting whether a given ligand will bind to a target have been used to select candidates for high throughput screens and are described in Section 1.4.1.

A complementary approach is fragment-based drug discovery (FBDD), where instead of using a large library of drug-like molecules to identify hits, a smaller library of low molecular weight molecules (< 300 Da) is screened against a target. Compared to HTS, FBDD has the advantage that it allows more efficient coverage of chemical space (Erlanson et al., 2004), resulting in hit-rates that are 10-1000 times higher than HTS campaigns (Schuffenhauer et al., 2005). Whilst fragment hits may bind with low affinity, they typically exhibit high ligand efficiency, increasing their chances of having favourable absorption and permeability properties (Hopkins et al., 2004). Fragment-based approaches have seen increasing use in academia and pharma over the past 20 years, with over 40 compounds in clinical trials and 4 launched drugs

(Denis et al., 2021).

Once one or more hits have been identified, whether by a high throughput- or fragment screen or another means, attempts will be made to make modifications to the hits in order to optimise a range of properties, including binding affinity, selectivity and ADME-T. This process is known as Hit-to-Lead, and typically follows the design-make-test-analyse cycle (Plowright et al., 2012), where the performance of molecules designed in the previous round will inform the designs for the next round.

If the scientists have access to a structure of the protein-ligand complex, they can utilise so-called "structure-based" approaches to optimise the hit molecule. By inspecting the protein-ligand complex, the scientists can determine which ligand functional groups are likely to be engaged in intermolecular interactions with the protein. Techniques such as biosteric replacement, where a functional group is replaced with a different functional group that is capable of making the same type of interaction with the protein, can be used to optimise the individual interactions (Langdon et al., 2010). Similarly, scaffold hopping involves replacing the core of the molecule with a substructure of similar size that exhibits similar properties. Alternatively, one might wish to replace a functional group in order to pursue a different interaction (e.g. replace a hydrophobic functional group with one that is capable of making a hydrogen bond). Knowledge of the binding site also allows optimisation of the ligand's size, as a larger binding pocket may be able to accommodate a larger ligand.

If the structure of the protein-ligand complex is not available, scientists can use ligand-based approaches to optimise hits. Whilst it might not be possible to use information about the binding site to optimise specific functional groups, one could optimise general attributes of the ligand, for example reducing the calculated logP to be less than 5 (Lipinski et al., 1997) or improving the molecule's synthetic accessibility. One could also leverage information about other molecules known to bind to the target protein as a basis for suggesting modifications, e.g. by training a ligand-based virtual

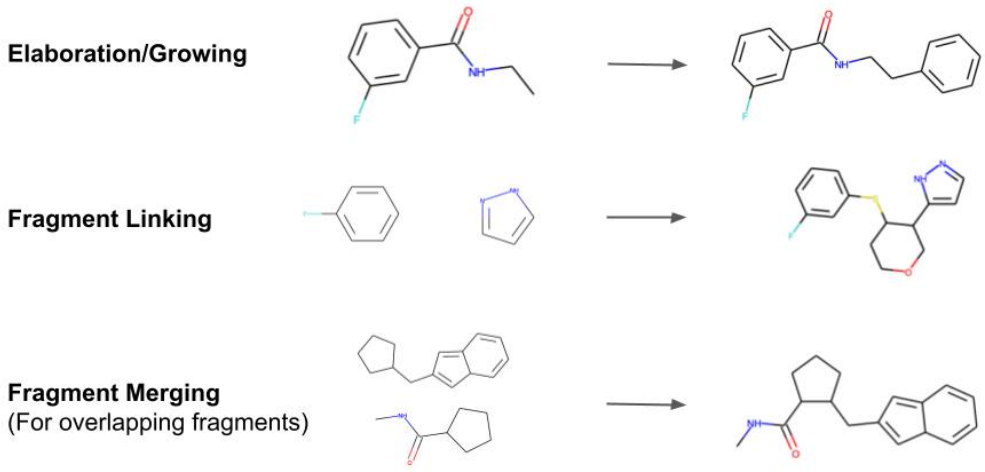

Figure 1.2: Different fragment-to-lead approaches. Scientists propose designs by accounting for the structure of the binding pocket and synthetic accessibility.

screening model (Section 1.4.1.2) and using it to identify promising modifications.

Fragment hits, which are usually smaller-than-drug-like, usually do not bind strongly enough to affect the target's function, and therefore require additional functional groups to be added to increase their potency. Three main approaches exist to achieve this: elaboration, linking and merging (Scott et al., 2012), illustrated in Figure 1.2. Each approach typically requires knowledge of the structure of the bound protein-fragment complex. Fragment elaboration takes a single bound fragment as input and adds functional groups whilst keeping the original fragment fixed. The addition of extra functional groups allows the resulting molecule to make further interactions with the target, increasing the binding affinity. Fragment linking involves constructing a molecular linker between two fragments that are concurrently bound in the same binding pocket; the linking process preserves the known interactions of the original fragments, offering a greater degree of certainty that the resulting molecule will bind with higher affinity than elaboration. Finally, fragment merging requires two or more fragments to bind in overlapping regions and involves the de-

sign of molecules which incorporate motifs from each fragment. Whilst linking and merging have the advantage of allowing researchers to use information about multiple fragment hits, elaboration remains applicable when only a single fragment is bound in a binding pocket, and information regarding other fragment hits can be incorporated into elaboration designs. As with HTS, it is theoretically possible to conduct a fragment-to-lead campaign without knowledge of the target's structure. However, as the size of the binding pocket is a critical factor in determining how many atoms can be added to a fragment, and the relative locations of different fragments inform whether they can be feasibly linked/merged, fragment-to-lead campaigns typically incorporate bound protein-fragment complexes in the decision making process.

Once a lead molecule, i.e. a compound which shows promise as a potential treatment for a disease, has been produced, it may undergo additional modifications to optimise relevant properties further, known as lead optimisation. A series of pre-clinical tests will then be used to the pharmacokinetic and pharmacodynamic properties of the resulting molecules, and to determine whether they are likely to be safe for human consumption (Kramer et al., 2007). Toxicity studies form a key part of the pre-clinical evaluation, where the compound is administered in escalating doses to animals. Toxicity testing encompasses a variety of different considerations, including mutagenicity testing, carcinogenicity testing and one-generation reproduction toxicity testing (Parasuraman, 2011). Such studies assist in the quanitifcation of a "No Observed Adverse Effect Level", which can be used as a basis for the calculating the staring dose in human trials. Whilst toxicology studies play a vital role in minimising drug-related adverse events to humans during the clinical trial process, Balls et al. (2019) reported that 20-25% of trials failing to progress from the first stage of clinical trials do so because of toxicology issues, emphasising the need for continued research in this area.

1.2.2 Clinical Trials

Before a therapeutic is licensed for use by humans, its safety and efficacy must be demonstrated through a sequence of clinical trials. The first clinical trial, known as Phase I, is used to establish whether a drug can be tolerated by humans, i.e. whether it can be administered without causing significant adverse events. Such trials are often performed open-label on small groups of healthy volunteers, whilst the treatment is administered using increasing doses to ascertain its maximum tolerable dose (Umscheid et al., 2011). A recent study reported that approximately 65% of medicines entered for Phase I trials are carried forward to the next round of trials (Wong et al., 2019).

If a drug is deemed to be tolerable then it may be progressed to the next stage of trials, known as Phase II. Phase II trials typically involve a larger number of participants than Phase I trials, and the participants normally have the disease of interest. They are used to allow investigators to better understand how the drug works in humans, in terms of its pharmacokinetic profile and dose response, and provides further evidence as to the drug's safety (Umscheid et al., 2011). The Phase II trials are also used to inform a number of aspects of a large-scale trial, including optimal dose size, administration routes, and endpoints (Umscheid et al., 2011). Approximately 58% of Phase II trials progress to the next round of trials.

Although Phase II trials seek to establish some evidence of efficacy, the small sample sizes used in such trials typically gives them limited statistical power, and a larger study is necessary to establish whether the drug can make a clinically meaningful impact on patients. Phase III trials usually involve between 300 and 3000 participants and aim to show that the drug outperforms either a placebo or the current standard of care (Umscheid et al., 2011). Following successful demonstration that the drug is safe and efficacious, it will be submitted to national regulatory agencies for approval. Wong et al. (2019) reported that approximately 14% of drugs commencing Phase I

trials between 2000-2015 successfully received approval.

Whilst the above pipeline represents a typical progression of a drug through clinical trials, the specific factors relating to a particular disease may necessitate a different approach; for example for a very rare disease, or where it was considered unethical to use a placebo as a control, it might be more appropriate to evaluate the performance of the drug against historical control data (Pocock, 1976). If the mechanism of action and safety profile of a drug are well understood (e.g. because it has already been licensed to treat another disease), then it is often possible to skip Phases I and II entirely (Food and Drug Administration, 2017).

1.3 Machine Learning

As described in Section 1.4, machine learning is used in drug discovery for rapidly generating molecules which are likely to bind to the target with high affinity, or prioritising such molecules from a large database. In this section, we describe several basic machine learning methods that are used throughout this thesis.

Machine learning concerns the automatic detection of patterns in data, and their application in decision making (Murphy, 2012). The development of such algorithms dates back to at least the late 19th century (Stanton, 2001), and machine learning can be broadly partitioned into three subfields: The first, *supervised learning*, generally concerns the prediction of a dependent variable conditional on one or more independent variables; The second *unsupervised learning*, focuses on dataset processing, where the aim is generally to understand the structure of a given dataset, e.g. by identifying redundant variables or clustering closely related examples. Finally, *reinforcement learning* concerns the training of an *Agent* to make decisions which are optimal with respect to a set of criteria. We consider each of these paradigms in detail below.

1.3.1 Supervised Learning

Let $D = \{(x_i, y_i)\}_{i=1}^n$ be a set of n pairs, where for each $i \in \{1, ..., n\}$, $x_i = [x_{i,1}, ..., x_{i,p}] \in \mathbb{R}^p$ and, for simplicity, $y_i \in \mathbb{R}$ (although y_i can also be vector-valued). We refer to y_i as a *label*, and each $x_{i,j}$ as a *feature*. In supervised learning, we are interested in fitting a function, $f : \mathbb{R}^p \to \mathbb{R}$, which takes the features as input and attempts to predict the labels as accurately as possible. Mathematically, we seek:

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(x_i)),$$

where $\mathcal{F}$ denotes a set of allowed functions and $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ denotes a *loss function*, which quantifies the accuracy of the prediction (a smaller loss represents a more accurate prediction). f^* is therefore the function which attains the smallest cumulative loss amongst all the functions in $\mathcal{F}$.

1.3.1.1 Linear Regression

To make the above concrete, we consider the simplest supervised learning algorithm, linear regression. In linear regression we predict the label as a weighted sum of the features, thus the space of permissible functions, $\mathcal{F}$, takes the form:

$$\mathcal{F} = \{f : \mathbb{R}^p \to \mathbb{R} | f(x) = \beta_0 + \beta_1 \times x_1 + ... + \beta_p \times x_p\}$$

To quantify the accuracy of a prediction, we use the *squared loss*, i.e. $L(y_i, f(x_i)) = (y_i - f(x_i))^2$, meaning that perfect predictions attain a loss of 0, and imperfect predictions attain a loss which is proportional to the squared distance between the true label and the prediction. Identifying the optimal linear regression function, f^*, therefore equates to finding the model parameters, $\beta_0, \beta_1, ..., \beta_p$, which attain the smallest total loss over the n examples. The optimal parameters can be obtained by computing

$$\frac{\partial}{\partial \beta_j}(\sum_{i=1}^{n}(y_i, f(x_i))^2)$$

for each $j = 0, ..., p$ and setting the resulting expressions to zero to attain the stationary points. Due to the convexity of the objective function with respect to the model parameters, linear regression has a unique, closed-form solution, which can be expressed as:

$$\hat{\beta} = (X^T X)^{-1} X^T y,$$

where $y = [y_1, ..., y_n]$ and $X = [x_1, x_2, ..., x_n]^T$, respectively.

1.3.1.2 Neural Networks

The simplicity of linear regression and the ease with which its parameters can be interpreted make it an attractive and widely used predictive model. However, in practice, we often wish to model processes which can not well approximated by a linear combination of dependent variables. Whilst we can incorporate non-linearities and interactions into a linear regression (e.g. by defining a new feature, say $x_{p+1} = x_1 \times x_2$ or $x_{p+1} = x_p^2$ and fitting the linear regression as described above), dataset augmentation strategies require explicit specification of the interactions and non-linearities which should be included in the model, which may be infeasible for complex, high-dimensional datasets.

Neural networks are a highly flexible class of predictive model which do not require the user to specify interactions or non-linearities in advance of fitting the model; instead, the model learns which interactions are important to accurately predicting the label whilst the model is being trained.

Following Bishop and Nasrabadi (2006), we construct the basic neural network as follows: First, for an input $x \in \mathbb{R}^p$, we construct M linear combinations of the input

$x = [x_1, ..., x_p]$ of the form:

$$a_j = \sum_{i=1}^{p} w_{ji}^{(1)} x_i + w_{j0}^{(1)},$$

where $j = 1, ..., M$. We refer to each $w_{ji}^{(1)}$ as a 'weight' and the $w_{j0}^{(1)}$ terms as 'biases'. Next, we apply a non-linear function to each weighted sum of the inputs, to obtain $z_j = h(a_j)$ for $j = 1, ..., M$. We call the z_j's 'hidden units', and they represent a 'new' dataset, where each feature is a non-linear function applied to a weighted sum of the original features. As each weight $w_{ji}^{(1)}$ is a learnable parameter which can be tuned to optimise the model fit, we can view of this first stage as a data processing step where the model generates a new feature set which will allow it to predict the label more accurately.

The next stage of the neural network model is to again compute a weighted sum:

$$a = \sum_{j=1}^{M} w_j^{(2)} z_j + w_0^{(2)} \tag{1.1}$$

where the weighted sum is taken over the hidden units computed in the previous stage. For regression problems, where our aim is to predict a continuous real-valued label, we typically return $\hat{y} = a$ as the model's output. If we consider equation 1.1, it bears a close resemblance to the linear regression equation described above, where the output is a weighted sum of the input features. As each z_j is a non-linear function of the original features, in a regression context we can therefore view a neural network as an extension of the linear regression model, where interactions between features and non-linearities are learnt by the model instead of being explicitly specified in advance. Neural networks for classification work in much the same way but we generally require the imposition of a non-linear function, $\sigma : \mathbb{R} \to [0, 1]$, to ensure that the model predictions lie in the range $[0, 1]$. More detail on the basic neural network model for classification can be found in Bishop and Nasrabadi (2006), Section 5.1.

Fitting a simple neural network for regression therefore equates to solving the following equation:

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(x_i)),$$

where L is again the squared loss $L(y_i, f(x_i)) = (y_i - f(x_i))^2$, and

$$\mathcal{F} = \{f : \mathbb{R}^p \to \mathbb{R} | f(x) = \sum_{i=1}^{M} w_j^{(2)} \times h(\sum_{i=1}^{p} w_{ji}^{(1)} x_i + w_{j0}) + w_0^{(2)}\}$$

Unlike linear regression, which has a unique closed form solution, the non-linear activation function makes the objective function challenging to optimise and we must resort to a numerical method, such as stochastic gradient descent, to optimise the objective function. The backpropagation algorithm allows efficient computation of the partial derivatives of the objective function and is described in Bishop and Nasrabadi (2006) Section 5.3.

The hidden units z_j together comprise a "hidden layer" of a neural network, where the inputs are transformed to better allow them to approximate the label. An easy extension of the simple neural network is the addition of further hidden layers, where the first hidden layer is transformed as follows:

$$z_j^{(2)} = h(\sum_{k=1}^{M} w_{ji}^{(2)} z_k^{(1)} + w_{j0}^{(2)}),$$

where $z_k^{(1)}$ are the hidden units in the first hidden layer. Whilst the specification of additional hidden layers can allow neural networks to learn more complex decision rules, they also bring an increased computational cost due to more parameters being added to the model and increase the risk of the model overfitting to the training data.

1.3.1.3 Graph Neural Networks

Whilst neural networks have proven adept functional approximators, the classical architecture described in the previous section, which treated each feature as an independent input, fails to take advantage of known correlations between features which can be useful in making more accurate predictions. Graphs can be used to model a wide variety of systems, including bodies of text (Malekzadeh et al., 2021), protein-protein interaction networks (Pellegrini et al., 2004) and molecules (Kearnes et al., 2016), prompting significant interest in developing neural networks that can naturally exploit the structure of graphs to improve their predictive accuracy (see Zhou et al. (2020) for a recent review). Here we describe the gated graph neural network model (GGNN) (Li et al., 2015), which propagates information between the nodes of an input graph and is used in chapters 2, 3 and 4 of this thesis.

Gated Graph Neural Network. Graph neural networks typically have two distinct phases. The first phase is an *information sharing* phase, where the information contained in a node is combined with the information contained in its neighbours' nodes. The second phase is the *readout phase*, where the updated node representations are used to make predictions. For each node, v, the initial hidden state, $h_v^{(1)}$, is set to be $h_v^{(1)} = x_v$, the attribute of the node. Next, for each node, at time step t, a *message* is computed from each of its neighbours, w:

$$a_{v,w}^{(t)} = A_{v,w}^T \times (h_w^{t-1})^T + b \tag{1.2}$$

Where $A_{v,w}$ is a learned matrix that depends on the edge label of the edge between v and w, and b is a learned vector. The vector $a_{v,w}^{(t)}$ represents the information from the neighbouring node which will be included in the next update, as:

$$h_v^{(t)} = \text{GRU}(h_v^{(t-1)}, a_v^{(t)}) \tag{1.3}$$

Where $a_v^{(t)} = \{a_{v,w}^{(t)} | w \in N(v)\}$, and GRU is the Gated Recurrent Unit proposed by Cho et al. (2014), which uses an update gate and a reset gate to better retain information that was provided to the model in previous steps.

Once equations 1.2 and 1.3 have been applied T times, the information sharing stage terminates, and the prediction can be obtained as $\hat{y} = f(\{h_v^T | v \in G\})$, i.e. by using the final hidden states of each node as an input to a neural network.

1.3.2 Generative Modelling

Given an input, $x \in X$, and target variable, $y \in Y$, the goal of supervised learning is to learn a function, $f : X \to Y$, such that $f(x)$ is close to y. This allows accurate prediction of the target based on the input features. Central to this thesis are generative models, which, given samples $x_1, ..., x_n \in X$, attempt to generate new samples $x_{n+1}, ..., x_{n+m}$ which could plausibly belong to the set X. For example, given a set of images of human faces, generate a set of images which contain new faces which do not belong to real people (Ranjan et al., 2018); or, given a set of texts, generate a piece of writing on a specified subject (Brown et al., 2020). In the context of *de-novo* molecular design, generative models have the potential to rapidly propose novel, drug-like molecules which are designed to be active against a specific target; in this section we outline a number of different generative model architectures and describe recent work in generative *de-novo* molecular design in Section 1.4.2.

Recurrent neural networks (RNN) are a widely used tool for sequential data. Whilst a typical neural network (Section 1.3.1.2) requires an input with a fixed number of features, sequential data, e.g. a string of text, might have a variable number of measurements across different examples, and it would be inconvenient to train a separate model for each input length. For a given sequence $x^{(1)}, x^{(2)}, ..., x^{(\tau)}$ we seek to accurately predict $x^{(t)}$ given $x^{(1)}, ..., x^{(t-1)}$ by learning a conditional probability distribution $p(x^{(t)} | x^{(1)}, ..., x^{(t-1)}, \theta)$, where θ is a parameter vector. However, this

formulation fails to avoid the problem of being applicable to sequential inputs of different lengths, as we are required to input $t-1$ conditioning variables. Instead, RNNs utilise a *hidden state*, $h^{(t-1)}$, which summarises all the information contained in the features $x^{(1)}, ..., x^{(t-1)}$ in a fixed-length vector and uses it to predict $p(x^{(t)}|h^{(t-1)}, \theta)$. As $h^{(t-1)}$ retains the same dimension irrespective of the value of t, we can use the same learned function for any input sequence. A simple RNN takes the following form: First we initialise the hidden state $h^{(0)}$, then compute

$$\begin{aligned} a^{(t)} &= b + W \times h^{(t-1)} \\ h^{(t)} &= tanh(a^{(t)}) \\ o^{(t)} &= c + V \times h^{(t)} \\ \hat{x}^{(t)} &= softmax(o^{(t)}) \end{aligned} \tag{1.4}$$

Where b and c are learned parameter vectors, V and W are learned parameter matrices, and $\hat{x}^{(t)}$ is the outputted probability distribution over the possible values that $x^{(t)}$ can take. Once the model has been trained, one can sample new sequences iteratively, attaining the value $x^{(t)}$ by sampling from the categorical distribution $\hat{x}^{(t)}$. A detailed overview of RNNs is given by Goodfellow et al. (2016).

A more general framework for generative modelling is the autoencoder. An autoencoder comprises two functions $\phi : X \to Z, \psi : Z \to X$, such that $||x - \psi(\phi(x))||$ is minimised. Whilst this task initially appears to have little value, if we specify that the dimensionality of Z must be smaller than the dimensionality of X, the function ϕ can be viewed as a means of compressing elements of X; in order for ϕ to be able to accurately reconstruct the input value, $\phi(x)$ must retain some of the information encoded in x. We refer to ϕ as an *encoder*, ψ as a *decoder*. Most modern autoencoders parameterise ϕ and ψ as deep neural networks; their ability to accurately approximate highly non-linear functions allow them to more accurately recreate the objective and encode a more informative compression of the data within the latent space, Z.

The widely known dimensionality reduction tool, principal components analysis (PCA) (Abdi and Williams, 2010), is a simple example of an autoencoder. Other autoencoders can similarly be used for dimensionality reduction/feature extraction (e.g. (Meng et al., 2017)), as the compressed representation encoded in the latent space is designed to encode as much information in the smaller space as possible.

There has been considerable recent interest in the use of autoencoders for synthetic data generation (e.g. (Ranjan et al., 2018; Wan et al., 2017)). The decoder ψ provides a map between a low-dimension space, Z and the original space, X, allowing one to sample from X by sampling $z \in Z$ and computing $\psi(z)$. Variational autoencoders (Kingma and Welling, 2013) include a Kullback-leibler divergence (Joyce, 2011) term in the autoencoder loss function which incentivises the distribution of points in latent space to resemble an isotropic Gaussian; this increases the likelihood that a sampled point $z \sim N(\mathbf{0}, \mathbf{I})$ will correspond to a realistic output.

An alternative approach to the VAE is a Generative Adversarial Network (GAN) (Goodfellow et al., 2020), which also simultaneously trains two neural networks, a *generator* and a *discriminator*. Given a set of training examples $x_1, ..., x_n \in X$ and random inputs $z_1, ..., z_n$, the generator $\xi : Z \to X$ generates examples $\hat{x}_1 = \xi(z_1), ..., \hat{x}_n = \xi(z_n)$. The discriminator $\kappa : X \to [0, 1]$ then attempts to determine whether each example $x_1, ..., x_n, \hat{x}_1, ..., \hat{x}_n$ is a genuine example or a generated one. The discriminator training objective rewards the discriminator for accurately separating the genuine examples from the generated ones, whereas the generator training objective rewards the generator if the discriminator is unable to accurately separate the genuine and generated examples. The two models are therefore trained in an adversarial fashion, forcing the generator to generate increasingly realistic examples in order to stop the discriminator from attaining strong accuracy. GANs have demonstrated excellent performance in the generation of image data (e.g (Han et al., 2018)), however in some instances have proven extremely different to train to give sensible

output (Nie and Patel, 2020).

1.3.3 Reinforcement Learning

Reinforcement learning concerns the training of an *agent* to take actions which are optimal with respect to a pre-specified reward function. It has been used for a variety of tasks which require the user to take sequential decisions in a high dimensional space, for example self-driving cars (Cao et al., 2021), heating and cooling systems (Jeen et al., 2022) and playing computer games (Mnih et al., 2013). Compared to supervised learning models, which are trained to make predictions based on a series of examples which are provided to it by the user, reinforcement learning models are only provided with a reward function and a set of rules governing the actions they are allowed to take. As such, reinforcement algorithms are not constrained in their learning by a limited training set and have been shown in some cases to learn to take optimal actions that human experts would not have considered (Silver et al., 2017).

Concretely, suppose that we begin at an initial state, $s_0 \in S$, where S is the set of all possible states. We must take an action, $a_1 \in A$, where A is a set of permissible actions, which leads us to a new state, s_1. After moving to the new state, we are given a reward $R_1 \in [0, \infty)$, which quantifies how favourable the move was. We continue this process for T steps, each time choosing an action a_t which leads us to a new state, s_t, producing a reward R_t. Starting from time t, we compute the total reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Where $\gamma \in [0, 1]$ represents a discounting factor which downweights rewards that are further into the future. The total reward G_t quantifies how optimal the actions taken were.

The process described above is often framed as a Markov Decision Process (Alagoz

et al., 2010), where the current state value contains all the required information to make a decision regarding which state to go to next. In such cases, reinforcement learning requires us to learn a conditional distribution $\pi_\theta(a|s)$, parameterised by θ, such that sampling our actions at each state from π_θ maximises the expected reward. We call such a distribution a *policy*, and we say the policy is deterministic if we always take the most probable action, given a state, or stochastic if we sample an action from $\pi_\theta(\cdot|s)$.

Following Weng (2018a) and Weng (2018b), we present a brief overview of policy gradient algorithms and the REINFORCE algorithm which we use in Chapter 2. First, we define the value function as the conditional expectation of the reward, given an initial state, $V(s) = \mathbb{E}_\pi[G_t|S_t = s]$, and the "Q-value" of a state-action pair as $Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. In both cases the expectation is dependent on the policy π_θ.

Policy gradient algorithms aim to determine the parameter vector θ which maximises

$$J(\theta) = \mathbb{E}_{\pi_\theta}[V(S)] = \sum_{s \in S} d_{\pi_\theta}(s) V_{\pi_\theta}(s) \tag{1.5}$$

where $d_{\pi_\theta}(s)$ is the stationary distribution of s under π. Simply, the policy gradient attempts to maximise the value function, averaged over all possible starting states. From the definition of the Q-value, we can rewrite equation 1.5 as:

$$J(\theta) = \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(a|s) Q_\pi(s, a) \tag{1.6}$$

We use a gradient ascent scheme to optimise $J(\theta)$, so we must obtain derivative of the objective with respect to θ: :

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(a|s) Q_\pi(s, a) \tag{1.7}$$

This derivative is computationally challenging, but we can use the policy gradient theorem (section 13.2 of Sutton and Barto (2018)) to remove the dependence of the derivative on d_{π_θ}:

$$\nabla_\theta J(\theta) \propto \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} \nabla_\theta(\pi_\theta(a|s)) Q_\pi(s,a) \tag{1.8}$$

Multiplying and dividing by $\pi_\theta(a|s)$, and using the fact that $\frac{d}{dx}(\log(f(x))) = \frac{df/dx}{f(x)}$, we get:

$$\nabla_\theta J(\theta) \propto \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(a|s) \nabla_\theta(\log(\pi_\theta(a|s))) Q_\pi(s,a) \tag{1.9}$$

This takes the form of an expectation:

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{s \sim d_\pi, a \sim \pi_\theta}[\nabla_\theta(\log(\pi_\theta(a|s))) G_t] \tag{1.10}$$

Note that we have exchanged $Q_\pi(s,a)$ for G_t, by the law of total expectation (i.e. for random variables $X, Y, \mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$. This process of finding the optimal policy is known as a REINFORCE algorithm, which we outline in Algorithm 1.

Algorithm 1 REINFORCE Algorithm for optimising a reward function

Require: $\alpha > 0$
  Initialize:
    θ, parameter vector
  for n = 1 to N **do**
    Generate trajectory, $s_0, a_1, s_1, ..., a_T, s_T$ using current π_θ
    for t = 1 to T **do**
      Calculate reward G_t
      Update policy parameters: $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta(a_t|s_{t-1})$
    end for
  end for

A variety of other policy gradient algorithms exist in addition to REINFORCE. For a brief overview the reader is referred to Weng (2018b).

1.4 Computer Aided Drug Discovery

Traditional hit-finding campaigns require experimental testing of a large number of compounds. This process is expensive and time consuming and there has been a significant, long standing interest in developing computational methods which can speed up this process. Of particular interest is the computational prediction of whether a molecule will bind to a target, the automatic generation of libraries of molecules which are likely to be active against a target, and the combination of these two approaches to realise a computational analogue of the design-make-test-analyse cycle. In this section we describe several existing approaches for virtual screening and molecule generation and the use of structural information in computational techniques.

1.4.1 Virtual Screening

Virtual screening offers an alternative to manually screening libraries of molecules for hits in a laboratory; the very low hit-rates associated with such screens typically require the use of extremely large libraries, and it is not uncommon for the cost of a high throughput screens to be hundreds of thousands of dollars (Dreiman et al., 2021). Whilst experimental validation still remains the gold standard for assessing whether or not a ligand will bind to a protein, virtual screening methods have played a useful role in a number of drug discovery campaigns in discarding molecules which are unlikely to bind and prioritising a smaller number for laboratory testing (e.g. (Gorgulla et al., 2020; Lyu et al., 2019; Stein et al., 2020)), dramatically reducing the cost required to identify hit molecules. Virtual screening models can either be classification models (classifying a molecule as a binder or non-binder) or regression models, which predict the binding affinity of a given molecule. As well as identifying a set of binders from a large library, virtual screening models can be used to help understand the quantitative structure-activity relationship (QSAR), allowing researchers to identify motifs which

contribute towards improved binding affinity; such insights can inform subsequent designs.

1.4.1.1 Molecular Similarity

One of the fundamental principles of drug discovery is that similar molecules are likely to have similar properties. As such, when assessing molecules with a computer, it is essential that we are able to quantitatively express the level of similarity between two molecules. Scientists have used molecular descriptors to summarise the properties of molecules for decades (Brugger et al., 1976), and a wide number of descriptors exist. The simplest set of molecular descriptors summarise properties of the whole molecule, such as its heavy atom count or logP. However, two molecules with equivalent heavy atom counts could be drastically different, making such statistics unsuitable for fully assessing molecular similarity.

An alternative type of molecular descriptor is a 2D molecular fingerprint, which fragments the query molecule and encodes information regarding the different fragments that are present in the molecule (see Duan et al. (2010) for a review of different 2D fingerprint schemes). A popular 2D fingerprint is the Morgan fingerprint (Rogers and Hahn, 2010), which grows a set of fragments by radially expanding from each heavy atom in the molecule, and mapping each distinct fragment to a different integer. The similarity of two molecules can then be assessed by comparing their respective fingerprints, e.g. by computing the Tanimoto similarity between them. Molecular fingerprints have the advantage of being quick to compute and they can therefore be used to rapidly screen large libraries of molecules.

A disadvantage of comparing two molecules based on whether they share a set of molecular substructures is that structurally distinct molecules are often capable of making the same interaction with a protein. For example, a pyridine fragment and carbonyl fragment would both be capable of facilitating a hydrogen bond with a

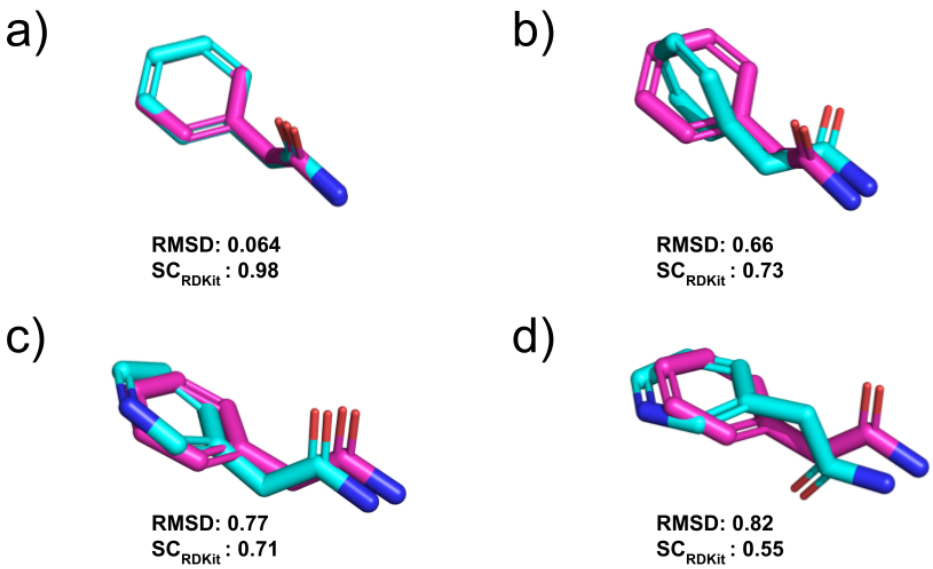

Figure 1.3: RMSD and SC_{RDKit} scores attained for different molecules and conformations. a) and b) illustrate a comparison of two identical molecules with different conformations; as the RMSD increases, the SC_{RDKit} score moves further away from 1. In c) and d) we compare two different molecules, where the benzene substructure has been replace with a pyridine.

protein donor atom, but they would be considered dissimilar by a fingerprint-based similarity metric. The SC_{RDKit} score (Figure 1.3) used by Imrie et al. (2020) assesses molecules based on their ability to make the same interactions by combining two existing 3D similarity functions. The first, proposed by Putta et al. (2005) quantifies the volumetric overlap between two 3D molecules, whilst the second, proposed by Landrum et al. (2006), takes each pharmacophore in a reference molecule and calculates the distance between the pharmacophore and the nearest pharmacophore of the same type in a query molecule. To obtain a high SC_{RDKit} score, two molecules must therefore be able to occupy a similar 3-dimensional space and be able to make the same intermolecular interactions. SC_{RDKit} score is sensitive to the input poses of the query and reference molecules; two identical molecules which had different conformations could obtain a very low SC_{RDKit} score if their conformations were sufficiently different. One way to mitigate this problem is to sample multiple poses for the input

molecules and take the highest obtained SC_{RDKit} score, although such an approach increases the computational complexity of the method.

1.4.1.2 Ligand-Based Virtual Screening

Ligand-based virtual screening (LBVS) depends on the principle that similar molecules will exhibit similar activity towards the same target and therefore, if a set of known binders already exists, they can be used to predict whether or not other molecules are likely to bind to the target. Compared to structure-based approaches, ligand-based approaches do not require a 3D structure of the protein-ligand complex and models can be constructed using assay data which is quicker and cheaper to obtain than experimental structure data but doesn't provide any insights as to the binding mechanism of the ligand.

The simplest ligand-based approach is a similarity search, which identifies molecules which attain an above-threshold 2D- or 3D similarity score with an existing active. Similarity search methods are most applicable for cases where only a small number of active ligands are known (Muegge and Mukherjee, 2016), where training a machine learning model would not be possible. Whilst recent years have seen machine learning models proposed for LBVS (e.g (Ramsundar et al., 2015; Wu et al., 2018)) their reliance on an existing database of known actives precludes their application to novel targets, which will have few or no known actives. Moreover, if the dataset of known actives is not sufficiently diverse, LBVS can lead to a high false negative rate if true actives are not sufficiently similar to the known actives used.

1.4.1.3 Structure-Based Virtual Screening

An alternative to LBVS is structure-based virtual screening (SBVS), which leverages information about the target structure in order to make predictions. Molecular docking tools (e.g. (Verdonk et al., 2003; Friesner et al., 2004; Morris et al., 2009;

Trott and Olson, 2010)) utilise a two-step process for predicting the binding affinity of a protein-ligand complex: First, a 3D ligand pose is generated and placed within the binding site, followed by an estimation of the binding affinity attained by the generated protein-ligand complex with a scoring function. In practice, for a given protein-ligand pair, one usually generates a set of protein-ligand complexes, and the docking function will attempt to generate poses which maximise the scoring function (for example by use of a genetic algorithm (Morris et al., 1998)).

Early scoring functions typically used a linear regression model to predict the binding affinity, where the linear regression comprised a set of hand-crafted features representing different intermolecular interactions and was trained on real-world protein-ligand binding data. However, there has been substantial recent interest in the creation of machine-learning based scoring functions; their ability ability to learn complex relationships from data without explicit parametric assumptions makes them an attractive alternative to classical scoring functions.

Early applications of ML algorithms (e.g. (Ballester and Mitchell, 2010; Durrant and McCammon, 2011)) took a protein-ligand interaction fingerprint as input and used an ML algorithm to predict its binding affinity or classify the input complex as a binder or non-binder. Such an approach has the advantage that a random forest (Breiman, 2001) or neural network would be able to learn complex interactions between features without explicit pre-specification, in contrast to a linear/logistic regression scoring function. However, fingerprint-based ML models are still dependent on a human-specified featurisation of the protein-ligand complex, inducing the risk that important information may be inadvertantly omitted from the featurisation. Whilst the model proposed by Ballester and Mitchell (2010) outperformed classical scoring functions on a multi-target evaluation, subsequent studies raised concerns surrounding their lack of sensitivity to changes in pose and their ability to generalise to new targets.

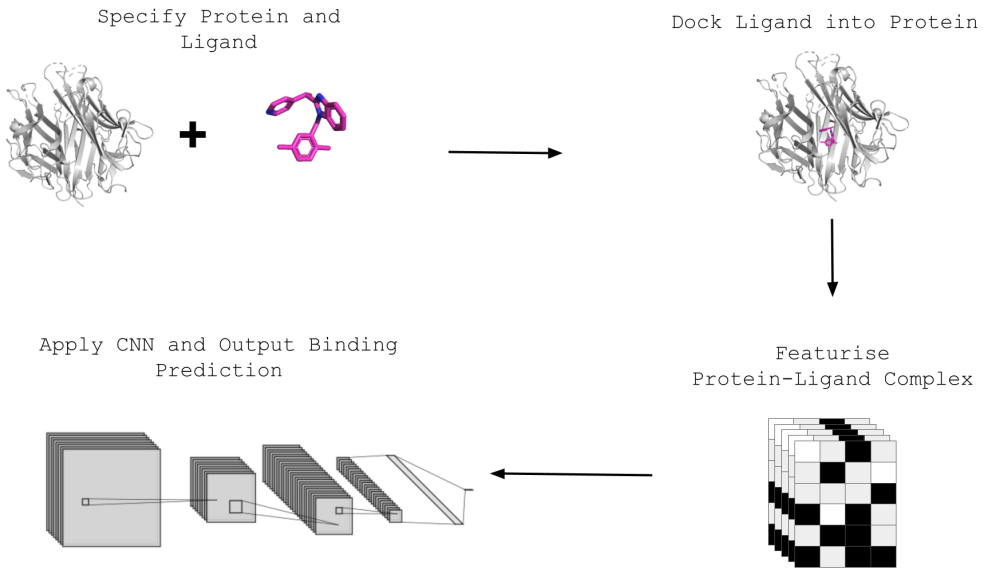

Figure 1.4: Schematic of protein-ligand scoring using Convolutional Neural Networks. A ligand is docked into the protein of interest, featurised and passed to a CNN. The CNN captures important spatial information and predicts whether the ligand is likely to bind to the protein in the pose provided to the model.

Inspired by the ability of convolutional neural networks (CNNs) to capture important spatial information on image recognition tasks, several authors sought to predict protein-ligand binding by using a docked protein-ligand complex as the input to a CNN (e.g. (Ragoza et al., 2017; Imrie et al., 2018)). Compared to existing fingerprint-based virtual screening models, it was hypothesized that CNN-based models, illustrated in Figure 1.4, would be better able to predict binding affinity in a similar way to how a expert might, by identifying ligand pharmacophores which have the potential to form interactions with protein residues. Pereira et al. (2016) computed a set of features (distances, partial atomic charges etc.) for each atom and employed a CNN to capture the spatial information in these features and classify molecules as actives or decoys. Three recent methods have utilised voxelised representations of the protein and ligand, with each voxel containing information regarding the presence or absence of different atom types (Ragoza et al., 2017; Jiménez et al., 2018; Imrie et al., 2018).

Two of these methods, (Ragoza et al., 2017; Imrie et al., 2018), were validated

on the DUD-E (Mysinger et al., 2012) dataset, a virtual screening dataset which contains experimentally validated actives and decoys obtained by sampling from ZINC (Sterling and Irwin, 2015), and the MUV dataset, (Rohrer and Baumann, 2009), a PubChem-derived (Kim et al., 2021) dataset which uses nearest neighbour analysis to ensure that target-specific datasets are unbiased with regard to analogue bias and artificial enrichment. Both methods demonstrated superior predictive performance on a virtual screening task compared to Autodock Vina (Trott and Olson, 2010) and existing machine learning scoring functions when the test set was derived from the DUD-E set. However the models performed comparably with existing scoring functions when tested on the MUV (Rohrer and Baumann, 2009) dataset, suggesting that, like other methods, these CNN models struggled to generalise to novel targets.

A follow-up study (Chen et al., 2019) illustrated that the strong performance exhibited by models trained and tested on the DUD-E dataset may well be driven by hidden biases in DUD-E which had allowed the model to detect systematic differences between the actives and decoys, rather than any ability of such models to learn protein-ligand interactions from the data. In fact it was shown that if the protein structure was removed from the input to these models only minimal degradation in performance was observed (Chen et al., 2019; Scantlebury et al., 2020). To address this issue, Scantlebury et al. (2020) proposed a dataset augmentation technique where the set of decoys used to train the model was augmented by active ligands which had been assigned random conformations and randomly rotated and translated. Under this formulation, to discriminate between the true actives and the perturbed actives labelled as decoys, the model would be forced to consider local protein structure. The performance of the model proposed by Scantlebury et al. (2020) was degraded substantially by the omission of the protein structure, illustrating that this model was more dependent on protein structure for making predictions.

Whilst deep learning-based protein-ligand scoring models have shown promise in

identifying high affinity binders, they require a protein-ligand complex to be provided as input. When the provided complexes are generated by docking protocols, this can negatively affect the performance of the downstream scoring models. Boyles et al. (2021) showed that training and testing structure-based scoring functions on docked poses led to worse performance than training on crystal poses. This is to be expected, as Autodock Vina (Trott and Olson, 2010) was able to generate a pose with < 2 Å RMSD compared to the crystal pose in 78% of examples. As the docking algorithm was unable to recover the correct binding pose in over a fifth of examples, it would be more challenging for a virtual screening model to learn the underlying biophysical rules which govern protein-ligand binding when training the model, and could result in active molecules being assigned as inactive due to an inaccurate input pose. Recent work (Stärk et al., 2022) has explored the use of generating protein-ligand complexes using deep learning, exhibiting competitive performance against classical docking algorithms.

An alternative to molecular docking for structure-based virtual screening is to use molecular dynamics (MD) simulations to estimate the free energy of binding. Whilst MD simulations require more precise preparation of the protein than molecular docking, they are typically more robust to the precise positions of the protein residues, as these can be relaxed during sampling (Cournia et al., 2017). As a result, unlike docking, MD simulations have the ability to identify when a residue side-chain might be likely to move to accommodate the ligand, allowing a more realistic depiction of the binding mode. In contrast to estimating the binding affinity of a protein-ligand complex as a weighted sum of interaction terms, one can use MD simulations to estimate the binding affinity as the difference in free energy between the bound state and the unbound state. Whilst computing the binding free energy from MD simulations can often give accurate estimations of the binding affinity, the computational expense of approximating the conformational distributions means that it is generally

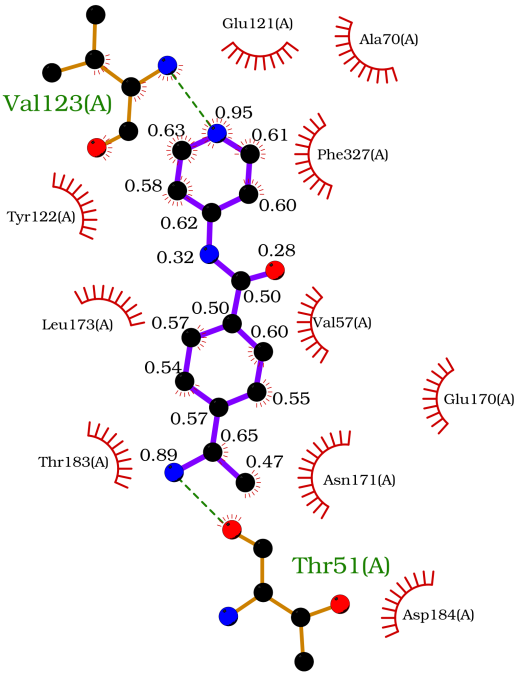

Figure 1.5: Example visualisation of attribution in protein-ligand scoring for cAMP-dependent Protein Kinase A in complex with ligand Y27 (PDB ID: 1Q8T), generated using LIGPLOT (Wallace et al., 1995). Atoms with scores close to 1 are considered to make an important contribution to the ligand binding to the protein; conversely, atoms with a score close to 0 are considered to reduce the probability of binding.

computational infeasible with very large libraries of ligands. In practice, one might compute the binding free energy of a small number of shortlisted ligands identified via docking, or compute the relative binding affinities of a series of ligands proposed in a lead optimisation campaign (Gumbart et al., 2013).

A key step in building greater trust in structure-based virtual screening methods is the development of methods which allow users to understand which features of a ligand meant that the model predicted it would bind (Figure 1.5 illustrates an example protein-ligand complex which has been assigned such attributions). Hochuli et al. (2018) proposed several methods for visualising the influence specific atoms had on a CNN model's classification, for example by predicting the binding proba-

bility using the whole protein-ligand complex and using the whole complex with one atom removed; the difference in the two scores is then considered to be the 'contribution' of that atom to the model's binding decision. Brown et al. (2021) proposed a method which takes a congeneric ligand series as input and predicts which substructures improve or degrade binding affinity relative to other ligands in the series, allowing easier iterative refinement. Additional methods for attribution have been proposed (McCloskey et al., 2019; Sundar and Colwell, 2020), although they do not consider the problem of protein-ligand binding and instead restrict their focus to the identification of pre-defined substructures within a molecule. Sanchez-Lengeling et al. (2020) recently proposed a set of metrics for the evaluation of attribution methods, with the aim of facilitating the development of more interpretable deep-learning models. The development of such methods is an important step in the further integration of protein-ligand scoring models into drug discovery campaigns, as existing models currently only predict whether a specific ligand will bind to a protein but give no insights as to which motifs play a key role in allowing binding to occur; accurate methods for attribution would more easily allow practitioners to retain important motifs and discard those which do not help the ligand to bind, and could be used to inform the kinds of molecules made by generative models.

1.4.1.4 Prediction of Synthetic Accessibility

Whilst it is vital that a potential drug is able to bind to the target protein with high affinity, it is equally important that any hit/lead molecules are synthetically accessible. Although different chemists might differ in their assessment of how challenging it would be to synthesise a given molecule, computational methods can help to discard unsuitable molecules and reduce the amount of manual inspection required of medicinal chemists. A range of different approaches have been proposed to quantify the synthetic accessibility of a query compound: Ertl and Schuffenhauer (2009) pro-

posed an approach which fragmented a molecule and computed the propensities of the constituent fragments in a large database of molecules, adopting the heuristic that fragments which appeared infrequently would be more challenging to synthesise. The final synthetic accessibility score was then calculated as a function of the fragment propensities and a complexity term for motifs that were known to be challenging (e.g. macrocycles). Coley et al. (2018) proposed a deep learning-based approach for predicting synthetic complexity (SCScore); starting with a database of known reactions, they argued that the product of any reaction should have a greater synthetic complexity (and therefore lower synthetic accessibility) than each of its constituent reactants. Their loss function rewarded the model for assigning a higher SCScore to a product than its reactants.

Orthogonal to the above approaches, which considered how challenging a molecule would be to synthesise, Sanchez-Garcia et al. (2022) proposed a model which predicted how expensive a molecule would be to purchase from an online catalogue. Their graph neural network exhibited substantially better correlation with the true compound price compared with existing synthetic accessibility models, making it a more appropriate tool for screening campaigns where the shortlisted molecules were to be purchased rather than synthesised in-house.

In addition to quantifying how synthetically accessible a molecule is, a scientist also needs to know *how* a molecule can be synthesised. Segler et al. (2018) proposed a Monte-Carlo Tree Search-based approach which took a molecule as input and returned a list of purchasable precursors. Each node in the search tree corresponds to a set of molecules derived from the original input molecule, and at each iteration the most promising node is selected. A neural network policy is used to select reaction templates for the molecules in the selected node, expanding the tree until a terminal state is reached. The starting leaf node is then assigned a score, which is backpropagated to the root node, and the next iteration commences. An open-source version of

the algorithm proposed by Segler et al. (2018) was released by Genheden et al. (2020), allowing users to rapidly automate synthesis planning for shortlisted molecules.

1.4.2 Generative Design

Given a sufficiently accurate model, and enough computational resource, it would in theory be possible to predict the binding affinity of every drug-like molecule for a given target and select the n molecules with the highest predicted affinity for experimental testing. However, in practice the size of chemical space makes such a screen computationally infeasible and as virtual screening models are unable to identify binders with perfect accuracy, even a very low false positive rate would lead to extremely low precision.

An alternative computational approach is therefore to attempt to sample from a subset of chemical space where the likelihood of sampling a binder is greatly improved. To achieve this, a large number of studies have proposed deep generative models for *de-novo* molecule design. As with virtual screening models, a range of different molecular representations have been applied as inputs. However, unlike virtual screening models, the output of a generative model must uniquely correspond to a molecule; it would not be sufficient, for example, to generate a molecular formula (e.g. $C_6H_{12}O_6$), as there many distinct molecules with an identical molecular formula. The most commonly adopted molecular representations in generative design are SMILES strings (Weininger, 1988) (Gómez-Bombarelli et al., 2018), (Olivecrona et al., 2017), which represent molecules as a string of characters and provide information about the connectivity of each atom and the type of each bond, and graphs (e.g. Liu et al. (2018), Jin et al. (2018)), which specify which atoms are connected and the bond type of each atom through an adjacency matrix. Whilst the SMILES representation allows scientists to take advantage of recent research in Natural Language Processing, they describe the 2D structure of a molecule, whereas a graph can easily be extended

to a 3-dimensional representation by providing each atom's coordinates to the graph. A recent benchmarking exercise conducted by Brown et al. (2019) compared the performance of a number of different generative models on a series of different 2D tasks, and found that SMILES-based and graph-based generative models achieved broadly similar performance.

Whilst the long term goal of generative design is to be able to generate a series of high affinity binders for a specific protein target, initial work in deep generative design focused on simpler tasks in order to generate the potential of deep generative models. Many early generative models (e.g. (Gómez-Bombarelli et al., 2018; Jin et al., 2018)) trained a variational autoencoder (Kingma and Welling, 2013) using a database of drug-like molecules (Sterling and Irwin, 2015) and employed gradient ascent or Bayesian Optimisation (Frazier, 2018) to identify regions of the VAE latent space which corresponded to molecules which were optimal with respect to a simple molecular property, such as the Quantitative Estimate of Druglikeness (QED) (Bickerton et al., 2012) or logP. Olivecrona et al. (2017) took a step closer to a model applicable to real-world drug discover projects by proposing a SMILES-based generative model and demonstrating that reinforcement learning allowed them to consistently generate molecules which were predicted to be active against DRD2 by a LBVS model. A similar approach was employed by Jeon and Kim (2020), who used reinforcement learning to optimise a docking score (Handoko et al., 2012), allowing it to be applied to targets without large numbers of existing actives.

An alternative approach to those described above is to create a conditional model which provides information about the protein's structure to the model when it is generating molecules (e.g. (Skalic et al., 2019b; Ragoza et al., 2022; Peng et al., 2022)). Such models are trained on existing protein-ligand complexes and, compared to learning to optimise a potentially noisy reward function via reinforcement learning, have the advantage that the model is able to learn important intermolecular interac-

tions directly from the data. A disadvantage of a conditional generative approach is that instead of using a dataset of drug-like molecules to train the generative model, structure-aware models such as LiGANN (Skalic et al., 2019b) require protein-ligand complexes as input, there are substantially fewer available examples with which such models can be trained; for example, LiGANN was trained on a total of 11256 examples, which is considerably smaller than the 250k ligands used to train the generative model proposed by Gómez-Bombarelli et al. (2018). One would expect that a smaller training set would impair the ability of a model to sample from a broad chemical space, and that it would be difficult to attain optimal performance with a very high dimensional model trained on a limited number of examples.

To increase the number of examples available for model training, the model proposed by Peng et al. (2022) used a total of 22.5 million cross-docked examples to train their generative model. Whilst a larger number of training examples would likely allow a generative model to more accurately approximate the complex rules governing intermolecular interactions, care must be taken when using cross-docked structures as a basis for learning; docking software will still place a ligand in a binding site even if the ligand is unable to bind to the receptor in reality. Training a generative model on cross-docked examples therefore induces the risk of the model learning to approximate a docking algorithm's attempt to place a ligand into a binding site, rather than to learning to generate ligands which are capable of making high energy interactions with the protein. Moreover, as discussed previously, even if a ligand does bind with high affinity to a protein it does not guarantee that the docked pose will resemble the true binding mode.

The generative models discussed above all generate molecules 'from scratch', making them more analogous to a high throughput screen than to a fragment-to-lead campaign. A number of authors have proposed models which generate molecules in a constrained fashion, where all generated molecules contain one or more pre-specified

substructures. For example, Lim et al. (2020) and Li et al. (2019) proposed deep generative models for scaffold decoration which added atoms to a scaffold whilst keeping the original scaffold fixed. Another model of this type, proposed by Arús-Pous et al. (2020), offered a greater degree of control over the generated molecules by allowing the specification of an exit vector.

As a complementary approach for scaffold decoration, Imrie et al. (2020) proposed DeLinker, which takes a pair of fragments as input and builds a molecular linker between them. Much of the work in this thesis builds upon the DeLinker generative model, so we describe the generative process in detail below.

1.4.2.1 DeLinker

DeLinker (Imrie et al., 2020) is a deep generative model for linker design that builds on the Constrained Graph Variational Autoencoder (CGVAE) proposed by Liu et al. (2018). The original DeLinker implementation takes two fragments as input, as well as their relative position, and generates a linker between them whilst keeping the original fragments fixed. The user is also required to specify the fragment exit vectors used by the model; i.e. the atoms which the linker will be connected to. Below we outline how DeLinker generates linkers and the procedure used to train it.

The original CGVAE model was trained on a dataset of molecules where the objective was to minimise the difference between the input molecule whilst enforcing a Gaussian distribution on the latent space. That is, for a chemical space, $\mathbb{M}$, and a set of input molecules $M_1, ..., M_n \in \mathbb{M}$, the objective was to identify two functions $\phi : \mathbb{M} \to Z$ and $\psi : Z \to \mathbb{M}$ such that $\sum_{i=1}^{n} \mathcal{L}(M_i, \psi(\phi(M_i)) + \mathcal{L}_{latent}$ was minimised, where $\mathcal{L}$ represents the reconstruction loss, $\mathcal{L}_{latent}$ represents the KL-divergence of the distribution of the latent space and the isotropic Gaussian distribution, and λ_{KL} is a scaling constant.

Whilst the aim of the CGVAE model was to learn to sample molecules which

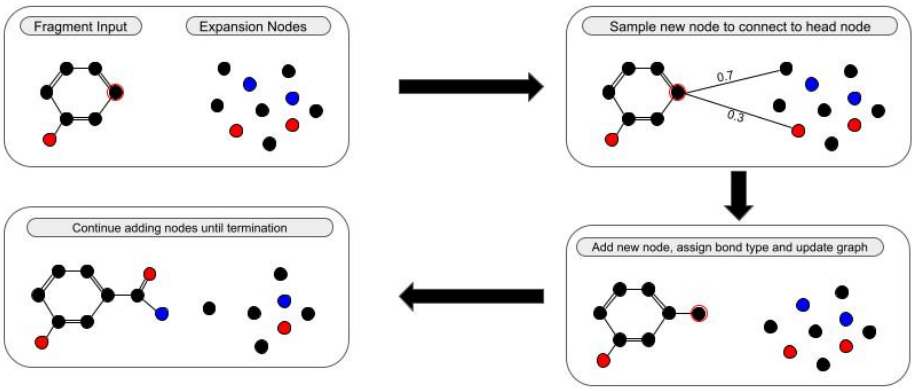

Figure 1.6: Schematic of the DeLinker Generative proposed by Imrie et al. (2020). The process starts with the input fragment and a 'sea' of atoms to be added. A graph neural network selects which atom should be added to the graph and the corresponding bond type. All atoms are added to the head node (illustrated by the red circle). Instead of adding a new atom to the graph, the network can opt to change the head node for the next node in the queue, so that new atoms are bonded to a new atom.

were highly similar to the input molecule, DeLinker takes two fragments as input and output a linked molecule. The training objective is therefore slightly different: For a set of pairs $(F_1, M_1), ..., (F_n, M_n)$, where F_i represents the input fragment pair from a chemical space $\mathbb{F}$ and M_i represents the ground truth molecule which contains both input fragments as substructures, DeLinker aims to identify functions $\phi : \mathbb{M} \rightarrow Z$ and $\psi : Z \times \mathbb{F} \rightarrow \mathbb{M}$ such that $\sum_{i=1}^{n} \mathcal{L}(M_i, \psi(\phi(M_i), F_i) + \lambda_{KL}\mathcal{L}_{latent}$ is minimised.

Once the model has been trained, one can generate linkers for a fragment pair F_{test} by replacing the latent code $z \in Z$ derived during training with a sample from an isotropic Gaussian, and computing $\psi(z, F_{test})$

Generative Process. We outline the process by which DeLinker takes a fragment pair, F and latent code z and produces a linked molecule. The fragments are converted to a graph representation, where each node represents an atom and each edge a bond. Each node, v contains a hidden state $\mathbf{z}_v$ and a one-hot encoded label $\mathbf{l}_v$, which represents the atom type of v. The graph is passed through a graph gated neural network (Li et al., 2015), which updates each node in the network with in-

formation from its neighbouring nodes. Next, a set of m vectors are sampled from an h-dimensional isotropic Gaussian distribution, where m is the maximum allowed linker length, and h is the size of the hidden state. These vectors represent a 'pool' of atoms, which the linker will be generated from, and are assigned an atom type by a learnt mapping f.

Starting from one of the fragment exit vector nodes, the linker is iteratively constructed atom-by-atom. A node is selected from the pool, connecting it to the fragment and being added to the node queue, the representation of each atom is updated to reflect the new graph, and a new atom from the pool is selected to be added to the fragment. This process continues until a special 'stop node' is sampled instead of a new atom, and the current head of the queue is removed and replaced by the next node in the queue. This process continues until the queue is empty, at which point the algorithm terminates and returns the largest connected component as the output molecule.

The decision of which node to add to the fragment is taken by a single-layer neural network; for a current queue-head node, v, at step t a feature vector $\phi^t_{v,u}$ is constructed for all possible candidate nodes, u, as:

$$\phi^t_{v,u} = [t, s^t_v, s^t_u, d_{v,u}, \boldsymbol{H}^0, \boldsymbol{H}^t, \boldsymbol{D}]$$

where $s^t_v = [z^t_v, l_v]$ is the concatenated hidden representation of node v after t steps and l_v is the associated atom type, $d_{v,u}$ is the graph distance between v and u, $\boldsymbol{H}^i$ is the average representation of all nodes at the ith time point, and $\boldsymbol{D}$ represents the structural information provided to the model (in this case, the euclidean distance and angle between the fragment exit vectors). The node to be added to the fragment is then sampled from the following distribution:

$$p(v \leftrightarrow u|\phi^t_{v,u}) = \frac{\exp(C(\phi^t_{v,u}))}{\sum_w \exp(C(\phi^t_{v,w}))},$$

where $C(\phi^t_{v,u})$ is the output of the neural network. Once a node has been selected, a further neural network uses $\phi^t_{v,u}$ to assign an bond type between the two atoms, and the hidden state of all nodes are updated to reflect their new local neighbourhoods. Valency restraints are enforced via a masking process, which ensures that the probability of selecting a node or edge which would violate valency constraints is set to zero.

1.5 Thesis Outline and Contributions

In this chapter, we gave a brief overview of the drug discovery process and computer aided drug design, in particular how machine learning methods have been used to computationally design and screen new molecules. In the remainder of this thesis, our primary focus is on the development of a novel deep learning-based tool for structure-aware fragment elaboration.

In Chapter 2, we attempt to generate elaborations to a fragment which exhibit a high 3D similarity to a ground truth elaboration. To achieve this, we employ reinforcement learning to finetune a pre-trained deep generative model for fragment elaboration. We demonstrate that the imposition of a structured curriculum allows the reinforcement learning step to generate a wider variety of highly similar elaborations, compared to optimising directly on the similarity function.

In Chapter 3, we explore the use of pharmacophoric constraints as an alternative to reinforcement learning in incentivising a deep generative model for fragment elaboration to generate elaborations with a specific pharmacophoric profile. On a large scale evaluation, we found that pharmacophoric constraints enabled the generative model to successfully recover the ground truth more often, and allowed it to generate

a diverse range of highly similar elaborations which were capable of making the same intermolecular interactions.

In Chapter 4, we describe the development of a web application which allows users to load a fragment or fragment pair into a 3D molecule viewer, select the atoms they wish to employ as exit vectors, and generate a set of elaborations/linkers which can be easily downloaded for further consideration.

In Chapter 5, we propose a method for extracting important structural information from a target protein and converting it to a pharmacophoric constraint of the form that we developed in the previous chapter. Given a target protein, we computed a Fragment Hotspot Map, which describe regions of the protein binding pockets which are likely to make a disproportionate contribution to binding affinity, and generated elaborations so that matching functional groups were placed within the hotspot regions. Our method requires only a bound fragment and protein structure, making it applicable to novel targets which don't have large numbers of existing actives, and is able to propose more ligand efficient elaborations than existing methods.

In Chapter 6, we consider the related problem of virtual screening, and investigate the ability of different virtual screening models to identify the functional groups responsible for binding. Through a synthetic generative process with a deterministic binding rule, we demonstrate that a deep-learning based virtual screening model is better able to identify the most important functional groups than a fingerprint-based model, and that, in addition to artificially inflating the predictive accuracy of virtual screening models, ligand-specific biases degrade their ability to identify important functional groups. This highlights the importance of the creation of high quality, unbiased datasets for virtual screening.

Finally, in Chapter 7, we summarise the results of this thesis and discuss possible avenues for future work.

Chapter 2

Using Reinforcement Learning to Generate Elaborations with a Specified Pharmacophoric Profile.

2.1 Preface

As discussed in the Introduction, the long term goal of my DPhil project was to develop a tool which could take a fragment-protein complex as input and use the protein structure to return a set of molecules which bound to the protein with high affinity. For simplicity, we divided the task into two smaller subtasks: First, establish a method by which we could incentivise a model to generate molecules from a specific region of chemical space, then develop a protocol for extracting relevant structural information from the protein and using it to decide which region of chemical space the model should sample from. For the first subtask, which we address in this chapter and the following chapter, we attempted to incentivise a generative model to generate elaborations with a high 3D similarity to a ground truth elaboration.

In this chapter we investigate whether reinforcement learning (RL) can be used

to train a model to reliably generate elaborations which are highly similar to the specified ground truth. We initially used a measure of 3D similarity as the reward function for a policy gradient algorithm but found that the high dimensionality of chemical space, combined with the non-linear nature of the reward landscape, made it challenging to optimise the similarity-based reward function directly. To address this, we developed a curriculum-based reward function which incentivised the model to sample from increasingly focused regions of chemical space. We found that this approach yielded more elaborations that were highly similar to the ground truth, compared optimising directly on the reward function.

Although reinforcement learning showed promise as a method for sampling from a constrained region of chemical space, we found that its performance varied substantially for different examples and that retraining the model for each new task increased the computational overhead and hindered its applicability. In the next chapter, we build from this work to directly incorporate the curriculum-based reward function into the generative model as a pharmacophoric constraint.

2.2 Background

As discussed in the Introduction, there has been substantial recent interest in applying deep learning models to the *de-novo* generation of molecules for drug discovery campaigns. A variety of different deep learning architectures have been proposed for this purpose, including Variational Autoencoders (Gómez-Bombarelli et al., 2018), Generative Adversarial Networks (Guimaraes et al., 2017) and Recurrent Neural Networks (Olivecrona et al., 2017), and several works have proposed methods for training generative models so that the generated molecules are optimised with respect to a generic scoring function.

Inspired by the successes of reinforcement learning on a variety of challenging

tasks (Silver et al., 2017; Mnih et al., 2013; Kober et al., 2013), a number of recent works have sought to use reinforcement learning to optimise a reward function, including the quantitative estimate of druglikeness (QED), (Bickerton et al., 2012), (Zhou et al., 2019), similarity to a ground truth molecule (Olivecrona et al., 2017), and a docking score against a particular target (Jeon and Kim, 2020). In this chapter, we consider the problem of using a deep generative model to generate elaborations to a fragment, and use reinforcement learning to incentivise the model to generate elaborations which are functionally equivalent to a ground truth elaboration. Motivated by the success of curriculum-based reinforcement learning on a variety of challenging problems ((Florensa et al., 2017), (Luo et al., 2020)), we explored whether a curriculum-based reward function could be used to improve the ability of an Agent to generate elaborations which were highly similar to the ground truth.

Our experiments demonstrate that using reinforcement learning to finetune a model allows it to generate more distinct highly similar elaborations than a purely supervised model. In addition, the specification of a structured curriculum allowed the model to generate more highly similar elaborations than when directly incorporating the similarity score into the reward function, illustrating the value of a curriculum-based approach.

Despite improving the ability of our generative model to sample elaborations from a focused subset of chemical space, we found that the requirement to finetune a model for each new target hindered its applicability, and the success of the reinforcement learning algorithm in learning to optimise the reward function varied substantially between examples, mirroring previous studies which have raised concerns surrounding the reproducibility of reinforcement learning (Henderson et al., 2018).

2.3 Methods

In this section we describe the generative model we use to generate elaborations to a specified fragment whilst keeping the fragment fixed. The model, based on the generative process defined by Imrie et al. (2020), is able to generate a wide range of chemically valid elaborations and was able to generate elaborations not included in the training set, illustrating the model's ability to generalise. However, the original model did not have the functionality to change the elaborations it proposed to meet a particular use case, e.g. generating elaborations which were likely to improve the binding affinity of a fragment against a particular target. We therefore describe two reinforcement learning-based approaches which allow us to finetune the model, incentivising it to generate elaborations which can make similar interactions compared to a ground truth.

2.3.1 Generative Process

For all experiments in this chapter, we used a generative process adapted from DeLinker (Imrie et al., 2020), which we described fully in the Introduction (Section 1.4.2.1). DeLinker takes two fragments as input and generates linkers between them in an atom-by-atom fashion, with valency constraints ensuring chemical validity. We repurposed DeLinker for fragment elaboration by modifying the preprocessing scripts to construct the training sets so that each example comprised a single fragment input and a ground truth elaboration (as opposed to a fragment-pair input and ground truth linker). In addition to a fragment pair, DeLinker also took as input the euclidean distance and angle between the two fragment exit vectors; this structural information aided it in designing linkers which were more appropriate for the binding pocket. This information was not provided to the model for our fragment elaboration experiments as, in contrast to fragment linking where the goal is to extend a structure

to a specified 3-dimensional point, there was no pre-defined 'end point' for our fragment elaboration experiments. However, as with the linking experiments conducted in Imrie et al. (2020), the generative model was provided with the number of atoms contained in the ground truth elaboration, making it more likely that the elaborations proposed by the generative model would be able to fit within the target binding site.

2.3.2 Supervised model

To train the model described above, we selected a 250k random subset of the Moses dataset (Polykovskiy et al., 2020). Each molecule was fragmented following the approach described by Hussain and Rea (2010), i.e. enumerating all cuts of acyclic single bonds that were not within functional groups, to yield a set of fragment-ground truth pairs for each molecule. We discarded all entries where the ground truth elaboration was smaller than two or greater than ten atoms, yielding a final training set of 669451 fragment-ground truth pairs. The network was trained using Tensorflow (Abadi et al., 2015) (V1.10) using the hyperparameters and atom types described in Imrie et al. (2020). Although the model was trained for fragment elaboration rather than fragment linking, for the remainder of this chapter we refer to this model as "DeLinker" to reflect the fact that it uses an analogous generative process to the DeLinker generative model (Imrie et al., 2020).

2.3.3 Reinforcement Learning

As discussed in Section 1.3.3, reinforcement learning (RL) concerns the training of an Agent who must choose an action $a \in \mathbb{A}$, given a certain state, $s \in \mathbb{S}$, where $\mathbb{A}$ is the set of all possible actions and $\mathbb{S}$ is the set of all possible states. The actions of the Agent are governed by a *policy*, $\pi_\theta(a|s)$, a probability distribution governed by a set of parameters, θ, which specifies the probability of taking each action conditional on a given state.

We consider an iterative process where, following an initial state, s_0, the Agent samples an action, a_1, from the policy $\pi_\theta(\cdot|s_0)$. The action a_1 gives rise to a new state, s_1, which is then used to sample a new action a_2 from $\pi_\theta(\cdot|s_1)$. This process continues until time T, yielding a final state, s_T. At each $t = 1, ..., T$, we can specify a *reward*, R_t, for how favourable the state s_t is, yielding a total reward $G_T = \sum_{t=1}^{T} R_t$. Our goal is to select the parameter set, θ, such that the average reward, $V_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}[G_t|s_t = s]$ is maximised.

For our problem, the initial state, s_0 corresponds to the initial fragment provided to the model. An action, a, represents either an atom being added to the fragment, a bond type between two atoms being selected, or the "stop node" being sampled. We formulate the problem as a Markov Decision Process, meaning that all decisions taken by the Agent have the Markov property, where:

$$\mathbb{P}(s_{t+1}|s_t) = \mathbb{P}(s_{t+1}|s_1, ..., s_t)$$

In other words, the current state contains all the information required to assess the suitability of the next action; previous states do not have any impact on the next action. In addition, as intermediate states may not necessarily correspond to a valid molecule (e.g. a partially constructed aromatic ring), we only compute the reward function at the final stage, i.e. $G_T = r_T$.

We follow the approach proposed by Olivecrona et al. (2017), using a REINFORCE (Williams, 1992) algorithm to finetune a pre-trained supervised model to make elaborations which are optimal with respect to a scoring function.

2.3.4 RL-enhanced model

As our goal was to generate elaborations with a high degree of similarity to the ground truth, for a given molecule m and ground truth m_{GT}, we specified our reward function

to take the following form:

$$r(m|m_{GT}) = \frac{\sigma S(m, m_{GT})}{L(m, m_{GT})^2},$$

where $S(\cdot, \cdot)$ is an arbitrary function that quantifies the similarity between two molecules (ranging between 0 and 1, with a score of 1 denoting a perfect match), $L(\cdot, \cdot)$ is the VAE loss function used to train the generative model (ranging between 0 and ∞, where a loss of 0 denotes perfect match), and σ denotes a scaling constant (we used $\sigma = 100$ for all experiments in this chapter). Elaborations with a high degree of similarity to the ground truth would therefore attain a high reward, meaning that the REINFORCE algorithm would incentivise the model to generate similar elaborations. For all examples in this chapter, we ran the REINFORCE algorithm for 500 iterations and saved the model when the running average of $S(m, m_{GT})$ reached 0.1, 0.2,... 1. For generating elaborations, we used the saved model corresponding to the highest running average. If the running average failed to reach 0.1 during the 500 iterations, we deemed the training to have failed.

2.3.5 Similarity Scoring Functions

The first similarity function we considered was the SC_{RDKit} function (described in Section 1.4.1.1), based on the work of Putta et al. (2005) and Landrum et al. (2006), which quantify the volumetric and pharmacophoric overlap of two molecules, respectively. This allows molecules which are structurally distinct but capable of forming the same interactions with the protein to attain a high similarity. We refer to the generative model finetuned using the SC_{RDKit} score as "DeLinker-SC".

As an alternative to directly optimising the SC_{RDKit} score, we considered an alternative curriculum-based scoring function. Curricula have proved useful in a variety of tasks where the reward function is sparse or where the Agent is prone to converging

to a local minimum (e.g. (Florensa et al., 2017), (Luo et al., 2020)). As the subset of elaborations which are highly similar to the ground truth is extremely small compared to the wider chemical space, and slight perturbations to an elaboration can significantly improve/impair its suitability, it is plausible that a structured curriculum might facilitate easier optimisation by the Agent. For illustrative purposes, we first describe a very simple similarity curriculum and then extend it to describe the curriculum used for our experiments in Section 2.4

2.3.5.1 "Counts" Curriculum

The simplest way to incentivise the model to generate elaborations which are more-similar-than-random to the ground truth would be to employ the following similarity scoring function:

$$S(m, m_{GT}) = \begin{cases} \alpha & \text{if } m \text{ and } m_{GT} \text{ have the same heavy atom count,} \\ 0 & \text{Otherwise.} \end{cases} \tag{2.1}$$

Compared to a random sample of chemical space, molecules with the same heavy atom count to the ground truth are more likely to have a substantial volumetric overlap, leading to an improved SC_{RDKit} score (Section 1.4.1.1). Due to the abundance of molecules with a particular heavy atom count, the generative process should be able to frequently to sample molecules which attain a non-zero reward, allowing the Agent to optimise the average reward.

However, in addition to proposing elaborations with a high volumetric overlap to the ground truth, we also require the generated elaborations to be capable of making similar interactions with the protein, which is not accounted for by Equation 2.1. A simple similarity function which could incentivise the model to sample elaborations with similar pharmacophores to the ground truth is defined in Algorithm 2.

Algorithm 2 Simple curriculum example as a set of nested if statements. The similarity score is increased step-wise if the molecule meets each successive criterion.

Require: $\beta > 0$, $\gamma > 0$, $\delta > 0$
  $S(m, m_{GT}) \leftarrow 0$
  if $\texttt{num\_HBA}(m) == \texttt{num\_HBA}(m_{GT})$ **then**
    $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \beta$
    if $\texttt{num\_HBD}(m) == \texttt{num\_HBD}(m_{GT})$ **then**
      $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \gamma$
      if $\texttt{num\_Aromatic}(m) == \texttt{num\_Aromatic}(m_{GT})$ **then**
        $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \delta$
      end if
    end if
  end if
    $S(m, m_{GT}) \leftarrow \frac{S(m, m_{GT})}{\alpha\beta+\gamma+\delta}$ ▷ Normalise so that score is in $[0, 1]$

Similar to the similarity function defined in equation 2.1, this simplistic similarity function is non-sparse and it would be simple for the Agent to learn to optimise. However, if the ground truth contains a relatively small number of Donor, Acceptor or Aromatic functional groups, the model may learn to make very simplistic elaborations comprising a small number of atoms, leading to a poor volumetric and pharmacophoric overlap.

A curriculum-based similarity function attempts to satisfy both similarity functions by first incentivising the model to sample from the subset of chemical space, A which optimises the first similarity function, then further incentivises the model to sample within a further subset $B \subset A$, where B is the subset of chemical space which optimises both similarity functions. An illustration of such an approach is shown in Figure 2.1. A curriculum-based reward function which combines the similarity functions defined in Equation 2.1 and Algorithm 2 is defined in Algorithm 3.

2.3.5.2 "Path" Curriculum

Here, we extend the curriculum described in Algorithm 3 to incentivise the DeLinker model to sample from a more focused region of chemical space. The exact placement

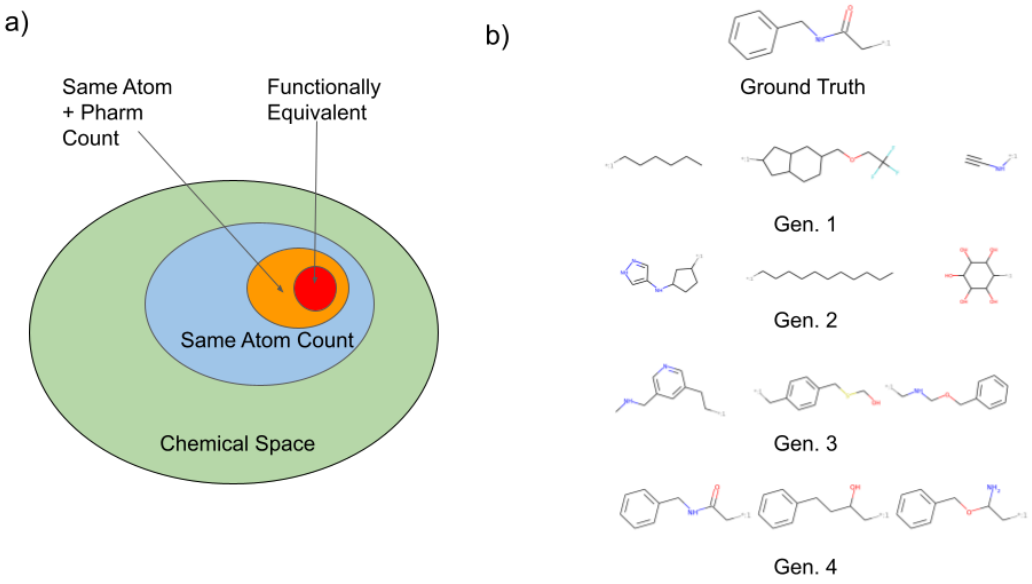

Figure 2.1: Illustration of curriculum-based reward function. a) Our goal is to start with a model which samples from a broad chemical space (Green) and incentivise it to sample elaborations from the Functionally Equivalent subset (Red). Curriculum learning incentivises the model to sample from progressively smaller subsets of chemical space. b) Illustration of a model learning to generate elaborations similar to a ground truth via a curriculum. Initially the model samples random elaborations, before learning to sample elaborations with the same heavy atom count as the ground truth. Next the model learns to sample elaborations with the correct number of pharmacophores and finally learns to place the pharmacophores so that the elaborations can make the same interactions as the ground truth.

Algorithm 3 Illustration of the Counts curriculum. A model trained using this curriculum would first learnt to generate elaborations with the correct atom count, then it would learn to generate elaborations with the correct number of pharmacophores.

Require: $\alpha > 0$, $\beta > 0$, $\gamma > 0$, $\delta > 0$
  $S(m, m_{GT}) \leftarrow 0$
  if $\texttt{num\_atoms}(m) == \texttt{num\_atoms}(m_{GT})$ **then**
    $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \alpha$
    if $\texttt{num\_HBA}(m) == \texttt{num\_HBA}(m_{GT})$ **then**
      $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \beta$
      if $\texttt{num\_HBD}(m) == \texttt{num\_HBD}(m_{GT})$ **then**
        $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \gamma$
        if $\texttt{num\_Aromatic}(m) == \texttt{num\_Aromatic}(m_{GT})$ **then**
          $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \delta$
        end if
      end if
    end if
  end if
    $S(m, m_{GT}) \leftarrow \frac{S(m, m_{GT})}{\alpha\beta+\gamma+\delta}$ ▷ Normalise so that score is in $[0, 1]$

of a functional group within an elaboration is an essential consideration in determining whether it will be able to make an interaction with the target. As such, a large proportion of elaborations which have the same number of pharmacophores as the ground truth would be unable to recreate the intermolecular interactions associated with the ground truth. A natural extension to the "Counts" curriculum similarity scoring function (Algorithm 3) is therefore to consider whether, in addition to having an equivalent number of pharmacophores, the pharmacophores may be able to make equivalent interactions with the target. To assess this, we compute the path distance between the fragment exit vector and each pharmacophore contained in the ground truth elaboration. In the "Path" curriculum similarity scoring function, if an elaboration has an equivalent number of pharmacophores to the ground truth and the path distance between the exit vector and a pharmacophore matches that of an equivalent pharmacophore in the ground truth, the score is incremented by some constant value. We describe the full scoring function in Algorithm 4 and illustrate the various stages of the curriculum in Figure 2.1.

Algorithm 4 Simple curriculum example as a set of nested if statements. The similarity score is increased step-wise if the molecule meets each successive criterion.

Require: $\alpha > 0$, $\beta_1 > 0$, $\gamma_1 > 0$, $\delta_1 > 0$
Require: $\beta_2 > 0$, $\gamma_2 > 0$, $\delta_2 > 0$
$S(m, m_{GT}) \leftarrow 0$
path_distances_hba_gt $\leftarrow$ [path_distance(a) for a in Get_HBAs(m_{GT})]
path_distances_hbd_gt $\leftarrow$ [path_distance(a) for a in Get_HBDs(m_{GT})]
path_distances_arom_gt $\leftarrow$ [path_distance(a) for a in Get_Aromatics(m_{GT})]
if num_atoms(m) == num_atoms(m_{GT}) **then**
  $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \alpha$
  if num_HBA(m) == num_HBA(m_{GT}) **then**
    $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \beta_1$
    for hba in Get_HBAs(m) **do**
      if path_distance(hba) in path_distances_hba_gt **then**
        $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \beta_2$
      end if
    end for
    if num_HBD(m) == num_HBD(m_{GT}) **then**
      $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \gamma_1$
      for hbd in Get_HBDs(m) **do**
        if path_distance(hbd) in path_distances_hbd_gt **then**
          $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \gamma_2$
        end if
      end for
      if num_Aromatic(m) == num_Aromatic(m_{GT}) **then**
        $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \delta_1$
        for arom in Get_Aromatics(m) **do**
          if path_distance(arom) in path_distances_arom_gt **then**
            $S(m, m_{GT}) \leftarrow S(m, m_{GT}) + \delta_2$
          end if
        end for
      end if
    end if
  end if
end if
normalise($S(m, m_{GT})$)

Whilst it is true that an equivalent path distance is a less accurate predictor of whether two functional groups would make the same interaction in a protein binding pocket than, say, calculating the euclidean distance of the two functional groups after they have been docked into the binding site, using the path distance to assess similarity has several advantages. First, computing the path distance between two atoms is computationally inexpensive compared to protein-ligand docking, allowing for considerably faster model training. In addition, as the DeLinker generative model generates elaborations in 2D without access to 3D atomic coordinates, scoring via path distances does not require the Agent to learn a complex function which approximates the 3D conformation of each molecule within the binding pocket. We refer to the model finetuned using the Path curriculum as "DeLinker-Curriculum".

2.3.6 Test Set

The Malhotra set (Malhotra and Karanicolas, 2017) is a set of paired ligands bound to a target in the PDB (Berman et al., 2000). For many of the ligand pairs, the smaller ligand is a substructure of the larger ligand. We can therefore view the larger ligand as a successful elaboration of the smaller one, and consider a scenario where we would like to generate alternative elaborations which preserve the interactions made by the larger ligand. We used the Malhotra set as a basis for constructing a test set to assess the ability of different models to make appropriate interactions with the protein. We included the examples where the larger ligand had between 2 and 10 more atoms than the smaller ligand, the smaller ligand was a substructure of the larger ligand, and additional structure in the larger ligand was attached to the smaller ligand via a single exit vector. To increase the size of the test set, we made the following modifications:

- If the larger ligand was between 11 and 14 atoms larger than the smaller ligand then we added a small number of atoms to the smaller ligand to reduce the

required elaboration to 10 atoms or fewer.

- If the additional structure in the larger ligand was attached to the smaller ligand via multiple exit points, we added structure to the smaller ligand so that only a single exit point was required.

- If the smaller ligand wasn't a substructure of the larger ligand but the maximal common substructure was only a single atom smaller than the smaller ligand, we modified the smaller ligand so that it was a substructure of the smaller ligand.

The above procedure led to a set of 50 fragment-ground truth pairs. A total of 9 and 11 examples failed to train for DeLinker-SC and DeLinker-Curriculum respectively, so the final test set comprised the 33 fragment-ground truth pairs for which both models successfully trained (shown in Table A.1). For each example, we generated 500 elaborations with the DeLinker, DeLinker-SC and DeLinker-Curriculum models.

2.3.7 Metrics

To assess the ability of each model to make appropriate elaborations, we computed a variety of different metrics:

- **Validity:** We considered an elaboration to be valid if the resulting molecule can be parsed by RDKit and at least one atom has been added to the fragment.

- **Uniqueness:** For a given fragment-ground truth example, we calculate the proportion of unique elaborations as the number of distinct elaborations divided by the total number of generated molecules.

- **Novelty:** We consider an elaboration to be novel if it was not contained in the training set as a ground truth elaboration.

Table 2.1: Results of experiments on the Malhotra test set. DeLinker-Curriculum generated a larger number of distinct above-threshold elaborations than both DeLinker and DeLinker-SC.

Metric	DeLinker	DeLinker-SC	DeLinker-Curriculum
Valid	100%	98.3%	100%
Unique	36.9%	31.5%	47.1%
Novel	62.9%	59.18%	61%
SC_{RDKit}			
>0.6	23.85	18.88	40.45
>0.7	16.79	8.73	26.64
>0.8	9.3	4.21	11.27
>0.9	0.7	0.88	1.27

For each example, we assessed the ability of each model to generate elaborations which were highly similar to the ground truth by computing the number of distinct elaborations which attained an SC_{RDKit} score above a series of thresholds. We report the average number of distinct above-threshold elaborations produced over all of the examples in the test set

2.4 Results and Discussion

We investigated the extent to which reinforcement learning could improve our generative model's ability to generate elaborations with a high degree of similarity to the ground truth. Our experiments on the Malhotra test set demonstrated that both RL-enhanced Agents were able to generate more highly similar elaborations than DeLinker, with the curriculum-based reward function enabling the Agent to generate more distinct elaborations with a high SC_{RDKit} score compared to the Agent optimised on the SC_{RDKit} score directly.

Table 2.1 shows the results of our experiments on the Malhotra test set. Almost all elaborations generated by all 3 models were considered to be valid, and all models were

able to generate a substantial proportion of elaborations not contained in the training set (DeLinker: 63%, DeLinker-SC: 59%, DeLinker-Curriculum: 61%), illustrating the ability of the models to generalise beyond the training set. DeLinker was able to produce a broad range of different elaborations for each example, attaining a uniqueness statistic of 37%. DeLinker-SC produced a smaller proportion of distinct elaborations (31.5%), but this was expected as the goal of the reinforcement learning finetuning was to incentivise the model to learn from a reduced subset of chemical space. Surprisingly, DeLinker-Curriculum generated a larger proportion of unique elaborations than DeLinker, despite also being incentivised to sample from a reduced subset of chemical space.

In terms of generating elaborations which were capable of making the same interactions as the ground truth elaboration, all models were able to generate elaborations with a high SC_{RDKit} score. For the SC_{RDKit} threshold of 0.6, which indicates a reasonably good match between the generated elaboration and the ground truth, DeLinker-Curriculum, on average, generated the most distinct above-threshold elaborations (40.45). Surprisingly, despite being trained to directly optimise the SC_{RDKit} score, DeLinker-SC (18.88) generated fewer distinct above-threshold elaborations than the purely supervised DeLinker model (23.85). We observed the same trends for the 0.7 and 0.8 thresholds, with the number of distinct above-threshold elaborations decreasing as the threshold increased.

All models struggled to consistently generate a broad range of elaborations which attained an SC_{RDKit} score of above 0.9. This can be explained in part by the relatively small heavy atom count of the ground truth elaborations (10 atoms or fewer), making it difficult to obtain a very high SC_{RDKit} score without exactly recreating the ground truth elaboration. Moreover, the relatively small elaboration size places extra weight on individual functional groups in the calculation of the pharmacophoric overlap, reducing the smoothness of the reward function and potentially making it

Table 2.2: Average total number of above-threshold elaborations generated by each model on the Malhotra test set. Despite generating a smaller number of distinct above-threshold elaborations than DeLinker-Curriculum, DeLinker-SC generated considerably more above-threshold elaborations in total.

Metric	DeLinker	DeLinker-SC	DeLinker-Curriculum
SC_{RDKit}			
>0.6	77.7	151.8	113.4
>0.7	60.4	125.4	73.5
>0.8	24.2	81.9	42.9
>0.9	03.6	54.36	19.09

more difficult to optimise directly.

Whilst DeLinker-SC tended to generate fewer distinct above-threshold elaborations than the other models, the average SC_{RDKit} score of the elaborations generated by DeLinker-SC (0.51) was greater than those generated by DeLinker (0.43) and DeLinker-Curriculum (0.47), respectively. In addition, if instead of computing the number of distinct above-threshold elaborations, we compute the total number of above-threshold elaborations (including duplicates), DeLinker-SC comfortably outperforms the other models (Table 2.2).

To explore the reason for this difference further, we consider an example from the Malhotra set in detail (Figure 2.2). Both DeLinker-SC and DeLinker-Curriculum were able to generate a range of elaborations which were highly similar to the ground truth phenol; of the 500 elaborations proposed by DeLinker-SC, 492 attained an SC_{RDKit} score greater than 0.7, compared to 57 of the elaborations proposed by DeLinker-Curriculum. However, when we considered the number of distinct above-threshold elaborations proposed by each model, DeLinker-SC (5) proposed considerably fewer than DeLinker-Curriculum (19). Of the 492 above-threshold elaborations, DeLinker-SC generated a single chlorobenzene elaboration, 97 benzene elaborations and 394 methylbenzene elaborations. Whilst repeatedly generating a simple, benzene-

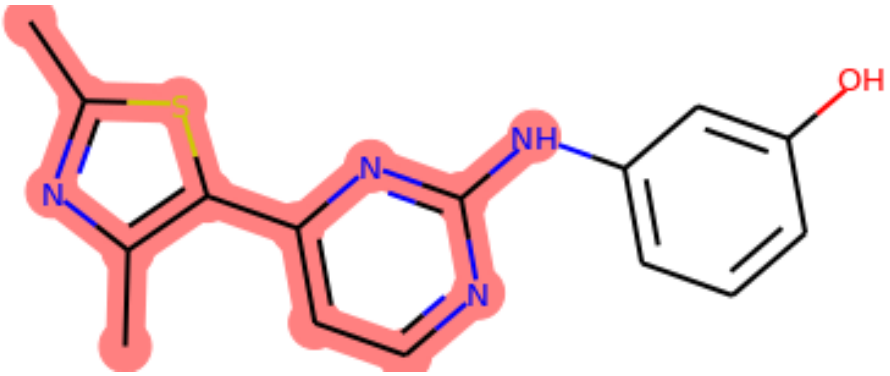

Figure 2.2: An example from the Malhotra test set. The highlighted substructure represents the starting fragment, whilst the ground truth elaboration comprises the remaining atoms. DeLinker-SC was able to attain moderately high SC_{RDKit} scores by repeatedly generating benzene or methylbenzene elaborations. Whilst such elaborations closely matched the phenol ground truth, the missing donor-acceptor group meant that the proposed elaborations would be unable to make the same interactions as the ground truth.

containing elaboration is an efficient way of maximising the average reward, it inhibits DeLinker-SC's ability to generate elaborations which are capable of making the same interactions as the ground truth. By contrast, as DeLinker-Curriculum must generate elaborations with the correct number of pharmacophores in order to attain higher rewards, it is less able to adopt the strategy of repeatedly making simplistic elaborations to maximise the average reward. We illustrate the most commonly made elaborations attaining an SC_{RDKit} score greater than 0.7 in Tables A.1-A.2.

Our experiments clearly show that the use of reinforcement learning, on average, improves the ability of a generative model to generate elaborations which are highly similar to the ground truth. However, we found that for a substantial proportion of cases, finetuning using reinforcement learning did not lead to an increased number of highly similar elaborations: For both DeLinker-SC and DeLinker-Curriculum, in 24% of cases, the finetuned generative model failed to generate any elaborations with an SC_{RDKit} score greater than 0.7, compared to 33% of examples for DeLinker.

2.5 Conclusion

In this chapter we have demonstrated that reinforcement learning can be used to incentivise a generative model to sample from a focused subset of chemical space, making them more applicable to a fragment-to-lead or R-group optimisation campaign, where the structure of the target informs which functional groups are likely to lead to an improvement in binding affinity. Overall, we found that the specification of a structured curriculum aided the Agent to learn to generate elaborations with a high 3D similarity to the ground truth. The value of curriculum learning has been further validated by subsequent work by Guo et al. (2022), who also found that the imposition of a curriculum gave improved results compared to a standard policy gradient algorithm.

Whilst using reinforcement learning proved to be particularly effective for some examples, it also exhibited several drawbacks which hindered its applicability to real world studies: First, the DeLinker-SC and DeLinker-Curriculum needed to be undergo RL-finetuning for each new fragment-ground truth pair, substantially increasing the computational overhead required to use them and making them more difficult to use out-of-the-box, particularly for scientists without training in working with machine learning algorithms. In addition, the performance of finetuning varied dramatically depending on the fragment-ground truth input, for a significant number of examples, both DeLinker-SC and DeLinker-Curriculum were unable to satisfactorily optimise the specified reward function. One potential way in which the Agent training could be improved would be to explore replacing the REINFORCE training algorithm with an alternative approach such as deep Q-learning (Mnih et al., 2015).

A further way which could aid the generative models in designing more appropriate interactions would be to provide a low dimensional representation of the binding pocket to the model when it generates elaborations. For fragment linking experiments, DeLinker (Imrie et al., 2020) was provided with the euclidean distance and angle

between the two fragment exit vectors to help it propose linkers which would fit within the binding site; whilst it was not possible to provide this information to the model for fragment elaboration experiments, an approach similar to that proposed by Skalic et al. (2019a), which generated elaborations conditional on a 3D shape, might improve the ability of the model to generate appropriate elaborations.

Although ultimately we found that the curriculum developed in this chapter was not a viable approach for consistently generating elaborations with a specified pharmacophoric profile, we believed the representation of molecular similarity based on the pharmacophorc counts and path distances that we developed in this chapter could be provided as an extra feature to the model when training in a self-supervised fashion. In this next chapter, we explore the use of pharmacophoric constraints as a model input, and show that they provide a fine degree of control over the molecules generated by the resulting generative model.

Chapter 3

Generating Focused Sets of Elaborations via Pharmacophoric Constraints.

3.1 Preface

This chapter contains work described in the following publication:

Fergus Imrie, **Thomas E. Hadfield**, Anthony R. Bradley, and Charlotte M. Deane (2021). Deep generative design with 3D pharmacophoric constraints. *Chemical Science*, 12(43), pp.14577-14589.

The work in this chapter was carried out in collaboration with Fergus Imrie, who developed the concept for deriving 3D pharmacophoric constraints and implemented the code for deriving 3D pharmacophoric constraints (as well as writing the DeLinker code described in Imrie et al. (2020)). I modified the code to derive 3D pharmacophoric constraints for scaffold elaboration, and conducted all scaffold elaboration/R-group optimisation experiments in this chapter. The work described in this chapter is my own unless explicitly mentioned.

In the previous chapter we described a curriculum-based reinforcement learning (RL) approach for training a deep generative model for fragment elaboration. Whilst we showed that finetuning a model using a policy gradient allowed for a degree of control over the elaborations generated by the model, our approach required us to finetune a pretrained model for each new fragment. In addition to increasing the computational overhead of the method, we also found that the quality of the retrained model was highly sensitive to the input fragment; in some cases the policy gradient was unable to adequately approximate the reward function and in other instances the model optimised the reward function by learning to repeatedly generate the same elaboration, hindering the model's applicability to a real-world fragment elaboration campaign.

In this chapter we investigated whether we could apply the representation of the ground truth elaboration used to derive the reward function in the previous chapter as an input to a self-supervised fragment elaboration model. Such a model would have the advantage of not requiring retraining for each use case and we hypothesized that self-supervised training would exhibit greater stability than the RL training.

On a challenging test set derived from the 2019 PDBBind set, we show that our model was frequently able to recover the known-active molecule even when the ground truth elaboration had been excluded from the training set. In addition the model was able to generate a wide range of distinct molecules with a high degree of shape- and pharmacophoric similarity to the ground truth. Concurrent to our work, Fergus Imrie had developed a scaffold replacement model which built on the DeLinker (Imrie et al., 2020) generative model by providing 3D pharmacophoric constraints. We extended that framework so that the model could be used for scaffold elaboration/R-group optimisation and compared its performance to our semi-supervised model, achieving broadly equivalent performance.

3.2 Background

As discussed in the Introduction (Section 1.2.1), after identifying molecules which bind to the target via a high throughput- or fragment screen, the resulting hits typically undergo an iterative process of refinement to optimise a range of properties including selectivity, toxicity and binding affinity. This multi-stage optimisation process usually involves the modification of one or more substructures within the hit molecule, either to facilitate additional interactions with the protein or to replace existing functional groups with similar functional groups which are capable of making the same inter-molecular interaction.

Whilst this process has historically been led by human experts, the recent development of generative models for *de-novo* molecular design (e.g (Gómez-Bombarelli et al., 2018; Jin et al., 2018; Liu et al., 2018)) has led to interest in proposing models for scaffold decoration ((Li et al., 2019; Lim et al., 2020)), where a user can input a molecular scaffold and the model rapidly generates a number of novel molecules which contain the scaffold of interest as a substructure. As discussed in the previous chapter, these methods do not allow the specification of explicit attachment points, making them more applicable to tasks surrounding the generation of molecules with a privileged scaffold or substructure as opposed to optimising specific R-groups within a molecule.

A recently proposed model, Scaffold-Decorator (Arús-Pous et al., 2020), allows the user to specify an explicit exit vector, allowing greater control over the elaborations produced. However, Scaffold-Decorator doesn't allow the user to condition the model to generate molecules from a specific region of chemical space (e.g. requiring the generated R-groups to facilitate a hydrophobic interaction with a specified protein residue), making it challenging to integrate into a real-world R-group optimisation campaign, where human experts would wish to explore designs which satisfied a specified design hypothesis.

An important step in increasing the applicability of generative models for R-group optimisation is developing the capability to sample elaborations which meet a specified design hypothesis. In this chapter, we investigate the extent to which the imposition of pharmacophoric constraints allows a generative model to sample elaborations from a focused region of chemical space. On two large scale evaluations, we demonstrate that imposing pharmacophoric constraints allowed us to generate considerably more elaborations that had a high 3D similarity to the ground truth, compared to existing, unconstrained, models for scaffold elaboration. On a case study derived from the literature (Borkin et al., 2016), we demonstrate the applicability of our tool for R-group optimisation, as our method retrospectively generated several elaborations which were experimentally shown to yield a substantial increase in potency.

3.3 Methods

3.3.1 Generative Process

As in the previous chapter, the generative model used to propose elaborations in this chapter is built on the DeLinker (Imrie et al., 2020) framework, outlined in the Introduction, where the user specifies a fragment and the model iteratively adds atoms to the fragment until termination without modifying the original fragment substructure. Whereas in the previous chapter we explored the use of a reinforcement learning training step to incentivise the model to generate elaborations with a particular pharmacophoric profile, in this chapter we investigate to extent to which pharmacophoric constraints can achieve the same goal.

3.3.2 Pharmacophoric Constraints.

When training under a traditional variational autoencoder (Kingma and Welling, 2013) framework, the model attempts to minimise the difference (according to a

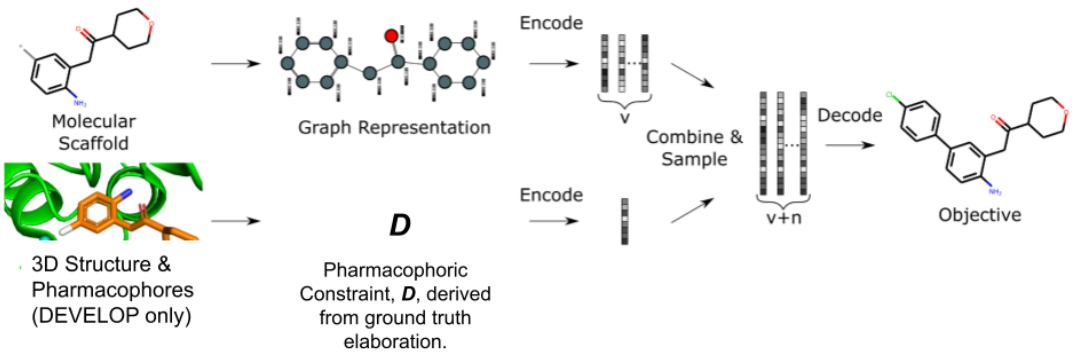

Figure 3.1: Schematic of the process used to train generative models in this chapter. As with the original DeLinker model described in Section 1.4.2.1, the model takes a fragment as input and attempts to generate the ground truth elaboration. When training, the model is also provided with a set of pharmacophoric constraints derived from the ground truth; using these constraints allows the model to more accurately approximate the ground truth elaboration. When generating elaborations, the user can provide their own pharmacophoric constraints, which the elaborations generated by the model will satisfy. Figure modified from Imrie et al. (2021).

specified metric) between the input and the output. In the context of molecular design, this training process forces the model to learn a compressed representation of chemical space, so that the decoder can create molecules which are reasonably similar to those in the training set. To allow us to have a greater degree of control over the elaborations proposed by the generative model, when training we provide the model with a low dimensional summary, $\boldsymbol{D}$, of the ground truth elaboration. The model can then exploit the information contained in $\boldsymbol{D}$ to generate elaborations which are more similar to the ground truth, helping to reduce the training loss. We illustrate a schematic of the training process in Figure 3.1. This approach has the benefit that, once the model is trained, if we wish to generate elaborations from a certain subset of chemical space, we can specify $\boldsymbol{D}$ so that the model generates elaborations from that subregion, as it has learnt to incorporate the information contained in $\boldsymbol{D}$ into the generative process. Compared to the reinforcement learning approach described in the previous chapter, this approach has the benefit that a trained model does not

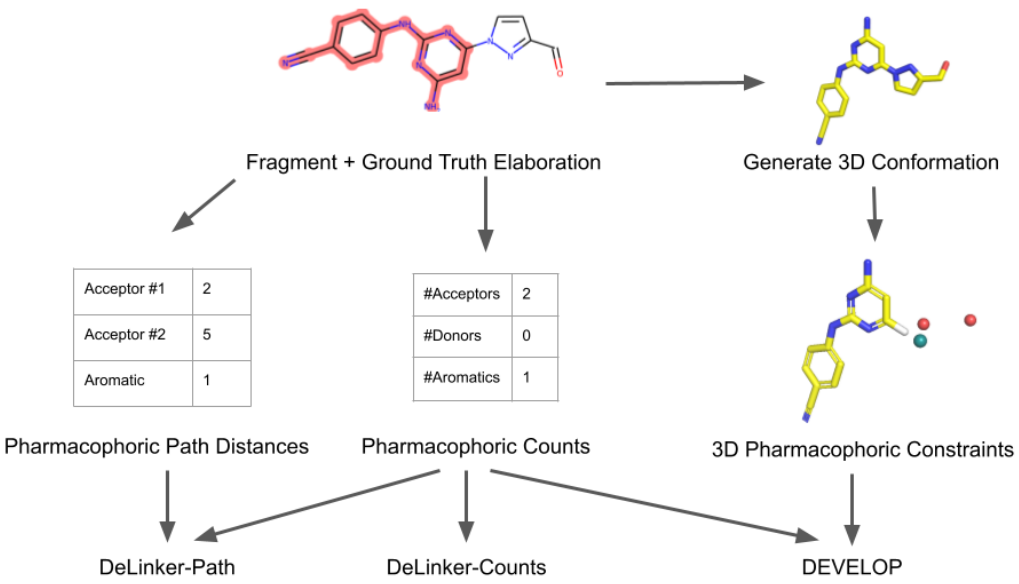

Figure 3.2: Different pharmacophoric constraints used in this chapter. The 'Counts' constraints are derived by calculating the number of Hydrogen Bond Acceptors (HBA), Hydrogen Bond Donors (HBD) and Aromatic groups are in the ground truth elaboration. The 'Path' constraints, are derived by calculating the path distance between the fragment exit vector and each functional group. The 3D pharmacophoric constraints are derived by calculating the 3D coordinates of each HBA, HBD and Aromatic group and providing them as input to a 3D CNN.

need to be finetuned for each novel target.

In this section we outline different constructions of the structural information vector, $\boldsymbol{D}$, which we use to focus the the generation of elaborations on specific subsets of chemical space. For this purpose, we make use of the pharmacophore framework (Schaller et al., 2020), defined in the Introduction. A schematic of the different pharmacophoric constraints used in this chapter is shown in Figure 3.2.

3.3.2.1 "Counts" Constraints

The Counts constraint is a simple 3-dimensional vector. During training it contains the number of Hydrogen Bond Acceptors, the number of Hydrogen Bond Donors and the number of Aromatic groups contained in the ground truth elaboration, allowing the generative model to more accurately approximate the ground truth to

minimise the training loss. As the model learns to associate the values within the Counts constraint vector with the number of pharmacophores present in the ground truth elaboration, one can use the Counts constraints vector when generating elaborations to bias the model to generate elaborations with the requested number of pharmacophores. We refer to the model which uses the Counts constraints vector as "DeLinker-Counts".

3.3.2.2 "Path" Constraints

The Counts constraint vector allows a degree of control over the elaborations made, but still allows the generative model to sample from a very broad subset of chemical space. A more prescriptive set of constraints is to specify the path distance between a pharmacophore and the fragment exit vector, allowing the user a fine degree of control over the placement of pharmacophores within the elaboration. We call these constraints "Path constraints" and refer to the model which uses both the Counts constraints and Path constraints as "DeLinker-Path".

3.3.2.3 3D Pharmacophoric Constraints

An alternative representation to the Path constraints which also allows precise specification of the position of each pharmacophore is to encode the 3D coordinates and type of each pharmacophore by means of a convolutional neural network (CNN). This is achieved by voxelising the input fragment and the ground truth pharmacophores, following the approach proposed by Sunseri and Koes (2020) and passing the voxel grid into a 3D CNN composed of three $3 \times 3 \times 3$ convolutional layers with ReLU activation, each followed by a $2 \times 2 \times 2$ max pooling layer, with the final convolutional layer followed by a global max pooling operation. We then apply dropout with probability 0.2 before a fully-connected layer produces the final structural information vector $\boldsymbol{D}$. Whereas for the Counts and Path constraints the function which

converts a given ground truth elaboration to $\boldsymbol{D}$ is fixed, with the 3D pharmacophoric constraints this function is learned during training, allowing the model to derive the representation of the pharmacophores which allows optimal reconstruction. We call the model which uses the Counts constraints and the 3D pharmacophoric constraints "DEVELOP" (**De**ep **V**ision **E**nhanced **L**ead **Op**timisation).

3.3.3 Model Hyperparameters

All models were trained for 10 epochs using the default hyperparameters outlined in by Imrie et al., (2020). A number of the hyperparameters are specified below:

- Learning rate: 0.001
- Hidden state dimension: 50
- λ_{KL} : 0.3

3.3.4 Datasets

3.3.4.1 Training Set

We used a 250k subset of ZINC (Sterling and Irwin, 2015) to generate our model training set. For each molecule, we enumerated all single cuts of acyclic bonds that were not within functional groups, yielding a set of molecular pairs. For each pair, the larger molecule was chosen as the fragment whilst the smaller molecule represented the ground truth elaboration. The set of pairs was passed through a filtering process which included a synthetic accessibility (Ertl and Schuffenhauer, 2009) filter and a PAINS (Baell and Holloway, 2010) filter, yielding a final training set of approximately 427k examples.

3.3.4.2 CASF Test Set

We used the procedure described above to construct a test set using ligands from the CASF-2016 set (Su et al., 2018). To increase the difficulty of the exercise, all examples where the ground truth elaboration comprised four or fewer atoms were removed, requiring the models to effectively use the constraints provided to sample from the appropriate region of chemical space in order to generate elaborations with a high degree of 3D similarity to the ground truth.

3.3.4.3 PDBBind Test Set

We constructed a further test set using ligands from the 2019 PDBBind Refined set (Liu et al., 2017). To assess the extent to which the different models were able to learn 'novel' chemistry (i.e. make elaborations which had not been shown to the model as part of the training set), we constructed the PDBBind set as described above but additionally omitted any examples where the ground truth elaboration was contained in the training set.

3.3.5 Metrics

We assessed the generated elaborations via a range of different metrics:

- **Validity:** We consider an elaboration to be valid if at least one atom has been added and the molecule can be parsed by RDKit (Landrum, 2006).
- **Novelty:** We define novelty as the proportion of generated elaborations which were not included in the training set.
- **Uniqueness:** The number of distinct elaborations proposed by the model divided by the total number of elaborations.

- **Pass 2D Filters:** The proportion of examples which passed the synthetic accessibility (Ertl and Schuffenhauer, 2009) and PAINS (Baell and Holloway, 2010) filters.

- **Recovery:** The proportion of examples in the test set where the ground truth elaboration was amongst the elaborations proposed by the model.

To assess the utility of the different models for R-group optimisation, we also assessed their ability to generate elaborations with high 3D similarity to the ground truth. We computed the SC_{RDKit} score used by Imrie et al. (2020), based primarily on the methods described in Putta et al. (2005) and Landrum et al. (2006), which quantifies the volumetric overlap of two molecules and the correspondence of pharmacophoric features between two molecules. For a given elaboration, we calculated the similarity to the ground truth as follows:

- First, we sampled a conformation for the ground truth using the procedure outlined in Ebejer et al. (2012) and generated a conformer for the starting fragment by mapping the coordinates of the fragment atoms in the ground truth molecule.

- Next, we generated a set of conformations for the generated molecule, again following Ebejer et al. (2012). For each conformation, we calculated the SC_{RDKit} between the ground truth elaboration and the generated elaboration.

- We reported the calculated SC_{RDKit} as the best score attained by any of the generated conformations, which reduced the risk of a highly similar elaboration attaining a low 3D similarity score as a result of the conformer generation process.

We reported the proportion of elaborations which attained an SC_{RDKit} score above a series of thresholds. An SC_{RDKit} score of above 0.6 reflects a good match, whilst a

score above 0.9 indicates that the two elaborations were almost identical.

3.3.6 Baselines

For each example in the CASF and PDBBind sets, we generated 250 elaborations using five different models. As baselines, we used two generative models without any constraints: DeLinker (Imrie et al., 2020) and Scaffold-Decorator (Arús-Pous et al., 2020), and compared their performance to that obtained by DeLinker-Counts, DeLinker-Path and DEVELOP respectively. As the Recovery and 3D similarity metrics rewarded models capable of sampling in the neighbourhood of chemical space occupied by the ground truth model, we would expect the models utilising generative constraints to outperform the unconstrained generative models on these metrics, but it was of interest to assess whether the constrained sampling approach degraded the novelty or uniqueness of the generated molecules.

3.4 Results and Discussion

In this section, we assess the ability of different generative models to propose elaborations which were highly similar to the ground truth. We compared three models which utilise pharmacophoric constraints to two baselines which do not. We found that the imposition of pharmacophoric constraints allowed the generative models to more often recover the ground truth and generate highly similar elaborations compared to the baselines. We also demonstrated the applicability of pharmacophoric constraints to real-world drug discovery tasks, via an R-group optimisation case study derived from the literature.

Table 3.1: CASF set results. See Section 3.3.5 for definitions of the metrics.

Metric	Scaffold-Decorator	DeLinker	DeLinker-Counts	DeLinker-Path	DEVELOP
Valid	99.9%	100.0%	100.0%	99.8%	99.8%
Unique	25.2%	74.2%	52.0%	37.3%	39.7%
Novel	2.4%	55.1%	50.7%	44.6%	43.4%
Recovered	18.9%	33.6%	45.5%	55.2%	58.7%
Pass 2D filters	98.4%	64.2%	67.5%	72.3%	71.7%
SC_{RDKit}					
>0.6	30.3%	22.4%	42.8%	68.9%	67.6%
>0.7	22.0%	13.9%	35.8%	58.9%	58.6%
>0.8	12.0%	5.3%	24.8%	43.0%	44.0%
>0.9	5.1%	3.1%	18.1%	30.5%	33.7%

3.4.1 Large Scale Experiments

Our experiments on the CASF and PDBBind test sets demonstrate the value of providing pharmacophoric constraints for R-group optimisation. Table 3.1 shows the performance of the different models on the CASF test; almost all elaborations generated by all of the generative models were considered valid, whilst DeLinker produced a substantially higher proportion of unique and novel elaborations than any of the models incorporating pharmacophoric constraints. These results are not particularly surprising, as the unconstrained DeLinker model is able to sample from a substantially larger region of chemical space than the constrained model and so the likelihood of repeatedly sampling the same molecule is reduced. Despite this, all of the constrained models were able to sample a considerable proportion of unique and novel elaborations and DeLinker-Path and DEVELOP produced a greater proportion of molecules passing the 2D filters than DeLinker or DeLinker-Counts.

Whilst Scaffold-Decorator was able to generate the highest proportion of filter passing elaborations, it generated by far the smallest number of unique elaborations, and only 2.4% of elaborations proposed by Scaffold-Decorated were not present in

Table 3.2: PDBbind set results.

Metric	Scaffold-Decorator	DeLinker	DeLinker-Counts	DeLinker-Path	DEVELOP
Valid	99.9%	100.0%	100.0%	99.9%	99.5%
Unique	23.4%	87.8%	81.6%	66.2%	76.2%
Novel	2.0%	71.1%	79.2%	77.1%	78.2%
Recovered	0.0%	1.0%	4.5%	14.6%	15.3%
Pass 2D filters	98.9%	55.3%	47.8%	52.3%	51.3%
SC_{RDKit} Generated					
>0.6	10.3%	7.6%	13.0%	33.3%	31.5%
>0.7	4.3%	2.7%	5.7%	17.1%	16.9%
>0.8	0.6%	0.7%	1.8%	6.4%	6.8%
>0.9	0.0%	0.1%	0.3%	1.5%	1.5%

the training set.

In terms of recovering the ground truth elaboration, DEVELOP (58%) and DeLinker-Path (55%) recovered a substantially higher proportion of examples than DeLinker-Counts (46%), DeLinker (33.6%) and Scaffold-Decorator (19%). In addition, DEVELOP and DeLinker-Path also generated far more elaborations with a high 3D similarity to the ground truth elaborations than the other models, with 34% of the elaborations generated by DEVELOP attaining an SC_{RDKit} score of above 0.9, compared to 3% and 5% for DeLinker and Scaffold-Decorator, respectively. This further reinforces the utility of imposing pharmacophoric constraints on generative models for R-group optimisation.

The PDBBind test set was more challenging than the CASF test set, as none of the ground truth elaborations were contained in the test set, forcing the models to generate novel elaborations in order to recover the ground truth. Table 3.2 shows the results obtained by different methods on the PDBBind set, with all methods again generated a very high proportion of valid elaborations, whilst the graph-based models (DeLinker, DeLinker-Counts, DeLinker-Path and DEVELOP) produced a higher pro-

portion of unique and novel elaborations compared to the CASF test set. We again observed that increasing the dimensionality of the pharmacophoric constraint reduced the proportion of unique elaborations. All models recovered a substantially smaller proportion of ground truth elaborations compared to the CASF test set. This was to be expected as the PDBBind test set was constructed to ensure that none of the ground truth elaborations were present in the training set, but we again observed that DEVELOP (15.3%) and DeLinker-Path (14.6%) successfully recovered more examples than DeLinker-Counts (5%) and DeLinker (1%). Scaffold-Decorator was unable to recover any of the ground truth elaborations in the PDBBind test set, but this is not particularly surprising as only 2% of elaborations produced by Scaffold-Decorator were not contained in the training set.

To assess whether the superior recovery rate was due to the fact that we only sampled a relatively small number of elaborations for each example, we sampled 5000 elaborations for each example in the PDBBind test set and monitored the change in recovery rate (Figure 3.3) across the different graph-based models. All models continued to recover more examples as more elaborations were produced, so in theory one might be able to recover all of the examples recovered by DEVELOP by simply sampling more elaborations with DeLinker. However Figure 3.3 indicates that one would need to sample several orders of magnitude more molecules to achieve this, illustrating the efficiency of employing pharmacophoric constraints.

In terms of generating elaborations with a high 3D similarity to the ground truth, we again found that the DEVELOP and DeLinker-Path attained a larger number of above-threshold elaborations than the other methods, with 1.5% of elaborations generated by each model attaining an SC_{RDKit} of greater than 0.9, compared to 0.3%, 0.1% and ¡0.1% generated by DeLinker-Counts, DeLinker and Scaffold-Decorator respectively. This provides further evidence that the imposition of pharmacophoric constraints improves the ability of generative models to sample from focused regions

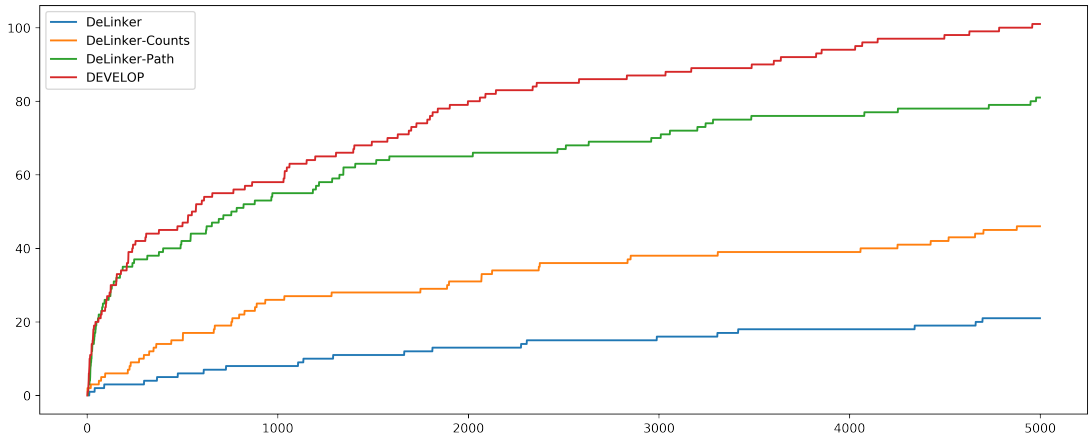

Figure 3.3: Number of PDBBind examples successfully recovered as the number of elaborations generated for each example is increased. Even when DeLinker generates a large number of elaborations, it is unable to recover as many examples as DEVELOP and DeLinker-Path when they generate a very small number of elaborations.

from chemical space, even when that region is not well represented in the training set.

3.4.2 Assessing the difference between DEVELOP and DeLinker-Path.

Whilst DEVELOP and DeLinker-Path achieved broadly comparable results across the CASF and PDBBind test sets, they often generated considerably different elaborations for the same example. To illustrate this, we present two examples from the PDBBind test set and consider the elaborations made by the different models.

The first example featured a 3-methyl-benzamide ground truth elaboration, requiring the models to add an aromatic group, a Hydrogen Bond Acceptor and a Hydrogen Bond Donor in the correct order, in order to attain a high SC_{RDKit} score. Whilst DEVELOP successfully recovered the ground truth elaboration and DeLinker-Path did not, DeLinker-Path generated a far greater proportion of elaborations with a high SC_{RDKit} score, compared to DEVELOP (Figure 3.4). The primary reason for this difference appears to lie in the correct placement of the aromatic group; of

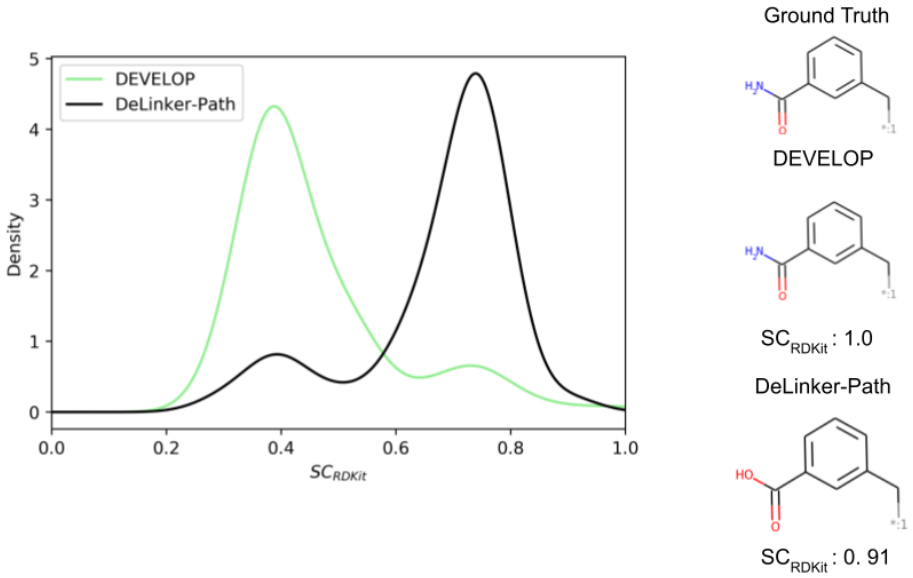

Figure 3.4: Left: Distribution of SC_{RDKit} scores obtained by DEVELOP and DeLinker-Path on a single example in the PDBBind test set. Right: Most similar elaborations to the ground truth generated by DEVELOP and DeLinker-Path, respectively. DEVELOP was able to sample the 3-methyl-benzamide ground truth elaboration, attaining an SC_{RDKit} score of 1.

the elaborations generated by DEVELOP which passed the 2D filters, only 25% of elaborations contained an aromatic group connected to the fragment by a methylene linker, compared to 85% of elaborations proposed by DeLinker-Path. A potential reason for this is that the pharmacophoric constraint supplied to DeLinker-Path explicitly encodes the path distance between the fragment exit vector and the first aromatic atom, making it easy for DeLinker-Path to learn to use this information when training compared to the 3D constraints provided to DEVELOP.

The second example concerned a difluoroanisole elaboration, requiring the generative models to construct a benzene ring with 3 substituents. This example was challenging for both DeLinker-Path and DEVELOP, with neither able to successfully recover the ground truth elaboration. However, of the elaborations which passed the 2D filters, DEVELOP generated eleven elaborations which contained an aromatic ring with 3 substituents and one which matched the ground truth substitution pattern,

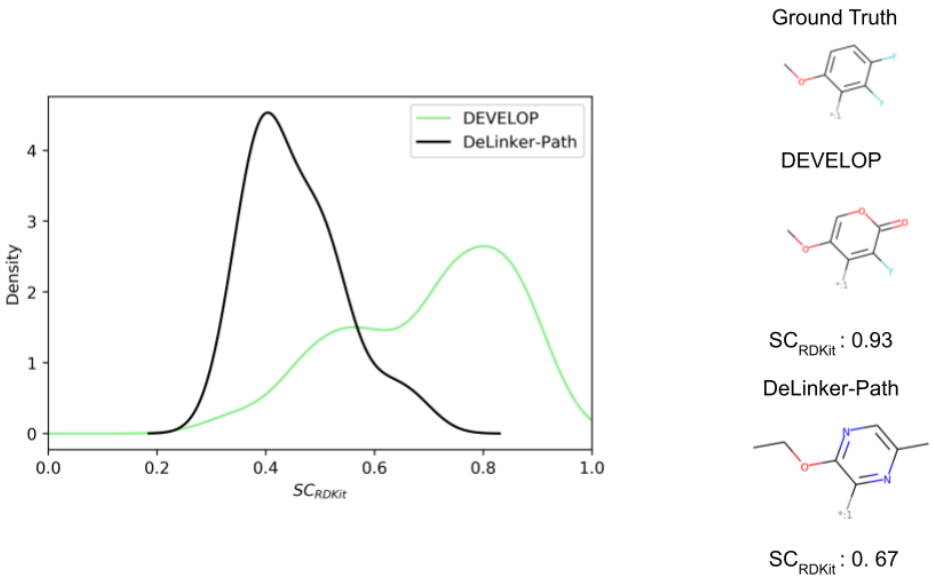

Figure 3.5: Left: Distribution of SC_{RDKit} scores obtained by DEVELOP and DeLinker-Path on a single example in the PDBBind test set. Note that the plotted distributions do not exactly match with the maxima of each distribution as they were obtained using kernel density estimation. Right: Most similar elaborations to the ground truth generated by DEVELOP and DeLinker-Path, respectively.

whereas DeLinker-Path was only able to generate one elaboration with an aromatic ring and three substituents and none which exactly matched the ground truth substitution pattern. Figure 3.5 shows the distribution of SC_{RDKit} scores attained by both generative models, with DEVELOP generating a larger proportion of highly similar elaborations to the ground truth. A possible explanation for the superior performance of DEVELOP on this example is that the 3D pharmacophoric constraints were able to naturally capture the topology of the ground truth elaboration, whereas the path lengths supplied to DeLinker-Path did not help the model realise that the ground truth elaboration was a single aromatic ring with multiple substituents.

The above results demonstrate that the different pharmacophoric constraints provided to DEVELOP and DeLinker-Path aid them in generating complementary sets of elaborations, and that one may be better suited than the other to generate a set of highly similar elaborations, depending on the composition of the ground truth elab-

oration. A possible future area of investigation would be to combine the constraints provided to each model into a single generative model.

3.4.3 DEVELOP Case Study

We further demonstrate the applicability of DEVELOP to R-group optimisation via a case study derived from the literature. Borkin et al. (2016) developed a thienopyrimidine class of compounds to block the protein–protein interaction between menin and mixed lineage leukemia (MLL) fusion proteins. This interaction plays an important role in acute leukemias with MLL translocations, making this an important drug target. The authors' previous work(Borkin et al., 2015) had led to the identification of a highly potent menin–MLL inhibitor (IC_{50}=31 nM, GI_{50}=0.55 μM, PDB ID: 4X5Z) but required further improvement of cellular activity and drug-like properties to develop compounds with potential therapeutic value. This was achieved via structure-based optimisation of substituents introduced to the indole ring (Figure 3.6a).

Following optimisation of several positions, the most potent compound displayed almost a seven-fold improvement in affinity in MLL-AF9 cells (GI_{50}=83 nM, PDB ID: 5DB3, Figure 3.6b, right), while other highly potent compounds demonstrated favourable drug-like properties, such as significant improvements in selectivity, reduced lipophilicity, and bioavailability.

The most significant modification to the original compound was the optimisation of the hydrogen bond interactions with Glu363 and Glu366 on menin. The indole nitrogen in the original molecule was involved in a hydrogen bond with the side chain of Glu363 but was partially solvent exposed and was not forming interactions with Glu366 (Figure 3.6a). This led the authors to explore a variety of substituents containing hydrogen bond donors. Two potent substitutions were an acetamide group (Figure 3.6b, left) and 4-methylpyrazole (Figure 3.6b, right).

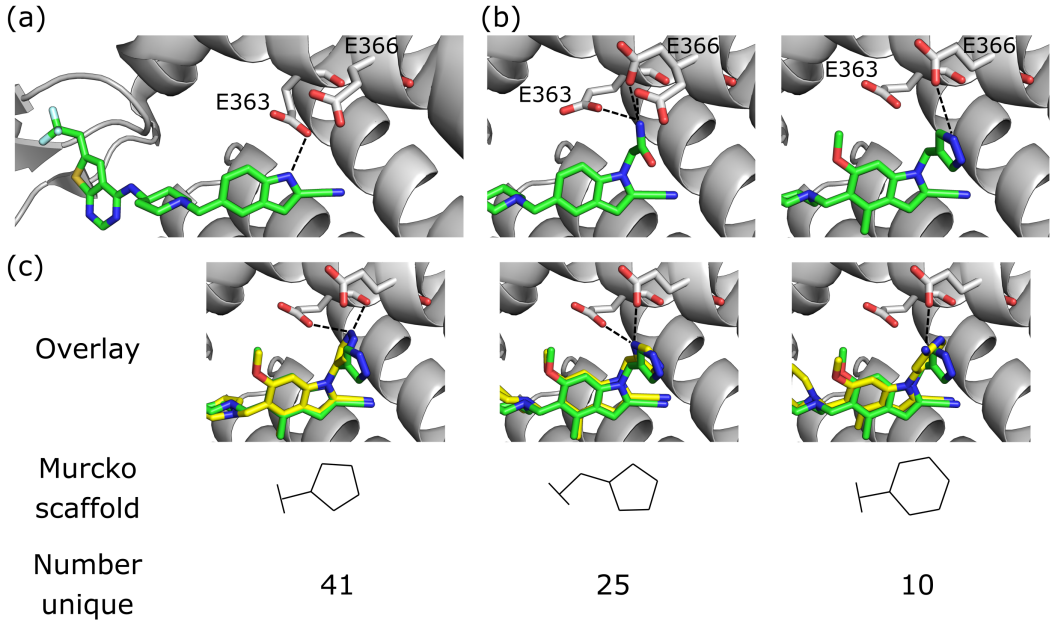

Figure 3.6: R-group optimisation case study. (a) Crystal structure (PDB ID 4X5Z) of the initial complex bound to menin. (b) Structure of two of the most potent optimised compounds (PDB IDs left 5DB2, right 5DB3). The dashed lines represent key interactions. (c) Overlay of the most potent optimised compound (green carbons, PDB ID 5DB3) and several compounds generated by DEVELOP (yellow carbons) that make similar hydrogen bonding interactions (dashed lines).

We investigated the ability of DEVELOP to propose R-groups that met the design hypothesis described in Borkin et al. (2016). In particular, we sought to design both aromatic and non-aromatic hydrogen bond donor groups that were able to make similar interactions to the R-groups that were experimentally tested. We derived 3D pharmacophoric profiles from the ligands in PDB IDs 5DB2 and 5DB3 to serve as input to DEVELOP. For the pharmacophoric profile derived from 5DB2, we generated 1000 R-groups with a maximum of four, five, and six atoms, whilst for the pharmacophoric profile derived from 5DB3 we generated 1000 molecules with a maximum of five, six, and seven atoms.

DEVELOP successfully recovered both of the experimentally-verified R-groups while generating many alternative molecules that could form similar interactions with menin. All methods were able to recover the acetamide R-group (Figure 3.6b, left). However, DEVELOP produced substantially more examples that matched the pharmacophoric profile (455) compared to both DeLinker-Counts (327) and DeLinker (103). All methods were also able to recover the 4-methylpyrazole R-group, although this elaboration was only generated once by DeLinker and DeLinker-Counts, compared to 61 times by DEVELOP. While generating the same molecule a large number of times is not necessarily desirable, the increased frequency of generation of the ground truth elaboration in this case has several benefits. First, the true R-group is one of a limited number of elaborations that matches the pharmacophores. Thus when this constraint is provided to the model, we would hope that it generates molecules that meet this constraint more frequently. Second, since all three generative models are stochastic, if a given molecule is only generated a small number of times, there is the possibility this is due to chance and this molecule would not be discovered in another instance. To exemplify this point, we repeated the generation step for the case study 10 times for each method and calculated in how many cases we recovered the 4-methylpyrazole R-group. In all cases, DEVELOP recovered the ground truth

molecule, whereas DeLinker and DeLinker-Counts only recovered the ground truth in 5/10 and 4/10 cases, respectively. Finally, we note that DEVELOP generated this elaboration with an overall frequency of c. 2% (61/3000), which represents a small fraction of all generated molecules. In addition, 237 of the elaborations generated by DEVELOP contained an aromatic system with a donor group linked to the indole via a methylene group compared to 50 for DeLinker and 11 for DeLinker-Counts.

We next sought to assess the alternatives to the pyrazole R-group (Figure 3.6b, right) that were proposed by DEVELOP. To validate the molecules proposed by DEVELOP, we docked the generated molecules containing an aromatic system and at least one donor group using GOLD (Verdonk et al., 2003) and checked whether the docked pose formed hydrogen bonding interactions with Glu363 or Glu366. Three elaborations, together with their Murcko scaffolds, are shown in Figure 3.6c (yellow carbons) overlayed with the pyrazole R-group (green carbons). All of the examples appear to fit within the pocket and were able to form hydrogen bonds with Glu363 or Glu366, consistent with the stated design hypothesis.

Finally, we scored the generated molecules which satisfied the specified pharmacophoric profiles using the smina (Koes et al., 2013) version of AutoDock Vina (Trott and Olson, 2010). For the first pharmacophoric profile, around one third of the molecules proposed by all methods obtained a predicted binding affinity greater than or equal to the corresponding ground truth elaboration, leading to DEVELOP proposing 152 such molecules, DeLinker 34 and Delinker-Counts 115 (Figure B.1). For the second pharmacophoric profile, DEVELOP (171 from 237) significantly more often (both as a proportion and in absolute terms) proposed molecules which satisfied the second pharmacophoric profile and had a predicted binding affinity at least as great as the ground truth compared to DeLinker (17 from 50) and DeLinker-Counts (5 from 11) (Figure B.2).

3.5 Conclusion

In this chapter we demonstrated that the imposition of pharmacophoric constraints allow greater control over the elaborations generated by a scaffold elaboration model. On the challenging PDBBind test set, both DEVELOP and DeLinker-Path were able to recover the ground truth elaboration considerably more often than the unconstrained DeLinker and Scaffold-Decorator models, as well as being better able to generate elaborations that exhibited high 3D similarity to the ground truth.

Compared to other approaches which allow users to incorporate existing actives into scaffold elaboration models (e.g. (Arús-Pous et al., 2020)), the 3D pharmacophoric constraints used by DEVELOP have the advantage that they are highly interpretable, allowing practitioners to understand the rationale behind the generated elaborations. Instead of conditioning on a set of pharmacophoric constraints, several recent works (e.g. (Skalic et al., 2019b; Ragoza et al., 2022; Green et al., 2021) have proposed methods which condition on the structure of the binding site and can generate molecules 'in 3D'. Whilst this is a promising direction of research, such methods must be trained using protein-ligand structures, meaning that there is a far smaller set of examples which can be used to train them, making it harder for the models to learn chemical principles. In addition, the imposition of protein structure into a model greatly increases its dimensionality, potentially inducing a risk of overfitting.

Whilst our experiments indicate that DEVELOP is a promising tool for *in silico* R-group optimisation, it is dependent on the existence of a known active to derive the 3D pharmacophoric constraints. This potentially hinders its applicability as a tool for elaborating fragments identified in a fragment screen of a novel target, as it might not be clear where the 3D pharmacophoric constraints should be specified.

We have made the code for DEVELOP available at `https://github.com/oxpig/DEVELOP`. Whilst the code is relatively straightforward to download and use for users

with a moderate amount of programming experience, the command line inteface might prove challenging to use for practitioners who are new to programming. In the next chapter we describe the development of a web application with an intuitive user interface which allows users to easily generate molecules using DeLinker and DEVELOP.

Chapter 4

Development of a Web Application for DeLinker and DEVELOP

4.1 Background

As discussed in the Introduction, drug discovery is an expensive, time consuming process, and computational techniques show promise as a means for developing drugs more quickly and cheaply (Ou-Yang et al., 2012). A factor that hinders the uptake of computational tools by the drug discovery community is that they are often challenging to install and use, particularly for scientists with limited programming experience. One way which computational techniques can be made more accessible is by the development of simple-to-use web applications, which allow prospective users to use a simple GUI and require no installation.

Within the context of deep generative models, which allow users to rapidly generate libraries of molecules to screen against a target, a number of authors have created webservers to facilitate easier use of their tools. Green and Durrant (2021) proposed a browser application for DeepFrag (Green et al., 2021), a structure-based model for fragment elaboration, whilst Skalic et al. (2019b) made their shape-based gen-

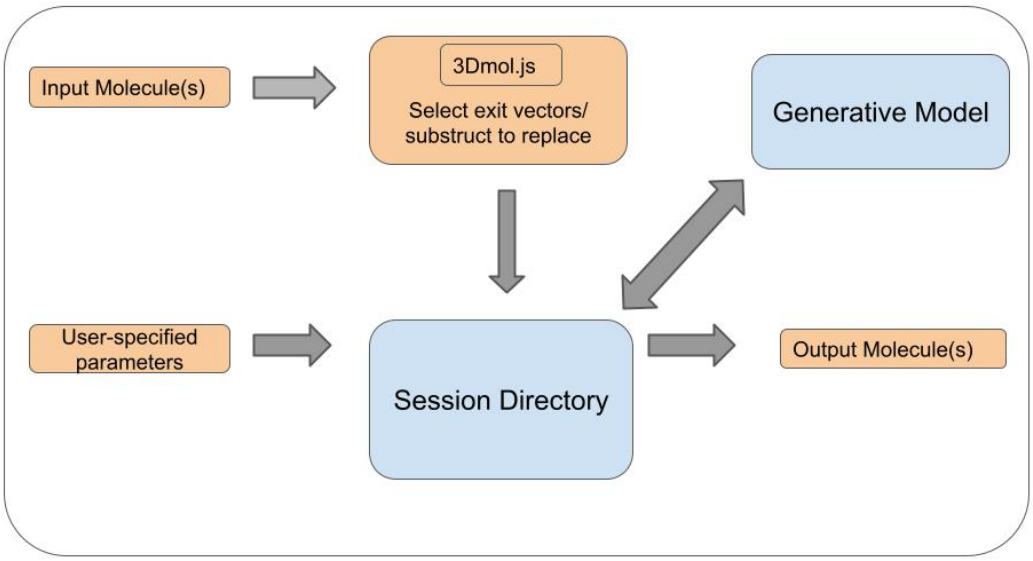

Figure 4.1: Schematic of our Web Application. Orange boxes denote processes which are carried out in the user's browser, whilst Blue boxes denote processes which are stored on our server.

erative model available as a webserver. In this chapter, we report the development a web application for two deep generative models developed in the Oxford Protein Informatics Group, DeLinker (Imrie et al., 2020) and DEVELOP (Imrie et al., 2021). DeLinker, described in detail in the Introduction, is a tool for fragment linking where the user provides a pair of fragments as input and DeLinker generates a set of linkers between the specified exit vectors. DEVELOP is a tool for scaffold replacement and R-group optimisation, described in Chapter 3; the user specifies a substructure which they wish to replace and, by leveraging a set of 3D pharmacophoric constraints, DEVELOP replaces the specified substructure with alternative substructures which are capable of making an equivalent set of interactions with the protein.

4.2 Application Overview.

A schematic of the application is shown in Figure 4.1. The application is powered by Flask (Grinberg, 2018), which takes the user input, provided through the 3D

molecule viewer 3Dmol.js (Rego and Koes, 2015) and text fields, and saves them in a directory created for the specific session. The inputs are then loaded into to the specified generative model (implemented in TensorFlow 1.10 (Abadi et al., 2015)), which generates a set of molecules and writes them to the session directory and loaded onto the web page so that they can be viewed and downloaded by the user. We provide details on each step of this process below.

4.3 3D Molecule Viewer

The user begins a session by loading a number of molecules into the 3D molecule viewer, 3Dmol.js (Rego and Koes, 2015). For DeLinker, the user loads one or two sdf files. If one molecule is loaded, then the user selects the atoms in the molecule which they wish DeLinker to replace (Figure 4.2a). By contrast, if two molecules are loaded, the user selects a single atom from each to act as an exit vector (Figure 4.2b). As DEVELOP requires an existing bound molecule with which to derive pharmacophoric constraints, the user should load a single molecule and select a substructure to be replaced; DEVELOP will then attempt to generate scaffolds/R-groups containing functional groups which are capable of making the same interactions as the selected substructure.

To allow the user to incorporate protein-specific information into their choice of inputs, users can additionally load a PDB file into the molecule viewer.

4.4 Preparing a Run

Once the user has selected a set of atoms, either as a pair of exit vectors if two small molecules have been loaded, or a substructure to be replaced if a single small molecule has been loaded, they should click the '`Confirm Selected Atom`' button, which stores the 3D position of the selected atoms. The user can alternatively click

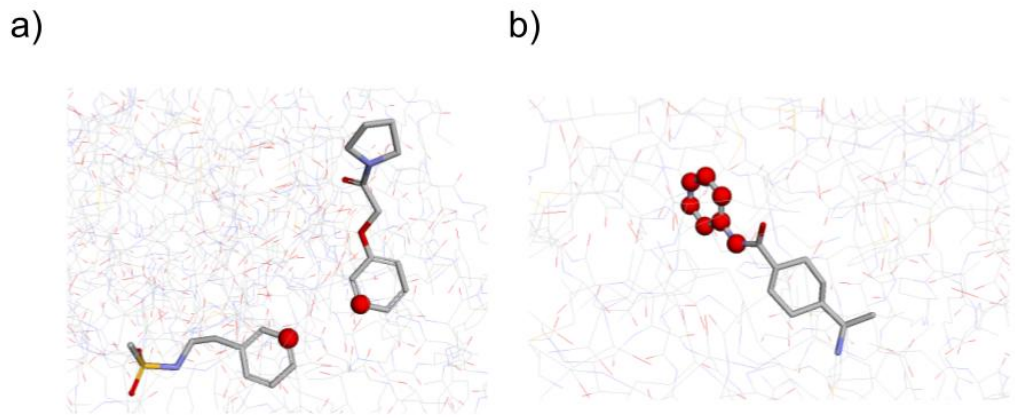

Figure 4.2: Examples of using 3Dmol.js to prepare input. a) Two fragments have been loaded as input, so the user must specify a single atom on each fragment to serve as the exit vectors. b) A single molecule has been provided as input and the user must select a substructure which DEVELOP will use to derive a set of pharmacophoric constraints.

'`Reset Image`' button to reload the 3D molecule viewer so that alternative atoms can be selected.

The application allows the user to specify two parameters for the model run: First, the user can specify the requested linker/elaboration length, allowing control over the size of the molecules which will be proposed. If a single molecule is provided as input, the default linker/elaboration length will be the size of the substructure that is selected to be replaced. For DeLinker, if two fragments are provided, there is no default linker length and this must be specified by the user.

Second, the user may specify the number of molecules they wish to be generated. Although DeLinker (Imrie et al., 2020) and DEVELOP (Imrie et al., 2021) were both shown to be capable of generating a diverse range of novel, synthetically accessible molecules, as the molecules are generated by a stochastic sampling process, a number of the generated molecules will be duplicates. These can be easily removed from the

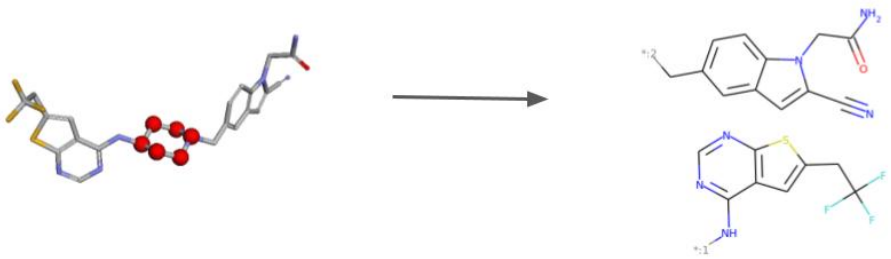

Figure 4.3: After the specifying the input to the generative model, the web application allows the user to inspect the fragment(s) that will provided to the model to ensure correct specification.

model output.

Before generating a set of molecules, the user can click the '`Check Input`' button, which generates an image of the fragment(s) which will be provided to the generative model (Figure 4.3).

4.5 Generating molecules

Once all of the model inputs have been submitted, the user can generate molecules by clicking '`Generate Linkers`' or '`Generate R-Groups`'. Table 4.1 illustrates the approximate amount of time DeLinker requires to generate 250 elaborations of length 4-10 with a single CPU core. The user can load an image of the most commonly generated molecules, or download the full list of generated molecules, written in SMILES format to a `csv` file.

Table 4.1: Time required for DeLinker to generate 250 linkers of various lengths for the example shown in Figure 4.3. As expected, DeLinker takes longer to run when sampling linkers of longer lengths, but it is able to rapidly generate hundreds of linkers which can be inspected and downloaded by the user for downstream use.

Linker Length	Runtime (secs)
4	29.31
5	31.82
6	37.67
7	41.88
8	45.93
9	49.95
10	54.33

4.6 Conclusion

We have developed a Flask-based webserver for generative models developed within the Oxford Protein Informatics Group. The webserver uses an intuitive user interface to allow the user to easily specify the required inputs to DeLinker or DEVELOP. In addition, our webserver allows users to generate molecules without downloading any software, facilitating easy access for members of the drug discovery community with limited exposure to computational techniques. The code for the web application can be found at `https://github.com/tomhadfield95/WebApp`, and the DeLinker and DEVELOP code can be found at `https://github.com/oxpig/DeLinker` and `https://github.com/oxpig/DEVELOP`, respectively.

In the previous chapters we have focused on developing methods for generating elaborations which are highly similar to a ground truth, first via reinforcement learning, then through the imposition of pharmacophoric constraints. Whilst these approaches enabled the generative models to sample from a focused region of chemical space, their use depends on the existence of a compound which binds with high affinity to the target protein, hindering their applicability in scenarios where no such

molecules exist. In the next chapter we present a method for automatically extracting a set of pharmacophoric constraints directly from the target protein's structure, and using them to generate elaborations designed to make high energy interactions with the protein.

Chapter 5

Incorporating Target-Specific Pharmacophoric Information Into Deep Generative Models For Fragment Elaboration.

5.1 Preface.

This chapter contains work described in the following publication: **Thomas E. Hadfield**, Fergus Imrie, Andy Merritt, Kristian Birchall and Charlotte M. Deane (2022). Incorporating Target-Specific Pharmacophoric Information Into Deep Generative Models For Fragment Elaboration. *Journal of Chemical Information and Modeling*. doi:10.1021/acs.jcim.1c01311. All work described in this chapter is my own unless explicitly stated otherwise.

An open-source implementation of this method is available at `https://github.com/oxpig/STRIFE`.

The structure of a protein is an essential consideration when designing molecules

which can bind with high affinity to produce a desired therapeutic effect. A ligand must be able to fit within the binding pocket and form a variety of different interactions with the protein and medicinal chemists carefully consider the structure of a protein when proposing ligands. However, although early generative models (e.g. (Gómez-Bombarelli et al., 2018; Liu et al., 2018; Jin et al., 2018; Olivecrona et al., 2017)) were able to produce valid, "druglike" molecules, they were unable to account for the target structure when generating molecules. As such, the vast majority of molecules proposed by such models would not bind to the target. Whilst one could theoretically sample an extremely large number of molecules from a structure-unaware generative model and computationally screen them to identify likely binders, this would quickly become computationally prohibitive in practice.

The development of generative models which can condition on the target structure is therefore a critical step in *in silico* molecular design. In the previous chapter we considered the problem of deriving pharmacophoric constraints from a known-active molecule to facilitate R-group optimisation or scaffold replacement. Whilst DEVELOP's (Imrie et al., 2021) generative model could be used to elaborate or link fragments identified in a fragment screen, in practice its dependence on the existence of a known active hinders its applicability to novel targets, as it would require the user to manually specify the pharmacophoric constraints. In this chapter we propose a novel algorithm for directly extracting information from the target's structure to generate a set of pharmacophoric constraints. The derived constraints can then be passed to a generative model with the aim of generating molecules which can bind with high affinity to the target.

We found that the inclusion of structural information allowed our model, STRIFE, to generate elaborations which were more ligand efficient than published generative models (Polishchuk, 2020; Arús-Pous et al., 2020) which do not incorporate target-specific information. STRIFE also outperformed an alternative structure-aware deep

generative for fragment elaboration (Green et al., 2021), and its structure processing step has the advantage of being easily interpretable, allowing users to understand the rationale behind its elaborations.

5.2 Background

Recent years have seen significant interest in developing machine learning models to rapidly generate and screen large numbers of molecules as potential drug candidates. Different authors have employed a range of different molecular representations, including SMILES (Gómez-Bombarelli et al., 2018), graphs (Liu et al., 2018), SELFIES (Krenn et al., 2020) and atomic density grids (Ragoza et al., 2022), and a number of different deep learning architectures, such as Generative Adversarial Networks (Guimaraes et al., 2017), Variational Autoencoders (Jin et al., 2018) and Recurrent Neural Networks (Olivecrona et al., 2017). With the aim of generating molecules with an optimal set of properties, several approaches have been proposed for multi-objective optimisation, including gradient descent (Gao et al., 2020), reinforcement learning (Olivecrona et al., 2017), Bayesian Optimisation (Jin et al., 2018) and Particle Swarm Optimisation (Winter et al., 2019).

Whilst early generative models typically generated a molecule 'from scratch', several authors have recently proposed deep learning-based methods to help improve the efficiency of fragment-to-lead campaigns. Graph-based approaches for scaffold elaboration were proposed by Lim et al. (2020) and Li et al. (2019), which provide a model with a fragment and generate a set of molecules which contain the original fragment as a substructure, whilst Arús-Pous et al. (2020) proposed a SMILES-based (Weininger, 1988) model, Scaffold-Decorator, which gave the user the ability to decide which atoms in the fragment should be used as an exit vector, allowing greater control over the types of elaborations generated. However, none of the above ap-

proaches allow for the specification of a preferred elaboration size, which, combined with their inability to account for protein structure when generating elaborations, means they cannot ensure that elaborations made by the model would be of an appropriate size to fit within the binding pocket. In the previous chapter, we proposed DEVELOP, a fragment-based generative model for linking and growing which built on the DeLinker (Imrie et al., 2020) model. DEVELOP allows the specification of pharmacophoric constraints and linker/elaboration length, providing a greater degree of control over the resulting molecules. In concurrent work to DEVELOP, Fialková et al. (2021) proposed LibINVENT, an extension to Scaffold-Decorator (Arús-Pous et al., 2020) which can be used to design core-sharing chemical libraries using only specific chemical reactions. LibINVENT also allows users to generate molecules with high 3D similarity to an existing active molecule via reinforcement learning. However, as with DEVELOP, LibINVENT is reliant on the existence of known actives to generate targeted sets of molecules, making it a more suitable tool for R-group optimisation than for designing compounds against a novel target.

Orthogonal to the generative approaches described above, several recent papers have proposed database-based approaches to compound design. A recent method, CReM, (Polishchuk, 2020) is based on the idea that a fragment within the context of a larger molecule can be interchanged with another fragment that has been observed to have the same local context in another molecule. CReM identifies potential elaborations by searching a database of molecules for fragments which have the same local context as the specified exit vector. Other recent database-based approaches incorporate protein-specific information: FragRep (Shan et al., 2020) takes a protein and ligand as input and enumerates modifications to the ligand by cutting the ligand into fragments and replaces a fragment with similar fragments from a database which would preserve the same protein-ligand interactions, whilst DeepFrag (Green et al., 2021) uses a structure-aware convolutional neural network to select the most

appropriate elaborations from a database of possible elaborations.

For the task of generating molecules 'from scratch', a number of authors have proposed generative models which extract information directly from the protein. Skalic et al. (2019b) used a GAN (Goodfellow et al., 2014) to generate ligand shapes complementary to the binding pocket which were then used to generate potential molecules by employing a shape-captioning network. Masuda et al. (2020) encoded atomic density grids into separate latent representations for ligand and protein and trained a model to generate 3D ligand densities conditional on the protein structure, which were then translated into discrete molecular structures. Whilst both papers demonstrated that the ligands generated by their respective models were dependent on the learned structural representations, the models do not facilitate the specification of a design hypothesis provided by a human expert or provide intuition as to why the generative model is proposing the molecules it is. Kim et al. (2020) used water pharmacophore models to learn the location of key protein pharmacophores which were then used to construct a training set of molecules with complementary pharmacophores. Whilst this approach would more readily integrate into standard drug-discovery efforts, it requires the training of a separate deep learning model for every target, as each target requires a training set of compounds which match the water pharmacophores.

In this work we propose STRIFE (**Str**ucture **I**nformed **F**ragment **E**laboration), a generative model for fragment elaboration which extracts interpretable and meaningful structural information from the protein and uses it to make elaborations. This is different to all existing fragment-based generative approaches which either extract information implicitly from known ligands or do not make use of any protein-specific information when generating molecules.

To allow straightforward integration into fragment-to-lead campaigns, STRIFE is readily customisable; in addition to the design hypotheses extracted directly from the protein, we provide a simple-to-use functionality which allows users to specify

their own design hypotheses and generate elaborations with the aim of satisfying a desired pharmacophore. In a large-scale evaluation derived from the CASF-2016 set (Su et al., 2018), we show that STRIFE offers substantial improvements over existing fragment-based models (Arús-Pous et al., 2020; Polishchuk, 2020). We further demonstrate the applicability of STRIFE to real-world FBDD campaigns through two fragment elaboration tasks derived from the literature. In the first, we make elaborations to a fragment bound to N-myristoyltransferase, a key component in rhinovirus assembly and infectivity, and show that STRIFE is able to generate several elaborations that are strikingly similar to a highly potent inhibitor (Mousnier et al., 2018). To demonstrate how user-specified design hypotheses can be incorporated into STRIFE, we consider the fragment-inspired small molecule inhibitor of tumour necrosis factor reported by O'Connell et al. (2019). In this example, the elaboration proposed by O'Connell et al. (2019) induces a substantial movement in a Tyrosine side chain. We manually specified a design hypothesis to explore side-chain flexibility, and successfully recovered the elaboration proposed by O'Connell et al. (2019), as well as a range of other elaborations which were predicted to induce a similar movement in the Tyrosine side chain.

5.3 Methods

We present our deep generative model for fragment elaboration, STRIFE, which requires the user to specify a target protein, a bound fragment, and the fragment exit vector. In Chapter 3, we demonstrated how the imposition of pharmacophoric constraints allowed a substantial degree of control over the types of functional groups added to a fragment. STRIFE builds on the approach proposed Chapter 3, where the pharmacophoric constraints were extracted from existing active molecules, by extracting pharmacophoric constraints directly from the protein, thereby extending

its applicability to a much broader range of targets. Pharmacophoric information is extracted by calculating a Fragment Hotspot Map (Radoux et al., 2016) (FHM), which describes regions of the binding pocket that are likely to make positive contribution to binding affinity. STRIFE then identifies pharmacophoric constraints which are likely to place a pharmacophore within a matching hotspot region and uses the pharmacophoric constraints to generate elaborations.

5.3.1 Fragment Hotspot Maps

We calculate FHMs using the Hotspots API (Curran et al., 2020) which implements the algorithm described by Radoux et al. (2016); in this work all FHMs were calculated using the default parameters given by Curran et al. (2020). An FHM is calculated as follows: Atomic Propensity Maps are calculated using SuperStar (Verdonk et al., 1999), which defines a grid covering the protein with equally spaced points 0.5 Å apart, and uses data from the Cambridge Structural Database (CSD) (Groom et al., 2016) to assign a propensity for a given probe type at each grid point; If an interaction between two groups at a certain distance and angle is particularly favourable then it will occur more frequently in structures stored in the CSD and be assigned a higher propensity score. Once an Atomic Propensity Map has been calculated an FHM is derived by first weighting the scores assigned to each grid point in proportion to how buried in the protein the grid point is.

The FHM scores are then calculated by using small chemical probes which take the form of an aromatic ring with different atoms in the substituent position; for the apolar hotspot maps the substituent is a methyl group, whilst for the acceptor and donor hotspot maps the substituent is a carbonyl and amine, respectively. The probes are translated to all grid points with weighted propensity scores above 15 and randomly rotated 3000 times about the center of the substituted atom. For each pose, each atom receives a score read from the weighted propensity map and the

probe scores are calculated as the geometric mean of the atom scores; as an atom receives a score of zero if it clashes with the protein, the geometric mean gives a score of 0 to any pose which clashes.

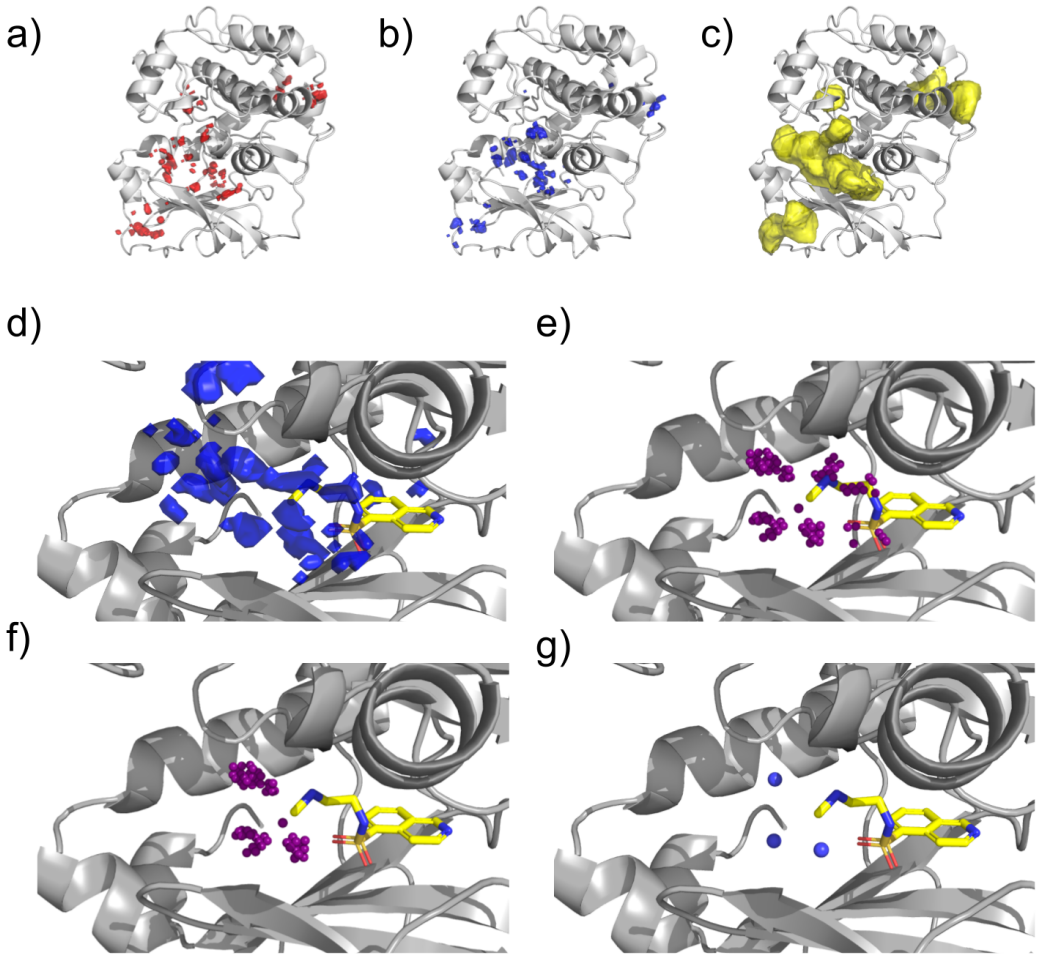

Figure 5.1: Processing Fragment Hotspot Maps. a) Acceptor Hotspot Map. b) Donor Hotspot Map. c) Apolar Hotspot Map. A matching pharmacophore placed within a hotspot has a chance of making a disproportionate contribution to binding affinity. d) An unprocessed donor hotspot map in the vicinity of the fragment of interest. e) Each sphere represents a voxel in the hotspot map. Voxels which are too far away from the fragment exit vector are discarded. f) Voxels which are closer to another fragment atom than the exit vector are removed. g) Voxels are clustered based on their position. STRIFE attempts to generate elaborations such that a matching ligand pharmacophore is in close proximity to a cluster centroid.

FHMs have a number of attractive properties. As only grid points with an above-threshold weighted propensity score are sampled, and the propensity scores are weighted by how buried in the protein they are, regions of the protein which are overly exposed are unlikely to be identified as hotspot regions. Additionally, because probe poses which clash with the protein attain a score of zero, any region identified as a fragment hotspot must be able to accommodate a molecule of reasonable size, meaning that the risk of attempting to satisfy a pharmacophore identified by the FHM which cannot be accessed by an elaboration is reduced.

5.3.2 FHM Processing.

For a protein target, STRIFE uses FHMs to guide the generative model in the placement of functional groups which can interact with the target. As the different hotspot maps are used for different purposes, they are processed slightly differently (Figure 5.1): the acceptor and donor hotspots are used to identify desirable pharmacophoric constraints, whilst the apolar maps are used to verify that the fragment is located in an appropriate binding site. For the apolar maps, we identify all grid points which have a value greater than 1 and discard all other points. Similarly, for the acceptor and donor maps, we retain all grid points which have a value greater than 10. Whilst Radoux et al. (2016) reported that values greater than 17 were generally predictive of fragment binding, we selected 10 as a threshold to obtain wider coverage; this parameter is simple to change to restrict the search to higher quality hotspots.

To process the acceptor and donor maps, all points which are less than 1.5 Å or greater than 5 Å from the fragment exit vector are discarded, to allow for elaborations of appropriate length; these distance thresholds were chosen to reflect the iterative nature of a fragment-to-lead campaign, where practitioners typically make a succession of small elaborations, but they can be altered by the user to admit longer or shorter elaborations.

A greedy clustering algorithm is employed to identify contiguous hotspot regions as follows: A cluster is initialised as a single point and all unclustered points which are within 1 Å of the grid point are added to the cluster. For each point in the cluster, the distance to all remaining unclustered points is calculated and any points which are within 1 Å are added to the cluster until no unclustered points can be added. Once a cluster has terminated, a new cluster is defined by selecting a single unclustered point, until all points have been assigned to a cluster. For each hotspot cluster, centroids are defined by computing the mean position of the points in the cluster. To reduce redundancy, if two cluster centroids are closer than 1.5 Å apart the cluster centroid corresponding to the smaller cluster is deleted. In addition, if a cluster is smaller than eight points it is removed, unless no clusters of eight or more points exist, in which case smaller clusters are retained.

We use the apolar maps to conduct a final filtering step, adopting the heuristic that a molecule which is entirely contained within an apolar hotspot region has a better chance of binding to the protein. Therefore, if an acceptor or donor cluster centroid is not contained within a hotspot region then it is filtered out. Additionally, if all fragment atoms are not contained within an apolar hotspot then we consider the fragment to be unsuitable for elaboration and terminate the algorithm. Whilst this might appear to be overly restrictive, in practice the apolar hotspot maps typically cover the majority of binding sites in a target and this filtering step can be easily negated if the user believes that a fragment is a suitable candidate for elaboration.

The final output of the processing scheme are the 3D coordinates of the remaining cluster centroids from the acceptor and donor maps (hereafter "pharmacophoric points"). In the subsequent molecule generation steps, our aim is to generate elaborations which place matching functional groups in close proximity to the pharmacophoric points. Whilst the above pipeline automates the process of defining pharmacophoric points, we also provide a simple-to-use functionality for users to define

their own pharmacophoric points, allowing them to pursue a range of different design hypotheses (see Section 5.3.6).

Next, we describe how STRIFE uses a set of pharmacophoric points to generate elaborations with complementary pharmacophores to the target.

5.3.3 Generative Model.

The generative model employed by STRIFE is similar to DEVELOP, described in Chapter 3, where the generative process is based upon the Constrained Graph Variational Autoencoder framework proposed by Liu et al. (2018). STRIFE differs from DEVELOP in the structural information $\boldsymbol{D}$ provided to the model when decoding molecules (see Section 5.3.4). Given a fragment, $\boldsymbol{f}$, and structural information, $\boldsymbol{D}$, elaborations are generated as follows: Representing $\boldsymbol{f}$ as a graph, each node v is assigned an $h-$ dimensional vector representation $\mathbf{z}_v$ and corresponding label l_v, denoting the atom type of the node. A set of K 'expansion nodes', $\mathbf{z}_{v_1}, ..., \mathbf{z}_{v_K}$ are generated by sampling from an $h-$dimensional standard normal distribution and each expansion node is assigned a label l_{v_k} by a linear classifier which takes $\mathbf{z}_{v_k}$ and $\boldsymbol{D}$ as input. The expansion nodes represent the possible atoms which may be appended to the fragment.

Starting from the fragment exit vector, the model samples a node to add to the graph from the set of expansion nodes. To choose whether to form a bond between node v and node u, we use a neural network which takes as input:

$$\phi^t_{v,u} = [t, s^t_v, s^t_u, d_{v,u}, \mathbf{H^0}, \mathbf{H^t}, \boldsymbol{D}]$$

Where:

- t is the number of time steps that have currently been taken.
- $s^t_v = [\mathbf{z^t_v}, \mathbf{l_v}]$ is the concatenation of latent vector and label at the t^{th} time step.

- $d_{u,v}$ is the graph distance between u, v
- $\mathbf{H^j}$ is the average of all latent vectors at the j^{th} time step.

After a new node has been added to the graph, a gated graph neural network (Li et al., 2015) is used to update the encodings for each node, to reflect its potentially altered neighbourhood. This iterative approach continues until termination where the final molecule is returned. A more detailed explanation of the generative model framework can be found in Section 1.4.2.1.

5.3.4 STRIFE Algorithm.

Above we described how STRIFE uses Fragment Hotspot Maps (FHMs) (Radoux et al., 2016) to obtain an interpretable representation of structural information and how, given a fragment, $\boldsymbol{f}$, and structural information, $\boldsymbol{D}$, we can generate elaborations to the fragment. Here we describe how these processes fit within the STRIFE algorithm. In particular, we outline how the 3D pharmacophoric points derived from the FHMs are converted to a representation of structural information, $\boldsymbol{D}$, which is used to generate elaborations.

The structural information $\boldsymbol{D}$ can be provided to the generative model in two different forms: The first is a coarse-grained pharmacophoric representation, where the model is simply provided with a vector containing the number of Hydrogen Bond Acceptors, the number of Hydrogen Bond Donors, and the number of Aromatic groups. The desired pharmacophoric profile of the generated elaborations can also be more precisely specified by adding the predicted path distances (the length of the shortest sequence of atoms connecting two points) from the exit vector to the pharmacophore, providing a greater degree of control over the types of elaborations made by the model. STRIFE utilises both of these representations of pharmacophoric information at different stages of the algorithm: In the Exploration phase (Figure 5.2a), STRIFE

uses the coarse-grained representation to generate a wide range of elaborations, which are then assessed for suitability. In the Refinement phase (Figure 5.2b), fine-grained pharmacophoric profiles are derived from the most suitable elaborations and are used to generate further elaborations. Additional details are provided below.

In a standard fragment elaboration campaign, where practitioners typically work in an iterative way, making small elaborations to a fragment which is then optimised before making additional elaborations to the optimised molecule. In this paper we demonstrate STRIFE generating elaborations which place a pharmacophore close to a single pharmacophoric point at a time. For example, if the set of pharmacophoric points contains one donor and one acceptor, STRIFE will attempt to produce a set of elaborations which include a donor in close proximity to the donor pharmacophoric point and a set of elaborations which place an acceptor in close proximity to the acceptor pharmacophoric point, but will not attempt to satisfy both pharmacophoric points simultaneously. STRIFE is capable of attempting to satisfy multiple pharmacophoric points simultaneously, but this is not recommended unless the pharmacophoric points have been manually specified or inspected by the user, as it may not be possible to simultaneously satisfy certain combinations of pharmacophores with a single elaboration. After obtaining a series of pharmacophoric points from the FHM, STRIFE proceeds as follows:

5.3.4.1 Exploration Phase.

STRIFE aims to generate a set of elaborations which contain functional groups in close proximity to a pharmacophoric point. To facilitate this, for each pharmacophoric point we predict the atom-length distance between the fragment exit vector and the pharmacophoric point using a trained support vector machine (Cortes and Vapnik, 1995). As the generative model requires the specification of a desired elaboration length, we use the atom-length prediction to control the length of elaborations pro-

posed by STRIFE. To allow for the inclusion of rings and side-chains in the elaboration, we use several different desired elaboration lengths; if the predicted atom distance is p, we generate elaborations with a requested length of up to $p+4$. As well as specifying a desired elaboration size, the generative model requires us to specify a desired pharmacophoric profile. In the Exploration phase we generate molecules using the coarse-grained pharmacophoric profile; as the coarse-grained pharmacophoric profile doesn't specify a desired path distance between the exit vector and the ligand pharmacophore, the pharmacophores in the elaborations proposed by the generative model will occupy a broad range of different positions in the binding pocket.

The proposed elaborations are filtered to ensure that they have the same pharmacophoric counts as provided in the coarse-grained pharmacophoric profile; we also apply the 2D filter proposed by Imrie et al. (2021) (see Section 5.3.7.1) to remove synthetically inaccessible molecules or elaborations containing PAINS (Baell and Holloway, 2010) substructures. The remaining molecules are docked using the constrained docking functionality in GOLD (Verdonk et al., 2003), where the structure of the fragment is provided as the constraint. Each molecule is docked 10 times and the top-ranked pose selected. For each top-ranked pose, we compute the distance between the 3D pharmacophoric point and a matching pharmacophore in the molecule. We then identify all molecules where the resulting distance is less than 1.5Å and select the five molecules for which the distance between pharmacophoric point and ligand pharmacophore is smallest. If less than five molecules exhibit a distance of less than 1.5Å, we select only molecules which fulfil this criteria.

5.3.4.2 Refinement Phase.

The molecules which exhibit a functional group in close proximity to a pharmacophoric point provide useful information, as they can be used to derive the more fine-grained representation of structural information which specifies the path dis-

tance between the exit vector and each ligand pharmacophore; as such, we refer to these molecules as "quasi-actives", because they play a similar role to known actives in existing generative models. Having obtained a set of quasi-actives for each pharmacophoric point and used them to derive a set of structural information vectors $\boldsymbol{D}_1, \boldsymbol{D}_2, ..., \boldsymbol{D}_n$, the user can either generate a fixed number of elaborations using each $\boldsymbol{D}_i$ or request a fixed total number of elaborations, where a structural information vector is randomly sampled from $\{\boldsymbol{D}_i\}_{i=1}^{n}$ for each elaboration. As before, the generated molecules are filtered and docked using the constrained docking functionality in GOLD (Verdonk et al., 2003). Finally, each unique molecule, m, is ranked by its ligand efficiency, computed as the docking score divided by the number of heavy atoms, allowing the user to quickly prioritise a small number of elaborations for consideration.

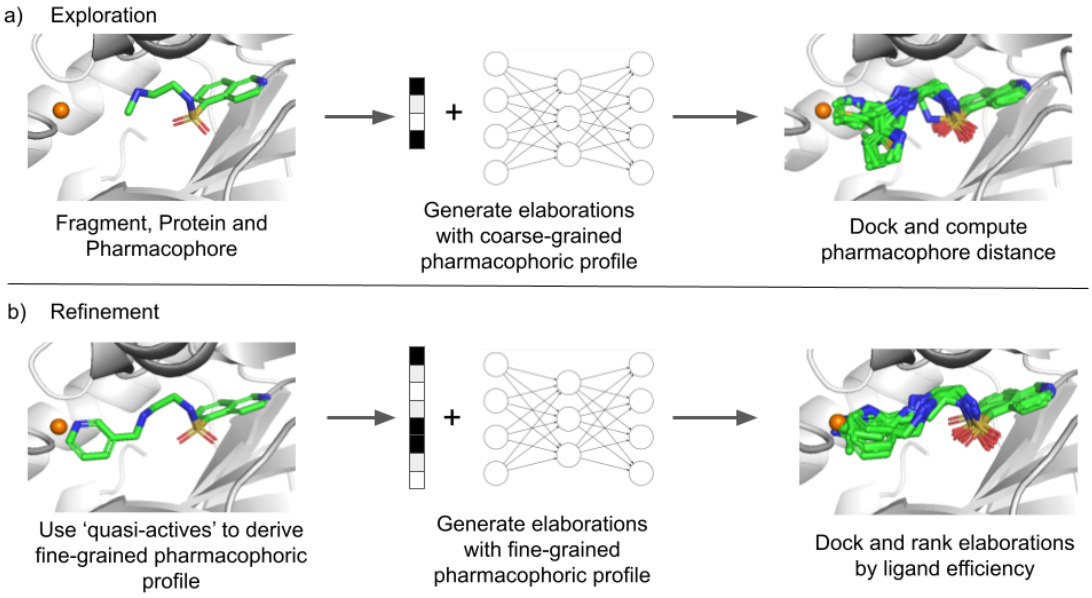

Figure 5.2: Illustration of how STRIFE generates elaborations which place pharmacophores close to a specified pharmacophoric point. a) STRIFE first generates elaborations using a coarse-grained pharmacophoric profile and docks them using the constrained docking functionality in GOLD (Verdonk et al., 2003). b) Elaborations which placed a matching pharmacophore in close proximity to the pharmacophoric point are used to derive a fine-grained pharmacophoric profile. STRIFE then generates elaborations using those pharmacophoric profiles; the resulting molecules are docked and ranked by their predicted ligand efficiency.

5.3.5 ADMET Filtering.

To reduce the likelihood of proposing molecules with undesirable ADMET characteristics, we provide an additional, optional filter that can be applied post hoc. The filter is based on the Quantitative Estimate of Druglikeness (QED) (Bickerton et al., 2012). The QED score is based on a range of factors (e.g. hydrophobicity, molecular weight) which are important ADMET considerations, so can be used to quickly flag molecules which might exhibit problematic ADMET characteristics.

As the QED associated with an elaborated molecule will be heavily dependent on the original fragment, we use the QED attained by the original fragment as a threshold for flagging a molecule proposed by STRIFE. In other words, a molecule is flagged if it is predicted to be 'less-druglike' than the original fragment. We also provide the option for the user to alter the flagging threshold.

5.3.6 Customisability.

Although STRIFE can automatically extract a set of pharmacophoric points from a protein, in a real-world drug discovery setting practitioners may wish to explore their own design hypotheses. To facilitate such usage, we provide a simple-to-use functionality which allows a user to manually specify the location of a pharmacophore in the context of the protein. The tool, shown in Figure 5.3, loads a lattice centered around the fragment exit vector into a molecule viewer: To manually specify their own pharmacophoric profiles, the user simply selects the lattice points corresponding to their desired pharmacophore location, saves the resulting object and runs STRIFE as usual.

5.3.6.1 Model Training.

To generate elaborations we used two generative models described in Chapter 3. In the Exploration phase, we used the DeLinker-Counts model (Section 3.3.2.1), whereas

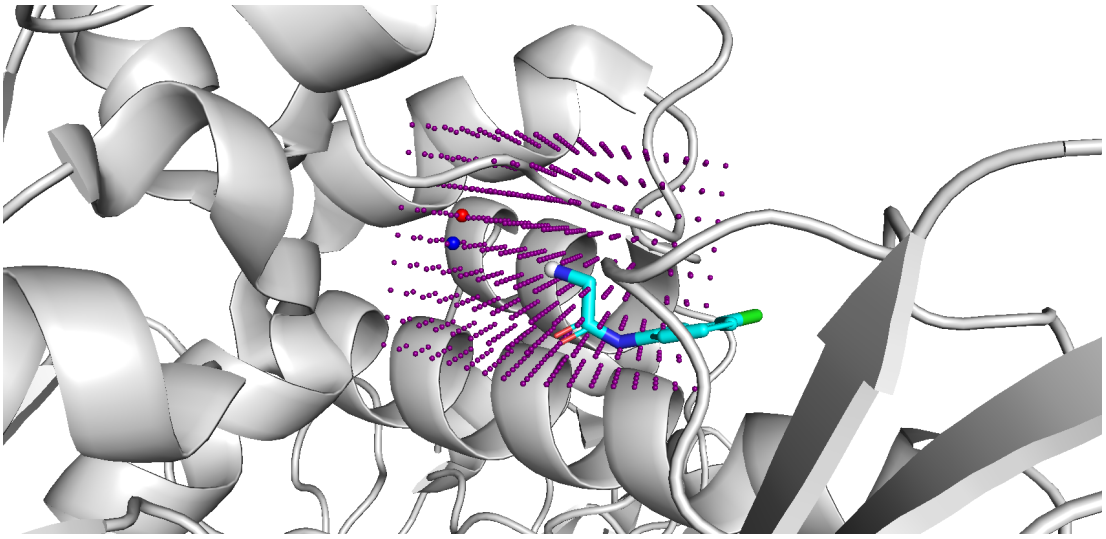

Figure 5.3: Example of how the pharmacophoric points provided to STRIFE can be customised using the molecule viewer PyMOL (Schrödinger, LLC, 2015). A lattice of points are centered about the fragment exit vector (denoted by the grey atom), and the user simply selects the point(s) they wish to denote as a pharmacophoric point and saves them in an SDF file. The red and blue points represent an Acceptor point and Donor point, respectively. STRIFE can then be run as usual and will attempt to make elaborations which places matching pharmacophores close to the user-specified pharmacophoric points.

in the Refinement phase we used the DeLinker-Path model (Section 3.3.2.2). We used the saved models used to attain the results in Chapter 3; i.e. the training set comprised a set of approximately 427,000 fragment-ground truth examples derived from ZINC (Sterling and Irwin, 2015).

5.3.7 Experiments.

We assessed the ability of our model to make appropriate elaborations using a test set derived from the CASF-2016 set (Su et al., 2018). This test set was constructed using the same procedure used to generate our training set and initially comprised 237 examples. As our aim was to assess the ability of our model to learn from the structural information supplied by the FHMs, we excluded from our test sets examples where the ground truth molecule was not contained within an apolar hotspot region and examples where no suitable pharmacophoric points could be identified by the

Hotspots algorithm. In addition, we filtered examples where STRIFE was unable to identify any quasi-actives. These filtering steps removed 109, 26 and 1 examples from the test set respectively, leaving a final test set of 101 examples (a full list is given in Table C.1). Whilst the filtering steps outlined above removed a substantial proportion of examples from our test set, the initial test set was constructed by fragmenting the ground truth ligand without consideration of the associated protein. As such, many of the fragments would not have been considered suitable candidates for elaboration.

Using the STRIFE pipeline (Figure 5.2), we sampled a set of 250 elaborations for each example in the test set. We compared STRIFE to four baselines: The deep generative model published by Arús-Pous et al. (2020), "Scaffold-Decorator", the database-based CReM (Polishchuk, 2020) and DeepFrag (Green et al., 2021), and a truncated version of the STRIFE algorithm (STRIFE_{NR}) which generated elaborations from the coarse-grained model (essentially only conducting the Exploration phase from Figure 5.2a and omitting the Refinement phase). We provided CReM with the same set of 250k molecules we used to derive the training sets for STRIFE, which was converted into a database of fragments using CReM's fragmentation procedure. The Scaffold-Decorator model was trained using the same set of examples as the STRIFE generative models. For DeepFrag, we used the saved model trained by the original authors in the original publication (Green et al., 2021); as the DeepFrag training process requires a fragment and associated protein structure for each example, it is not possible to use the 250k subset of ZINC (Sterling and Irwin, 2015) to train the DeepFrag model. As DeepFrag is trained on an entirely separate region of chemical space compared to all other baselines, including protein-ligand complexes included in the CASF-2016 set, it is difficult to objectively compare it to the other methods. We include the results from DeepFrag primarily for completeness to give an indication as to how STRIFE compares to another structure-aware approach.

5.3.7.1 Evaluation Metrics.

For our experiments on the CASF test set, we report several standard 2D metrics in line with those reported in Chapter 3:

- **Validity**: Proportion of generated molecules which could be parsed by RDKit (Landrum, 2006) and for which at least one atom was added to the fragment.
- **Uniqueness**: The proportion of distinct molecules generated by the model, calculated as the number of distinct molecules divided by the total number of molecules.
- **Novelty**: The proportion of generated molecules for which the elaboration was not included in the model training set.
- **Passed 2D Filters**: The proportion of generated molecules which passed a set of 2D filters. A generated molecule was filtered out if the SAScore (Ertl and Schuffenhauer, 2009) of the generated molecule was higher (harder to synthesise) than the SAScore associated with the fragment, if the elaboration contained a non-aromatic ring with a double bond or if the molecule failed to pass any of the pan-assay interference (PAINS) (Baell and Holloway, 2010) filters.

We did not compute the proportion of unique or novel associations proposed by CReM, as CReM does not allow the specification of a desired number of elaborations: CReM returns the set of elaborations contained in the database deemed 'reasonable', meaning that all elaborations proposed by CReM are by design unique. As CReM proposes molecules from a fixed vocabulary of possible elaborations, none of the elaborations proposed by CReM could be considered novel. Similarly, we did not compute novelty or uniqueness values for DeepFrag and for each example attributed 250 elaborations to DeepFrag by selecting the elaborations ranked 1-250 (from a vocabulary of 5k possible elaborations) by DeepFrag's own ranking method.

In addition to the 2D metrics proposed above, we report an additional 2D metric based upon QED (Bickerton et al., 2012), to assess whether attempting to satisfy the identified pharmacophoric points impacts STRIFE's ability to generate druglike elaborations:

- **ΔQED**: The average difference in the QED attained by the elaborated molecules and the original fragment, calculated as: $\Delta\text{QED} = \frac{1}{n}\sum_{j=1}^{n}\Delta\text{QED}_j$, where $\Delta\text{QED}_j = \frac{1}{k}\sum_{i=1}^{k}\text{QED}(\text{mol}_{ij}) - \text{QED}(\text{frag}_j)$.

To assess the ability of STRIFE to generate elaborations capable of forming promising interactions with the target, we used the constrained docking functionality in GOLD (Verdonk et al., 2003) to dock each generated ligand 10 times and calculated the docking score of the top-ranked pose for each ligand. To mitigate the tendency of classical scoring functions to favour larger molecules over smaller ones (Boittier et al., 2020), we calculated the ligand efficiency of each molecule by dividing the docking score by the number of heavy atoms. To account for the variation in docking scores across different targets, we standardise the ligand efficiencies attained by a model on a specific example to have zero mean and unit variance, applying the same transformation to the ground truth ligand efficiency. For the j^{th} example, we compute $\Delta\text{SLE}_{\alpha,j} = \text{SLE}_{\alpha,j} - \text{SLE}_{GT,j}$, where $\text{SLE}_{\alpha,j}$ is the average standardised ligand efficiency of the top α ranked molecules and $\text{SLE}_{GT,j}$ is the standardised ligand efficiency of the corresponding ground truth. If α is specified as greater than the total number of elaborations for which the ligand efficiency was computed (as only molecules which pass the 2D filters are docked), we use the average standardised ligand efficiency of all such molecules. We average over all examples to obtain $\Delta\text{SLE}_\alpha = \frac{1}{n}\sum_{j=1}^{n}\Delta\text{SLE}_{\alpha,j}$. ΔSLE_α (Standardised Ligand Efficiency Improvement) only considers a subset of the molecules generated by each model, mirroring how a large number of molecules produced by a generative model would be assessed in a

real-world fragment-to-lead campaign, where it is unlikely that a human expert would manually inspect hundreds of lowly ranked molecules.

As CReM is unable to return a fixed number of elaborations, we calculated three sets of summary statistics for CReM, each using a different subset of the test set. In all cases, if CReM returned more than 250 elaborations for a specific example, we sampled a set of 250 elaborations from the larger set:

- The set of examples for which CReM returned 250 elaborations ($n = 45$).
- The set of examples for which CReM returned 50 or more elaborations ($n = 62$).
- The set of examples for which CReM returned at least one elaboration ($n = 82$).

We present the results for the first set in Table 5.1 and compare the results between the three subsets in Appendix C (Table C.2): in the case where we included all examples with at least one elaboration, the ΔSLE_α values were substantially degraded by the subset of examples where only a small number of elaborations were proposed.

5.4 Results and Discussion

We assessed the ability of STRIFE to propose elaborations to fragments by incorporating meaningful pharmacophoric information into the generative process. Through a large scale evaluation on a test set derived from the CASF-2016 set (Su et al., 2018), we show that STRIFE is able to generate a wide range of chemically valid elaborations, many of which were not contained in the training set. In addition, in terms of generating elaborations which exhibit high ligand efficiency, STRIFE substantially outperforms existing computational methods for fragment elaboration (Arús-Pous et al., 2020; Polishchuk, 2020), illustrating the advantages of incorporating structural information into the generative model. We demonstrate the applicability of STRIFE

Table 5.1: Comparison of CReM, Scaffold-Decorator, DeepFrag, STRIFE$_{NR}$ and STRIFE on the CASF test set (see Section 5.3.7.1 for definitions of the metrics). **Bold** indicates the best value obtained across the different methods.

Metric	CReM	Scaffold-Decorator	DeepFrag	STRIFE$_{NR}$	STRIFE
Valid	**100**%	99.98%	**100**%	99.5%	98.96%
Unique	N/A	32.78%	N/A	**56.96**%	37.31%
Novel	N/A	4.23%	N/A	**55.65**%	49.21%
Pass 2D filters	66.06%	**98.2**%	78.47%	73.81%	75.38%
ΔQED	-0.148	**-0.05**	-0.125	-0.09	-0.086
ΔSLE_{20}	-0.029	0.1	-0.177	0.44	**0.512**
ΔSLE_{50}	-0.489	-0.222	-0.56	0.078	**0.164**
ΔSLE_{100}	-0.992	-0.572	-0.979	-0.316	**-0.228**

to real-world fragment-to-lead campaigns using two case studies derived from the literature; in particular, we show how STRIFE can be used to explore design hypotheses including side-chain movement.

5.4.1 Large Scale Experiments.

Our experiments on the CASF set demonstrate the benefits of including structural information in the generative process (Table 5.1). All methods generated chemically valid elaborations in more than 99% of cases, illustrating their ability to apply basic valency rules. Scaffold-Decorator, the SMILES-based, structure-unaware generative model proposed by Arús-Pous et al. (2020), generated the smallest proportion of unique molecules (33%). STRIFE$_{NR}$, a truncated version of the STRIFE algorithm which terminates before the Refinement phase so doesn't account for the location of fragment hotspots, generated a greater proportion of unique elaborations (57%) than STRIFE (37%). However this is to be expected as the Refinement phase of the algorithm attempts to sample elaborations from a greatly reduced chemical space compared to the Exploration phase.

Illustrating its ability to generalise beyond the information provided in the training set, almost half (49%) of the elaborations proposed by STRIFE were not contained in the training set. By contrast, only 4% of the elaborations generated by Scaffold-Decorator were novel, suggesting that it relies more heavily on the training set when making elaborations. Almost all of the elaborations proposed by Scaffold-Decorator (98%) passed the set of 2D filters, compared to 75% of elaborations generated by STRIFE and 74% by STRIFE$_{NR}$. As nearly all of the elaborations proposed by Scaffold-Decorator were contained in the training set, which itself was filtered to remove undesirable elaborations, the high pass rate of 2D filters is unsurprising.

STRIFE obtained the 2nd highest ΔQED value amongst the different methods, behind Scaffold-Decorator, suggesting that attempting to satisfy the pharmacophoric points extracted from the FHM did not unduly affect the ability of STRIFE to propose druglike elaborations. We note that on average, the molecules proposed by all methods were 'less druglike' than the corresponding fragment. This is not entirely surprising as none of the models were trained to optimise the QED score, but all methods were able to produce a substantial number of elaborations that were more druglike than the original fragment (Table 5.2).

Method	QED Flag
CReM	0.233
Scaffold-Decorator	**0.401**
DeepFrag	0.204
STRIFE$_{NR}$	0.28
STRIFE	0.283

Table 5.2: Proportion of examples generated by each molecule which attained a higher QED than the associated fragment.

On ΔSLE, which assesses the ability of models to generate elaborations which are more ligand efficient than the ground truth ligand, models that incorporate structural information proposed more ligand efficient elaborations. When considering the top 20

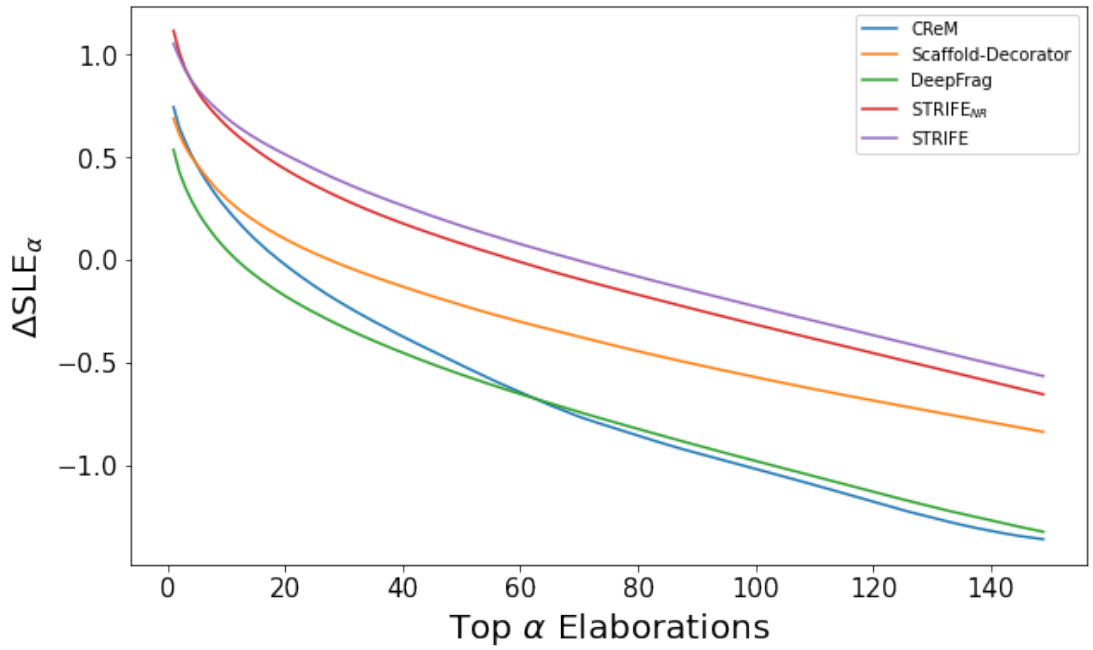

Figure 5.4: Effect of changing the proportion of elaborations made by each method on the Standardised Ligand Efficiency Improvement (ΔSLE_α). As more elaborations are included in the calculation, the average ligand efficiency is degraded compared to the ground truth. STRIFE was able to attain larger ΔSLE_α than all other methods for almost all values of α.

elaborations, the elaborations generated by CReM ($\Delta\text{SLE}_{20} = -0.029$) and Scaffold-Decorator ($\Delta\text{SLE}_{20} = 0.1$) were on average less ligand efficient than the ground truth, in contrast to STRIFE$_{NR}$ ($\Delta\text{SLE}_{20} = 0.44$) and STRIFE ($\Delta\text{SLE}_{20} = 0.512$). These results indicate that the fine-grained pharmacophoric profiles extracted during the Refinement phase allow STRIFE to generate more ligand efficient elaborations, as the model more often generates elaborations which place pharmacophores in close proximity to a pharmacophoric point. We computed the ΔSLE_α attained by each model for a broad range of α values (Figure 5.4) and observed the same trend, although the average ligand efficiency obtained by all models was lower than the ground truth ligand efficiency for larger values of α.

In terms of the proportion of all generated elaborations which were more ligand efficient than the ground truth, STRIFE achieved the largest number, with 26% of

elaborations obtaining a higher ligand efficiency than the ground truth, compared to 22%, 17% and 12% for STRIFE$_{NR}$, Scaffold-Decorator and CReM (when considering examples with 250 elaborations) respectively.

5.4.2 Comparison with DeepFrag.

As mentioned above, it is difficult to objectively compare the performance of DeepFrag to the other baselines. As training examples for DeepFrag require a protein-fragment complex, DeepFrag may be hindered by the relatively scarcity of such structures (the Binding MOAD database (Smith et al., 2019) used to construct their training set contains approximately 40000 structures) and is therefore restricted to a much smaller region of chemical space compared to the other models. On the CASF test set, DeepFrag obtained the lowest ΔSLE$_{20}$ and ΔSLE$_{50}$ values across all models and only attained a better ΔSLE$_{100}$ value than CReM. It is somewhat surprising that DeepFrag isn't able to leverage the structural information provided to it to design more ligand efficient elaborations than the structure unaware Scaffold-Decorator; we provide additional information about the elaborations proposed by DeepFrag in Appendix C.2.

5.4.3 Fragment-Based Design of an N-myristoyltransferase Inhibitor.

Rhinovirus is a pathogen which plays a key role in complications arising in a variety of important respiratory diseases, including asthma, chronic obstructive pulmonary disease (COPD) (Ritchie et al., 2015) and cystic fibrosis (Kieninger et al., 2013). Several studies (Marc et al., 1989; 1990) have reported that the host cell's N-myristoyltransferase (NMT) supports capsid assembly and infectivity, making NMT a potential antiviral drug target.

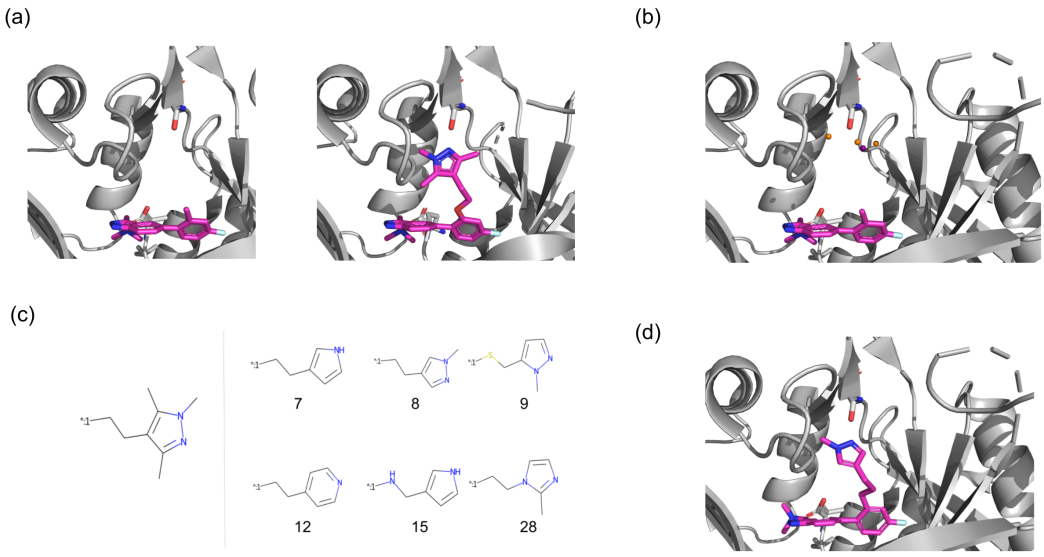

Figure 5.5: Fragment elaboration case study. (a) Left: Crystal structure (PDB ID 5O48) of the fragment bound to P.vivax NMT. Right: Crystal structure (PDB ID 5O6H) of the optimised compound bound to Human NMT1. The trimethylpyralzole facilitates an interaction with the residue S319. (b) Processed pharmacophoric points from the Fragment Hotspot Map calculated on P.vivax NMT. The orange spheres correspond to hydrogen bond acceptor points whilst the purple sphere corresponds to a hydrogen bond donor point. (c) The elaboration proposed by Mousnier et al. (2018) (left) compared to several elaborations proposed by STRIFE which satisfied the same design hypothesis (right). The number underneath each elaboration corresponds to the rank assigned to it by STRIFE. (d) Docked pose of one of our elaborations, which appears to be capable of forming the same hydrogen bond interaction with S319.

Following a fragment screen against NMT from the human malaria parasite *Plasmodium falciparum* (Bell et al., 2012), Mousnier et al. (2018) identified a fragment-like compound, IMP-72 (Figure 5.5a), with weak ($IC_{50} = 20\mu M$) activity against Human NMT1 (HsNMT1). The binding mode of IMP-72 was originally determined in NMT from the malaria parasite *P.vivax* (PvNMT), but as the fragment's key interactions involved residues which are conserved in human NMTs, it was considered to be a viable starting point for the development of an HsNMT1 inhibitor. The authors noted that IMP-72 bound in a region complementary to a previously identified quinoline inhibitor (Goncalves et al., 2012), MRT00057965, however closer inspection of the overlaid binding modes precluded a fragment merging strategy. To address this the

authors constructed a simplified quinoline fragment, IMP-358, which could recapitulate the same interactions as MRT00057965 (S319 in PvNMT and S405 in HsNMT1) without clashing with IMP-72. Despite exhibiting weak inhibition of HsNMT1 (17% at a concentration of $100\mu M$), IMP-358 facilitated a synergistic inhibition alongside IMP-72, with the potency of IMP-72 increasing 300-fold for HsNMT1 in the presence of IMP-358. The authors developed a further compound, IMP-917, derived by replacing IMP-358 with a trimethylpyrazole group which was then linked to IMP-72 with an ether linker. Compared to IMP-72, IMP-917 exhibited a 1500-fold improvement in potency ($IC_{50} = 0.013\mu M$) and retained the key interactions made by both IMP-72 and IMP-358. Finally, the authors made slight modifications to the core of IMP-917 and used the resulting compound to show that NMT inhibition completely prevents rhinoviral replication without inducing cytotoxicity, thereby identifying a potential drug target.

We investigated the ability of STRIFE to propose molecules that could satisfy the design hypothesis put forward by Mousnier et al. (2018). Instead of iteratively refining the original quinoline fragment and constructing a linker, we viewed the task as an elaboration problem and sought to propose elaborations which could form interactions with S319. As input to STRIFE, we provided the SMILES string of IMP-72, the exit vector we wished to make elaborations from and the crystal structure of PvNMT (PDB ID 5O48). Although our aim was to design compounds for HsNMT1, we did not have access to a crystal structure of IMP-72 bound to HsNMT1 so given the high degree of conservation of NMTs across species, we considered it preferable to use the *P.vivax* NMT as opposed to docking IMP72 into the crystal structure of IMP-917 in complex with HsNMT1. We used STRIFE to generate 250 elaborations for IMP-72, which we docked using the constrained docking functionality in GOLD (Verdonk et al., 2003), ranking each compound by its ligand efficiency. Figure 5.5C shows the structure added to IMP-72 to create IMP-917 and several highly ranked

elaborations proposed by STRIFE which appear to be capable of interacting with the Serine residue in the same way. Despite only generating a total of 250 compounds, some of the molecules proposed by STRIFE bear a striking resemblance with the trimethylpyrazole elaboration proposed by Mousnier et al. (2018). A list of all unique elaborations generated by STRIFE can be found in the Appendix (Figures C.3-C.7)

5.4.4 Customisability Case Study: Small Molecule Inhibition of Tumour Necrosis Factor.

Whilst structure-aware generative models are increasingly being proposed, existing models incorporate such information through a single static structure, making them unable to account for the possibility that a side-chain may move to interact with a ligand. By utilising the flexible docking functionality in GOLD (Verdonk et al., 2003), STRIFE allows the user to explore design hypotheses where a specified side-chain moves; we illustrate how by considering a fragment-elaboration example from the literature.

O'Connell et al. (2019) developed a small molecule inhibitor of tumour necrosis factor (TNF), a cytokine which has been shown to be a key factor in several autoimmune diseases, by making elaborations to a weakly binding fragment. The first elaboration allowed the formation of a hydrogen bond between the appended pyridyl group and the residue $Y119^{A}$, which moved substantially in order to make the interaction, yielding a 2500-fold improvement in binding affinity (Figure 5.6a).

The magnitude of the $Y119^{A}$ side-chain movement presents a challenge for a generative model, as it would not be possible for a structure-aware model to predict that the side-chain would move, and if it was predicted by a chemist that the residue would be likely to move to form a hydrogen bond then it would not be possible to communicate such information to the generative model. Whilst STRIFE is unable to predict the movement of specific side-chains in advance, if a human expert has reason

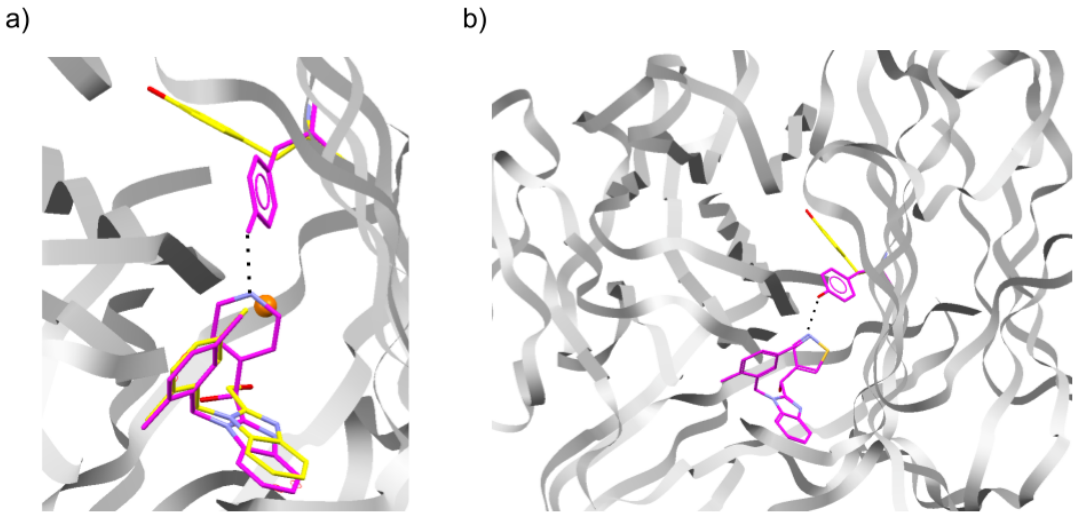

Figure 5.6: Visualisation of flexible docking using Hermes (Groom et al., 2016) a) Fragment (yellow carbons, PDB ID: 6OOY) with elaborated molecule (magenta carbons, PDB 6OOZ) reported by O'Connell et al. (2019). The side chain of $Y119^{A}$ moved substantially to form a hydrogen bond. The orange sphere represents a user-specified pharmacophoric point which we provided as input to STRIFE. b) An example of one of the molecules generated by STRIFE that appears to satisfy the specified design hypothesis. The molecule was docked into the fragment crystal structure (PDB ID: 6OOY, magenta side-chain is the predicted conformation) using the flexible docking functionality in GOLD, and supports the hypothesis that the side chain might move to accommodate the ligand.

to believe a side-chain might move to accommodate a ligand, it is able to generate molecules which satisfy such a design hypothesis. This can be done by manually specifying a pharmacophoric point (see Section 5.3.6) such that a ligand pharmacophore placed at those coordinates would be able to interact with the residue side-chain if it were to move in the hypothesised fashion. Under this set-up, STRIFE attempts to generate molecules with pharmacophores close to the user-specified pharmacophoric point and uses the flexible docking functionality in GOLD (Verdonk et al., 2003) to dock the molecules whilst allowing the residue of interest to move freely; the user can then identify high scoring elaborations which were predicted to form the desired interaction with the protein.

To assess the ability of STRIFE to generate molecules which satisfied the design hypothesis specified by O'Connell et al. (2019), we manually specified a pharmacophoric point (Figure 5.6a) and generated 250 elaborations using the same procedure as for our other experiments. To allow GOLD's genetic algorithm to adequately explore the larger solution space created by side-chain flexibility, we generated 100 poses per molecule and used the highest scoring pose to calculate the corresponding ligand efficiency; further details of the flexible docking protocol can be found in the Appendix.

STRIFE successfully recovered the highly potent pyridyl elaboration proposed by O'Connell et al. (2019), whilst also proposing a wide range of structural analogues which appeared to be capable of making a similar hydrogen bond interaction with $Y119^A$. In particular, the most common elaboration proposed by the model was a pyridine with a meta substitution pattern. In total, 49 of the 250 elaborations contain a pyridine substructure, whilst 125 elaborations included a hydrogen bond acceptor that was also part of an aromatic group. Elaborations comprising a six-membered aromatic ring with a hydrogen bond acceptor were not scored amongst the most ligand efficient using GOLD's PLP scoring function, which generally rated pyrazole analogues or elaborations with hydrophobic groups more highly. However, consistent with the observed bound crystal structure, for both the ground truth pyridyl elaboration and several highly-ranked elaborations which met the stated design hypothesis, the side-chain of $Y119^A$ moved substantially to accommodate the proposed elaboration; an example is shown in Figure 5.6b and further details of the elaborations proposed by STRIFE can be found in the Appendix (Figures C.8, C.9).

The above analysis was dependent on manually choosing the location of the pharmacophoric point. To assess STRIFE's robustness to the exact positioning of the pharmacophoric point, we constructed a lattice of pharmacophoric points in the binding pocket (Figure 5.7) and used each one in turn as an input to STRIFE. As expected,

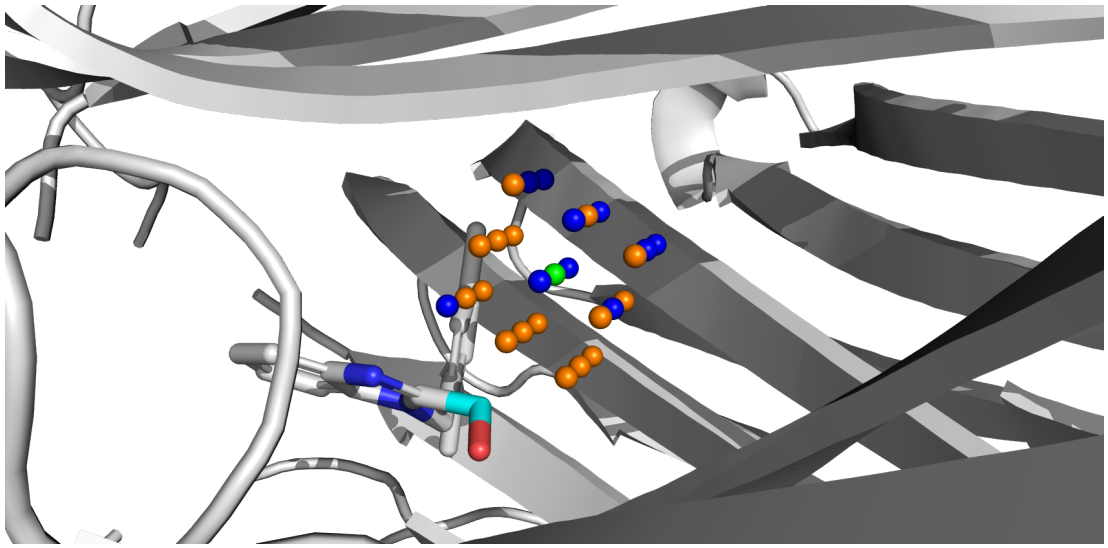

Figure 5.7: A lattice of pharmacophoric points in the binding pocket. The cyan coloured carbon denotes the fragment exit vector, whilst the green pharmacophoric point represents the original pharmacophoric point used in the case study. The blue points are the pharmacophoric points where STRIFE successfully recovered the ground truth elaboration, whilst the orange points denote that STRIFE did not recover the ground truth elaboration (STRIFE also recovered the pyridyl elaboration using the green pharmacophoric point).

modifying the position of the pharmacophoric point affected the types of elaborations proposed; pharmacophoric points that were placed closer to the fragment exit vector tended to produce shorter elaborations than when the pharmacophoric point was further away (Figure C.10).

STRIFE successfully recovered the ground truth pyridyl elaboration for 11 of the 27 different pharmacophoric points, demonstrating its robustness to the exact placement of pharmacophoric points. However, it is of greater interest to assess how often STRIFE was able to generate elaborations with an equivalent pharmacophoric profile to the pyridyl ground truth, as such elaborations would likely have the best chance of exhibiting similar behaviour to the pyridyl elaboration. For each pharmacophoric point in the lattice, we calculated the number of elaborations proposed by STRIFE which were of length 5 or 6 and contained an aromatic Hydrogen Bond Acceptor and plotted the values against the distance between the fragment exit vector and pharma-

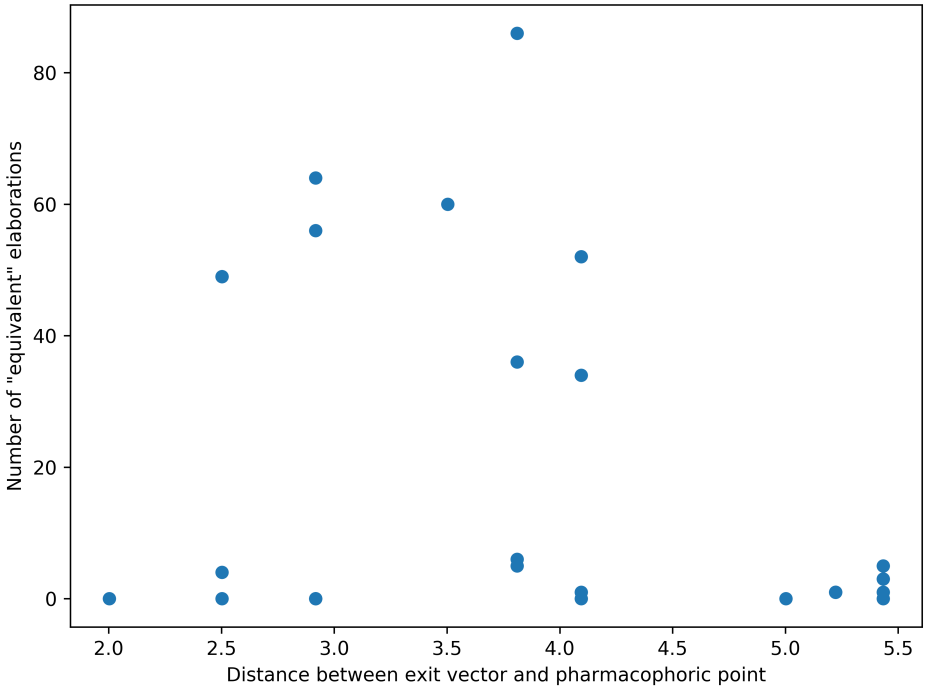

Figure 5.8: When the pharmacophoric point was placed very close to, or far away from, the exit vector, STRIFE proposed a small number of elaborations of length 5 or 6 with an aromatic HBA. However it produced a much larger number of such elaborations when the pharmacophoric point was between 3 and 4 Å away.

cophoric point (Figure 5.8). Figure 5.8 shows that when the pharmacophoric point was placed between 3 and 4 Å away from the exit vector, STRIFE was usually able to generate a sizable number of elaborations which had an equivalent pharmacophoric profile to the pyridyl ground truth. This suggests that STRIFE is fairly robust to the precise placement of the pharmacophoric point (it doesn't need to be placed in an exact spot in order to generate sensible elaborations) but also that the placement of the pharmacophoric point does strongly affect the elaborations produced.

In summary, despite only making a small number of elaborations we were able to use the pharmacophoric information provided to make a range of plausible elaborations which satisfied the specified design hypothesis. In practice, predicting if and how a side chain may move is often extremely difficult but in such cases STRIFE can

be used to assess the plausibility of such a movement and provide starting points for a fragment-to-lead campaign.

5.5 Conclusion

We have proposed a model for fragment elaboration which derives meaningful information from the target into the generative process; unlike other generative models for fragment elaboration, STRIFE can incorporate target-specific information without using an existing active (although information from existing actives can easily be incorporated).

Currently, STRIFE uses information from FHMs which guide the placement of hydrogen bond acceptors and donors within the appended structure. Although hydrogen bonds between ligand and protein often lead to large improvements in binding affinity, they are by no means the only consideration when making elaborations to a fragment; the framework could easily be expanded to explicitly account for properties such as hydrophobicity and aromaticity, allowing a greater degree of control over the design process. Whilst we have used FHMs to extract important information from the protein, one could also use alternative Pharmacophore Interaction Fields, such as GRAILS (Schuetz et al., 2018) or T2F (Mortier et al., 2018), to extract similar information which could then be supplied to the generative model. A further limitation of the default implementation of STRIFE is that it does not seek to simultaneously satisfy multiple pharmacophoric points within a single elaboration, potentially curtailing its ability to generate highly efficient elaborations in some scenarios. However, fragment elaboration campaigns generally involve incrementally making small additions to the molecule and STRIFE provide the functionality to attempt to simultaneously satisfy multiple pharmacophoric points (whether FHM-derived or manually specified), should the user wish to.

Whilst we have used a two stage Exploration-and-Refinement approach to generate the final set of molecules, using a coarse-grained pharmacophoric profile followed by a fine-grained one, an alternative approach would be to use a single stage where elaborations are generated using a large number of potential pharmacophoric profiles. However, we believe that our two-stage approach is likely to be more computationally efficient, as generating elaborations using a series of different fine-grained pharmacophoric profiles, without any assessment of the suitability of such profiles, would lead to docking large amounts of unsuitable molecules.

STRIFE ranks the final set of generated molecules by their predicted ligand efficiency, calculated by docking each molecule and dividing the docking score by the number of heavy atoms in the molecule. Whilst docking scores are known to not correlate perfectly with experimental binding affinities (e.g. (Su et al., 2018)), they have successfully been used in a variety of scenarios to quickly screen large libraries and prioritise small numbers of compounds for experimental validation (Stein et al., 2020; Lyu et al., 2019; Gorgulla et al., 2020) and can give a useful indication to a human expert over whether a molecule is likely to bind to the target. If a user wished to use an alternative metric to rank the molecules produced by STRIFE, they would easily be able to do so.

Compared to existing structure-unaware models for fragment elaboration, the STRIFE algorithm carries a moderate up-front computational cost in calculating an FHM and identifying the set of quasi-actives (between 30-60 minutes on a desktop computer, in most cases). However, the most significant computational expense when generating a large number of elaborations is the docking of each generated molecule to estimate its ligand efficiency. As the quasi-actives only need to be identified once for a given fragment, the computational cost associated with STRIFE is therefore broadly comparable to other methods when generating large sets of molecules.

Although STRIFE is capable of being applied with minimal user input, one area

which requires user specification is the choice of fragment and the associated exit vector. In practice, screening a fragment library may reveal dozens of weakly binding hits, yielding a large set of fragment-exit vector pairs to be explored; STRIFE could readily sample exhaustively from each fragment and exit vector, however a future avenue of research would be to develop a prioritisation scheme capable of identifying promising starting points for a fragment-to-lead campaign, to allow a more efficient allocation of resources.

An advantage of the representation of structural information that STRIFE extracts from the target is that it is extremely easy for a user to interpret. Whilst this is useful in allowing the user to understand why STRIFE generates the kinds of elaborations it does for a specific target, it also allows the user to easily specify their own design hypotheses. As such, we hope that STRIFE will be useful both in cases where a practitioner wishes to automatically generate a set of elaborations to a fragment bound to a novel target and in cases where they wish to rapidly enumerate a set of elaborations that conform to a specific design hypothesis and can be used as a basis for further designs. We have released the code for STRIFE (`https://github.com/oxpig/STRIFE`) so that it can be used by practitioners on their own projects.

Chapter 6

Understanding The Ability of Virtual Screening Models To Capture Important Spatial Information

6.1 Background

In the previous chapters we have described the development of a variety of approaches for computationally enumerating a series of molecules. In this chapter we move on a stage from attempting to suggest molecules that bind to considering the problem of protein-ligand virtual screening. A large number of different approaches to this problem have been proposed, with varying levels of success. In order to investigate the extent to which current virtual screening models are able to identify the functional groups responsible for protein-ligand binding, we develop a new assessment method.

Early virtual screening models (e.g. (Trott and Olson, 2010; Verdonk et al., 2003)) generated a ligand pose and used physics-based or empirical scoring functions to

estimate the binding affinity of a protein-ligand complex. Such scoring functions are typically calculated as the linear combination of a series of scores associated with different interactions, such as hydrophobic interactions or hydrogen binds between atoms. Whilst traditional scoring functions have successfully identified high affinity binders in a variety of different studies, approximating the binding affinity as a linear combination of different interactions is a crude approximation of the true binding process, where the contribution of a particular interaction might vary based on other interactions.

To sidestep manual specification of a scoring function, subsequent approaches encoded a protein-ligand complex as a molecular fingerprint and used them as an input to a machine learning algorithm, for example a neural network ((Durrant and McCammon, 2011)) or a random forest ((Ain et al., 2015)). As these algorithms do not require the user to pre-specify interactions between different features, it was argued that they would be able to learn the biophysical rules which govern protein-ligand binding directly from the data, allowing for more accurate prediction.

Inspired by the remarkable ability of deep learning models to capture important spatial information in other fields, the most recent set of virtual screening models (e.g. (Pereira et al., 2016; Ragoza et al., 2017; Imrie et al., 2018)) have tended to use deep-learning architectures, representing the protein-ligand complex either as a graph or as a 3D image. It is hypothesized that such models are better able to identify important protein-ligand interactions and will be more generalizable to novel targets than fingerprint-based models or models which depend on a set of hand-picked features.

Despite their ability to capture important spatial information, Chen et al. (2019) raised concerns surrounding the susceptibility of deep learning-based models to ligand-specific biases. In particular, they demonstrated that removing all protein information from a deep learning-based virtual screening model did not degrade its performance,

illustrating that the model was not capturing important spatial information but rather learning to classify examples based on ligand-specific biases. In addition, Sieg et al. (2019) showed that ML algorithms are able to exploit distributional differences between the actives and inactives in DUD-E to achieve inflated predictive accuracy, whilst recent studies (Ragoza et al., 2017; Imrie et al., 2018) have shown that classical models illustrated comparable or better predictive performance than deep learning models, calling into question whether deep learning models are actually able to capture important spatial information and use it to identify important intermolecular interactions.

To investigate the extent to which ML algorithms are able to accurately assign importance to individual atoms when making a prediction, several recent works (McCloskey et al., 2019; Sundar and Colwell, 2020; Matveieva and Polishchuk, 2021) have proposed synthetic datasets which labelled ligands as 'active' if they contained a pre-defined molecular substructure. Using an attribution technique, such as Integrated Gradients (Sundararajan et al., 2017), it is then possible to compare the model-assigned atom importances to the ground truth atom labels to assess whether the atoms which comprised the pre-defined substructure were identified as the most important atoms. While these studies provided valuable insights on the ability of ML algorithms to identify important features, they did not assess whether the ML algorithms were able to capture important spatial information or identify intermolecular interactions.

Whilst several authors have used attribution techniques on real-world data to uncover important functional groups (Hochuli et al., 2018; Scantlebury et al., 2020), it is often difficult to ascertain the precise contribution of each atom in an experimentally obtained protein-ligand complex. Combined with the difficulty in manually curating a large-scale test set, it is currently infeasible to objectively assess the attribution performance of ML algorithms on real-world virtual screening tasks.

To address this, in this chapter we propose a protocol for generating a synthetic dataset which mimics the mechanics of protein-ligand binding. As the label of each example is determined by a known deterministic binding rule, we can precisely specify which functional groups, if any, in the ligand are responsible for binding. Although our deterministic binding rule is considerably simpler than the mechanics of real-world protein-ligand recognition, it allows us to assess whether an ML algorithm is able to capture important spatial information and use it when making predictions.

Using our synthetic dataset, we assess the ability of a fingerprint-based virtual screening model to correctly predict the label of each example and to accurately assign importance to each ligand atom. We then investigate the effect that common real-world artefacts such as ligand-specific biases, and incorrect labelling have on the model's ability to identify important functional groups and assess whether the model's performance is sensitive to the specified fingerprint parameters. We compare the results of our fingerprint-based model to a recently published deep learning model, demonstrating that the deep learning-based model exhibits several advantages.

6.2 Methods

As it is not possible to experimentally exactly quantify the extent to which an intermolecular interaction contributes to protein-ligand binding on real-world data, we propose two protocols for generating synthetic data where the contribution of any atom can be computed exactly. Whilst the synthetic data we generate gives only a coarse-grained approximation of real-world protein-ligand binding, it nevertheless allows us to assess the ability of virtual screening models to capture important spatial information and apply it to making predictions. We are also able to ensure that our synthetic datasets are free of ligand-specific bias and other extraneous factors which may inflate a model's predictive performance on a test set while degrading its gen-

eralizability to novel targets. This allows us to quantify the effect of such real-world factors on model generalizability.

6.2.1 Generating a synthetic protein-ligand complex.

We define a "synthetic protein" to be the set $\{(x_i, y_i, z_i, t_i)|i = 1, ..., m\}$, where each element, which we call a "synthetic residue", comprises 3D coordinates (x_i, y_i, z_i) and an associated type, t_i. After specifying a ligand with a 3D conformation, we construct a synthetic protein as follows:

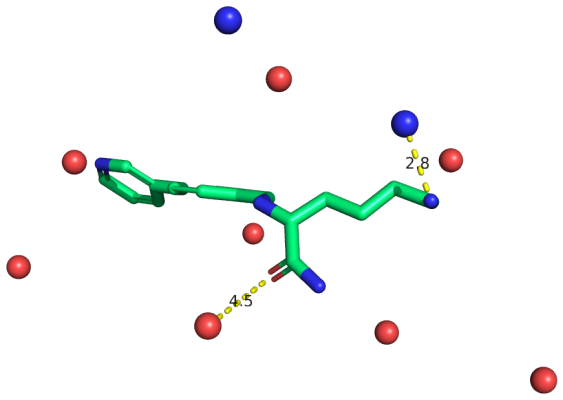

Figure 6.1: Example of synthetic protein-ligand complex. The blue spheres represent synthetic residues with type 'Hydrogen Bond Donor' (HBD), whilst the red spheres represent synthetic residues with type 'Hydrogen Bond Acceptor' (HBA). The amine group is 2.8 Å away from a synthetic residue with type HBD; as the distance is less than the 4 Å specified by the deterministic binding rule, we consider this example to be active. While the carbonyl is 4.5 Å away from the nearest synthetic residue of with type HBA (and therefore does not interact with it), under the Polar generative process only a single interaction is needed for binding.

- We first define a box around the ligand. To obtain the x-axis of the box we identify the minimum and maximum x-coordinates, x_{min} and x_{max}, over all ligand atoms and define the x-axis as $[x_{min} - 5, x_{max} + 5]$. The y-axis and z-axis are obtained in the same way.

- We then sample a number of coordinates uniformly within the box, such that the density of points is invariant of the box size. That is, we sample m points, where $m = a_{coef} * b$ for box volume b.

- For each set of coordinates, we randomly sample an associated type to create a synthetic residue.

- If any synthetic residue is within 2Å of a ligand atom, it is deleted.

- The synthetic residues are filtered further so that no two synthetic residues are within 3Å of each other.

- To reduce the risk of inducing ligand-specific bias dependent on the number of functional groups present in a ligand, we sample the number of synthetic residues, n_{res}, as $n_{res} = n_{ops}/n_{lig}$, where n_{ops} is a constant and n_{lig} is the number of ligand functional groups which can interact with a protein (which varies according to the generative process used to generate the synthetic protein, see below).

6.2.2 Polar generative process.

The type of each synthetic residue determines which ligand functional groups it is able to interact with. In the Polar dataset we restrict the synthetic residue types to "Hydrogen Bond Acceptor" (HBA) and "Hydrogen Bond Donor" (HBD), and say that a ligand atom interacts with a synthetic residue if:

- Their types match; e.g. the synthetic residue has type HBA and the ligand atom is a Hydrogen bond acceptor, and

- The distance between the synthetic residue and ligand atom is below a specified threshold. For all experiments in this work we specified a threshold of 4 Å.

For a given synthetic protein-ligand complex, if any ligand atom interacts with a synthetic residue, we say the complex is active, otherwise it is inactive. An example of a synthetic protein-ligand complex generated using the Polar generative process is shown in Figure 6.1.

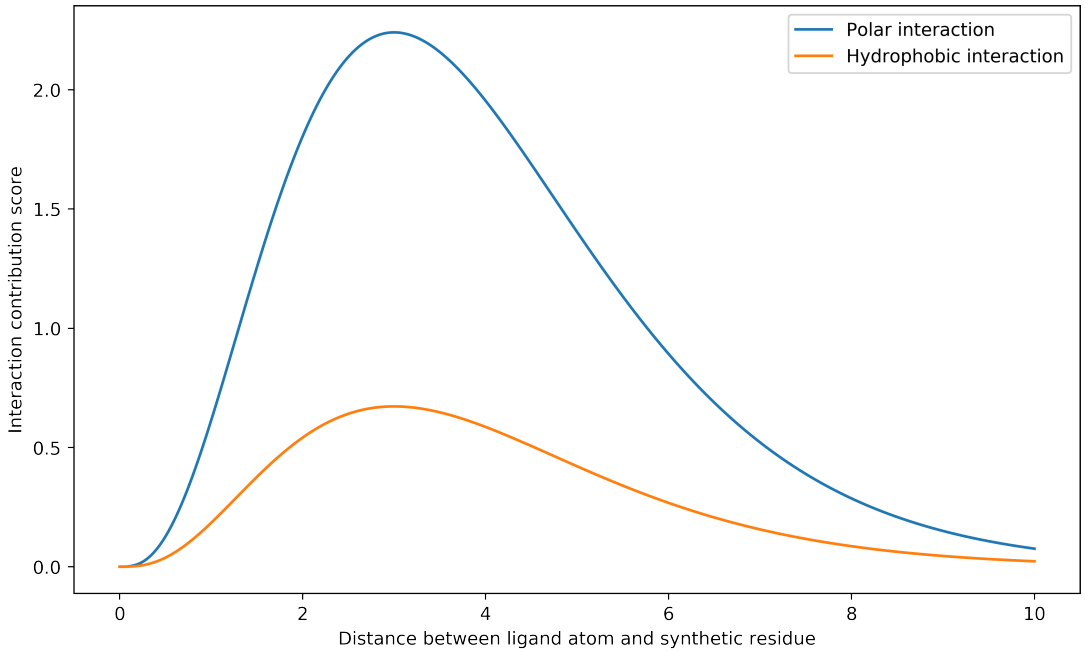

Figure 6.2: Non-linear functions used to determine the interaction score of a synthetic residue-ligand atom interaction. Different functions are used for Polar interactions and Hydrophobic interactions, making it more difficult for the model to learn the deterministic binding rule.

6.2.3 Contribution-based generative process.

While the synthetic protein-ligand complexes generated by the Polar generative process only requires a single interaction to be classified as active, in practice a ligand typically needs to make several different interactions in order to bind with high affinity. We therefore propose a different binding rule which assigns a score to each synthetic residue-ligand atom pair and classifies the complex depending on the cumulative score. In the Contribution dataset we extend the synthetic protein to include "Hydrophobic" synthetic residues as well as the HBA and HBD synthetic residues

present in the Polar dataset. The synthetic protein is generated in the same way as when using the Polar generative process, with each synthetic residue type sampled at uniform from the set $\{HBA, HBD, Hydrophobic\}$.

To label a synthetic protein-ligand complex, we consider synthetic residue-ligand functional group pairs where the ligand functional group has type "Acceptor", "Donor" or "Hydrophobe", assigned by RDKit (Landrum, 2006). If the synthetic residue and ligand functional group do not have matching types, we define their interaction score as 0, otherwise their interaction score is calculated as a non-linear function of the interaction type and the euclidean distance between the synthetic residue and ligand atom (Figure 6.2). The sum of all pairwise interaction scores is calculated, and we label the example as "active" if the summed interaction score exceeds a pre-specified threshold. For all experiments in this work, we used a score threshold of 4.

6.2.4 Machine learning algorithms.

We used two different ML algorithms to classify examples as active or inactive. The first model was a random forest (RF), which took Morgan fingerprints (Rogers and Hahn, 2010) or Protein-Ligand Extended Connectivity fingerprints (PLECs) (Wójcikowski et al., 2019) as input. We were able to represent our synthetic protein-ligand complexes as PLECs without needing to modify the Open Drug Discovery Toolkit (Wójcikowski et al., 2015) (oddt) codebase.

6.2.4.1 Morgan Fingerprints.

Morgan fingerprints are a vector representation of a molecule, calculated in an iterative fashion by first assigning an identifier to each atom and then updating the identifier to incorporate information from the identifiers of the atom's neighbours. This updating process is repeated a number of times so that information about atoms which are not immediate neighbours of an atom can be included in its identifier; in

this work, all Morgan fingerprints are calculated using the RDKit (Landrum, 2006) implementation with a radius of 2.

Morgan fingerprints, by design, do not incorporate any spatial information or any information about the target. In this work, the models trained using Morgan fingerprints serve as a baseline to assess whether a model can attain strong predictive accuracy using solely ligand-based features. Predictive performance that was substantially better than random would suggest that significant ligand-specific biases were present in the dataset, whereas close-to-random predictive performance would indicate that the active and inactive ligand sets were drawn from approximately the same population.

This allows us to quantify the extent to which a model might be susceptible to ligand-specific biases, as we would not expect models without access to spatial information to be able to learn the deterministic binding rules and therefore better-than-random predictive accuracy would likely be due to biases in the training set.

6.2.4.2 PLEC Fingerprints.

PLEC fingerprints encode important spatial information as follows: First, all ligand atom-protein atom pairs which are closer together than a specified cutoff are identified, and an integer radius is specified for both ligand and protein (which need not be the same length). For each qualifying ligand atom-protein atom pair, several molecular substructures are identified:

- The first substructure pair consists of the ligand atom and the protein atom.
- Next, the ligand substructure consisting of the ligand atom and all of its neighbors and the corresponding protein substructure make up the second substructure pair
- This process continues, such that the nth substructures contain all atoms with

a path distance of n or fewer atoms to the original atom.

- In the event that the ligand radius is smaller than the protein radius, the largest ligand substructure is paired with all larger protein substructures (and vice versa if the ligand radius is larger).

The substructure pairs assist in describing the physio-chemical properties of the root atoms. Once all such substructure pairs have been identified for all qualifying ligand-atom pairs, PLEC encodes the information in a fingerprint using a standard hashing algorithm. All PLEC fingerprints were computed using the oddt (Wójcikowski et al., 2015) implementation with a ligand radius of 3 and a protein radius of 0 (as the synthetic residues are represented as a single, unconnected atom).

As ligand atom-protein atom pairs which are within a specified threshold are explicitly encoded within the fingerprint, we would expect models trained using PLECs to perform strongly on the Polar tasks when the PLEC distance cutoff closely matches the distance specified by the deterministic binding rule. However, as PLEC doesn't encode any more detailed spatial information, we would expect that it would be difficult to learn the non-linear deterministic binding rule used by the Contribution generative process.

6.2.4.3 Model naming.

We refer to models trained using a Morgan fingerprint as `RF_Morgan`. Models trained using PLEC fingerprints are referred to `RF_PLEC`. `RF_PLEC_n`, where $\mathtt{n} \in [2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6]$ refers to an RF model trained with a specific PLEC distance cutoff.

6.2.4.4 Equivariant Graph Neural Network

We compared the performance of the `RF_PLEC` models to a recently proposed deep learning-based approach, "EGNN". (Jack Scantlebury, Lucy Vost; Personal Communication; 20APR2022). The network is based on the E(n)-Equivariant Graph Neural

Networks proposed by Satorras et al. (2021) and offers invariance to rotations and translations of the input. In contrast to the fingerprint-based `RF_PLEC` models, the EGNN takes as input the 3D coordinates of each protein and ligand atom, in addition to a one-hot encoding of the atom type. When constructing the input graph, where each node is an atom, two nodes are connected by an edge according to the following rules:

- two ligand atoms are connected by an edge if they are within 2 Å of each other.
- two protein atoms are connected by an edge if they are within 2 Å of each other.
- a protein atom and a ligand atom are connected by an edge if they are within 10 Å of each other.

For intra-molecular edges, the 2 Å cutoff connects atoms which are covalently bonded, whilst the inter-molecular edges connect atoms which might potentially interact. Whilst the EGNN therefore enforces a protein atom-ligand atom distance cutoff in the same way as PLEC, its purpose is to reduce the dimensionality of the input graph by not connecting atoms which are too far away from each other to interact with non-negligible strength, whereas the PLEC distance cutoff only connects atoms which might interact strongly. As such, the EGNN would be unable to utilise the distance cutoff to discover high energy interactions and must instead learn to identify them from the atomic coordinates and atom types. The EGNN also applies a further distance cutoff, where any receptor atom which was not within 6 Å of any ligand atom was ignored; this reduced the dimensionality of the input graph by ignoring residues which were not part of the binding pocket. However, as we constrained each synthetic protein to be within a box defined as $[x_{min} - 5, x_{max} + 5] \times [y_{min} - 5, y_{max} + 5] \times [z_{min} - 5, z_{max} + 5]$, where x_{min} was the smallest ligand atom x-coordinate and the other values were defined similarly, the

vast majority of synthetic residues would be within 6 Å of at least one ligand atom and so this cutoff would have minimal impact on the performance of the EGNN.

6.2.5 "ZINC" dataset.

Before exploring the effect of common real world artefacts such as ligand-specific bias and labelling errors on model attribution performance, we constructed a baseline dataset to assess the ability of different ML algorithms to learn the deterministic binding rules outlined above. We obtained a set of 10k ligands from ZINC (Sterling and Irwin, 2015) and used the Polar and Contribution generative processes to generate two synthetic datasets (the "Polar_ZINC" and "Contribution-ZINC" datasets, respectively).

We sampled 500 examples from each dataset to serve as a test set, and used the remaining examples to train each model. We trained 8 different `RF_PLEC` models, varying the PLEC distance cutoff by 0.5 Å from 2.5 Å to 6 Å.

6.2.6 Inducing ligand-specific bias.

As mentioned above, Chen et al. (2019) showed that the presence of ligand-specific biases allowed virtual screening models to disregard the provided protein-specific information and still achieve strong predictive accuracy. The models were able to do this by constructing a decision rule which classified examples based on 2D ligand features rather than intermolecular interactions. We hypothesized that a dependence on ligand-specific biases would severely degrade the ability of a virtual screening model to identify important functional groups.

To explore the effect of ligand-specific bias on attribution performance, we constructed a set of synthetic protein-ligand complexes where each ligand was taken from the Directory of Useful Decoys (Mysinger et al., 2012). We selected the five DUD-E targets containing the most ligands and used the ligands to construct five synthetic

datasets. For each ligand, we extracted its true label from DUD-E and generated a synthetic protein using the Polar generation process outlined above. If the synthetic protein-ligand complex had a different label than the true label assigned to the ligand, we generated a new synthetic protein and recalculated the label until the true label and the label of the synthetic protein-ligand complex matched. If, after generating 100 synthetic proteins we were unable to attain the true label, we discarded the ligand. We used the same approach to derive an additional 5 synthetic datasets from the 5 LIT-PCBA (Tran-Nguyen et al., 2020) targets with the most ligands. We used ligands from the LIT-PCBA dataset in addition to the DUD-E ligands as the target-specific datasets were subjected to a debiasing algorithm. It was therefore of interest to assess whether the models trained on LIT-PCBA ligands would generalise more effectively than models trained on DUD-E ligands. For the LIT-PCBA datasets, as each dataset exhibited extreme class imbalance, with far more decoy molecules than actives, we subsampled the set of decoys so that each dataset contained 10000 decoys.

By constructing the synthetic virtual screening datasets in this way, an ML algorithm would be able to classify examples by using the spatial information in the synthetic protein-ligand complexes or by learning ligand-specific biases, mimicking the choice offered to an algorithm trained on the real DUD-E datasets.

6.2.7 Labelling errors.

Real-world data can occasionally include labelling errors, where active molecules can inadvertantly be labelled as inactive or vice-versa. To assess the models' robustness to such errors, we generated datasets where the label of each example in the training set was changed with a pre-specified probability, p. We used the following values for $p : [0.01, 0.02, 0.05, 0.1, 0.25, 0.5]$ and assessed the ability of the resulting models to identify the most important ligand atoms in the presence of such errors.

6.2.8 "PDBBind" dataset.

We used ligands from the PDBbind set (Liu et al., 2017) (v. 2019) to construct two external test sets, one using the Polar generative process and one using the Contribution generative process. To reduce the risk of inflated performance due to overfitting, any ligand with a Tanimoto similarity of more than 0.8 with any ligand in any of the above training sets was discarded and we also discarded any very small molecules (fewer than 15 heavy atoms). Of the remaining ligands, we randomly selected 500 and generated a synthetic protein-ligand complex from each ligand. As the purpose of the test set was to assess the ability of the models to identify functional groups which were responsible for binding, we ensured that each example was "active" by resampling the synthetic protein for each ligand until it was active according to the deterministic binding rule.

6.2.9 Ranking the importance of ligand atoms.

Following Hochuli et al. (2018), we used atom masking to rank the importance assigned to a ligand atom by a particular model. For the i^{th} atom in a molecule, m, we calculate the masking score as $s_i = \text{score}(m) - \text{score}(m \backslash i)$, where score$(m)$ is the prediction given by the model for m and $m \backslash i$ denotes the molecule m where the i^{th} atom has been deleted. For the `RF_PLEC` models, we delete an atom by replacing it with a dummy atom, whilst for the EGNN we delete the corresponding node from the input graph. As higher scoring molecules are classified as active examples, masking assigns a high level of importance to atoms whose omission drastically reduces the model's confidence that an example has an active label.

6.2.10 Evaluation metrics.

For datasets generated using the Polar process, where each atom either contributes to binding or is not involved at all, we would hope that an attribution method would give the highest rank to atoms involved in binding, allowing users to identify the most important atoms by their attribution scores. We propose an 'Attribution AUC', where a ranking of atoms which places all binding ligand atoms at the top receives a score of 1, a ranking which places all binding ligand atoms at the bottom receives a score of 0, and all other rankings receive a score according to the following heuristic:

We define a 'change' to be the transposition of two adjacent rows in a dataframe sorted by the model attributions. We calculate the number of changes required to rank the ligand atoms correctly (all binding atoms ranked at the top), and the number of changes required to rank the ligand atoms correctly in the worst case scenario (all binding atoms ranked at the bottom).

$$1 - \frac{\#\text{ changes needed}}{\text{worst case }\#\text{ changes}}$$

For datasets generated using the Contribution process, we calculated the Spearman's Rank Correlation Coefficient between the true ligand atom contributions and the model attributions for each example. To assess the extent to which the models were able to identify the most important atoms, for a specified score threshold, t, we computed the average rank (assigned by the model attributions) of all atoms which had a true contribution greater than t.

In addition to the above metrics which assess the ability of a model to correctly identify which atoms were responsible for binding, we also computed the accuracy and area under the Precision-Recall curve (AU PRC) attained by the models on the held out test sets.

6.3 Results and Discussion

We generated synthetic datasets with deterministic binding rules (see Sections 6.2.2, 6.2.3), in order to quantify the extent to which different ML algorithms were able to accurately identify important intermolecular interactions. When training on the Polar_ZINC dataset, we demonstrate that fingerprint-based Random Forest models were frequently able to identify the ligand atoms responsible for binding, but that they were heavily dependent on the precise cutoff distance parameter used to generate the PLEC fingerprints. When trained on the Contribution_ZINC dataset, which employed a more realistic deterministic binding rule, the `RF_PLEC` models performed substantially worse than the deep learning-based EGNN, which was better able to identify the most important functional groups. When exposed to common real-world artefacts such as ligand-specific bias and labelling errors, both models exhibited a substantial drop in attribution performance.

6.3.1 Ligand-only model performance.

In line with previous work (Chen et al., 2019; Scantlebury et al., 2020), we trained several models where no synthetic protein-specific information was provided to the model. The performance of these models allowed us to assess whether the datasets exhibited any ligand-specific biases, as the deterministic binding rule was dependent on both ligand and synthetic-protein. If a ligand-only model performed substantially better than random on a given dataset this dataset is highly likely to be susceptible to ligand-specific biases; we would also expect structure-based models trained using the same ligands to be susceptible to the same biases as those models also have access to ligand-specific information. Trained on the Polar_ZINC dataset, the `RF_Morgan` model attained a predictive accuracy of 0.52, indicating that without target-specific information the Random Forest was unable to accurately classify the examples. By

Table 6.1: Performance of the `RF_Morgan` model on different datasets. Predictive accuracy substantially better than random suggests that the datasets may suffer from ligand-specific bias. Accuracy denotes the proportion of correctly classified examples, whereas Random Accuracy denotes the accuracy that would have been obtained by assigning all examples the most common label (= max(% actives, % inactives)). AU-PRC denotes the area under the Precision-Recall curve. Balanced Accuracy and Balanced AU-PRC denote the respective accuracy and area under the Precision-Recall curve when the model was trained using an equivalent number of actives and inactives.

Dataset	Random Accuracy	Accuracy	AU-PRC	Balanced Accuracy	Balanced AU-PRC
ZINC	0.504	0.52	0.53	N/A	N/A
DUDE-AA2AR	0.93	1.0	1.0	0.984	0.996
DUDE-DRD3	0.972	0.998	1.0	0.978	0.995
DUDE-FA10	0.97	1.0	1.0	0.992	1.0
DUDE-MK14	0.976	0.998	1.0	0.994	1.0
DUDE-VGFR2	0.984	1.0	1.0	0.99	0.999
LIT-ALDH1	0.613	0.76	0.809	0.768	0.806
LIT-FEN1	0.956	0.958	0.584	0.778	0.883
LIT-MAPK1	0.964	0.964	0.292	0.692	0.797
LIT-PKM2	0.944	0.952	0.755	0.79	0.901
LIT-VDR	0.928	0.942	0.6	0.772	0.87

contrast, when training models using any of the DUD-E datasets, all models attained a predictive accuracy substantially better than random (Table 6.1), suggesting that the models were susceptible to ligand-specific biases. Several of the LIT-PCBA datasets attained a similar predictive accuracy to random when trained using Morgan Fingerprints, possibly indicating that these targets were not susceptible to ligand specific bias and the models were using the substantial class imbalance to inflate predictive accuracy. However when training an `RF_Morgan` model for each target using a balanced dataset, where the number of actives and inactives were the same, each model attained an accuracy considerably above 0.5, illustrating that the models were able to still able to extract information from the ligand to make accurate predictions.

6.3.2 Performance of `RF_PLEC` models on Polar_ZINC dataset.

Having validated that our Polar_ZINC dataset was not susceptible to ligand-specific biases, we included the synthetic proteins in the training process. To assess how sensitive the `RF_PLEC` models were to the choice of PLEC distance cutoff, we trained eight distinct `RF_PLEC` models, incrementing the PLEC distance cutoff by 0.5 Å between 2.5 Å and 6 Å. We found that all `RF_PLEC` models attained a substantially better predictive accuracy than the `RF_Morgan` model on the Polar_ZINC dataset (Table 6.2), indicating that the inclusion of structural information into the model enabled the random forest to better approximate the deterministic binding rule. Unsurprisingly, the `RF_PLEC_4` (the same cutoff as the deterministic binding rule) model obtained the highest predictive accuracy.

The `RF_PLEC` models exhibited a similar trend in terms of Attribution AUC, with the `RF_PLEC_4` model most often correctly identifying the atoms responsible for binding and attribution performance. We observed that performance degraded substantially as the PLEC distance cutoff diverged from 4Å, suggesting that the `RF_PLEC` models were highly dependent on the precise specification of the PLEC distance cut-

Table 6.2: Performance of different RF_PLEC models on the Polar_ZINC dataset. Accuracy denotes the proportion of correctly classified examples, AU-PRC denotes the area under the Precision-Recall curve, and Attribution AUC reflects the ability of a model to correctly identify the ligand atoms responsible for binding. As expected, the best performing model was RF_PLEC_4, which uses the same distance cutoff as the Polar deterministic binding rule.

Model	Accuracy	AU-PRC	Attribution AUC
RF_PLEC_2.5	0.618	0.655	0.566
RF_PLEC_3	0.702	0.784	0.654
RF_PLEC_3.5	0.788	0.893	0.773
RF_PLEC_4	**0.948**	**0.988**	**0.885**
RF_PLEC_4.5	0.806	0.857	0.859
RF_PLEC_5	0.786	0.801	0.778
RF_PLEC_5.5	0.752	0.792	0.733
RF_PLEC_6	0.72	0.768	0.697

off.

6.3.3 Sensitivity of PLEC to distance cutoff.

To better understand the relationship between the PLEC distance cutoff and attribution performance, we consider a synthetic protein-ligand complex in detail. The original complex (shown in Figure 6.3a) is labelled as active and has a single synthetic residue-ligand atom pair with complementary type and a distance below the deterministic binding threshold. We perturbed the synthetic residue to take a range of locations between 1.5 Å and 10 Å away from the matching ligand atom (Figure 6.3b). We featurized each protein-ligand complex using PLEC (using a range of PLEC distance thresholds, incrementing by 0.5 Å from 2.5Å - 6Å) and used masking to determine the importance of each ligand atom and calculated the rank of the ligand atom involved in binding. Figure 6.4 shows the relationship between synthetic residue-ligand atom distance and the rank assigned to the binding ligand atom by the highest performing RF_PLEC models (RF_PLEC_4, RF_PLEC_4.5 & RF_PLEC_5). The

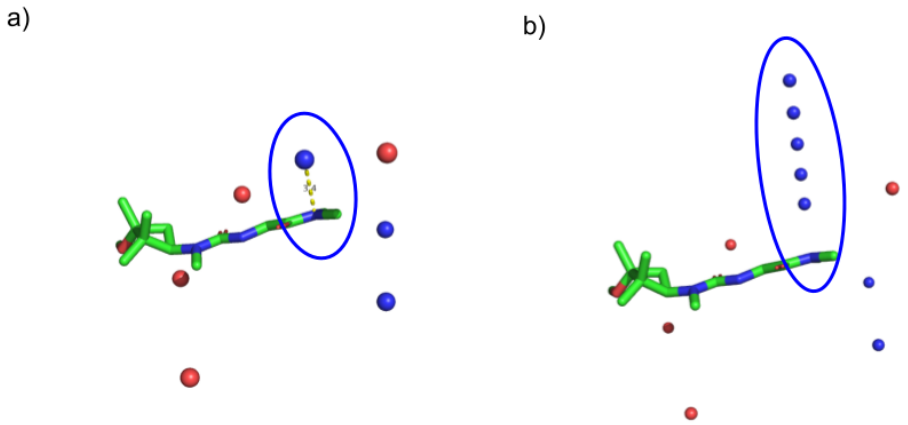

Figure 6.3: a) The original synthetic protein-ligand complex generating using the Polar generative process, labelled as active as a result of the circled interaction. b) Perturbed examples. We generated 50 synthetic protein-ligand complexes, which were all identical apart from the HBD synthetic residue contained in the circle, whose position was perturbed in relation to the ligand HBD with which it interacts. The five spheres within the circle illustrate five of the 50 positions occupied by the interacting residue; for the examples where the synthetic residue-ligand atom distance was greater than 4 Å, we would hope that the model did not consider the ligand HBD to be important as it no longer contributes to binding.

figure shows that the RF_PLEC models assign a high level of importance to the key ligand atom when the synthetic residue-ligand atom distance is less than the respective PLEC distance cutoff, and assigns a reduced level of importance when the distance is greater than the PLEC distance cutoff. In particular, when using a PLEC cutoff threshold of 4.5 or 5 Å, the ligand atom is assigned a high rank when the synthetic residue-ligand atom distance is greater than the true binding threshold but less than the PLEC distance cutoff. This demonstrates the sensitivity of the RF_PLEC models to the exact specification of the specified distance cutoff; it is not able to encode a precise level of spatial information and, given that for real protein-ligand complexes different interactions can take place at different distances, suggests that models trained using PLEC or based on distance-based cutoffs are unable to retain fine-grained spatial information.

6.3.4 Performance on Contribution dataset.

We next examined the performance of the methods on our Contribution dataset. As with the dataset generated by the Polar generative process, we first fit an RF_Morgan model on the Contribution_ZINC dataset to assess whether the models were susceptible to ligand-specific bias. The RF_Morgan model attained an accuracy of 0.47, indicating that the synthetic protein-aware models would have to use spatial information to achieve strong predictive performance. All of the RF_PLEC models attained a low Spearman's Rank Correlation Coefficient, ranging from 0.006 (RF_PLEC_2.5) to 0.093 (RF_PLEC_6). Figure 6.5 demonstrates that, in terms of identifying the ligand atoms which made the largest contribution to binding, several of the RF_PLEC models only outperformed a random ranking by a slight margin and none of the models were consistently able to assign a high rank to the functional groups attaining scores over 2.5. This suggests that, whilst the PLEC fingerprints were often able to encode a sufficiently detailed level of spatial information for the simplistic Polar tasks,

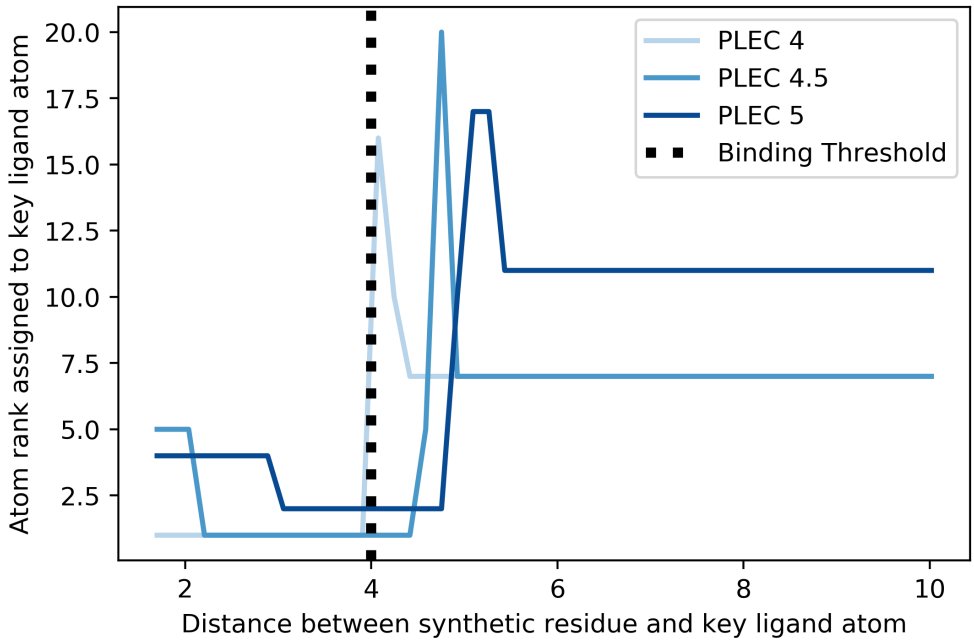

Figure 6.4: Illustration of how the relative importance of the key ligand atom changes as the distance between it and the perturbed synthetic residue changes. Despite the synthetic residue and ligand atom no longer interacting when the distance between them is greater than 4 Å, the `RF_PLEC_4.5` and `RF_PLEC_5` models continued to rank the ligand atom highly until the synthetic residue-ligand atom distance was greater than the respective PLEC distance cutoff. This illustrates the sensitivity of PLEC to the specification of the distance cutoff and suggests that it is unable to encode a high level of spatial information.

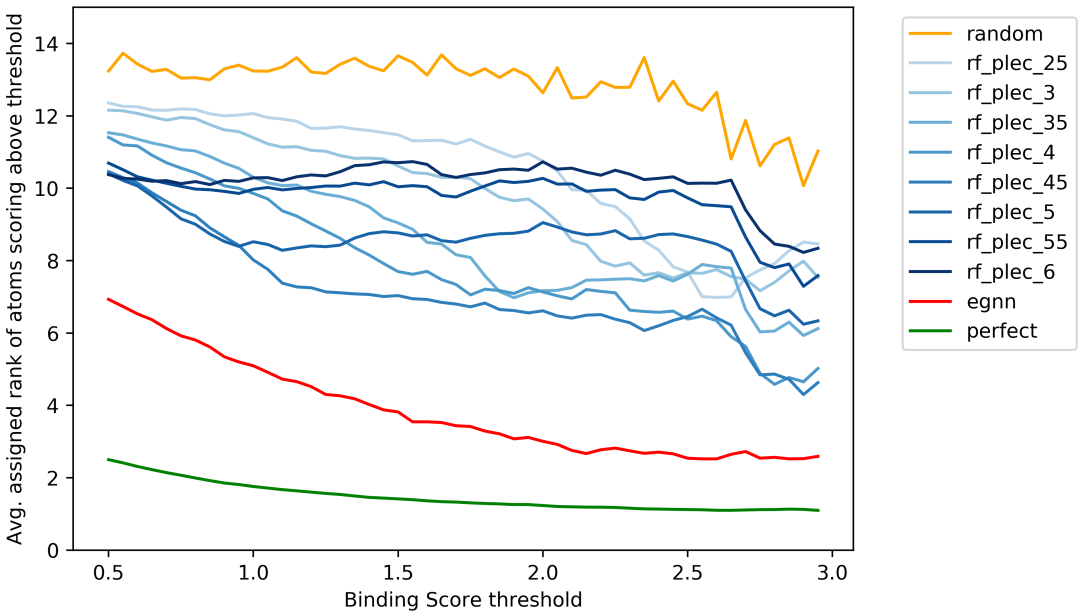

Figure 6.5: The average rank assigned to all ligand atoms attaining a score above a specified threshold. Perfect is the curve attained when ranking atoms by the true atomic contribution, whilst random is the curve obtained when the atoms are ranked completely at random.

its representation of spatial information is inadequate to learn the more complicated Contribution binding rule. As the Contribution binding rule is itself considerably simpler than the rules which govern real-world protein-ligand binding, it is likely that PLEC (and other fingerprint-based methods which are based on a binary distance cutoff) are ill-equipped to ascertain which functional groups make a key contribution towards binding.

6.3.5 Assessment of real world factors on attribution performance.

Although the above results suggest that fingerprint-based models struggle to learn complex binding rules, several previous studies (e.g. (Wójcikowski et al., 2017; Imrie et al., 2018)) have reported that fingerprint-based methods have attained strong predictive accuracy on real-world virtual screening datasets such as DUD-E (Mysinger

et al., 2012). In addition, Sieg et al. (2019) demonstrated that in the presence of distributional differences between classes, ML algorithms are able to attain high predictive accuracy without being provided any target-specific information. Therefore, it is of interest to assess the extent to which real-world factors such as ligand-specific biases and labelling errors degrade the ability of virtual screening models to identify important ligand functional groups.

6.3.5.1 Ligand-specific bias.

Table 6.3: Performance of the `RF_PLEC_4` model on different datasets. Accuracy denotes the proportion of correctly classified examples, whereas Random Accuracy denotes the accuracy that would have been obtained by assigning all examples the most common label (= max(% actives, % inactives). AU-PRC denotes the area under the Precision-Recall curve. Balanced Accuracy and Balanced AU-PRC denote the respective accuracy and area under the Precision-Recall curve when the model was trained using an equivalent number of actives and inactives.

Dataset	Random Accuracy	Accuracy	AU-PRC	Balanced Accuracy	Balanced AU-PRC
ZINC	0.504	0.948	0.988	N/A	N/A
DUDE-AA2AR	0.93	0.992	0.987	0.942	0.992
DUDE-DRD3	0.972	0.992	0.974	0.904	0.981
DUDE-FA10	0.97	0.992	0.971	0.934	0.993
DUDE-MK14	0.976	0.99	0.994	0.928	0.985
DUDE-VGFR2	0.984	0.992	0.92	0.926	0.987
LIT-ALDH1	0.613	0.948	0.987	0.946	0.988
LIT-FEN1	0.956	0.968	0.839	0.858	0.948
LIT-MAPK1	0.964	0.972	0.672	0.822	0.923
LIT-PKM2	0.944	0.974	0.976	0.916	0.984
LIT-VDR	0.928	0.98	0.931	0.928	0.979

Table 6.1 illustrates that all of the DUD-E and LIT-PCBA datasets were able to achieve substantially better-than-random performance using only the information encoded in Morgan fingerprints, suggesting that those datasets were susceptible to ligand-specific bias. When target-specific information is provided to the model by

Table 6.4: Attribution AUC obtained by the different RF_PLEC models and the EGNN on the Polar_ZINC dataset. The RF_PLEC models trained on the unbiased ZINC_Polar dataset consistently attained a larger Attribution AUC than those trained on datasets susceptible to ligand-specific bias, suggesting that if a model learns to classify examples based on ligand-specific features, its ability to learn the true binding rule is impaired.

Dataset	RF_PLEC_4	RF_PLEC_4.5	RF_PLEC_5	EGNN
ZINC	0.885	0.859	0.779	0.851
DUDE-AA2AR	0.863	0.757	0.733	0.838
DUDE-DRD3	0.829	0.676	0.628	0.691
DUDE-FA10	0.858	0.673	0.555	0.816
DUDE-MK14	0.824	0.632	0.605	0.73
DUDE-VGFR2	0.772	0.616	0.559	0.605
LIT-ALDH1	0.855	0.813	0.731	0.925
LIT-FEN1	0.826	0.687	0.623	0.868
LIT-MAPK1	0.806	0.609	0.591	0.698
LIT-PKM2	0.807	0.734	0.665	0.82
LIT-VDR	0.849	0.71	0.647	0.882
Random	0.503	0.503	0.503	0.503

substituting a Morgan fingerprint for a PLEC fingerprint, the models trained on DUD-E and LIT-PCBA data again achieved strong predictive accuracy (Table 6.3). Despite this, we found that training the models on DUD-E and LIT-PCBA ligands degraded the ability of the random forest to accurately identify the ligand atoms responsible for binding. Table 6.4 shows the Attribution AUC values attained by the highest performing RF_PLEC models when trained on a variety of different datasets and tested on the external PDBBind set.

When training on the DUD-E datasets, each of the RF_PLEC_4 models attained an attribution AUC value marginally below that obtained by the corresponding model trained on the ZINC dataset. However, we observed a considerably larger difference between the attribution AUC values obtained by the RF_PLEC_4.5 and RF_PLEC_5 models trained using the DUD-E datasets and the corresponding models trained using

the ZINC ligands. Indeed, two of the RF_PLEC_5 models attained an Attribution AUC (FA10: 0.555, VGFR2: 0.559) which was only marginally higher than the random baseline (0.503). We observed a similar trend for the RF_PLEC models trained using LIT-PCBA ligands.

Whilst the attribution AUC quantifies the ability of the models to assign a high rank to all interactions, we also investigated the ability of the RF_PLEC models to identify at least one interaction. Figure 6.6 shows the proportion of examples where a interacting ligand atom was ranked within the top N ligand atoms by the model, as N varies. We observed a similar trend as when considering Attribution AUC, where the RF_PLEC_4 models trained on ZINC ligands only marginally outperformed those trained on DUD-E or LIT-PCBA ligands, but observed a considerable difference when using RF_PLEC_4.5 and RF_PLEC_5 models. Compared to the models trained on ZINC ligands, several RF_PLEC_4.5 and RF_PLEC_5 models assigned an interacting atom the highest rank on approximately 50% fewer examples, further illustrating that ligand-specific biases can degrade attribution performance.

The above results suggest that, although the RF_PLEC models were provided with the necessary spatial information, their ability to learn the deterministic binding rule was impaired by the presence of ligand-specific bias. Whereas the performance of the RF_PLEC_4.5 and RF_PLEC_5 models was severely degraded when using the DUD-E and LIT-PCBA models, the RF_PLEC_4 models only saw a slight decrease in Attribution AUC compared to the corresponding ZINC model. A possible explanation for the greater robustness of the RF_PLEC_4 models could be that as the 4 Å PLEC cutoff corresponds exactly with the distance threshold used for the deterministic binding rule, synthetic residue-ligand atom pairs which are closer together than 4 Å are always encoded in the PLEC fingerprint, making it relatively simple for the RF to correctly identify atoms which contribute to binding. For the other PLEC cutoff distances which do not perfectly match the true binding threshold, it is more difficult to

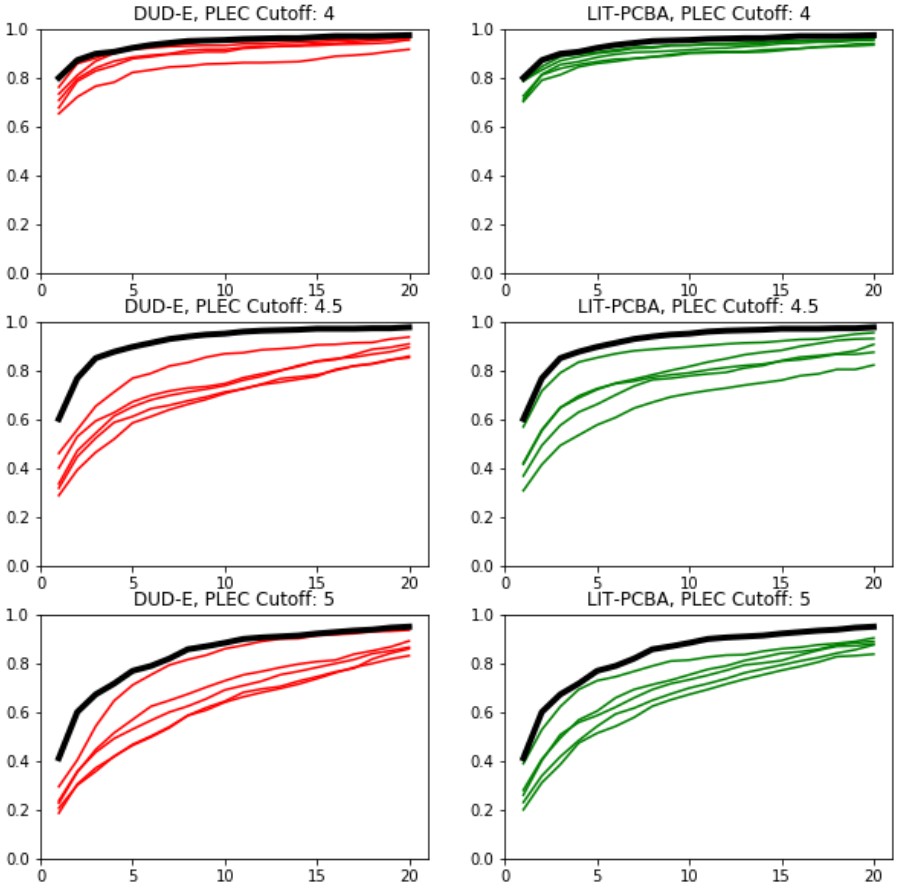

Figure 6.6: Ability of different `RF_PLEC` models to identify binding atoms. The x-axis of each plot shows a rank, N, and the y-axis displays the proportion of interactions where the top-ranked atom involved in binding was ranked in the top N ligand atoms by the model. The thick black line denotes the performance of the `RF_PLEC` models trained using ZINC ligands, whilst the red and green lines show the performance of the models when trained using DUD-E and LIT-PCBA ligands respectively.

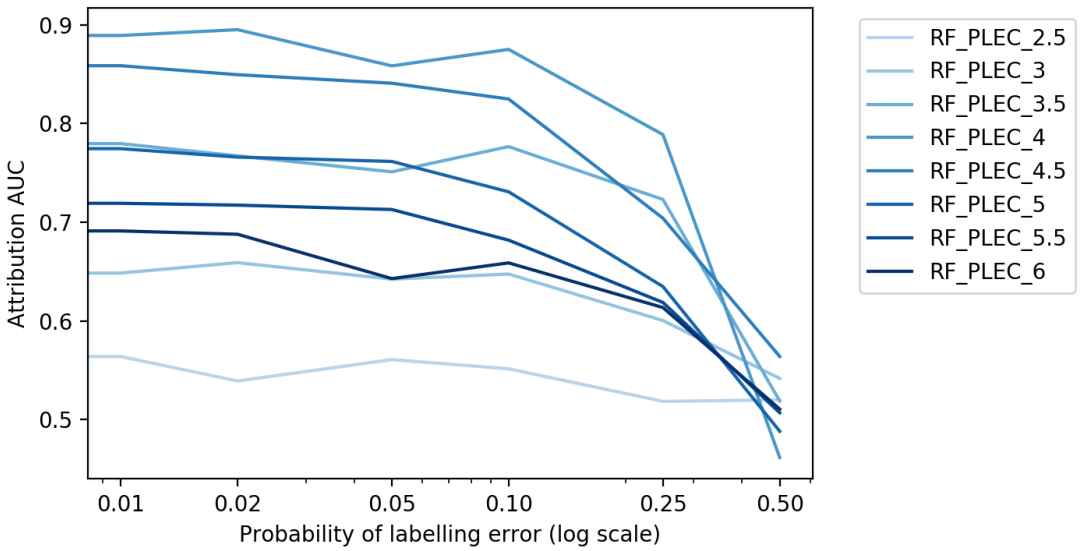

Figure 6.7: Change in Attribution AUC on the Polar_ZINC dataset as noise was introduced into the training set. The x-axis is plotted on the logarithmic scale to allow visualization of the Attribution AUC associated with small labelling error.

accurately classify examples based on the spatial information encoded in the PLEC fingerprints, perhaps meaning that it is simpler for the RF to learn to classify examples based on ligand-specific biases, rather than the provided spatial information.

6.3.5.2 Labelling Errors.

We computed the Attribution AUC for each `RF_PLEC` model and for a variety of labelling error rates (Figure 6.7). As expected, when approximately 50% of examples were mislabelled, the labels were essentially assigned at random and therefore the models were unable to learn the deterministic binding rule. Surprisingly, the models were robust to moderate labelling error rates, with all `RF_PLEC` models exhibiting similar performance when 10% of examples were mislabelled compared to when all examples were labelled correctly.

6.3.6 Performance of EGNN model.

When trained using the Polar_ZINC dataset, the EGNN model attained an accuracy of 0.886, a AU-PRC of 0.948 and an Attribution AUC of 0.851. Whilst the EGNN performed slightly worse than the `RF_PLEC_4` and `RF_PLEC_4.5` models, the EGNN was only provided with the atomic coordinates and had to learn the deterministic binding rule solely from the data. In contrast, the distance cutoffs used for the `RF_PLEC_4` and `RF_PLEC_4.5` models were either identical to or very close to the threshold of 4 Å used by the deterministic binding rule, making it much easier for the `RF_PLEC_4` and `RF_PLEC_4.5` models to identify binding.

When trained and tested on datasets using the Contribution generative process we found that, in contrast to the Polar generative process, the EGNN comfortably outperformed all of the `RF_PLEC` models. Whilst the Spearman's rank correlation (0.177) obtained by the EGNN was not particularly strong, it was still substantially higher than any correlation attained by the `RF_PLEC` models. In terms of ranking the most important atoms highly, the EGNN outperformed all `RF_PLEC` models (Figure 6.5), assigning a substantially higher average rank to all above-threshold atoms for all thresholds. This result is not particularly surprising as the `RF_PLEC` models do not capture a precise encoding of each atom's position, only recording whether the distance between two atoms is below a pre-specified threshold, making it difficult for the `RF_PLEC` models to approximate the non-linear deterministic binding rule. We would therefore expect the EGNN to perform better on this task, as it is not constrained by a particular featurization and can learn the deterministic binding rule from the data.

When trained on the DUD-E or LIT-PCBA datasets, we found that in several cases the attribution AUC attained by EGNN was substantially degraded compared to the ZINC dataset. The remaining datasets yielded performance on a par with, or substantially better than, the ZINC dataset. In the majority of cases, the EGNN's

ability to learn the true deterministic binding rule was only slightly affected by the ligand-specific biases. This suggests that whilst models which struggled to learn the true binding rule took advantage of ligand-specific biases to inflate their predictive accuracy, models which were able to accurately approximate the deterministic binding rule still learnt to classify examples based on intermolecular interactions in the presence of ligand-specific bias. As such, the development of models which can capture important spatial information should be a priority for the virtual screening community, as such models are likely to generalise more effectively even if trained on biased training data.

6.4 Conclusion

We have proposed a framework for assessing the ability of different ML algorithms to attribute binding at the atomic level. By simulating synthetic datasets we are able to define a ground truth for which atoms in a synthetic protein-ligand complex are responsible for binding, which is not possible with real-world data. Despite several recent studies suggesting that fingerprint-based machine learning algorithms achieve comparable performance with current deep learning methods, we found that a deep learning-based EGNN was better able to elucidate which atoms were responsible for 'binding' when the training datasets were not susceptible to ligand-specific biases, suggesting that they should have greater ability to generalize to novel targets. When we trained our models on datasets containing ligand-specific bias, we found that on the whole the models' ability to accurately assign atomic importance was diminished, consistent with the notion that dataset-specific bias degrades model generalizability.

We found that the drop in attribution performance attained by the EGNN was substantially smaller than the corresponding drop in performance observed for the `RF_PLEC_4.5` and `RF_PLEC_5` models and similar to that observed for the `RF_PLEC_4`

model, suggesting that models which can more easily learn the true binding rule are less susceptible to the effects of bias.

Our analysis has a number of limitations. The deterministic binding rules used in both the Polar and Contribution generative processes are significantly simplified compared to real-world protein-ligand binding. However, our objective was to assess the extent to which different ML algorithms were able to capture and apply relevant spatial information and our relatively simple generative processes were sufficient to highlight the difficulties of explicitly encoding a precise level of 3D information into a fingerprint. As even more precise structural information is required to decide whether a ligand would bind to a real protein with high affinity, we would expect fingerprint-based models to struggle to identify important functional groups even more on real-world data.

Whilst there is significant interest in developing models which can predict protein-ligand binding with high accuracy, there is a strong need for models which can identify the functional groups which contribute most heavily towards binding; the identification of key functional groups would allow easier iterative optimisation of lead compounds. In addition, models which demonstrated an understanding of biophysical rules would be more likely to be accepted by human experts.

Chapter 7

Concluding Remarks and Future Work

Recent years have seen an explosion of interest in the application of machine learning methods to drug discovery. Researchers have attempted to apply machine learning to almost all parts of the drug discovery process, and it promises to add value in a range of areas, by automating previously time-consuming tasks and providing insights that were not obvious to human experts. This thesis has primarily focused on the incorporation of target-specific information into deep learning models for drug discovery. In this chapter we summarise the key findings of our work, and highlight areas for future research.

7.1 Deep Generative Models for Molecule Generation

In Chapter 2, we used reinforcement learning to finetune a deep generative model for fragment elaboration, incentivising the model to generate elaborations which were highly similar to a ground truth elaboration. Whilst directly employing a similarity-

based reward function enabled the model to generate a large proportion of highly similar elaboration, we found that the resulting models were often unable to generate a diverse range of highly similar elaborations and that the policy gradient algorithm could get stuck in a local minimum where it repeatedly generated a simplistic elaboration which attained a moderate similarity score. To address this, we proposed the use of a structured curriculum, where the model learned to sample from increasingly smaller regions of chemical space in order to incrementally increase the reward. Compared to directly optimising the similarity function, training via a structured curriculum allowed the model to generate a wider range of distinct highly similar elaborations.

In Chapter 3, we investigated the use of pharmacophoric constraints as an alternative to reinforcement learning in generating elaborations which were highly similar to a ground truth. We found that employing pharmacophoric constraints allowed a fine degree of control over the elaborations produced by the model and, in two large-scale evaluations, outperformed recently published unconstrained generative models both in recovering the ground truth elaboration and in generating highly similar elaborations. Our results validated the effectiveness of our model as a tool for R-group optimisation campaigns.

In Chapter 4, we described the development of a Flask-based (Grinberg, 2018) web application for DeLinker (Imrie et al., 2020) and DEVELOP (Imrie et al., 2021). The application allows users to generate elaborations/linkers via a simple-to-use GUI and without installing anything, with the generated molecules available to download as a CSV file.

Finally, in Chapter 5, we proposed an approach for extracting meaningful and interpretable structural information from a target protein and converting it to a set of pharmacophoric constraints. Deriving pharmacophoric constraints directly from a protein structure rather than an existing active greatly extends the applicability of

our model, as it can be used to elaborate fragments obtained in a fragment screen of a novel target, which might not have any known active drug-like molecules. In a large-scale evaluation, the resulting model, STRIFE, was able to generate more ligand efficient elaborations than existing models which did not incorporate target-specific information. Moreover, STRIFE outperformed a recently published structure-aware model for fragment elaboration (Green et al., 2021), illustrating the effectiveness of the FHM processing protocol.

Whilst STRIFE is able to generate highly ligand efficient elaborations by extracting information directly from the target protein and allows users to pursue their own design hypotheses, there are several interesting directions for future work. First, whilst STRIFE uses 3D structural information to derive the pharmacophoric constraints, the generated molecules are generated in 2D, without any associated atomic coordinates. Replacing the CGVAE-like generative process (Liu et al., 2018) with a 3D EGNN-based generative process could remove the need for the Exploration phase of the STRIFE algorithm, thereby greatly improving its computational efficiency.

Moreover, STRIFE currently generates molecules in an independent and identically distributed fashion. An interesting extension of STRIFE would be to allow the incorporation of information about the activity of previously proposed molecules into the generative process. For example, after the first round of generations, if elaborations which targeted a particular hotspot did not lead to an improvement in binding affinity when tested experimentally, that hotspot could be removed from consideration for the next round of elaborations.

Finally, STRIFE requires the user to specify a single fragment and exit vector in order to generate elaborations. For a fragment screen with a large number of hits, the number of fragment-exit vector pairs might be too large for a human expert to consider feasibly consider all the options in detail. It might therefore be of interest to develop a protocol which could use STRIFE to automatically prioritise fragments

for consideration via a sampling scheme such as Bayesian Optimisation.

7.2 Structure-Based Virtual Screening

In Chapter 6, we proposed a novel protocol for generating a synthetic dataset with a deterministic binding rule. An important step in the development of structure-based virtual screening models is that they are able to identify the functional groups most responsible for binding. This is important not only because it allows the model to assist in the iterative development of lead molecules, but also because evidence that a model is able to correctly learn the biophysical rules governing protein-ligand binding reduces the chance that the model's generalisability is hampered by ligand-specific biases introduced during training. We found that an EGNN that was able to capture important spatial correlations in the input was better able to identify the most important functional groups than a fingerprint-based model which encoded a low-dimensional representation of structural information. This highlights the value of developing models which are able to capture and use 3-dimensional information when making predictions. Moreover, whilst the presence of ligand-specific biases often leads to strong predictive accuracy, we found that it degraded the ability of virtual screening models to identify the most important functional groups; underscoring the importance of creating high quality datasets for virtual screening.

There are a number of avenues for future research. Whilst our synthetic data generation process provided useful insights about the ability of virtual screening models to apply spatial information, our deterministic binding rules are considerably simpler than the rules which govern real-world protein-ligand binding and a natural next step would be to develop a synthetic protein and deterministic binding rule which more closely resembled the real world. In addition, a potential method for increasing the generalisability of virtual screening models would be to include an attribution

term in the loss function when training, meaning that in order to attain a low loss, the generative model would have to correctly identify atoms which make important contributions to protein-ligand binding.

Appendix A

Using Reinforcement Learning to Generate Elaborations with a Specified Pharmacophoric Profile.

A.1 Malhotra Test Set

	Fragment	Ground Truth Elaboration
0	[*:1]N1CCC(n2cnc3cnc4[nH]ccc4c32)CC1	c1ccc(cc1)C[*:1]
1	[*:1]c1cc(Br)cc2cc(C(=O)O)oc12	N[C@H]1CC[C@H](CC1)N[*:1]
2	[*:1]c1ccc2c(c1)CN(C(=O)c1cc(C(C)C)c(O)cc1O)C2	CN1CCN(CC1)C[*:1]
3	[*:1]NS(=O)(=O)c1ccccc1	NS(=O)(=O)c1nnc(s1)[*:1]
4	[*:1]C=C(Oc1cccc(C(=O)O)c1O)C(=O)O	c1ccc(cc1)[*:1]
5	[*:1]c1ccc([C@@H]2C[C@@H]2c2cc(=O)n(C)c(N)n2)cc1	Cc1cccc(c1)[*:1]
6	[*:1][NH2+][C@H](CC)c1ccc(Cl)c(Oc2ccccc2)c1F	C[C@@H](CC(=O)NCC[NH3+])[*:1]
7	[*:1]c1ccc2nc(N)sc2c1	CCOC(=O)[*:1]
8	[*:1]c1ccc(-c2ccc3cc[nH]c3c2)cc1CNc1cccnc1N	c1ccc(CO[*:1])nc1
9	[*:1]CC(=O)OCC[N+](C)(C)C	C[N+](C)(C)CCOC(=O)C[*:1]
10	[*:1]Nc1nccc(-c2sc(C)nc2C)n1	Oc1cccc(c1)[*:1]
11	[*:1]c1ccc2c(-c3nc4ccccc4[nH]3)n[nH]c2c1	COc1cc(ccc1O)[*:1]
12	[*:1]c1cc2cc(OC)ccc2[nH]1	CS(=O)(=O)NC(=O)[*:1]
13	[*:1]CCc1ccc(S(N)(=O)=O)cc1	Cc1cc(C)[n+](c(C)c1)[*:1]
14	[*:1]c1cc(C=CC(=O)O)c(O)cc1OC	CC(C)C[C@H](O)[*:1]
15	[*:1]c1cccc(-c2nc3cc(C(N)=[NH2+])c(F)cc3[nH]2)c1[O-]	C[C@@H]1CCCC[C@@H]1O[*:1]
16	[*:1]ON=C1O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1NC(=O)CCC	*C(=O)Nc1ccccc1
17	[*:1]c1cc(N)cc2c(=O)[nH]c(N)nc12	c1c[nH]c(n1)SC[*:1]
18	[*:1]Cc1cc(N)cc2c(=O)nc(N)[nH]c12	c1c[nH]c(n1)S[*:1]

19	[*:1]N(C(C(=O)NO)C(C)C)S(=O)(=O)c1ccc(OC)cc1	*Cc1cccnc1
20	[*:1]NS(=O)(=O)c1ccc2c(c1)CNC(CF)C2	*CCC(F)(F)F
21	[*:1]S(=O)(=O)NC(=O)c1cc2cc(OC)ccc2[nH]1	Cc1ccc(nc1)[*:1]
22	[*:1]C(=O)Nc1c[nH]nc1C(=O)Nc1ccc(F)cc1	Fc1cccc(F)c1[*:1]
23	[*:1]S(=O)(=O)c1ccsc1C(=O)O	*Nc1ccc(Cl)cc1
24	[*:1]COC(=O)c1cccc(S(=O)(=O)Nc2nnc(S(N)(=O)=O)s2)c1	O=[N+]([O-])OCCC[*:1]
25	[*:1]CC(=O)Nc1ccc(Br)cc1C(=O)N1C[C@@H](C)C[C@@H](C)C1	O=C(C[*:1])N1CCOCC1
26	[*:1]c1ccc(S(N)(=O)=O)cc1F	O=C(Cc1cccs1)N[*:1]
27	[*:1]c1ccc([C@H]2CCCN2C(=O)c2cc(Cl)c(O)cc2O)cc1	*CN1CCC(F)(F)C1
28	[*:1]CCCn1c(N)nc2cc(Cl)ccc21	CN(C(=O)[*:1])C1CCCCC1
29	[*:1]c1cc(I)c(O)c(CN2CCC(N(CC)CC)CC2)c1	C(#C[*:1])CNc1ccccc1
30	[*:1]C1c2ccccc2-c2c(-c3nc4ccncc4[nH]3)cccc21	O=C(N[*:1])c1ccncc1
31	[*:1]N1CCC(Nc2cccc(-c3sc(C(=O)O)c(OCC(=O)O)c3Br)c2)CC1	*S(=O)(=O)Cc1ccccc1
32	[*:1]c1c2nc(NC)[nH]c2cc2c(=O)[nH]c(N)nc12	C1CCC(C1)CNCC[*:1]

Table A.1: Examples derived from the Malhotra set.

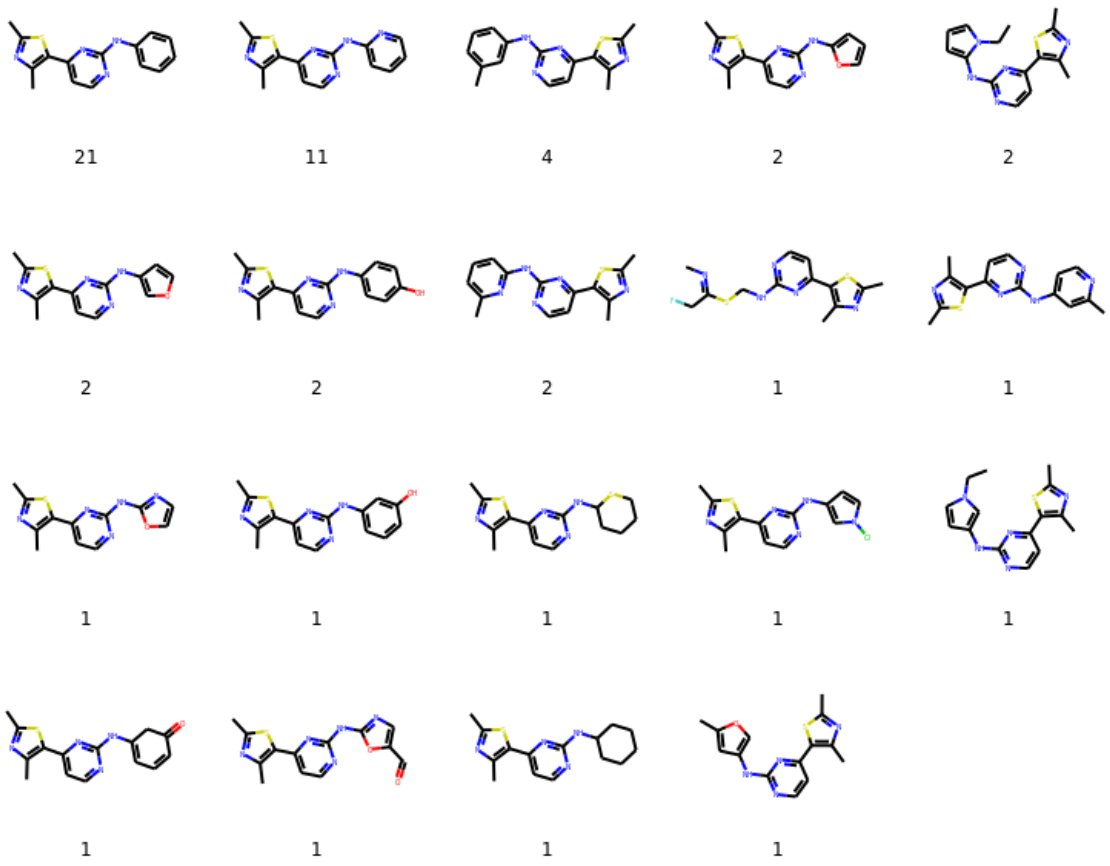

Figure A.1: Distinct elaborations generated by DeLinker-Curriculum which attained an SC_{RDKit} score of greater than 0.7 with the phenol ground truth. The number beneath each molecule denotes the number of times it was generated by DeLinker-Curriculum (from a total of 500 elaborations).

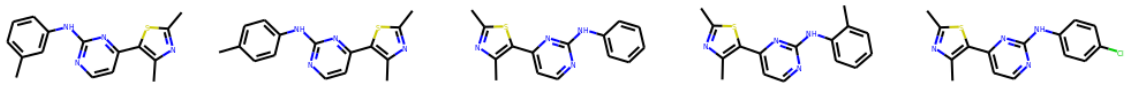

Figure A.2: Distinct elaborations generated by DeLinker-SC which attained an SC_{RDKit} score of greater than 0.7 with the phenol ground truth. The number beneath each molecule denotes the number of times it was generated by DeLinker-SC (from a total of 500 elaborations).

Appendix B

Generating Focused Sets of Elaborations via Pharmacophoric Constraints.

B.1 Docking of molecules generated in the R-group optimisation case study.

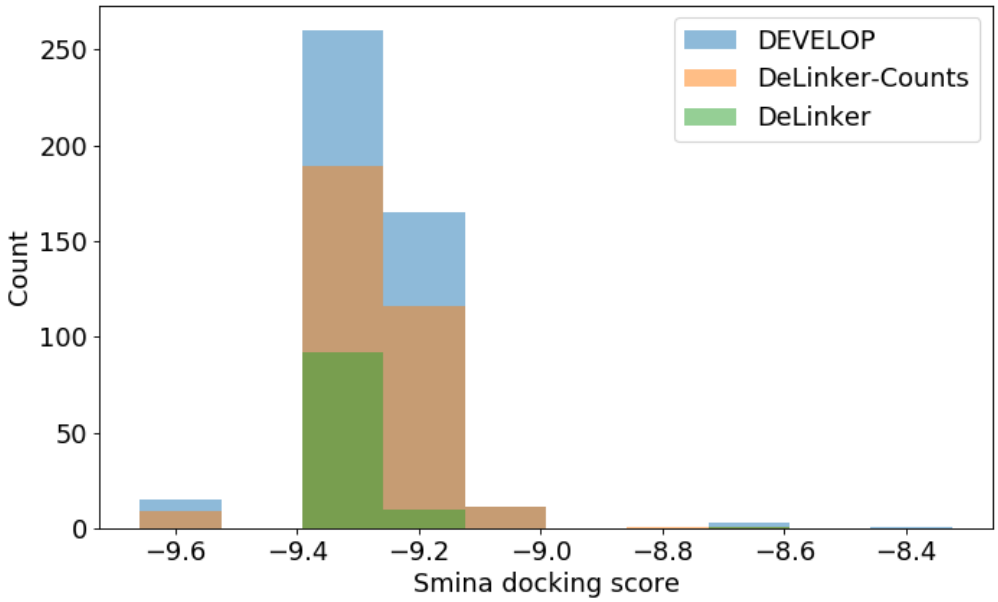

Figure B.1: R-group optimisation case study. Predicted binding affinities from docking using the smina (Koes et al., 2013) version of AutoDock Vina (Trott and Olson, 2010) for molecules that satisfied the pharmacophoric profile of the acetamide group.

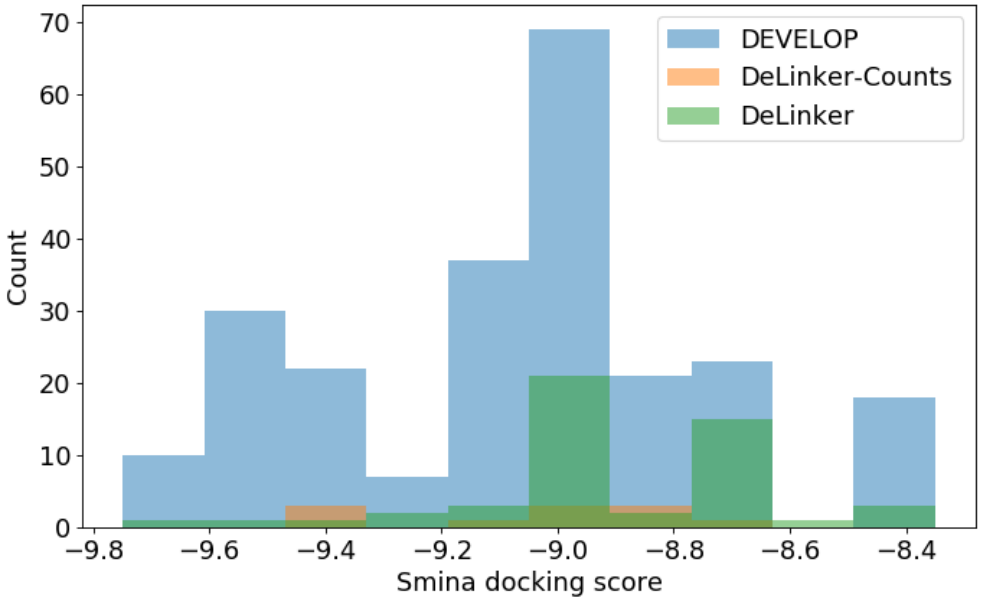

Figure B.2: R-group optimisation case study. Predicted binding affinities from docking using the smina (Koes et al., 2013) version of AutoDock Vina(Trott and Olson, 2010) for molecules that satisfied the pharmacophoric profile of the 4-methylpyrazole elaboration.

Appendix C

Incorporating Target-Specific Pharmacophoric Information Into Deep Generative Models For Fragment Elaboration.

C.1 Examples in the CASF Test Set

PDB ID	Fragment SMILES String
1pxn	CNc1nc(C)c(s1)-c1ccnc(n1)N[*:1]
1q8t	CC([NH3+])C1CCC(CC1)C(=O)N[*:1]
1uto	c1ccc(cc1)[*:1]
1ydt	O=S(=O)(NCC[NH2+]C[*:1])c1cccc2cnccc12
1ydt	O=S(=O)(NCC[NH2+]CC=C[*:1])c1cccc2cnccc12
1z95	CC(O)(CS(=O)(=O)c1ccc(F)cc1)C(=O)Nc1ccc(C#N)c(c1)[*:1]
2br1	COc1ccc(cc1)-c1oc2ncnc(c2c1-c1ccc(cc1)OC)[*:1]
2br1	COc1ccc(cc1)-c1oc2ncnc(N[*:1])c2c1-c1ccc(cc1)OC
2br1	COc1ccc(cc1)-c1c(oc2ncnc(NCCO)c12)[*:1]
2brb	c1ccc(cc1)-c1oc2ncnc(N[*:1])c2c1-c1ccccc1
2brb	c1ccc(cc1)-c1oc2ncnc(c2c1-c1ccccc1)[*:1]
2c3i	CC(=O)c1cccc(c1)-c1cnc2ccc(nn12)NC[*:1]
2c3i	CC(=O)c1cccc(c1)-c1cnc2ccc(nn12)N[*:1]
2c3i	CC(=O)c1cccc(c1)-c1cnc2ccc(nn12)[*:1]
2cet	OCC1C(O)C(O)C(O)c2nc(cn21)C[*:1]
2cet	OCC1C(O)C(O)C(O)c2nc(cn21)CC[*:1]
2fvd	CS(=O)(=O)N1CCC(CC1)Nc1ncc(C(=O)[*:1])c(N)n1
2fvd	COc1ccc(F)c(F)c1C(=O)c1cnc(nc1N)N[*:1]
2p15	CC12CCC3c4ccc(O)cc4CCC3C1CCC2(O)C=C[*:1]
2p15	CC12CCC3c4ccc(O)cc4CCC3C1CCC2(O)/C=C/c1ccccc1[*:1]
2pog	Oc1cccc2OC(C3CCCC3c12)[*:1]
2qe4	COCc1cc(O)cc2c1OC(C1CCCC21)[*:1]
2w66	OCC1[NH2+]CC(O)C(C(O)C1O)[*:1]
2wbg	OC1C(O)C(O)N2/C(=N/CCC[*:1])OCC2C1O

2wbg	OC1C(O)C(O)N2/C(=N/C[*:1])OCC2C1O
2wbg	OC1C(O)C(O)N2/C(=N/CCCC[*:1])OCC2C1O
2wbg	OC1C(O)C(O)N2/C(=N/CC[*:1])OCC2C1O
2wbg	OC1C(O)C(O)N2/C(=N/CCCCC[*:1])OCC2C1O
2wn9	COc1cc(O)ccc1/C=C1\CCCN=C1[*:1]
2xbv	O=C(Nc1ccc(cc1F)-n1ccccc1=O)C1C[NH+](CC(F)F)CC1[*:1]
2xbv	O=C(Nc1ccc(cc1F)-n1ccccc1=O)C1C[NH+](CC(F)F)CC1C(=O)N[*:1]
2xbv	O=C(Nc1ccc(Cl)cn1)C1C[NH+](CC(F)F)CC1C(=O)Nc1ccc(cc1F)[*:1]
2xj7	OC1CC[NH+]2CC(C(O)C(O)C12)[*:1]
2yki	O=C(NC1c2ccccc2-c2c(cccc21)-c1nc2ccncc2[nH]1)[*:1]
2yki	O=C(NC1c2ccccc2-c2c1cccc2[*:1])c1ccnc2[nH]ccc12
2ymd	Oc1ccc2[nH]cc(c2c1)[*:1]
2zb1	Cc1nnc(o1)-c1ccc(C)c(c1)-c1ccc(cc1)C(=O)N[*:1]
2zb1	Cc1nnc(o1)-c1ccc(C)c(c1)-c1ccc(cc1)C(=O)NC[*:1]
2zb1	Cc1nnc(o1)-c1ccc(C)c(c1)-c1ccc(cc1)[*:1]
2zb1	Cc1ccc(cc1-c1ccc(cc1)C(=O)NCC1CC1)[*:1]
3acw	OC1(C[NH+]2CCC1CC2)c1ccc(cc1)[*:1]
3aru	Cn1cnc2c1c(=O)n(CC[*:1])c(=O)n2C
3aru	Cn1cnc2c1c(=O)n(C[*:1])c(=O)n2C
3aru	Cn1cnc2c1c(=O)n(CCCC[*:1])c(=O)n2C
3aru	Cn1cnc2c1c(=O)n(CCC[*:1])c(=O)n2C
3ary	c1cc2oc(cc2cc1[*:1])C1=NCCN1
3b1m	COc1cc(O)c2c(OC3=CC(O)=C(C(C)=O)C(=O)C32C)c1C(=O)NC[*:1]
3b5r	CC(O)(CO[*:1])C(=O)Nc1ccc(C#N)c(c1)C(F)(F)F
3b65	CC(O)(CO[*:1])C(=O)Nc1ccc(C#N)c(I)c1
3b68	CC(O)(CO[*:1])C(=O)Nc1ccc(c(c1)C(F)(F)F)[N+](=O)[O-]
3b68	CC(O)(COc1ccc(cc1)[*:1])C(=O)Nc1ccc(c(c1)C(F)(F)F)[N+](=O)[O-]

3b68	CC(=O)Nc1ccc(cc1)OCC(C)(O)C(=O)Nc1ccc(c(c1)[*:1])[N+](=O)[O-]
3coy	CC(C)(C)C([NH3+])C(=O)NS(=O)(=O)OCC1OC(C(O)C1O)[*:1]
3coy	Nc1ncnc2c1ncn2C1OC(COS(=O)(=O)NC(=O)C([NH3+])[*:1])C(O)C1O
3coz	Nc1ncnc2c1ncn2C1OC(COS(=O)(=O)NC(=O)C([NH3+])[*:1])C(O)C1O
3coz	CC(C)C([NH3+])C(=O)NS(=O)(=O)OCC1OC(C(O)C1O)[*:1]
3e92	Cc1cc(ccc1-c1cc(ccc1C)C(=O)NC1CC1)[*:1]
3e92	Cc1nnc(o1)-c1ccc(c(C)c1)-c1cc(ccc1C)[*:1]
3fur	O=S(=O)(Nc1cc(Cl)c(Oc2cnc3ccccc3c2)c(Cl)c1)[*:1]
3fur	O=S(=O)(Nc1cc(Cl)c(O[*:1])c(Cl)c1)c1ccc(Cl)cc1Cl
3g0w	Cc1c(ccc(C#N)c1Cl)/N=C1\OC(C2C(O)CCN12)[*:1]
3g2n	O=C(NC1OC(CO)C(O)C(O)C1O)[*:1]
3lka	COc1ccc(cc1)[*:1]
3nx7	COc1ccc(cc1)S(=O)(=O)N(CCO)[*:1]
3nx7	COc1ccc(cc1)S(=O)(=O)N(CCO)C[*:1]
3qgy	c1ccc(cc1)-c1cc2c(ccc3cnc(nc32)Nc2cccc(c2)[*:1])s1
3qgy	c1ccc(cc1)-c1cc2c(ccc3cnc(nc32)N[*:1])s1
3rsx	Nc1ccc2cc(ccc2n1)[*:1]
3u8k	c1ncc(cc1N1CCC[NH2+]CC1)[*:1]
3u8n	Brc1ncc(cc1[*:1])N1CCC[NH2+]CC1
4dli	Nc1ccc2c(nc(nc2c1)-c1ccccc1)[*:1]
4dli	Nc1ccc2c(nc(nc2c1)[*:1])NC1CC1
4dli	Nc1ccc2c(nc(nc2c1)-c1ccccc1)N[*:1]
4e6q	c1cc2c(ncc3ncn(c32)C2CC[NH+](CC2)C[*:1])[nH]1
4ea2	C/C(=C/C[NH2+]CC[NH2+]C1C2CC3CC(C2)CC1C3)C[*:1]
4eky	O=C1NC(=O)N(CC1=C=C[*:1])C1OC(CO)C(O)C(O)C1O
4eor	c1nc2c(nc(nc2[nH]1)Nc1ccc(cc1)[*:1])OCC1CCCCC1
4eor	NS(=O)(=O)c1ccc(cc1)Nc1nc(c2nc[nH]c2n1)[*:1]

4eor	NS(=O)(=O)c1ccc(cc1)Nc1nc(OC[*:1])c2nc[nH]c2n1
4eor	NS(=O)(=O)c1ccc(cc1)Nc1nc(O[*:1])c2nc[nH]c2n1
4f3c	Nc1ncnc2c(c[nH]c12)C[NH+]1CC(O)C(CSC[*:1])C1
4f9w	c1ccc2cc(ccc2c1)-c1nnc(cc1-c1ccncc1)[*:1]
4f9w	CN(C)c1cc(c(nn1)-c1ccc2ccccc2c1)[*:1]
4ih7	O=C1N=CC=CC1c1cccc(c1)[*:1]
4ivb	N#CC1CCC(CC1)n1c(nc2cnc3[nH]ccc3c21)[*:1]
4ivd	N#CCCC1CCC(CC1)n1c(nc2cnc3[nH]ccc3c21)[*:1]
4ivd	CC(O)c1nc2cnc3[nH]ccc3c2n1C1CCC(CC1)[*:1]
4ivd	CC(O)c1nc2cnc3[nH]ccc3c2n1C1CCC(CC1)C[*:1]
4lzs	CCc1c([nH]c(C)c1[*:1])C(=O)NC
4lzs	CCc1c([nH]c(C)c1C(C)=O)[*:1]
4m0y	NC(=O)Nc1cn(nc1[*:1])-c1cccc2ccccc12
4m0z	COc1ccc2cccc(c2c1)-n1cc(c(n1)C(N)=O)[*:1]
4m0z	COc1ccc2cccc(c2c1)-n1cc(NC(N)=O)c(n1)[*:1]
4mgd	Oc1ccc(cc1)C(c1ccc(O)cc1)[*:1]
4qac	Nc1nc(cc(n1)[*:1])-c1ccc(cc1)C(F)(F)F
4twp	CNC(=O)c1ccccc1Sc1ccc2c(C=C[*:1])[nH]nc2c1
4twp	C(=C/c1[nH]nc2cc(ccc12)Sc1ccccc1[*:1])\c1ccccn1
4twp	CNC(=O)c1ccccc1Sc1ccc2c([nH]nc2c1)[*:1]
5c2h	Cc1c(Cl)nc(nc1NC[*:1])OCCCc1ccc2ccccc2n1
5c2h	Cc1c(Cl)nc(nc1N[*:1])OCCCc1ccc2ccccc2n1
5c2h	Cc1nc(C)c(CNc2nc(nc(Cl)c2C)OCCC[*:1])s1

Table C.1: Examples used for the large scale evaluation.

Metric	All Examples	> 50 Elaborations	250 Elaborations
Valid	**100**%	**100**%	**100**%
Unique	N/A	N/A	N/A
Novel	N/A	N/A	N/A
Pass 2D filters	**67.44**%	66.94%	66.06%
ΔQED	**-0.104**	-0.124	-0.148
ΔSLE_{20}	-2.015	-0.037	**-0.029**
ΔSLE_{50}	-2.443	-0.536	**-0.489**
ΔSLE_{100}	-2.812	-1.024	**-0.992**

Table C.2: Comparison of results obtained by CReM, on different sets of examples.

C.2 Performance of DeepFrag

C.2.1 Most Highly Ranked Elaborations

Below we present the elaborations most frequently identified by DeepFrag as the top-ranked elaboration (based on its own scoring function rather than predicted ligand efficiency). The number below each molecule represents the number of times DeepFrag identified the elaboration as the top-ranked one.

Figure C.1 indicates that DeepFrag consistently favours either very simple elaborations or very large ones, calling into question exactly how DeepFrag uses the structural information provided to it as one would not expect the same very large to be the optimal choice for a variety of different fragments and binding pockets. Of the 101 examples in our testing set, DeepFrag proposed a total of 19 different top-ranked elaborations compared to 64 proposed by STRIFE. We show the top-ranked elaborations proposed by STRIFE in Figure C.2.

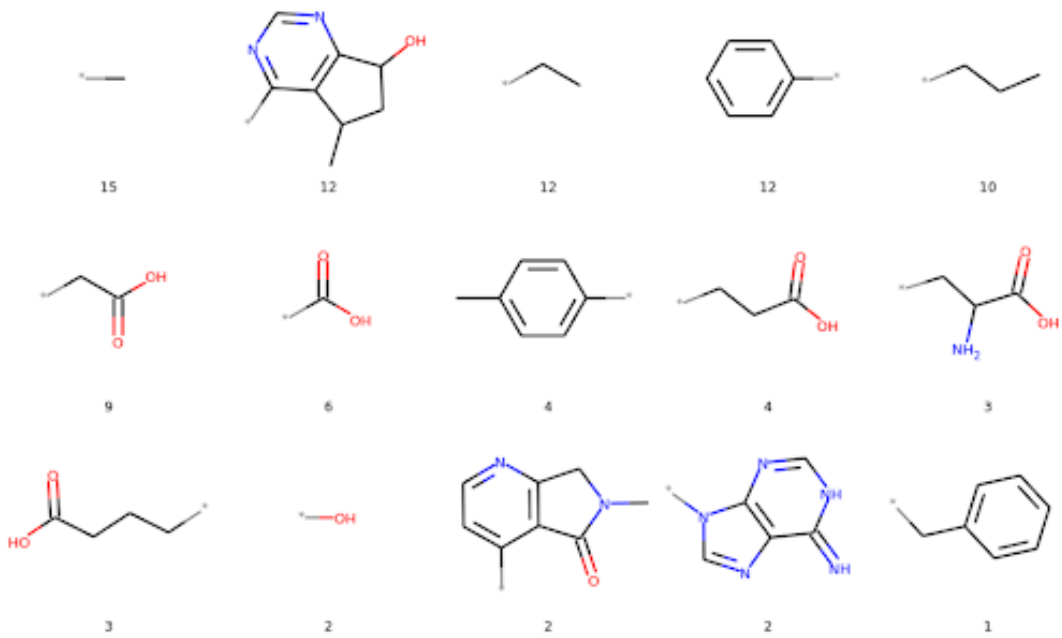

Figure C.1: The 15 most common top-ranked elaborations proposed by DeepFrag (Dummy atom indicates exit vector). The number beneath each elaboration denotes the number of times the elaboration was the top-ranked one.

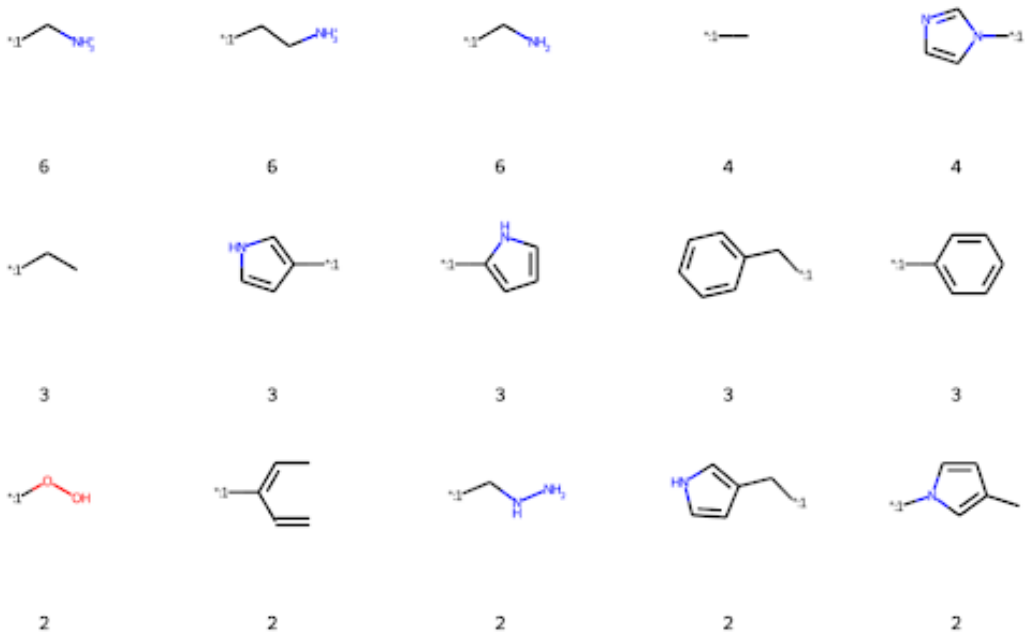

Figure C.2: The 15 most common top-ranked elaborations proposed by STRIFE. Most of the pictured elaborations are of relatively short length, reflecting the smaller chemical space available when attempting to satisfy a pharmacophoric point close to the fragment exit vector.

C.2.2 Generated Molecules

We present all unique molecules generated in both case studies, ranked by their ligand efficiency.

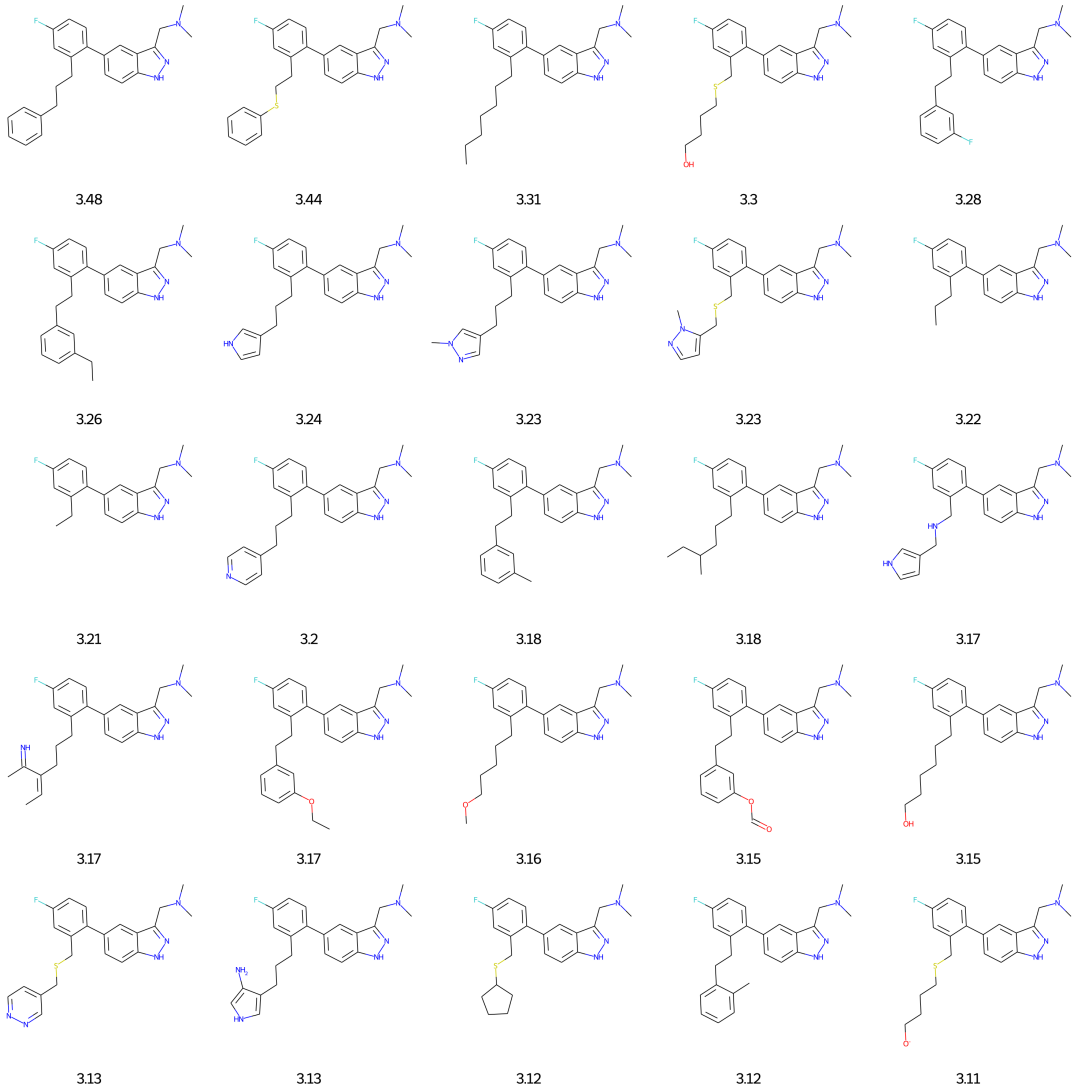

Figure C.3: Unique molecules generated on first case study with associated ligand efficiency: Rank 1-25

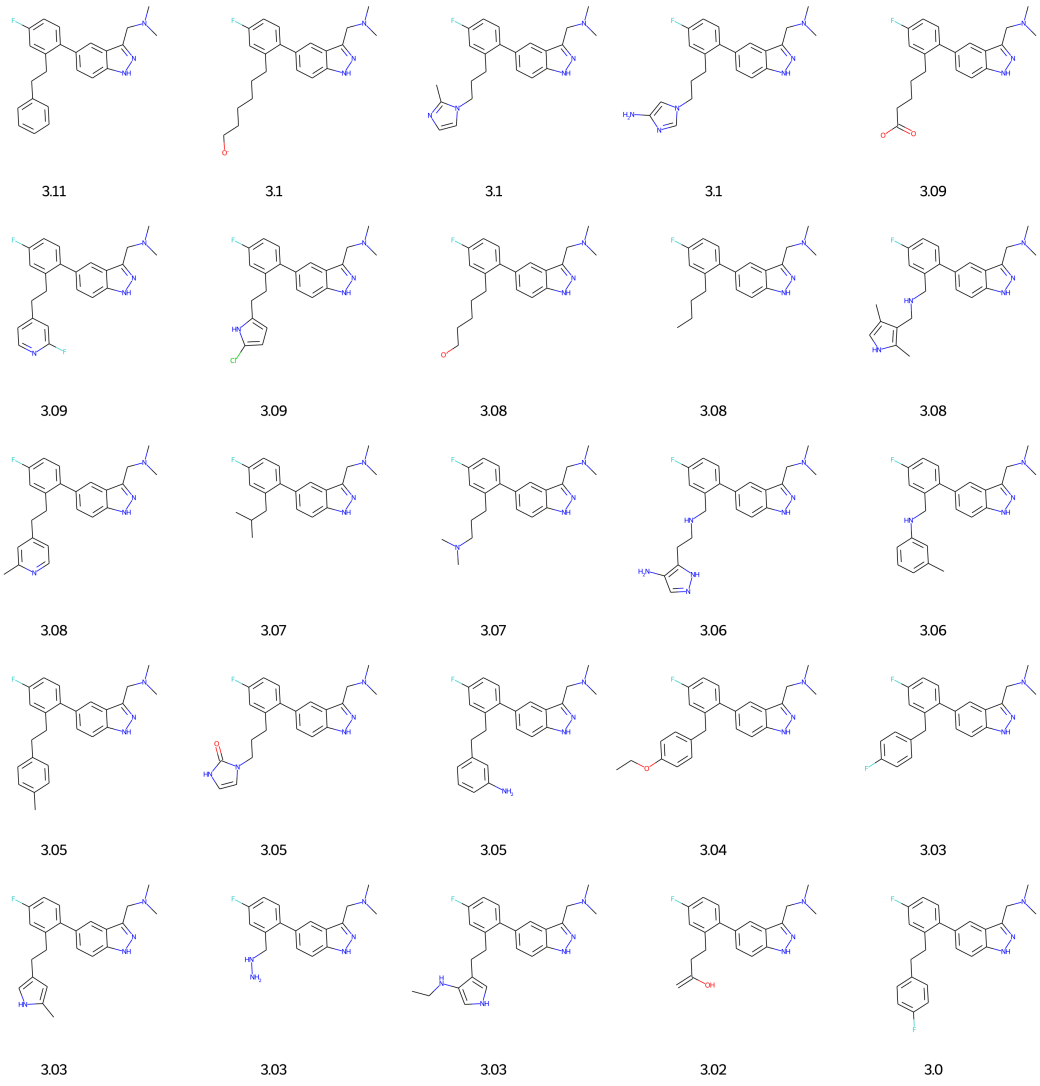

Figure C.4: Unique molecules generated on first case study with associated ligand efficiency: Rank 26-50

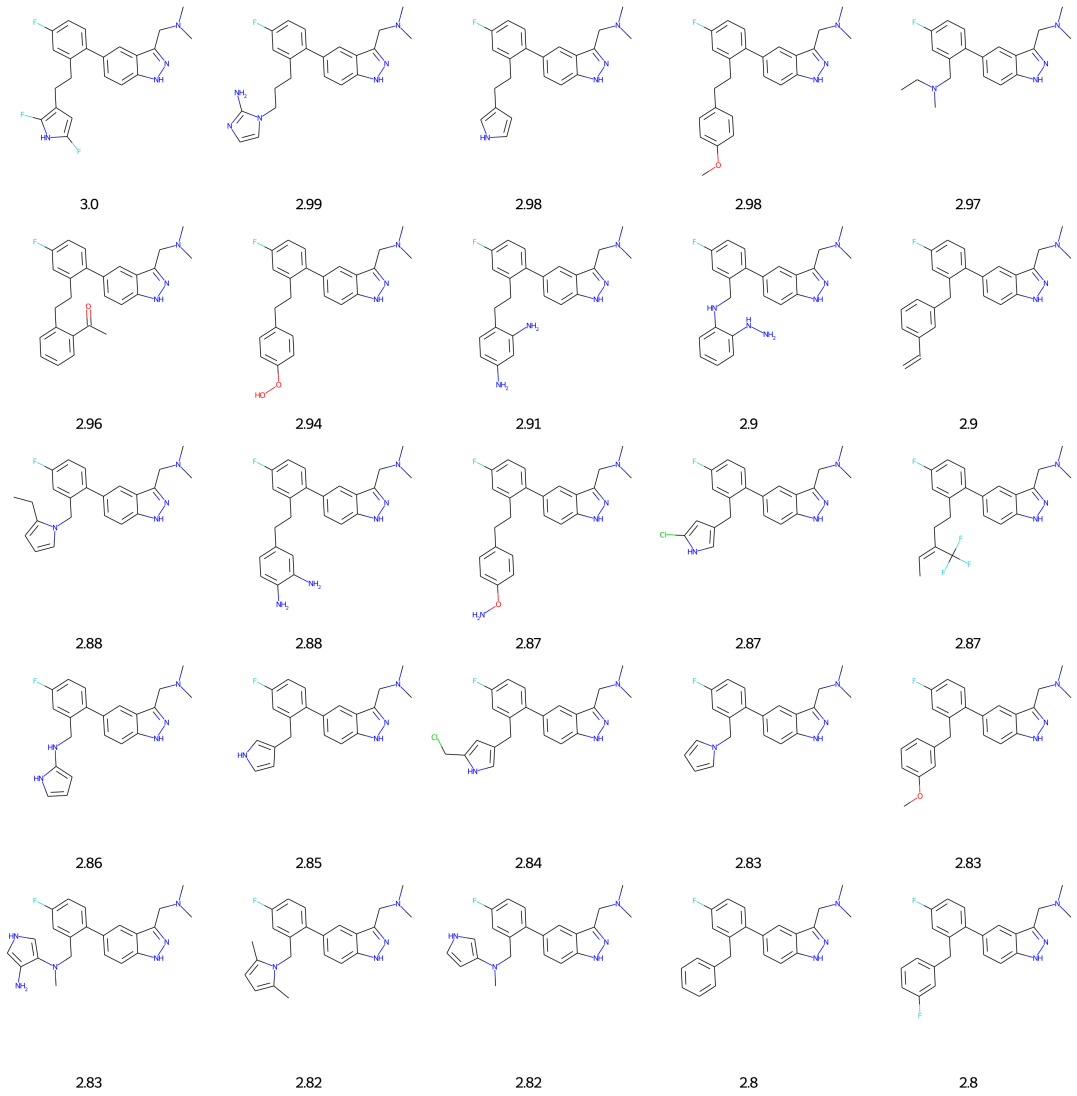

Figure C.5: Unique molecules generated on first case study with associated ligand efficiency: Rank 51-75

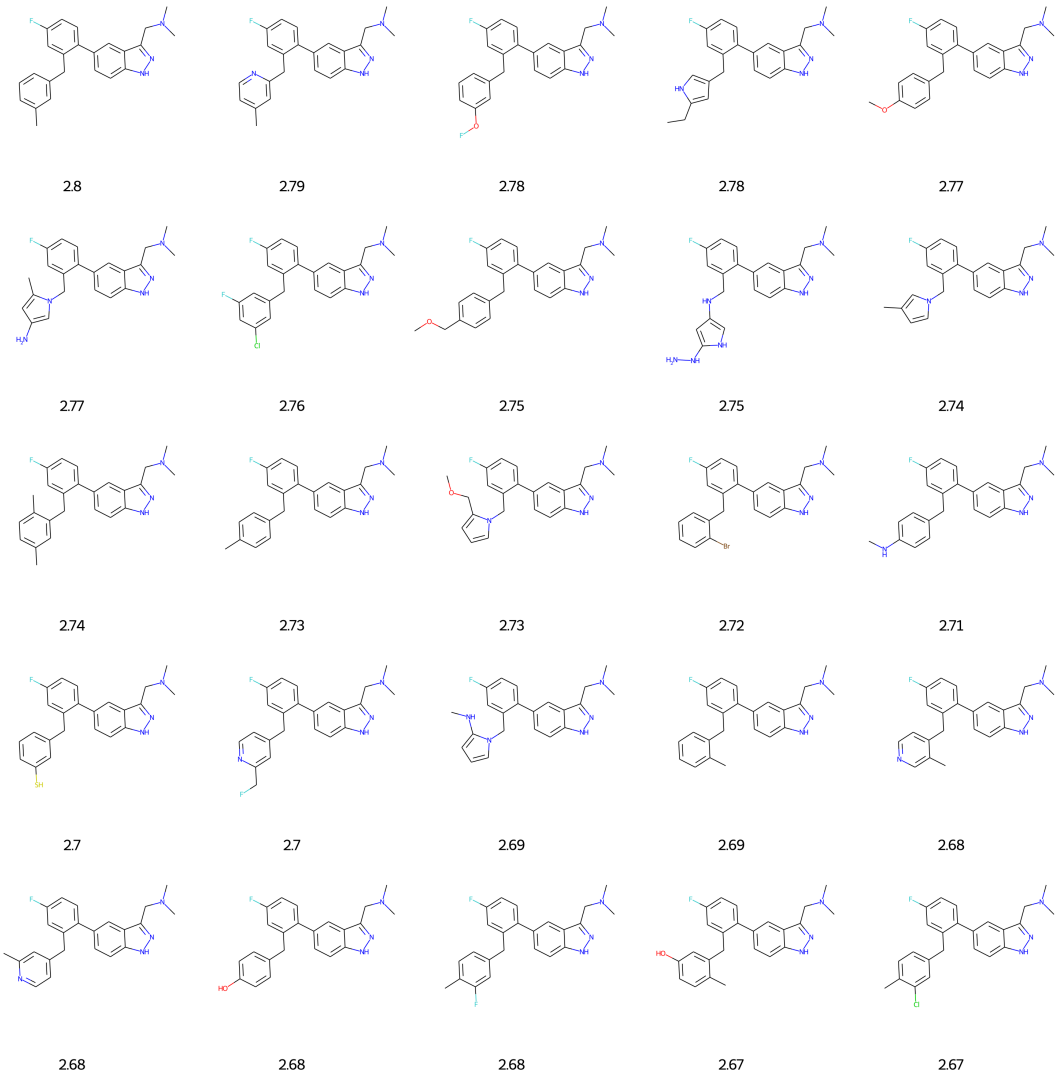

Figure C.6: Unique molecules generated on first case study with associated ligand efficiency: Rank 76-100

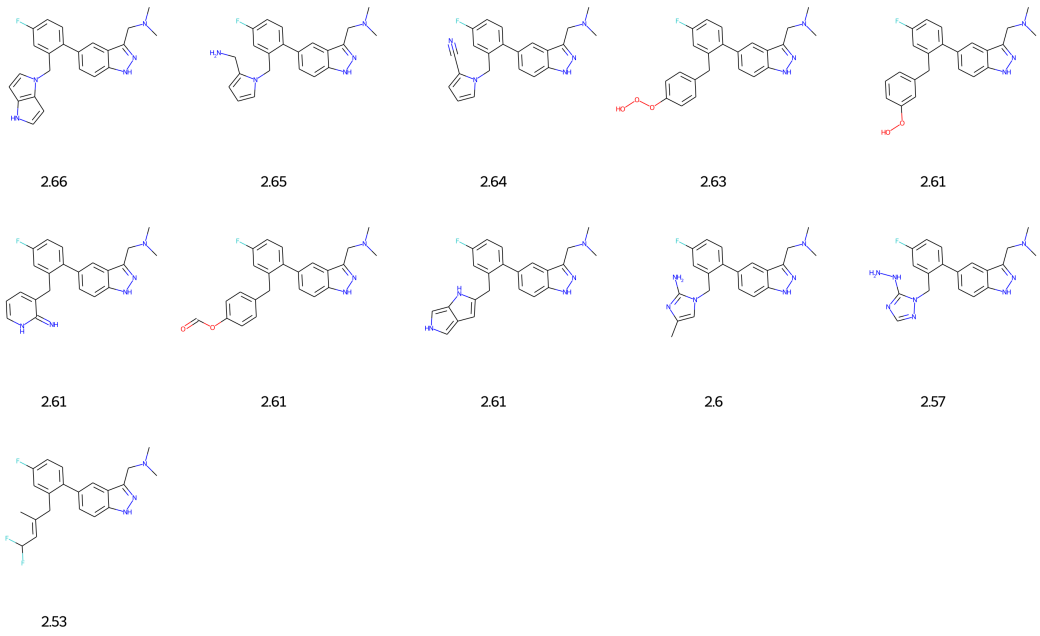

Figure C.7: Unique molecules generated on first case study with associated ligand efficiency: Rank 101-111

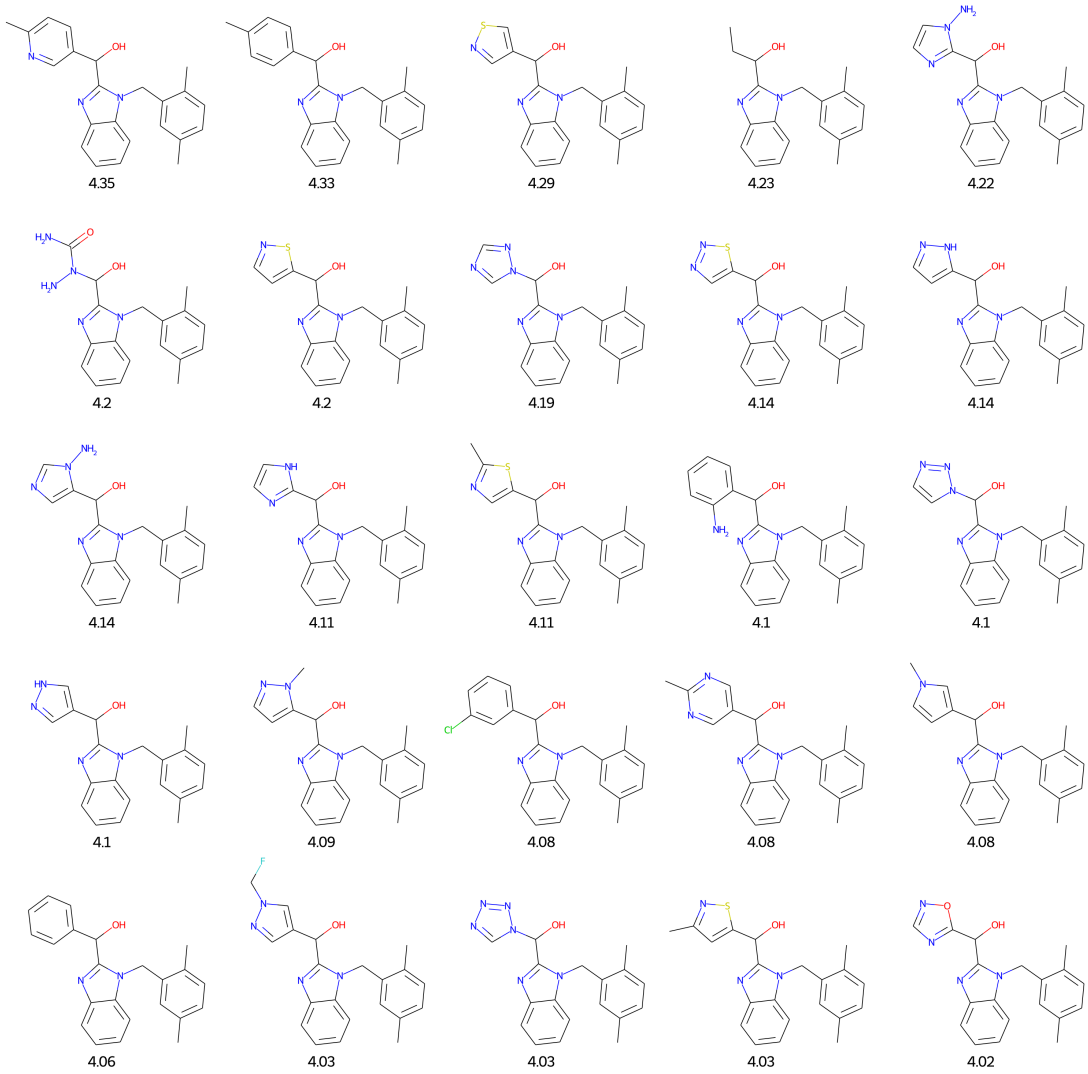

Figure C.8: Unique molecules generated on second case study with associated ligand efficiency: Rank 1-25

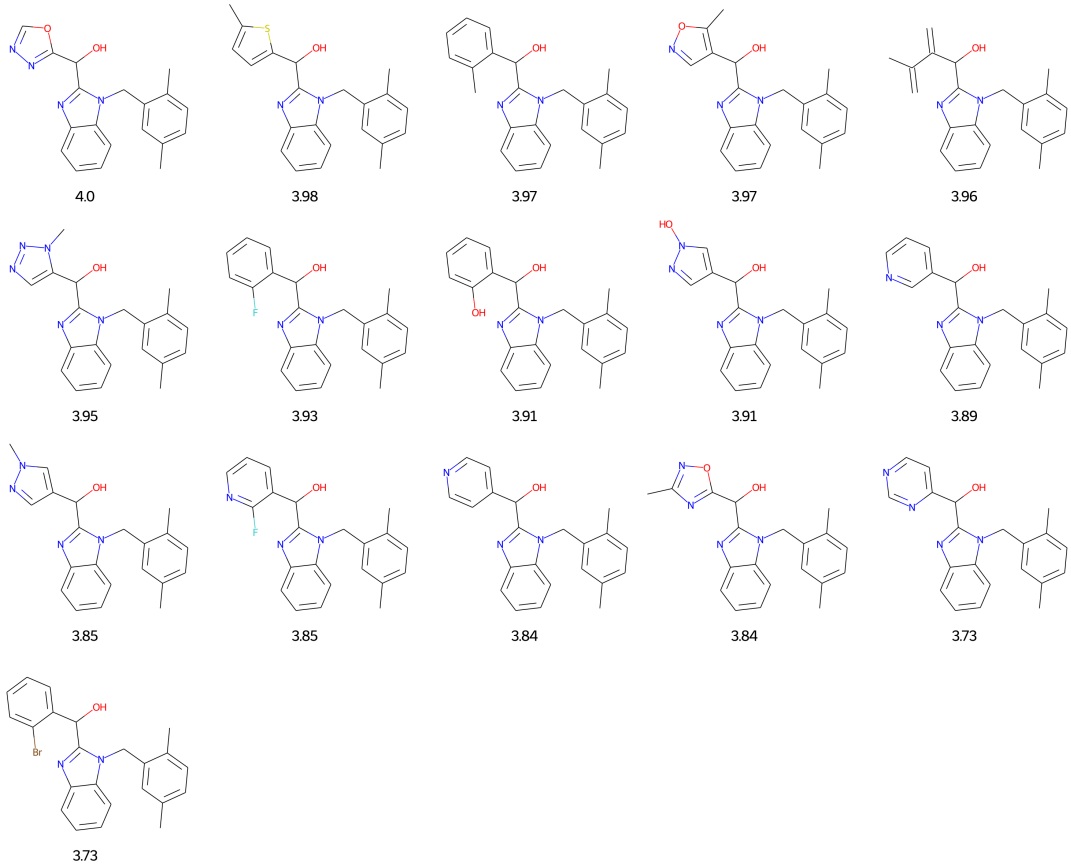

Figure C.9: Unique molecules generated on second case study with associated ligand efficiency: Rank 26-44

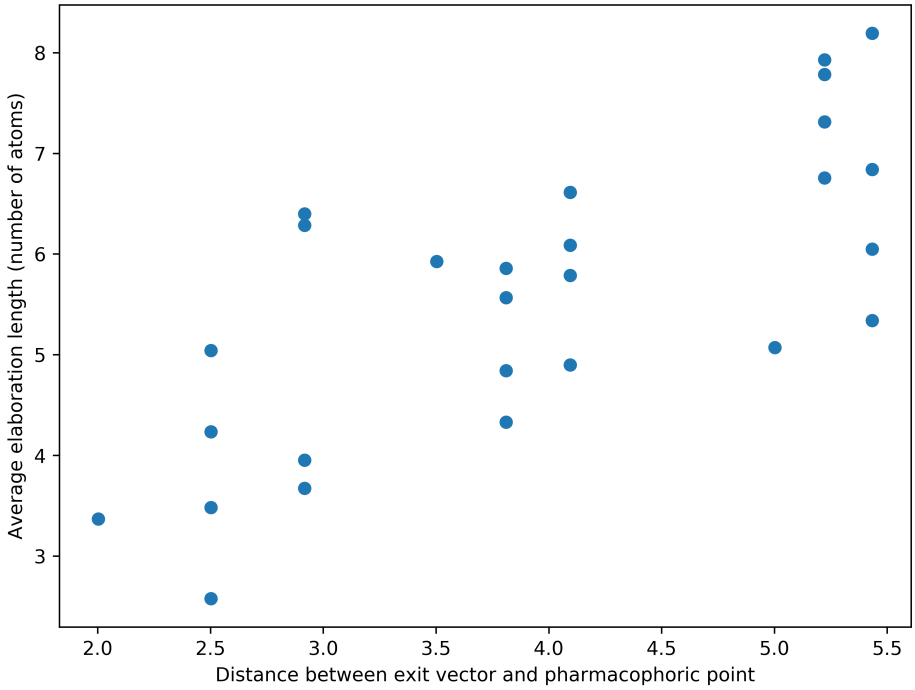

Figure C.10: The average elaboration size increases as the pharmacophoric point moves further away from the fragment exit vector

C.2.3 Perturbuing the Pharmacophoric Point In The Customisability Case Study.

C.3 GOLD Flexible Docking Configuration File

The following configuration file was used to carry out flexible docking in GOLD. Verdonk et al. (2003).

```
  GOLD CONFIGURATION FILE


  AUTOMATIC SETTINGS
autoscale = 0.1


  POPULATION
popsiz = auto
select_pressure = auto
n_islands = auto
maxops = auto
niche_siz = auto


  GENETIC OPERATORS
pt_crosswt = auto
allele_mutatewt = auto
migratewt = auto


  FLOOD FILL
radius = 10
origin = 0 0 0
do_cavity = 1
floodfill_atom_no = 0
cavity_file = <path/to/cavity/file>.mol2
```

```
floodfill_center = cavity_from_ligand 10 atoms

  DATA FILES
ligand_data_file <path/to/ligands>.sdf 100
param_file = DEFAULT
set_ligand_atom_types = 1
set_protein_atom_types = 0
directory = <output_directory>
tordist_file = DEFAULT
make_subdirs = 0
save_lone_pairs = 1
fit_points_file = fit_pts.mol2
read_fitpts = 0

  FLAGS
internal_ligand_h_bonds = 0
flip_free_corners = 0
match_ring_templates = 0
flip_amide_bonds = 0
flip_planar_n = 1 flip_ring_NRR flip_ring_NHR
flip_pyramidal_n = 0
rotate_carboxylic_oh = flip
use_tordist = 1
postprocess_bonds = 1
rotatable_bond_override_file = DEFAULT
solvate_all = 1
```

```
  TERMINATION
early_termination = 0
n_top_solutions = 3
rms_tolerance = 1.5

  CONSTRAINTS
force_constraints = 0
constraint scaffold <path/to/scaffold>.mol2 5.0000

  COVALENT BONDING
covalent = 0

  SAVE OPTIONS
save_score_in_file = 1 comments
save_protein_torsions = 1
concatenated_output = <output_directory>/docked_ligands.sdf
output_file_format = MACCS

  FITNESS FUNCTION SETTINGS
initial_virtual_pt_match_max = 3
relative_ligand_energy = 1
gold_fitfunc_path = plp
score_param_file = DEFAULT

  PROTEIN DATA
protein_datafile = <path/to/protein>/6ooy_protein.pdb
```

```
rotamer_lib
  name TYR119
 chi1 1586 1585 1587 1586
 chi2 1585 1586 1589 1590
 chi3 1586 1589 1590 1591
 rotamer 0 (60) 0 (180) 0 (180)
end_rotamer_lib

rotamer_lib
  name LEU120
 chi1 1607 1606 1608 1607
 chi2 1606 1607 1610 1611
 chi3 1607 1610 1611 1613
 rotamer 0 (60) 0 (180) 0 (180)
end_rotamer_lib
```

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning On Heterogeneous Systems. Software available from tensorflow.org.

Abdi, H. and Williams, L. J. (2010). Principal Component Analysis. Wiley Interdiscip. Rev. Comput. Stat., 2(4):433–459.

Ain, Q. U., Aleksandrova, A., Roessler, F. D., and Ballester, P. J. (2015). Machine-Learning Scoring Functions To Improve Structure-Based Binding Affinity Prediction And Virtual Screening. Wiley Interdiscip. Rev. Comput. Mol. Sci., 5(6):405–424.

Alagoz, O., Hsu, H., Schaefer, A. J., and Roberts, M. S. (2010). Markov Decision Processes: A Tool For Sequential Decision Making Under Uncertainty. Med Decis Making, 30(4):474–483.

Arús-Pous, J., Patronov, A., Bjerrum, E. J., Tyrchan, C., Reymond, J.-L., Chen, H., and Engkvist, O. (2020). SMILES-Based Deep Generative Scaffold Decorator For De-Novo Drug Design. J. Cheminf., 12(1):38.

Baell, J. B. and Holloway, G. A. (2010). New Substructure Filters For Removal Of Pan Assay Interference Compounds (PAINS) From Screening Libraries And For Their Exclusion In Bioassays. J. Med. Chem., 53(7):2719–2740.

Ballester, P. J. and Mitchell, J. B. (2010). A Machine Learning Approach To Predicting Protein–Ligand Binding Affinity With Applications To Molecular Docking. Bioinformatics, 26(9):1169–1175.

Balls, M., Bailey, J., and Combes, R. D. (2019). How Viable Are Alternatives to Animal Testing in Determining the Toxicities of Therapeutic Drugs?

Bell, A. S., Mills, J. E., Williams, G. P., Brannigan, J. A., Wilkinson, A. J., Parkinson, T., Leatherbarrow, R. J., Tate, E. W., Holder, A. A., and Smith, D. F.

(2012). Selective Inhibitors Of Protozoan Protein N-Myristoyltransferases As Starting Points For Tropical Disease Medicinal Chemistry Programs. PLoS Negl Trop Dis, 6(4):e1625.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. Nucleic Acids Res., 28(1):235–242.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. (2012). Quantifying The Chemical Beauty Of Drugs. Nat. Chem., 4(2):90–98.

Bishop, C. M. and Nasrabadi, N. M. (2006). Pattern Recognition And Machine Learning, volume 4. Springer.

Blay, V., Tolani, B., Ho, S. P., and Arkin, M. R. (2020). High-Throughput Screening: Today's Biochemical And Cell-Based Approaches. Drug Discov. Today, 25(10):1807–1821.

Boittier, E. D., Tang, Y. Y., Buckley, M. E., Schuurs, Z. P., Richard, D. J., and Gandhi, N. S. (2020). Assessing Molecular Docking Tools To Guide Targeted Drug Discovery Of CD38 Inhibitors. Int. J Mol. Sci., 21(15):5183.

Borkin, D., He, S., Miao, H., Kempinska, K., Pollock, J., Chase, J., Purohit, T., Malik, B., Zhao, T., Wang, J., Wen, B., Zong, H., Jones, M., Danet-Desnoyers, G., Guzman, M., Talpaz, M., Bixby, D., Sun, D., Hess, J., Muntean, A., Maillard, I., Cierpicki, T., and Grembecka, J. (2015). Pharmacologic inhibition of the menin-mll interaction blocks progression of mll leukemia in vivo. Cancer Cell, 27(4):589 – 602.

Borkin, D., Pollock, J., Kempinska, K., Purohit, T., Li, X., Wen, B., Zhao, T., Miao, H., Shukla, S., He, M., Sun, D., Cierpicki, T., and Grembecka, J. (2016). Property Focused Structure-Based Optimization Of Small Molecule Inhibitors Of The Protein–Protein Interaction Between Menin And Mixed Lineage Leukemia (MLL). J. Med. Chem., 59(3):892–913.

Boyles, F., Deane, C. M., and Morris, G. M. (2021). Learning From Docked Ligands: Ligand-Based Features Rescue Structure-Based Scoring Functions When Trained On Docked Poses. J. Chem. Inf. Model.

Breiman, L. (2001). Random Forests. Mach. Learn., 45(1):5–32.

Brown, B. P., Mendenhall, J., Geanes, A. R., and Meiler, J. (2021). General Purpose Structure-Based Drug Discovery Neural Network Score Functions With Human-Interpretable Pharmacophore Maps. J. Chem. Inf. Model., 61(2):603–620.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (2019). GuacaMol: Benchmarking Models for De Novo Molecular Design. J. Chem. Inf. Model., 59(3):1096–1108.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language Models Are Few-Shot Learners. Proc. Adv. Neural Inf. Process. Syst., 33:1877–1901.

Brugger, W., Stuper, A., and Jurs, P. C. (1976). Generation Of Descriptors From Molecular Structures. J. Chem. Inf. Comput. Sci., 16(2):105–110.

Cao, Z., Xu, S., Peng, H., Yang, D., and Zidek, R. (2021). Confidence-Aware Reinforcement Learning For Self-Driving Cars. IEEE Trans. Intell. Transp. Syst.

Chen, L., Cruz, A., Ramsey, S., Dickson, C. J., Duca, J. S., Hornak, V., Koes, D. R., and Kurtzman, T. (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.

Chhabra, M. (2021). Biological Therapeutic Modalities. In Translational Biotechnology, pages 137–164. Elsevier.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder For Statistical Machine Translation. ArXiv Preprint ArXiv:1406.1078.

Coley, C. W., Rogers, L., Green, W. H., and Jensen, K. F. (2018). SCScore: Synthetic Complexity Learned from a Reaction Corpus. J. Chem. Inf. Model., 58(2):252–261.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. Mach. Learn., 20(3):273–297.

Cournia, Z., Allen, B., and Sherman, W. (2017). Relative Binding Free Energy Calculations in Drug Discovery: Recent Advances and Practical Considerations. J. Chem. Inf. Model., 57(12):2911–2937.

Curran, P. R., Radoux, C. J., Smilova, M. D., Sykes, R. A., Higueruelo, A. P., Bradley, A. R., Marsden, B. D., Spring, D. R., Blundell, T. L., Leach, A. R., Pitt, W. R., and Cole, J. C. (2020). Hotspots API: A Python Package For The Detection Of Small Molecule Binding Hotspots And Application To Structure-Based Drug Design. J. Chem. Inf. Model., 60(4):1911–1916.

Denis, J. D. S., Hall, R. J., Murray, C. W., Heightman, T. D., and Rees, D. C. (2021). Fragment-Based Drug Discovery: Opportunities For Organic Synthesis. RSC Med. Chem., 12(3):321–329.

Dreiman, G. H., Bictash, M., Fish, P. V., Griffin, L., and Svensson, F. (2021). Changing The HTS Paradigm: AI-Driven Iterative Screening For Hit Finding. SLAS Discov, 26(2):257–262.

Duan, J., Dixon, S. L., Lowrie, J. F., and Sherman, W. (2010). Analysis And Comparison Of 2D Fingerprints: Insights Into Database Screening Performance Using Eight Fingerprint Methods. J. Mol. Graph. Model., 29(2):157–170.

Durrant, J. D. and McCammon, J. A. (2011). NNScore 2.0: A Neural-Network Receptor–Ligand Scoring Function. J. Chem. Inf. Model., 51(11):2897–2903.

Eaton, B. E., Gold, L., and Zichi, D. A. (1995). Let'S Get Specific: The Relationship Between Specificity And Affinity. Chem. Biol., 2(10):633–638.

Ebejer, J.-P., Morris, G. M., and Deane, C. M. (2012). Freely Available Conformer Generation Methods: How Good Are They? J. Chem. Inf. Model., 52(5):1146–1158.

Erlanson, D. A., McDowell, R. S., and O'Brien, T. (2004). Fragment-Based Drug Discovery. J. Med. Chem., 47(14):3463–3482.

Ertl, P. and Schuffenhauer, A. (2009). Estimation Of Synthetic Accessibility Score Of Drug-Like Molecules Based On Molecular Complexity And Fragment Contributions. J. Cheminf., 1(1):1–11.

Fialková, V., Zhao, J., Papadopoulos, K., Engkvist, O., Bjerrum, E. J., Kogej, T., and Patronov, A. (2021). LibINVENT: Reaction-Based Generative Scaffold Decoration For In Silico Library Design. J. Chem. Inf. Model., 62(9):2046–2063.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. (2017). Reverse Curriculum Generation For Reinforcement Learning. In Proc. Conf. Robot Learn., pages 482–495. PMLR.

Follmann, M., Briem, H., Steinmeyer, A., Hillisch, A., Schmitt, M. H., Haning, H., and Meier, H. (2019). An Approach Towards Enhancement Of A Screening Library: The Next Generation Library Initiative (NGLI) At Bayer—against All Odds? Drug Discov. Today, 24(3):668–672.

Food and Drug Administration (2017). 22 case studies where phase 2 and phase 3 trials had divergent results. `https://www.fda.gov/media/102332/download`, Accessed: 20-Aug-2022.

Frazier, P. I. (2018). A Tutorial On Bayesian Optimization. ArXiv Preprint ArXiv:1807.02811.

Friesner, R. A., Banks, J. L., Murphy, R. B., Halgren, T. A., Klicic, J. J., Mainz, D. T., Repasky, M. P., Knoll, E. H., Shelley, M., Perry, J. K., et al. (2004). Glide: A New Approach For Rapid, Accurate Docking And Scoring. 1. Method And Assessment Of Docking Accuracy. J. Med. Chem., 47(7):1739–1749.

Funatsu, K. and Sasaki, S.-I. (1988). Computer-Assisted Organic Synthesis Design And Reaction Prediction System,"AIPHOS". Tetrahedron Comput. Methodol., 1(1):27–37.

Gao, K., Nguyen, D. D., Tu, M., and Wei, G.-W. (2020). Generative Network Complex For The Automated Generation Of Drug-Like Molecules. J. Chem. Inf. Model., 60(12):5682–5698.

Genheden, S., Thakkar, A., Chadimová, V., Reymond, J.-L., Engkvist, O., and Bjerrum, E. (2020). AiZynthFinder: A Fast, Robust And Flexible Open-Source Software For Retrosynthetic Planning. J. Cheminf., 12(1):1–9.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. ACS Cent. Sci., 4(2):268–276.

Goncalves, V., Brannigan, J. A., Whalley, D., Ansell, K. H., Saxty, B., Holder, A. A., Wilkinson, A. J., Tate, E. W., and Leatherbarrow, R. J. (2012). Discovery Of Plasmodium Vivax N-Myristoyltransferase Inhibitors: Screening, Synthesis, And Structural Characterization Of Their Binding Mode. J. Med. Chem., 55(7):3578–3582.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, Proc. Adv. Neural Inf. Process. Syst., volume 27. Curran Associates, Inc.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative Adversarial Networks. Comms. Assoc. Comp. Mach., 63(11):139–144.

Gorgulla, C., Boeszoermenyi, A., Wang, Z.-F., Fischer, P. D., Coote, P. W., Das, K. M. P., Malets, Y. S., Radchenko, D. S., Moroz, Y. S., Scott, D. A., Fackeldey, K., Hoffman, M., Iavniuk, I., Wagner, G., and Arthanari, H. (2020). An Open-Source Drug Discovery Platform Enables Ultra-Large Virtual Screens. Nature, 580(7805):663–668.

Green, H. and Durrant, J. D. (2021). Deepfrag: An Open-Source Browser App For Deep-Learning Lead Optimization. J. Chem. Inf. Model., 61(6):2523–2529.

Green, H., Koes, D. R., and Durrant, J. D. (2021). DeepFrag: A Deep Convolutional Neural Network For Fragment-Based Lead Optimization. Chem. Sci.

Grinberg, M. (2018). Flask Web Development: Developing Web Applications With Python. " O'Reilly Media, Inc.".

Groom, C. R., Bruno, I. J., Lightfoot, M. P., and Ward, S. C. (2016). The Cambridge Structural Database. Acta Crystallogr. B, 72(2):171–179.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). Objective-Reinforced Generative Adversarial Networks (ORGAN) For Sequence Generation Models. ArXiv Preprint ArXiv:1705.10843.

Gumbart, J. C., Roux, B., and Chipot, C. (2013). Standard Binding Free Energies From Computer Simulations: What Is The Best Strategy? J. Chem. Theory Comput., 9(1):794–802.

Guo, J., Fialková, V., Arango, J. D., Margreitter, C., Janet, J. P., Papadopoulos, K., Engkvist, O., and Patronov, A. (2022). Improving de novo molecular design with curriculum learning. Nat. Mach. Intel., 4(6):555–563.

Han, C., Hayashi, H., Rundo, L., Araki, R., Shimoda, W., Muramatsu, S., Furukawa, Y., Mauri, G., and Nakayama, H. (2018). GAN-based Synthetic Brain MR Image Generation. In IEEE Int. Sym. Biomed. Im., pages 734–738. IEEE.

Handoko, S. D., Ouyang, X., Su, C. T. T., Kwoh, C. K., and Ong, Y. S. (2012). QuickVina: Accelerating AutoDock Vina Using Gradient-Based Heuristics For Global Optimization. IEEE/ACM Trans. Comput. Biol. Bioinform., 9(5):1266–1272.

Hansch, C., Maloney, P. P., Fujita, T., and Muir, R. M. (1962). Correlation Of Biological Activity Of Phenoxyacetic Acids With Hammett Substituent Constants And Partition Coefficients. Nature, 194(4824):178–180.

Heemstra, H. E., van Weely, S., Büller, H. A., Leufkens, H. G., and de Vrueh, R. L. (2009). Translation Of Rare Disease Research Into Orphan Drug Development: Disease Matters. Drug Discov. Today, 14(23-24):1166–1173.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep Reinforcement Learning That Matters. In Proc. AAAI Conf. Art. Intel., volume 32.

Hochuli, J., Helbling, A., Skaist, T., Ragoza, M., and Koes, D. R. (2018). Visualizing Convolutional Neural Network Protein-Ligand Scoring. J. Mol. Graph. Model., 84:96–108.

Hopkins, A. L., Groom, C. R., and Alex, A. (2004). Ligand Efficiency: A Useful Metric For Lead Selection. Drug Discov. Today, 9(10):430–431.

Hussain, J. and Rea, C. (2010). Computationally Efficient Algorithm To Identify Matched Molecular Pairs (MMPs) In Large Data Sets. J. Chem. Inf. Model., 50(3):339–348.

Imrie, F., Bradley, A. R., van der Schaar, M., and Deane, C. M. (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):2319–2330.

Imrie, F., Bradley, A. R., van der Schaar, M., and Deane, C. M. (2020). Deep Generative Models For 3d Linker Design. J. Chem. Inf. Model., 60(4):1983–1995.

Imrie, F., Hadfield, T. E., Bradley, A. R., and Deane, C. M. (2021). Deep Generative Design With 3D Pharmacophoric Constraints. Chem. Sci., 12:14577–14589.

Jeen, S. R., Abate, A., and Cullen, J. M. (2022). Low Emission Building Control With Zero-Shot Reinforcement Learning. ArXiv Preprint ArXiv:2208.06385.

Jeon, W. and Kim, D. (2020). Autonomous Molecule Generation Using Reinforcement Learning And Docking To Develop Potential Novel Inhibitors. Sci. Rep., 10(1):1–11.

Jiménez, J., Skalic, M., Martinez-Rosell, G., and De Fabritiis, G. (2018). K Deep: Protein–Ligand Absolute Binding Affinity Prediction Via 3d-Convolutional Neural Networks. J. Chem. Inf. Model., 58(2):287–296.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction Tree Variational Autoencoder For Molecular Graph Generation. In Int. Conf. Mach. Learn., pages 2323–2332. PMLR.

Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2002). Molecular Biology Of The Cell. Garland Science, 4.

Jones, S. and Thornton, J. M. (1996). Principles Of Protein-Protein Interactions. Proc. Natl. Acad. Sci. U. S. A., 93(1):13–20.

Joyce, J. M. (2011). Kullback-Leibler Divergence. In International Encyclopedia Of Statistical Science, pages 720–722. Springer.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly Accurate Protein Structure Prediction With AlphaFold. Nature, 596(7873):583–589.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular Graph Convolutions: Moving Beyond Fingerprints. J. Comput. Aided Mol. Des., 30(8):595–608.

Kieninger, E., Singer, F., Tapparel, C., Alves, M. P., Latzin, P., Tan, H.-L., Bossley, C., Casaulta, C., Bush, A., Davies, J. C., Kaiser, L., and Regamey, N. (2013). High Rhinovirus Burden In Lower Airways Of Children With Cystic Fibrosis. Chest, 143(3):782–790.

Kim, M., Park, K., Kim, W., Jung, S., and Cho, A. E. (2020). Target-Specific Drug Design Method Combining Deep Learning And Water Pharmacophore. J. Chem. Inf. Model.

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., et al. (2021). PubChem In 2021: New Data Content And Improved Web Interfaces. Nucleic Acids Res., 49(D1):D1388–D1395.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. ArXiv Preprint ArXiv:1312.6114.

Kirsch, D. R. (2020). Therapeutic Drug Development And Human Clinical Trials. In Biotechnology Entrepreneurship, pages 339–358. Elsevier.

Knott, G. J. and Doudna, J. A. (2018). CRISPR-Cas Guides The Future of Genetic Engineering. Science, 361(6405):866–869.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement Learning In Robotics: A Survey. Int. J. Rob. Res., 32(11):1238–1274.

Koes, D. R., Baumgartner, M. P., and Camacho, C. J. (2013). Lessons Learned in Empirical Scoring with smina From the CSAR 2011 Benchmarking Exercise. J. Chem. Inf. Model., 53(8):1893–1904.

Kramer, J. A., Sagartz, J. E., and Morris, D. L. (2007). The Application Of Discovery Toxicology And Pathology Towards The Design Of Safer Pharmaceutical Lead Candidates. Nat. Rev. Drug. Discov., 6(8):636–649.

Kremer, M. (2002). Pharmaceuticals And The Developing World. J. Econ. Perspect., 16(4):67–90.

Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. (2020). Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation. Mach. Learn. Sci Tech., 1(4):045024.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet Classification With Deep Convolutional Neural Networks. Proc. Adv. Neural Inf. Process. Syst., 25.

Landrum, G. (2006). RDKit: Open-Source Cheminformatics.

Landrum, G. A., Penzotti, J. E., and Putta, S. (2006). Feature-Map Vectors: A New Class Of Informative Descriptors For Computational Drug Discovery. J. Comput. Aided Mol. Des., 20(12):751–762.

Langdon, S. R., Ertl, P., and Brown, N. (2010). Bioisosteric Replacement And Scaffold Hopping In Lead Generation And Optimization. Mol. Inform., 29(5):366–385.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied To Document Recognition. Proc. IEEE, 86(11):2278–2324.

Li, Y., Hu, J., Wang, Y., Zhou, J., Zhang, L., and Liu, Z. (2019). Deepscaffold: A Comprehensive Tool For Scaffold-Based De Novo Drug Discovery Using Deep Learning. J. Chem. Inf. Model., 60(1):77–91.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated Graph Sequence Neural Networks. ArXiv Preprint ArXiv:1511.05493.

Lim, J., Hwang, S.-Y., Moon, S., Kim, S., and Kim, W. Y. (2020). Scaffold-Based Molecular Design With A Graph Generative Model. Chem. Sci., 11(4):1153–1164.

Lipinski, C. A., Lombardo, F., Dominy, B. W., and Feeney, P. J. (1997). Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings. Adv. Dr. Deliv. Rev., 23(1-3):3–25.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. L. (2018). Constrained Graph Variational Autoencoders For Molecule Design. ArXiv Preprint ArXiv:1805.09076.

Liu, Z., Su, M., Han, L., Liu, J., Yang, Q., Li, Y., and Wang, R. (2017). Forging The Basis For Developing Protein–Ligand Interaction Scoring Functions. Acc. Chem. Res., 50(2):302–309.

Luo, S., Kasaei, H., and Schomaker, L. (2020). Accelerating Reinforcement Learning For Reaching Using Continuous Curriculum Learning. In Proc. Int. Jt. Conf. Neural Netw., pages 1–8. IEEE.

Lyu, J., Wang, S., Balius, T. E., Singh, I., Levit, A., Moroz, Y. S., O'Meara, M. J., Che, T., Algaa, E., Tolmachova, K., Shoichet, B. K., Roth, B. L., and Irwin, J. J. (2019). Ultra-Large Library Docking For Discovering New Chemotypes. Nature, 566(7743):224–229.

Malekzadeh, M., Hajibabaee, P., Heidari, M., Zad, S., Uzuner, O., and Jones, J. H. (2021). Review Of Graph Neural Network In Text Classification. In Proc. Ann. Ubiq. Comp. Elec. Mob. Comm. Conf., pages 0084–0091. IEEE.

Malhotra, S. and Karanicolas, J. (2017). When Does Chemical Elaboration Induce A Ligand To Change Its Binding Mode? J. Med. Chem., 60(1):128–145.

Marc, D., Drugeon, G., Haenni, A.-L., Girard, M., and van der Werf, S. (1989). Role Of Myristoylation Of Poliovirus Capsid Protein VP4 As Determined By Site-Directed Mutagenesis Of Its N-Terminal Sequence. EMBO J., 8(9):2661–2668.

Marc, D., Masson, G., Girard, M., and van der Werf, S. (1990). Lack Of Myristoylation Of Poliovirus Capsid Polypeptide VP0 Prevents The Formation Of Virions Or Results In The Assembly Of Noninfectious Virus Particles. J. Virol., 64(9):4099–4107.

Masuda, T., Ragoza, M., and Koes, D. R. (2020). Generating 3D Molecular Structures Conditional On A Receptor Binding Site With Deep Generative Models. ArXiv Preprint ArXiv:2010.14442.

Matveieva, M. and Polishchuk, P. (2021). Benchmarks For Interpretation Of QSAR Models. J. Cheminf., 13(1):1–20.

McCloskey, K., Taly, A., Monti, F., Brenner, M. P., and Colwell, L. J. (2019). Using Attribution To Decode Binding Mechanism In Neural Network Models For Chemistry. P. Natl. Acad. Sci., 116(24):11624–11629.

Meng, Q., Catchpoole, D., Skillicom, D., and Kennedy, P. J. (2017). Relational Autoencoder For Feature Extraction. In Proc. Int. Jt. Conf. Neural Netw., pages 364–371. IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari With Deep Reinforcement Learning. ArXiv Preprint ArXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-Level Control Through Deep Reinforcement Learning. Nature, 518(7540):529–533.

Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., and Olson, A. J. (1998). Automated Docking Using A Lamarckian Genetic Algorithm And An Empirical Binding Free Energy Function. J. Comput. Chem., 19(14):1639–1662.

Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., and Olson, A. J. (2009). AutoDock4 And AutoDockTools4: Automated Docking With Selective Receptor Flexibility. Journal Of Computational Chemistry, 30(16):2785–2791.

Mortier, J., Dhakal, P., and Volkamer, A. (2018). Truly Target-Focused Pharmacophore Modeling: A Novel Tool For Mapping Intermolecular Surfaces. Molecules, 23(8):1959.

Mousnier, A., Bell, A. S., Swieboda, D. P., Morales-Sanfrutos, J., Pérez-Dorado, I., Brannigan, J. A., Newman, J., Ritzefeld, M., Hutton, J. A., Guedán, A., Asfor, A. S., Robinson, S. W., Hopkins-Navratilova, I., Wilkinson, A. J., Johnston, S. L., Leatherbarrow, R. J., Tuthill, T. J., Solari, R., and Tate, E. W. (2018). Fragment-Derived Inhibitors Of Human N-Myristoyltransferase Block Capsid Assembly And Replication Of The Common Cold Virus. Nat. Chem., 10(6):599–606.

Muegge, I. and Mukherjee, P. (2016). An Overview Of Molecular Fingerprint Similarity Search In Virtual Screening. Expert Opin. Drug Discov., 11(2):137–148.

Mullard, A. (2022). 2021 FDA Approvals. Nat. Rev. Drug. Discov.

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT press.

Mysinger, M. M., Carchia, M., Irwin, J. J., and Shoichet, B. K. (2012). Directory Of Useful Decoys, Enhanced (DUD-E): Better Ligands And Decoys For Better Benchmarking. J. Med. Chem., 55(14):6582–6594.

Nie, W. and Patel, A. B. (2020). Towards a Better Understanding and Regularization of GAN Training Dynamics. In Uncertainty in Artificial Intelligence, pages 281–291. PMLR.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (2017). Molecular De-Novo Design Through Deep Reinforcement Learning. J. Cheminf., 9(1):1–14.

Ou-Yang, S.-s., Lu, J.-y., Kong, X.-q., Liang, Z.-j., Luo, C., and Jiang, H. (2012). Computational Drug Discovery. Acta Pharmacol. Sin., 33(9):1131–1140.

O'Connell, J., Porter, J., Kroeplien, B., Norman, T., Rapecki, S., Davis, R., McMillan, D., Arakaki, T., Burgin, A., Fox Iii, D., et al. (2019). Small Molecules That Inhibit TNF Signalling By Stabilising An Asymmetric Form Of The Trimer. Nat. Comms., 10(1):1–12.

Parasuraman, S. (2011). Toxicological Screening. J. Pharmacol. Pharmacother., 2(2):74.

Pellegrini, M., Haynor, D., and Johnson, J. M. (2004). Protein Interaction Networks. Expert Rev. Proteomics, 1(2):239–249.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., and Ma, J. (2022). Pocket2Mol: Efficient Molecular Sampling Based On 3D Protein Pockets. ArXiv Preprint ArXiv:2205.07249.

Pereira, J. C., Caffarena, E. R., and Dos Santos, C. N. (2016). Boosting Docking-Based Virtual Screening With Deep Learning. J. Chem. Inf. Model., 56(12):2495–2506.

Plowright, A. T., Johnstone, C., Kihlberg, J., Pettersson, J., Robb, G., and Thompson, R. A. (2012). Hypothesis Driven Drug Design: Improving Quality And Effectiveness Of The Design-Make-Test-Analyse Cycle. Drug Discov. Today, 17(1-2):56–62.

Pocock, S. J. (1976). The Combination Of Randomized And Historical Controls In Clinical Trials. J. Chronic Dis., 29(3):175–188.

Polishchuk, P. (2020). CReM: Chemically Reasonable Mutations Framework For Structure Generation. J. Cheminf., 12(1):1–18.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. (2020). Molecular Sets (MOSES): A Benchmarking Platform For Molecular Generation Models. Front. Pharmacol., 11:565644.

Putta, S., Landrum, G. A., and Penzotti, J. E. (2005). Conformation Mining: An Algorithm For Finding Biologically Relevant Conformations. J. Med. Chem., 48(9):3313–3318.

Radoux, C. J., Olsson, T. S., Pitt, W. R., Groom, C. R., and Blundell, T. L. (2016). Identifying Interactions That Determine Fragment Binding At Protein Hotspots. J. Med. Chem., 59(9):4314–4325.

Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J., and Koes, D. R. (2017). Protein–Ligand Scoring With Convolutional Neural Networks. J. Chem. Inf. Model., 57(4):942–957.

Ragoza, M. T., Masuda, T., and Koes, D. R. (2022). Generating 3D Molecules Conditional On Receptor Binding Sites With Deep Generative Models. Chem. Sci.

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., and Pande, V. (2015). Massively Multitask Networks For Drug Discovery. ArXiv Preprint ArXiv:1502.02072.

Ranjan, A., Bolkart, T., Sanyal, S., and Black, M. J. (2018). Generating 3D Faces Using Convolutional Mesh Autoencoders. In Proc. Comput. Vis. ECCV, pages 704–720.

Rego, N. and Koes, D. (2015). 3Dmol. Js: Molecular Visualization With WebGL. Bioinformatics, 31(8):1322–1324.

Reymond, J.-L. (2015). The Chemical Space Project. Acc. Chem. Res., 48(3):722–730.

Reynolds, R. C., Ananthan, S., Faaleolea, E., Hobrath, J. V., Kwong, C. D., Maddox, C., Rasmussen, L., Sosa, M. I., Thammasuvimol, E., White, E. L., et al. (2012). High Throughput Screening Of A Library Based On Kinase Inhibitor Scaffolds Against Mycobacterium Tuberculosis H37Rv. Tuberculosis, 92(1):72–83.

Ritchie, A. I., Farne, H. A., Singanayagam, A., Jackson, D. J., Mallia, P., and Johnston, S. L. (2015). Pathogenesis Of Viral Infection In Exacerbations Of Airway Disease. Ann. Amer. Thorac. Soc., 12(Supplement 2):S115–S132.

Rogers, D. and Hahn, M. (2010). Extended-Connectivity Fingerprints. J. Chem. Inf. Model., 50(5):742–754.

Rohrer, S. G. and Baumann, K. (2009). Maximum Unbiased Validation (MUV) Data Sets For Virtual Screening Based On PubChem Bioactivity Data. J. Chem. Inf. Model., 49(2):169–184.

Sanchez-Garcia, R., Havasi, D., Takács, G., Robinson, M. C., von Delft, F., and Deane, C. M. (2022). CoPriNet: Deep Learning Compound Price Prediction for Use In De Novo Molecule Generation and Prioritization.

Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L., and Wiltschko, A. (2020). Evaluating Attribution for Graph Neural Networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, Proc. Adv. Neural Inf. Process. Syst., volume 33, pages 5898–5910. Curran Associates, Inc.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) Equivariant Graph Neural Networks. In Proc. Int. Conf. Mach. Learn., pages 9323–9332. PMLR.

Scantlebury, J., Brown, N., Von Delft, F., and Deane, C. M. (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):3722–3730.

Schaller, D., Šribar, D., Noonan, T., Deng, L., Nguyen, T. N., Pach, S., Machalz, D., Bermudez, M., and Wolber, G. (2020). Next Generation 3D Pharmacophore Modeling. Wiley Interdiscip. Rev. Comput. Mol. Sci., 10(4):e1468.

Schneider, G., Lee, M.-L., Stahl, M., and Schneider, P. (2000). De Novo Design Of Molecular Architectures By Evolutionary Assembly Of Drug-Derived Building Blocks. J. Comput. Aided Mol. Des., 14(5):487–494.

Schrödinger, LLC (2015). The PyMOL molecular graphics system, version 1.8.

Schuetz, D. A., Seidel, T., Garon, A., Martini, R., Kürbel, M., Ecker, G. F., and Langer, T. (2018). GRAIL: Grids Of Pharmacophore Interaction Fields. J. Chem. Theory Comput., 14(9):4958–4970.

Schuffenhauer, A., Ruedisser, S., Marzinzik, A., Jahnke, W., Selzer, P., and Jacoby, E. (2005). Library Design For Fragment Based Screening. Curr. Top. Med. Chem., 5(8):751–762.

Scott, D. E., Coyne, A. G., Hudson, S. A., and Abell, C. (2012). Fragment-Based Approaches In Drug Discovery And Chemical Biology. Biochemistry, 51(25):4990–5003.

Segler, M. H., Preuss, M., and Waller, M. P. (2018). Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. Nature, 555(7698):604–610.

Shan, J., Pan, X., Wang, X., Xiao, X., and Ji, C. (2020). FragRep: A Web Server For Structure-Based Drug Design By Fragment Replacement. J. Chem. Inf. Model., 60(12):5900–5906.

Shu, L., Blencowe, M., and Yang, X. (2018). Translating GWAS Findings to Novel Therapeutic Targets for Coronary Artery Disease. Front. Cardio. Med., 5:56.

Sieg, J., Flachsenberg, F., and Rarey, M. (2019). In Need Of Bias Control: Evaluating Chemical Data For Machine Learning In Structure-Based Virtual Screening. J. Chem. Inf. Model., 59(3):947–961.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering The Game Of Go Without Human Knowledge. Nature, 550(7676):354–359.

Skalic, M., Jiménez, J., Sabbadin, D., and De Fabritiis, G. (2019a). Shape-Based Generative Modeling For De Novo Drug Design. J. Chem. Inf. Model., 59(3):1205–1214.

Skalic, M., Sabbadin, D., Sattarov, B., Sciabola, S., and De Fabritiis, G. (2019b). From Target To Drug: Generative Modeling For The Multimodal Structure-Based Ligand Design. Mol. Pharm., 16(10):4282–4291.

Smith, R. D., Clark, J. J., Ahmed, A., Orban, Z. J., Dunbar Jr, J. B., and Carlson, H. A. (2019). Updates To Binding MOAD (Mother Of All Databases): Polypharmacology Tools And Their Utility In Drug Repurposing. J. Mol. Biol., 431(13):2423–2433.

Stanton, J. M. (2001). Galton, Pearson, And The Peas: A Brief History Of Linear Regression For Statistics Instructors. J. Stat. Educ., 9(3).

Stärk, H., Ganea, O., Pattanaik, L., Barzilay, R., and Jaakkola, T. (2022). Equibind: Geometric Deep Learning For Drug Binding Structure Prediction. In Proc. Int. Conf. Mach. Learn., pages 20503–20521. PMLR.

Stein, R. M., Kang, H. J., McCorvy, J. D., Glatfelter, G. C., Jones, A. J., Che, T., Slocum, S., Huang, X.-P., Savych, O., Moroz, Y. S., Stauch, B., Johansson, L. C., Cherezov, V., Kenakin, T., Irwin, J. J., Shoichet, B. K., Roth, B. L., and Dubocovich, M. L. (2020). Virtual Discovery Of Melatonin Receptor Ligands To Modulate Circadian Rhythms. Nature, 579(7800):609–614.

Sterling, T. and Irwin, J. J. (2015). ZINC 15–Ligand Discovery For Everyone. J. Chem. Inf. Model., 55(11):2324–2337.

Su, M., Yang, Q., Du, Y., Feng, G., Liu, Z., Li, Y., and Wang, R. (2018). Comparative Assessment Of Scoring Functions: The CASF-2016 Update. J. Chem. Inf. Model., 59(2):895–913.

Sundar, V. and Colwell, L. (2020). Attribution Methods Reveal Flaws In Fingerprint-Based Virtual Screening. ArXiv Preprint ArXiv:2007.01436.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic Attribution For Deep Networks. In Proc. Int. Conf. Mach. Learn., pages 3319–3328. PMLR.

Sunseri, J. and Koes, D. R. (2020). Libmolgrid: Graphics Processing Unit Accelerated Molecular Gridding For Deep Learning Applications. J. Chem. Inf. Model., 60(3):1079–1084.

Sutton, R. S. and Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT press.

Tran-Nguyen, V.-K., Jacquemard, C., and Rognan, D. (2020). LIT-PCBA: An Unbiased Data Set For Machine Learning And Virtual Screening. J. Chem. Inf. Model., 60(9):4263–4273.

Trott, O. and Olson, A. J. (2010). AutoDock Vina: Improving The Speed And Accuracy Of Docking With A New Scoring Function, Efficient Optimization, And Multithreading. Journal Of Computational Chemistry, 31(2):455–461.

Umscheid, C. A., Margolis, D. J., and Grossman, C. E. (2011). Key Concepts Of Clinical Trials: A Narrative Review. Postgrad Med, 123(5):194–204.

Verdonk, M. L., Cole, J. C., Hartshorn, M. J., Murray, C. W., and Taylor, R. D. (2003). Improved Protein–Ligand Docking Using GOLD. Proteins, 52(4):609–623.

Verdonk, M. L., Cole, J. C., and Taylor, R. (1999). SuperStar: A Knowledge-Based Approach For Identifying Interaction Sites In Proteins. J. Mol. Biol., 289(4):1093–1108.

Villar, H. O. and Hansen, M. R. (2009). Design Of Chemical Libraries For Screening. Expert Opin. Drug Discov., 4(12):1215–1220.

Wallace, A. C., Laskowski, R. A., and Thornton, J. M. (1995). LIGPLOT: A Program To Generate Schematic Diagrams Of Protein-Ligand Interactions. Protein Eng. Des. Sel., 8(2):127–134.

Wan, Z., Zhang, Y., and He, H. (2017). Variational Autoencoder Based Synthetic Data Generation For Imbalanced Learning. In Proc. IEEE Symp. Ser. Comput. Intell., pages 1–7. IEEE.

Weininger, D. (1988). SMILES, A Chemical Language And Information System. 1. Introduction To Methodology And Encoding Rules. J. Chem. Inf. Comp. Sci., 28(1):31–36.

Weng, L. (2018a). A (long) peek into reinforcement learning. lilianweng.github.io. Accessed 27SEP2022.

Weng, L. (2018b). Policy gradient algorithms. lilianweng.github.io. Accessed 27SEP2022.

Whitford, D. (2013). Proteins: Structure And Function. John Wiley & Sons.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms For Connectionist Reinforcement Learning. Mach. Learn., 8(3):229–256.

Winter, R., Montanari, F., Steffen, A., Briem, H., Noé, F., and Clevert, D.-A. (2019). Efficient Multi-Objective Molecular Optimization In A Continuous Latent Space. Chem. Sci., 10(34):8016–8024.

Wójcikowski, M., Ballester, P. J., and Siedlecki, P. (2017). Performance Of Machine-Learning Scoring Functions In Structure-Based Virtual Screening. Sci. Rep., 7(1):1–10.

Wójcikowski, M., Kukiełka, M., Stepniewska-Dziubinska, M. M., and Siedlecki, P. (2019). Development Of A Protein–Ligand Extended Connectivity (PLEC) Fingerprint And Its Application For Binding Affinity Predictions. Bioinformatics, 35(8):1334–1341.

Wójcikowski, M., Zielenkiewicz, P., and Siedlecki, P. (2015). Open Drug Discovery Toolkit (ODDT): A New Open-Source Player In The Drug Discovery Field. J. Cheminf., 7(1):1–6.

Wong, C. H., Siah, K. W., and Lo, A. W. (2019). Estimation Of Clinical Trial Success Rates And Related Parameters. Biostatistics, 20(2):273–286.

Wouters, O. J., McKee, M., and Luyten, J. (2020). Estimated Research And Development Investment Needed To Bring A New Medicine To Market, 2009-2018. Jama, 323(9):844–853.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). MoleculeNet: A Benchmark For Molecular Machine Learning. Chem. Sci., 9(2):513–530.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph Neural Networks: A Review Of Methods And Applications. AI Open, 1:57–81.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. (2019). Optimization Of Molecules Via Deep Reinforcement Learning. Sci. Rep., 9(1):1–10.