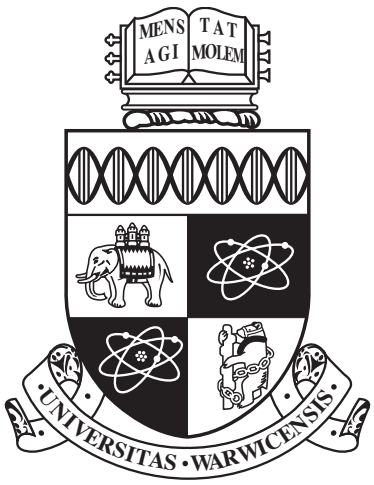

A Materials Science-inspired Paradigm to Predict the Physical Stability of Amorphous Drugs

by

Trent Barnard

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy in Mathematics of Systems

University of Warwick, Department of Mathematics

May 2023

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments

I would like to extend my heartfelt gratitude to my supervisor Gabriele for his invaluable support, guidance, and mentorship throughout my PhD journey. Gabriele has been an exceptional supervisor, who has gone above and beyond in helping me achieve my academic and personal goals.

From day one, he has been an unwavering source of encouragement, inspiration, and motivation. His extensive knowledge, expertise, and passion for research have been contagious, and he has continuously challenged me to push beyond my limits and strive for excellence in my work.

Gabriele's mentorship has been instrumental in shaping my academic career. His insightful feedback, constructive criticism, and attention to detail have been crucial in improving the quality of my research, and his guidance has helped me navigate through the various challenges that arise during a PhD program.

Moreover, Gabriele's unwavering availability, patience, and understanding have been a source of comfort during times of stress and uncertainty. He has been a supportive and compassionate mentor, who has consistently demonstrated his commitment to my personal and professional growth.

I feel incredibly fortunate to have had Gabriele as my supervisor, and I am grateful for the opportunities he has provided me with to learn and grow as a researcher. I am confident that the skills, knowledge, and experiences I have gained under his guidance will serve me well in my future endeavors.

I would also like to give a special thanks to my friends Ella, Lal, and my boy Oscar for being wholesome and keeping me sane throughout my corrections.

Declarations

The work presented here is my own, except where stated otherwise. This thesis has been composed by myself and has not been submitted for any other degree or professional qualification.

Chapter 4 has been published as

1. González-Jiménez, Mario, et al. "Understanding the emergence of the boson peak in molecular glasses." Nature Communications 14.1 (2023): 215.

Chapter 5 has been published as

1. Barnard, Trent, et al. "Leveraging genetic algorithms to maximise the predictive capabilities of the SOAP descriptor." Molecular Systems Design & Engineering (2023).

Parts of Chapter 6 have been published as

1. Barnard, Trent, et al. "Less may be more: an informed reflection on molecular descriptors for drug design and discovery." Molecular Systems Design & Engineering 5.1 (2020): 317-329.

2. Barnard, Trent, et al. "Combining Machine Learning and Molecular Simulations to Predict the Stability of Amorphous Drugs." Submitted to The Journal of Chemical Physics

Abstract

Amorphous drugs have gained attention as a promising alternative to crystalline formulations due to their ability to enhance solubility. However, ensuring the physical stability of amorphous drugs is critical for successful commercialisation. Unfortunately, predicting the timescale of crystallisation for amorphous drugs is challenging. To address this problem, machine learning models can be developed to predict the physical stability of amorphous drugs.

This study presents methodological advancements in using molecular dynamics simulations to develop machine learning models for predicting the crystallisation tendency of amorphous drugs. The study develops and computes solid-state descriptors that capture the dynamic properties of the amorphous phase and complements the traditional single-molecule descriptors commonly used in quantitative structure-activity relationship (QSAR) models. We have also specifically focused on a particular molecular glass to gain insights into the dynamical properties of materials similar to amorphous drugs.

The results show that the use of molecular simulations as a tool to enrich the traditional machine learning paradigm for drug design and discovery can lead to high accuracy in predicting the physical stability of amorphous drugs. The net result of this work is an improvement over the state-of-the-art in predicting the crystallisation tendency of amorphous drugs.

Chapter 1

Introduction

1.1 Amorphous drugs

Most modern drugs are packaged as crystalline formulations[1]. The molecules within these formulations have long range order and a relatively high density. The crystalline structure has significant effects on the physical properties of the drug, such as solubility, stability and bioavailability [2]. The molecules in a crystalline drug form a lattice-like structure which typically exhibits strong intermolecular attraction. This leads to poor solubility and bioavailability, two of the most important properties of pharmaceutical drugs. Studies have shown that over 40% of drugs that have been approved and almost 90% of molecules used for pharmaceutical drugs are categorised as poorly water soluble [3, 4].

Amorphous formulations of drugs represent a viable, novel way forward to improve the solubility of modern drug formulations. Amorphous drugs have been gaining popularity and a resurgence of interest in them has arisen as the number of poorly soluble molecules with therapeutic properties continues to grow [5]. Amorphous drugs lack a defined crystal structure. They instead exist as a disordered, non-crystalline solid. Because of this, they present several benefits in comparison to crystalline drugs:

1. **Improved solubility**: Amorphous drugs typically have higher solubility than their crystalline counterparts [6, 7, 8]. This is because the lack of a crystal lattice structure allows for the drug molecules to more readily interact with solvents. Recent studies [8] show that amorphous materials may be up to 1,600 times more soluble than crystalline materials.

2. **Enhanced bioavailability**: Because of their improved solubility, amorphous drugs may be more readily absorbed by the body, leading to higher bioavailability and po-

tentially more effective treatment.

3. **Faster onset of action**: Due to their increased solubility and bioavailability, amorphous drugs may act more quickly than crystalline drugs [9, 10]. It is, however, important to note that the actual onset of action will depend on several factors, such as the specific drug and the patients individual characteristics.

4. **Greater formulation flexibility**: Studies suggest [11, 8] that amorphous drugs can be more easily incorporated into different formulations, such as tablets, capsules, or suspensions, as they do not require a specific crystal structure to maintain their properties. This is because the amorphous form of a drug can be more readily dissolved in solvents, allowing for easier processing and formulation. Additionally, the lack of a specific crystal structure can also allow for greater flexibility in designing drug delivery systems with specific properties, such as sustained release or targeted delivery[8].

While amorphous drugs do have several advantages, they also have disadvantages that can make their development and formulation challenging.

1. **Lack of stability**: Amorphous drugs have a tendency to revert back to their crystalline form. This process is known as crystallisation and poses a challenging problem for amorphous pharmaceuticals. The main reasons for this are 1) Amorphous drugs are generally much less thermodynamically stable than crystalline drugs. Since the amorphous form has a greater free energy than the crystalline form, it tends to transition towards its more stable state over time [12]. 2) Since amorphous drugs lack a well defined structure, the molecules are free to rearrange themselves and move over time. This is known as structural relaxation [13] and can lead to changes in physical properties, such as its solubility and propensity for crystallisation. This crystallisation process is discussed in detail in section 1.1.1.

2. **Manufacturing challenges:** The production of amorphous drugs can be more challenging than crystalline drugs, requiring specialized manufacturing techniques such as spray drying, freeze-drying, and milling. This can increase the manufacturing cost and may limit the availability of the drug. We discuss some of the ways amorphous drugs are produced in section 1.1.2.

3. **Formulation challenges:** The formulation of amorphous drugs can be more challenging than crystalline drugs, as their properties can change depending on the formulation conditions. This can make it more difficult to achieve consistent dosing and can require additional testing and characterization to ensure product quality [14].

The main focus of this work is to try and better understand the first disadvantage i.e. the lack of stability.

1.1.1 Crystallisation

Crystallisation, in the context of amorphous drugs, refers to the process by which the amorphous form of a drug converts to a crystalline form [9]. Crystallisation can occur spontaneously, or it can be induced by various external factors such as changes in temperature, humidity, or mechanical stress [15, 16]. The factors that influence the crystallisation process are multifaceted, and the mechanisms involved are still not completely understood. However, several studies have reported on the impact of different parameters on the crystallization process, such as the drug's molecular structure, the degree of supersaturation, and the presence of impurities or additives [9]. The control and prevention of crystallisation in amorphous drug formulations are critical factors that need to be considered during the drug development process. Understanding the underlying mechanisms of crystallisation and identifying strategies to minimise its occurrence can help improve the stability and efficacy of amorphous drug formulations.

Due to the fact that the amorphous phase is metastable with respect to the crystalline phase, amorphous drugs are thermodynamically unstable. This means that they have a tendency to crystallise over time to get to a lower energy state. This crystallisation will mitigate a lot of the benefits of amorphous drugs. It can also lead to changes in chemical properties as well as decreased solubility and bioavailability.

It is not hard to imagine a case where a drug that has recrystallised can be of danger to a patient. If the patient is not aware that the drug has recrystallised and they take their regular dose, because of the reduced bioavailability, the drug will not be as effective. This could have dangerous consequences for the patient.

We can attempt to delay the crystallisation of the drug by adopting different strategies. Most amorphous formulations on the market are not pure amorphous solids, but instead amorphous solid dispersions (ASDs) [17]. An ASD is a technique used to increase the solubility of a drug. An ASD is created by distributing the amorphous substance within a polymer matrix. This polymer matrix helps to stabilise the amorphous substance and prevent it from crystallising. ASDs can be created using various different techniques including spray drying and co-precipitation [18, 19]. An example of an ASD drugs currently on the market is Tricor which is used to lower cholesterol. This contains the active ingredient fenofibrate which is one of the molecules in our dataset, and it is formulated as an ASD with the polymer polyvinylpyrrolidone [20]. Another drug in our dataset is celecoxib which is an

anti-inflamatory drug that is prepared as an ASD by mixing it with a stabilising polymer, commonly hydroxypropyl methylcellulose [21]. By formulating an ASD, the stability, solubility and bioavailability is improved [22, 23]. However, care must be taken when selecting the polymer and the method of manufacturing the drug as they can have a significant impact on the stability and efficacy [24]. Further care must be taken when storing the drug, since the temperature and humidity can have an effect on the stability of the ASD [25].

All amoprhous solids will eventually crystallise. As such, it is important that we have some estimate of how long this will take. In some cases it may only be a few minutes, whereas in other cases it could take several years for the amorphous solid to return to its crystalline form. Gauging how long this will take is not a particularly easy task. A variety of methods exist that allow us to determine which phase a drug is in, and if it has crystallised. Differential scanning calorimetry (DSC) [26] allows us to observe when a phase change has occured, and diffraction techniques such as x-ray diffraction (XRD) are the tools of the trade for looking into the structure of molecular solids. More information on these procedures is given in section 1.1.2. These are experiments that would have to take place in a lab, so for pharmaceuticals on the shelf it is obviously not practical to regularly test if the drug has recrystallised. This is why we need a method to reliably predict the stability of amorphous pharmaceuticals, so we can have some idea of their shelf life. When we speak of the stability of a given drug in this work, we refer to its propensity to crystallise within a certain timescale.

1.1.2 How amorphous drugs are made

There exist several experimental approaches to craft amorphous drugs. In this section we describe some of the more commonly used techniques, and outline how the experimental data we use was generated. There are a number of ways in which an amorphous drug can be prepared:

1. **Melt-quenching:** This involves heating the drug to a temperature higher than its melting point (T_m) so that it forms a liquid. This liquid is then rapidly cooled by immersing it in liquid nitrogen, placing it on a cold surface, or blowing cold gas over it. If the drug is cooled rapidly enough, it is prevented from reorganising itself into its crystalline form, and an amorphous solid is formed. To create an ASD the exact same process is used except the amorphous drug is melted together with a polymer. This technique is simple, effective, and can be used with a wide range of drugs and polymers. However, the high temperatures used can lead to drug degradation [27].

2. **Spray drying:** This is a process used to create ASDs. Spray drying involves dissolving the amorphous drug and the polymer in a solvent. This solution is then fed into a spray drying chamber and atomised using a spray nozzle. The atomised droplets are dried by a stream of hot gas intended to evaporate the solvent. This leaves behind a fine powder that contains the ASD. This powder has a high surface area which can enhance the solubility and bioavailability [18], however there may be some solvent residue left in the final product which could be a concern.

3. **Freeze drying:** Another method to form an ASD is again by dissolving the drug and polymer is a solvent and then freezing. This frozen mixture is placed in a vacuum and the solvent is removed using a technique called sublimation. This is where a solid is converted directly into a gas without transitioning to the liquid phase. This again results in a fine powder containing the ASD and has similar strengths and weaknesses to spray drying.

The experimental data generated for use in this work [28, 29] was created using melt-quenching and here we describe the exact methodology used. To begin, the molecules were heated up at 10° C/min until the temperature is 2° C/min above T_m. To ensure the system is fully melted, the temperature is now kept constant for 2 minutes. When the system is in this melted state it exhibits no long range order but it will crystallise when it is left alone. Note that there are exceptions to this rule, we look at one such example in chapter 4.

To prevent crystallisation from happening the system is rapidly cooled at a rate of −20° C/min to a very low temperature of −70° C. Since this cooling is done very quickly, a glass is created. By supercooling the melt in this way, the glass is in a metastable state that exhibits the long range disorder of a liquid, and thus the increased solubility that we seek, while also not being thermodynamically stable.

The importance of the rate at which the system is cooled can not be understated. Since the transition to a glass is not a thermodynamic phase transition (T_g), the cooling speed makes a difference to the glass transition temperature, as well as other physical properties. A faster cooling rate will typically produce a glass with a greater volume and lower density than a glass produced using a slower cooling rate. The rate of cooling also affects the glass transition temperature whereby a slower cooling rate typically leads to a lower T_g. This concept is illustrated in figure 1.1. Another important aspect of the cooling rate is that if it is too slow, the melt will crystallise instead of forming a glass. This is one of the benefits of using computational molecular dynamics (MD) simulations as opposed to physical experiments. When we use MD we are able to cool the melt at a rate that is orders of magnitude faster than what we would be able to realistically do in a lab. This allows us to observe a

simulation of a glass when we may not be able to actually create that glass in the real world, although this simulation may be unphysical.

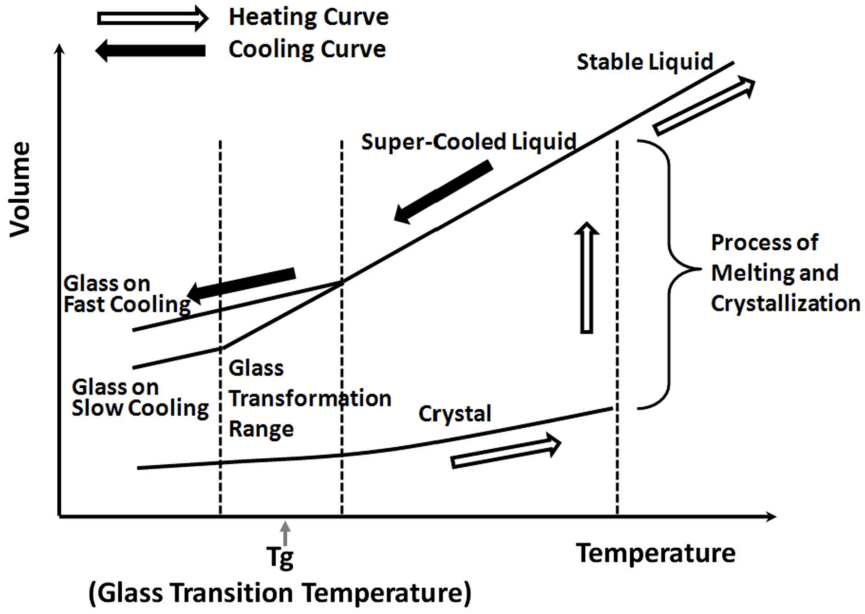

Figure 1.1: The effects on different cooling rates on a typical glass [30]

In the publications that we used to gather our data, the authors measure physical properties of a selection of drug-like molecules. These properties are described in section 1.1.4. Although all that is needed to create an amorphous solid is the melting and supercooling that has been described, for the purpose of determining these particular properties, an additional round of heating is used. After the melt has been supercooled, the system is heated up using heat-flux differential scanning calorimetry (DSC). DSC measures the heat flow in a sample of material that is being heated up and compares it to the heat flow in a reference sample. If the sample has undergone a phase transition, then the amount of heat required to keep the test sample at the same temperature as the test sample will change. This tells us what state the test sample is in. This is useful when determining the crystallisation class as it lets us know whether the sample crystallises or stays amorphous upon reheating. It also lets us determine the glass transition temperature if the test sample is heated up until a glass transition is observed.

1.1.3 Using Machine Learning to Predict Stability

The idea of predicting the stability of amorphous drugs using machine learning (ML) is not an entirely novel one and there is a fair amount of research on this topic already [31, 32]. The bottleneck, when it comes to making progress in this area is the lack of data. Firstly, the process of crystallisation has to take place before we know the stability of a drug. In some cases, this can take years to happen. Secondly, as previously mentioned, the stability of a drug is affected by many different variables such as the temperature, the humidity, and the method used to create the amorphous drug. This causes issues when trying to determine the stability, as the experimental values are not necessarily consistent. As a result of these challenges, the dataset we have for amorphous drug stability is extremely small, our largest contains only 137 molecules.

Even though the set of data we have about amorphous stability is small, there have still been attempts to try and predict it. Most of these attempts are made using treating the stability as the response variable in fairly simple models. For example, the work by Nurzyńska et al. [33] developed a model that takes measured molecular, thermodynamic, and kinetic parameters from a dataset of 25 molecules and develops a multiple linear regression model to predict stability. The best model created in this work had an adjusted R^2 of 0.7 while using descriptors such as glass transition temperature, enthalpy of fusion, and lipophilicity. The paper also claims that it correctly predicted the stability of 60% of the molecules in the unseen test set. Although this may seem impressive, nurzyńska et al. defined a correct prediction as one within 4 days of the true stability, and although this may seem reasonable, their unseen test dataset only contained 5 molecules and three of them had a stability of one day or less, so perhaps the criteria for an accurate prediction should be less forgiving.

The current state of the art in machine learning for pharmaceutical drugs is characterised by a burgeoning field that holds immense promise for revolutionising drug discovery and development. Machine learning techniques, particularly deep learning and various flavors of neural networks, have gained prominence in the pharmaceutical industry [34, 35, 36]. These algorithms are being used to analyse vast datasets, including chemical structures [37], genomic data [38], clinical trial results [39, 40], and medical records [41], to expedite the identification of potential drug candidates and predict their properties, safety, and efficacy. Furthermore, machine learning models are playing a pivotal role in target identification [42], drug repurposing [43], and optimising clinical trial designs [44], thus streamlining the drug development pipeline. Additionally, generative models are being leveraged for de novo molecule design [45], allowing for the creation of novel compounds

with desired properties. The integration of artificial intelligence and machine learning into pharmaceutical research has the potential to significantly reduce the time and cost associated with bringing new drugs to market, ultimately leading to more efficient and personalised healthcare solutions. However, challenges such as data quality, interpretability, and regulatory considerations still need to be addressed as this field continues to evolve.

In this work we take a different approach to try and predict the stability of amorphous drugs. Instead of using only a handful of easily calculable properties, we opt to generate actual 3D models of drugs using molecular dynamic simulations. Using these models, we can generate more advanced descriptors relative to both the structural and dynamical properties of the drugs. By using this approach we are able to emulate how the drug-like molecules would behave under certain conditions. In turn, this can give us access to descriptors that would otherwise have to be calculated experimentally.

1.1.4 T_g and class

Due to the extremely limited amount of reliable data on the stability of amorphous drugs, we seek to first predict two key properties that are closely related to the stability of amorphous pharmaceuticals. The first property we look at is the glass transition temperature (T_g).The glass transition temperature describes the temperature at which an amorphous solid transforms from a rigid, non-flowing state, known as a glass, to a more fluid-like low-viscosity state (or vice versa). More concretely, the T_g is typically defined as the temperature at which the shear viscosity of the system is equal to 10^{12} centipoise (cP) [46]. It is important to note that unlike e.g. the melting temperature (T_m), which is fixed, the glass transition temperature is not a thermodynamic property. This is due to the fact that T_g can change depending on various factors such as the heating/cooling rate, or the method used to prepare the amorphous substance.

T_g is defined as the temperature at which an amorphous material, such as a polymer or a glass, transitions from a hard, brittle, glassy state to a more rubbery or viscous state, without actually melting.

When the material is at a temperature below the T_g, it is in a glassy state, whereby the molecules are frozen in place. This low molecular mobility causes the material to be rigid and brittle. The mobility increases at temperatures above the T_g and the material becomes more flexible and elastic until it transitions into a liquid at T_m.

1.2 Overview

T_g is a key property in the context of amorphous formulations[28, 12, 48] in that (i.) it affects the propensity of the system to form a disordered solid as opposed to a crystal in the first place [49] and; (ii.) it correlates to a good extent with the physical stability of the amorphous phase [50], which needs to not re-crystallize over the typical timescales involved with the shelf-life of a marketed pharmaceutical. When T_g is higher, the drug is more stable and less likely to crystallise [51]. The higher T_g indicates a higher degree of molecular mobility restriction and reduced free volume. Conversely, a low T_g could mean that the drug is too unstable to use.

The second property we focus on in this work is the so-called crystallisation class. This is a classification system developed by Baird et al. [52] whereby 51 organic molecules were separated into 3 distinct classes based on their propensity to crystallise during the heat/cool/heat cycle described in section 1.1.2. These classes are defined as:

1. Crystallisation is observed whilst cooling the melt at temperatures greater than the T_g.

2. No crystallisation is observed when cooling from the melt to below the T_g. However, the system will crystallise when reheated above the T_g. This is perhaps the most interesting class of amorphous drugs from a fundamental perspective.

3. No crystallisation is observed at all throughout the heat/cool/heat cycle.

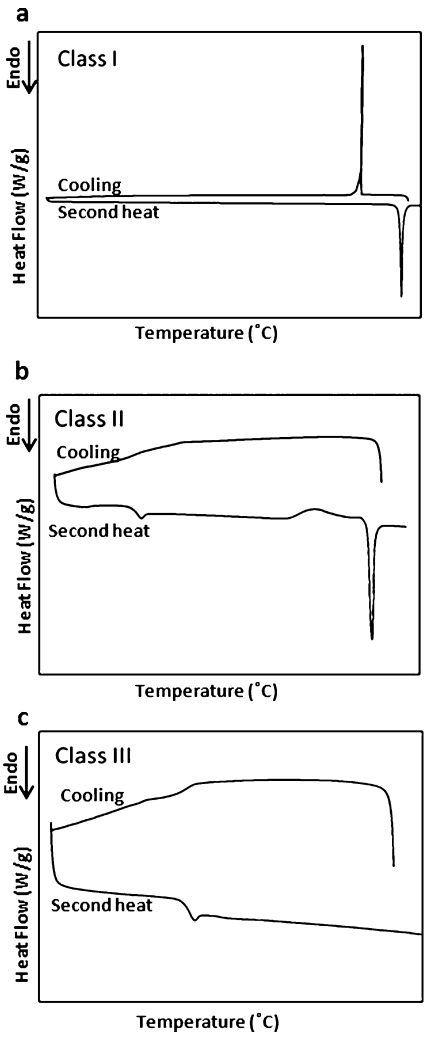

Figure 1.2: DSC thermograms [47]. The spikes in the thermograms indicate crystallisation. We see that Class 1 molecules crystallise on cooling, Class 2 molecules crystallise when reheated, and Class 3 molecules do not crystallise at all.

An illustrative diagram showing the difference in heat flow between these three classes is shown in figure 1.2. These classes are useful for the formulation and development of amorphous drugs. If the drug-like molecule falls into Class

1 we know that it will not make a good amorphous drug, so it is not worth synthesising and testing in a lab. If the molecule is a class 3 molecule then we can deduce that it could be a good candidate for an amorphous drug. The ability to accurately predict this class would represent a step forward in being able to increase the efficiency of bringing amorphous drugs to market.

Overall, the objectives of this work are to develop a method of accurately and reliably predicting which crystallisation class a molecule will fall into, and to also develop some sort of paradigm for estimating the glass transition temperature of a given molecule.

The present work is structured into six chapters, each of which aims to address different aspects of the research project. The current chapter provides the background and context for the study, outlining the problem under investigation and describing the datasets used in the analysis. In this chapter, we provide details on the sources of data, the methods of data collection and preparation, and the characteristics of the datasets that were utilized in the study.

The second chapter is dedicated to explaining the methods employed in the study. This chapter delves into the theoretical and practical aspects of the methods used to analyze the data, including statistical and mathematical techniques, as well as computational tools. In this chapter, we explore the strengths and limitations of the methods used and how they were applied in the context of the study.

The third chapter focuses on the machine learning protocol and workflow. Here, we describe the process of designing and implementing machine learning algorithms to analyze the datasets. This chapter provides details on the steps involved in data preprocessing, feature selection, model building, and performance evaluation.

Chapter four presents a specific example that is particularly relevant for understanding the dynamical and structural properties of molecular liquids and the changes they undergo as they approach the glass transition. In this chapter, we discuss the latest developments in this area and their significance for the broader field of amorphous drugs.

The fifth chapter outlines the genetic algorithm framework used in the study. This chapter explains the design and implementation of the genetic algorithm used to optimize the machine learning models. Here, we provide an in-depth discussion of the key features of the genetic algorithm and its performance in optimizing the models.

Finally, the last chapter presents the results of the study. This chapter outlines the findings of the analysis, including the insights gained from the data, the performance of the machine learning models, and the implications of the study for the broader field of amorphous drugs. Additionally, the chapter discusses the limitations of the study and suggests future research directions. Overall, this work offers a comprehensive overview of the re-

search project, covering the entire process from data collection and analysis to the development of machine learning algorithms and the presentation of the findings.

This work goes beyond the current state-of-the-art by harnessing a novel set of descriptors, so far unexplored in the context of predicting the stability of amorphous pharmaceuticals. In doing so, it not only advances the scientific understanding in this domain but also enhances the precision and reliability of predictions pertaining to crystallisation classification. By incorporating the techniques showcased in this work, we aim to unlock deeper insights into the complexities of molecular structures and also seeks to optimise the performance of these descriptors in predicting the stability of amorphous drugs, pushing the boundaries of what is currently achievable in the field.

1.3 Datasets

The majority of the results in this work pertain to two similar datasets:

- **Amorphous T_g [Amo-Reg]**: This is a dataset we have curated from literature data (refs [28, 29]). These are the only sources that were available to us that contained datasets of drug-like molecules and their experimental T_g. Fortunately, the experimental methodology to compute the T_g is the same for both of these works and is described in section 1.1.2. This contains 136 molecules with the entries consisting of the name of the molecule, a representation of the molecule in the form of Simplified Molecular Input Line Entry System (SMILES) [53], and the experimental value for T_g.

- **Amorphous Class [Amo-Class]**: This dataset is a subset of the Amo-Reg dataset and is compiled from literature [47]. This dataset contains the crystallisation class (see section 1.1.4) of 124 molecules present in the Amo-Reg dataset. This data allows us to explore the behaviour of certain drug-like molecules when they are heated and rapidly cooled in an attempt to create amorphous drugs.

The majority of our results are generated through these datasets. Although Amo-Class is a subset of Amo-Reg, they are treated as separate datasets. This means that we do not use the experimental T_g to predict the class or vice-versa. We have chosen to use these as our main datasets since the overlap in molecules allows us to minimise computational costs for molecular dynamics simulations as the same simulations can be used for both T_g and class tasks.

We summarise in Fig. 1.3 some information about the target properties of both datasets. The distribution of T_g across the whole Amo-Reg dataset (Fig. 1.3a) appears to be peaked

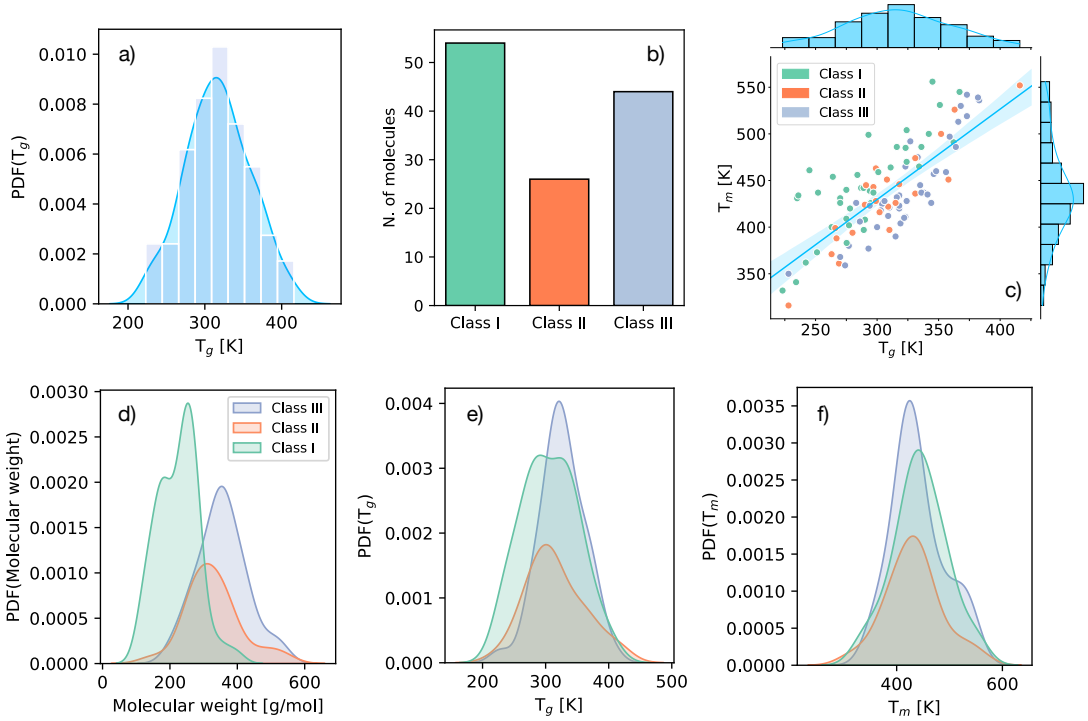

Figure 1.3: a) Probability density function (PDF) of the T_g re: the Amo-Reg dataset. The continuous distribution has been obtained via a kernel density estimation (KDE). b) The population of each crystallisation class re: the Amo-Class dataset. The smallest molecule is urea with 8 atoms and the largest is ritonavir with 98. c) The correlation between T_g and T_m re: the Amo-Reg dataset. The marginal distributions refers to all the available data, notwithstanding the crystallisation class. d) PDFs (via KDE) of the molecular weight re: the Amo-Reg dataset. e) PDFs (via KDE) of T_g re: the Amo-Reg dataset, with information about each different crystallisation class. f) PDFs (via KDE) of T_m re: the Amo-Reg dataset.

at around room temperature. However, the T_g of molecules belonging to Class III is rarely < 250 K (Fig. 1.3e), which is consistent with the assumption that molecules characterised by high T_g are less prone to crystallise. Note that Class II is substantially under populated if compared with either Class I or III (Fig. 1.3b) - an issue we will discuss in greater detail in the following chapters. It is also instructive to look at the distributions of the molecular weight across the different classes, reported in Fig. 1.3d) specifically, Class I molecules appear to be characterised by, on average, lower molecular weight if compared to molecules in either Class II or Class III. This suggests that smaller molecules have a stronger tendency to crystallise. Intuitively, this might be explained in terms of diffusivity, as smaller molecules tend to be characterised by higher self-diffusion coefficients, which in turn might facilitate

Chemical element	Amo-Reg		Amo-Class	
	Atoms	Molecules	Atoms	Molecules
H	2138	136	2092	124
C	2138	136	1874	124
O	410	133	355	121
N	227	98	205	89
F	48	19	38	16
Cl	43	28	40	25
S	37	31	34	28
P	1	1	1	1

Table 1.1: Frequency by which chemical elements appears in either the Amo-Reg or the Amo-Class datasets (see text). We report the overall occurrence of a given chemical element ("Atoms" columns) as well as the number of molecules containing a given chemical element ("Molecules" columns).

the crystallisation process. In contrast, is is challenging to extract any meaningful trend from the distributions of T_m (Fig. 1.3f), despite the fact that there exist a strong correlation between T_m and T_g, as illustrated in Fig. 1.3c).

In terms of the chemical composition of the drug molecules in the dataset, we have summarised in Table 1.1 the frequency by which the relevant chemical elements appears in either the Amo-Reg or the Amo-Class datasets. The relative populations of these chemical elements are in line with those expected when considering small drug-like organic molecules.

Throughout this work we also use a number of other datasets to validate our techniques and see how applicable they are to datasets not related to amorphous pharmaceuticals. Some tasks benefit from larger datasets, and it is important for testing methods such as our genetic algorithm that we validate them on different types of data. The other datasets we use are:

- **Lipohilicity [Lipo]**: this dataset is publicly available via the *moleculenet.ai*project [54]. It contains $\sim$ 4000 molecular structures as SMILEstrings [55] and their corresponding lipophilicity [56], measured experimentally as octanol/water distribution coefficients (logD at pH 7.4). In the context of pharmaceuticals, the lipophilicity of a certain drug provides a measure of its affinity for a lipid environment - thus including the cellular membrane. It is a majorly important biophysical target, as it affects the pharmacokinetics and the absorption of many drugs formulations.

- **Hepatocytes [Hepa]**: this dataset has been provided to us by AstraZeneca. It contains $\sim$ 400 molecular structures as SMILES strings and their corresponding human hepa-

tocytes intrinsic clearance (clint) [57], measured experimentally as log(Volume/Time). Clint values quantify the ability of the human liver (particularly of the hepatocytes cells that constitute more than half of it) to remove a given drug: as the liver plays a very important role in dictating drug metabolism in our bodies, clint values are considered as crucial biological targets for drug design. We note that this is a very "challenging" dataset, in that it features only a small number of data points while dealing with exceptionally complex biomedical activity.

- **Solubility [Sol]**: this dataset contains 6,119 drug-like molecules and their solubility. The solubility (S) i.e. the extent to which a chemical substance can dissolve in a solvent and form a homogeneous solution. It is customarily represented using the base 10 logarithm as $\log S$, with S in moles per litre units [58, 59]. Based on the information contained in Refs [60, 61], we believe that the solubility values in question refer to the thermodynamic solubility of these molecules. This dataset was curated by merging several sub-datasets containing solubility values characterised by an uncertainty inferior to 0.4 $\log S$ so as to maximise the reliability of the experimental data (a notorious issue when dealing with solubility measures) quality. This particular threshold in terms of uncertainty corresponds to the standard deviation relative to the sets of experimental measurements of S obtained for the same compounds by different research groups [62]. Prior to use, we discarded 35 compounds that were either inorganic (i.e. they contained no C atoms) or contained counter-ions.

- **QM7b [QM]**: In order to validate the robustness of the results we have obtained in chapter 5, we have used an additional dataset, namely the QM7b dataset[63, 64]. This dataset is routinely used in the context of machine learning for molecular properties. It is a relatively low-noise dataset that contains 7,211 molecules and features 13 target properties - we have chosen to focus on the polarisability.

- **Stability [Stab]:** This dataset is a subset of the Amo-Reg dataset and it contains the stability of 20 molecules. The stability is measured in days and this dataset is used to see how well our models can predict the time taken for amorphous drugs to crystallise

Chapter 2

Methods

In this chapter, we present an overview of the two types of descriptors, namely single-molecule and solid-state descriptors, and describe the general process of obtaining them. Subsequently, we provide a detailed, step-by-step description of this process. We then proceed to discuss each descriptor individually, elucidating its calculation and relevance in our work. Finally, we detail the optimisation techniques employed to enhance the performance of these descriptors.

2.1 Generating molecular models

An innovative aspect of this study is the utilisation of MD simulations to obtain descriptors that cannot be obtained by examining single molecules in isolation. This is particularly noteworthy since the data sets available for many QSAR models, including those in our possession, typically only offer information about the molecular structure in the form of SMILES strings. To generate a three-dimensional model for each drug molecule in the AmoReg dataset, hydrogen atoms were added where necessary and OpenBabel [65] was employed. Following this, relevant topologies and force field parameters were obtained using CGenFF (version 4.6) based on the CHARMM36 force field (version July 2021) [66, 67, 68, 69].

For our MD simulations, we utilised the GROMACS package in version 5.1.4 without GPU acceleration and in single precision. Periodic boundary conditions were applied in all three Cartesian directions.The hardware used for these simulations is the same used for the machine learning described in Sec 3.1. We assumed that both the glass and liquid phases are isotropic and adopted cubic simulation boxes accordingly. The dimensions of the boxes were determined by the equilibrium density of each system. In general, the edge of the cubic boxes for the glass phases ranged between four and 6 nm. We set the cutoff for

both van der Waals and electrostatic interactions to 12 Å, with van der Waals interactions being switched to zero between 10 and 12 Å. We constrained the hydrogen bonds within each molecular species using the P-LINCS algorithm. The equations of motion were integrated using a leap-frog integrator with a time step of 2 fs. For sampling in the NVT and NPT ensemble, we employed the Bussi-Donadio-Parrinello thermostat and the Berendsen barostat with coupling constants of 0.5 and 4 ps, respectively. While the Berendsen barostat may not be the most accurate barostat for sampling the isothermal-isobaric ensemble [70], we chose it due to the frequent and substantial changes in simulation conditions that occurred during our study. This decision was made to ensure the robustness of the simulations over the accuracy of more complex barostats.

In the following sections, we provide a detailed account of the precise steps undertaken in the MD protocol, while concurrently illuminating our rationale for the selection of simulation parameters.

2.2 Molecular dynamics

In order to step away from the general methods used to predict T_g, as well as access additional information that can only be accessed by looking at models of the glasses, we have simulated the process of making every drug-like molecule in our database into an amorphous system using computer simulations.

There are various types of computer simulations used to study the behavior of molecules and materials. One common approach is Monte Carlo simulation, which uses random sampling to model the behavior of complex systems. In Monte Carlo simulations, a large number of random samples are generated and used to estimate the behavior of the system. This approach is commonly used in fields such as finance, engineering, and physics.

In this work, we use molecular dynamics (MD) simulations, which employ numerical methods to model the behavior of individual atoms and molecules over time. These simulations typically use classical mechanics to solve the equations of motion.

In an MD simulation, the forces acting on the particles in a system cause their positions and velocities to change at each time step (dt). One approach to determining the forces between particles in MD simulations is to use a potential energy function, such as the Lennard-Jones potential or Coulombic interactions between charged particles. While these methods are commonly used, they may not provide the most accurate characterization of the forces. A more accurate approach involves solving the Schrödinger equation to obtain the potential energy function and forces. However, this method is computationally expensive and not practical for simulating large systems like molecular glasses, especially when

using ab-initio methods. The timescales for ab-initio simulations can be much longer than for classical MD, due to the increased computational demands. Therefore, while ab-initio simulations may provide more accurate results, they may not be feasible for simulating large systems on practical timescales.

These simulations have the capacity to investigate diverse phenomena, ranging from the dynamics of large proteins and complex biomolecular systems to the behavior of small molecules [71, 72]. These simulations prove to be particularly advantageous in predicting the properties of new materials that are yet to be synthesized and assessing the performance of materials under various conditions, such as alterations in temperature or pressure. Furthermore, MD simulations enable the simulation of scenarios that may be infeasible or exorbitantly expensive to achieve in real life, such as extremely high or low temperatures. Because of this, they have developed into a crucial tool in many branches of engineering and science, including as biophysics, materials science, and, in our case, drug development. They offer an effective way to comprehend the fundamental molecular mechanisms that control the behaviour of complex systems and to forecast how these systems will behave in various scenarios.

Using Newton's equations of motion to determine the paths of individual atoms in a system based on their initial positions and velocities, as well as the forces acting upon them, is the core idea underpinning MD simulations. Spheres or particles are used to represent the atoms in MD simulations, and force fields are used to characterise the interactions between atoms. A force field is a mathematical function that models the interatomic forces between atoms in a molecular system. The force field is used to compute the potential energy of the system at every time step of the simulation. By integrating the equations of motion using the potential energy, the positions and velocities of the atoms are propagated forward in time, allowing for the simulation of the system's dynamic behavior.

A force field consists of a set of parameters that define the nature of the interactions between atoms in the system. These parameters include the strength and range of the non-bonded van der Waals forces, which govern the attraction and repulsion between atoms, as well as the strength and directionality of the bonded forces, such as covalent bonds, angle bending, and dihedral rotation. These parameters are typically obtained from experimental data or quantum chemical calculations and are fitted to reproduce the observed behavior of the molecules of interest. Force fields are often specific to a particular class of molecules, such as proteins or nucleic acids, and can vary in their accuracy and applicability to different molecular systems.

The simulation is typically run by breaking time into small time steps, and for each time step, the atoms' positions and velocities are modified in accordance with the forces

acting on them. The simulation can capture the dynamics and behaviour of the system over a period of time by repeating this process for thousands or even millions of time steps.

Our MD simulations are designed to emulate the manufacturing process that is outlined in section 1.1.2 whereby the molecule is heated up to a melt, rapidly cooled, and then reheated. In order for us to do this, we need to know the T_g of each molecule. Unfortunately, an artifact of compiling our data from different sources is that we do not have uniform information about each molecule. As a result, the T_g is missing from a significant portion of the molecules we are looking at. To ensure fairness with all descriptors, it would not make sense for us to get the T_g from a different source since the T_g can be calculated in a number of different ways that can impact the results you get. In fact, when attempting to obtain the glass transition temperature of molecules through literature or online databases, it is a frequent occurrence for multiple distinct values to be reported.

In order to make the T_g we use for each molecule as consistent and fair as possible, we endeavour to calculate the T_g computationally using MD simulations. More detail on this process can be found in section 2.2. We note that table 2.1 explains this process in a more digestible way and would suggest that the reader refers to this table when reading section 2.2.

After we have computed T_g for each molecule, we then perform a very similar MD protocol to simulate the creating of an amorphous system of the given molecule. Unfortunately, this results in doing two very similar MD simulations for each molecule. This is a costly process but it is one that can not be avoided. As a result of this effort we have a dataset where the T_g is calculated in a consistent way, and all the steps of the MD are done at temperatures proportional to their calculated T_g.

In the following sections we detail the exact process that we use to create both the calculated T_g, and the MD simulation protocol that uses that T_g to get the descriptors that require simulation.

2.2.1 Calculating the forces

The selection of a force field is a crucial aspect of MD simulations, and in this study, we opted for the CHARMM36 force field [66]. The decision to use this force field was based on our prior experience with it and our success in using it to study a range of problems, including ice formation at biological interfaces [73, 74, 75, 76, 77] and molecular glasses [78].

It should be noted that the purpose of this study was to extract descriptors for building ML models, rather than obtaining accurate parameterisation of each molecule in the dataset. Therefore, we argue that optimising the parameterisation of over 100 molecules

in an efficient manner is not feasible in this context, regardless of the choice of force field. Additionally, while penalties associated with the parameterisation of each drug molecule were available, we chose not to include them in our ML models. This decision could have impacted the predictive power of the solid-state descriptors, as discussed in the following sections, but it did not impede us from utilising these descriptors to enhance the overall accuracy of our ML models.

The calculation of the force acting upon each particle is computationally expensive and is of order $\mathcal{O}(n^2)$ since we are considering the sum of all pairwise interactions for each particle. It is possible to speed up this process by estimating long range force interactions and only calculating interactions within a certain cutoff (except for in the case of electrostatic interactions), but obviously this leads to reduced accuracy. Note that in the case of electrostatic interactions, because they decay very slowly, we use the Particle Mesh Ewald method. More information on this method can be found in [79]

2.2.2 Initialising the system

We begin the simulation by placing all the molecules in the system within a box. The box is subdivided into a cubic lattice with a molecule being placed into individual sections of the lattice, see figure 2.5 d) for a visualisation of this. This prevents molecules from overlapping with each other and creating systems that are physically impossible. Each particle is then assigned a velocity that is taken randomly from a pre-defined distribution and subsequently shifted so that the system has a total momentum of zero. The velocities are then scaled to achieve a mean kinetic energy of a given temperature.

2.2.3 Calculating the velocities

The velocities of particles can be calculated using a variety of methods, the most common is the Verlet algorithm which we use for all the simulations in this work. We begin by noting that the force acting on each particle is calculated by solving Newtons second law,

$$\bar{F} = m\bar{a} \tag{2.1}$$

where $\bar{F}$ is the force acting on the particle, given by the force field, m is the mass of the particle, and $\bar{a}$ is the acceleration of the particle that is trivially computed by rearranging equation 2.1.

As mentioned in section 2.2.2, the velocities are initialised randomly from a distribution. So we now have the initial position $\bar{x}(0)$, velocity $\bar{v}(0)$, and acceleration $\bar{a}(0)$ of each

particle.

Following this we can work out the position of a particle at some timestep Δt as

$$\bar{x}(\Delta t) = \bar{x}(0) + \bar{v}(0)\Delta t + \frac{1}{2}\bar{a}(0)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{2.2}$$

which is a Taylor expansion of the coordinates of a particle around time t. Higher-order Taylor series expansions can be used to increase the accuracy, however, this is generally not necessary and will lead to greater computational expense.

The position of each particle can be updated using the following equation

$$\bar{x}(t + \Delta t) = 2\bar{x}(t) - \bar{x}(t - \Delta t) + \bar{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{2.3}$$

It is now possible to get the position of a particle at a given timestep. From this, the velocity can be calculated by working out the rate of change of position, given by

$$\bar{v}(t) = \frac{\bar{x}(t + \Delta t) - \bar{x}(t - \Delta t)}{2\Delta t} \tag{2.4}$$

where $\bar{v}(t)$ is the velocity of the particle at time t.

Since the initial velocities are generated using a random distribution, it is likely that the initial velocities are not accurate representations of the true velocities of the particles in the system. It is also possible that the thermal energy may not be evenly distributed amongst the particles. Structurally, the system may also not be representative of something that is physically plausible. Equilibration allows the system to reach a representative state where the particles are in realistic positions.

2.2.4 Compressing the system

After initial velocities are determined and the initial conditions of the system are met we begin the initial equilibration process. The MD simulation is run at a high temperature and pressure for 10ns, this was achieved at 1,000 K and 1,000 bar for the systems in this work, one of the benefits of MD simulations are that we can construct systems that would be either extremely expensive or physically impossible to create. By running the system at such a high temperature and pressure, we compress the system in a way that allows the molecules to be packed in a way that is similar to what we would typically see in reality.

There are other methods to achieve this, one of the more common ways is to use software specifically designed to achieve a similar result, such as PACKMOL [80]. Similar to our compression protocol, this software packs the molecules into a system in a way that

satisfies various spacial restrictions imposed by user defined input.

2.2.5 Cooling the system down

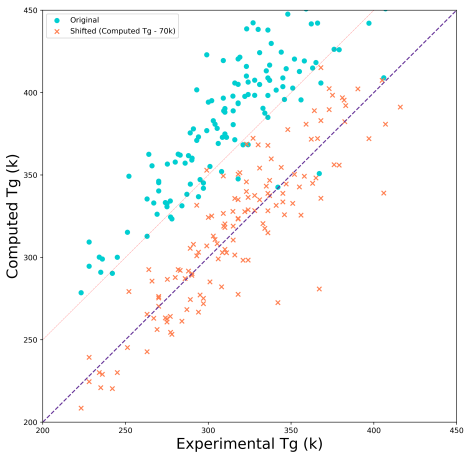

Figure 2.1: Correlation between the experimental T_g and the computed T_g of molecules in our dataset. The computed T_g values have been uniformly shifted by -70K to account for systematic errors and this adjustment has reduced the MSE from 5,425 to 481. The purple dashed line simply represents the best possible result, whereas the red dotted line shows the experimental T_g shifted by 50K . Since our simulations start at 50K below the computed T_g, all points to the right of the red dotted line will start the MD simulation below their experimental T_g. Details of how the computed T_g values were obtained are given in section 2.3.7

Once we are satisfied that the system conformation is realistic and well-behaving, we now cool the system slowly to the temperature that we wish to begin the heat/cool/heat cycle from. This is done while instead of keeping the pressure constant as in section 2.2.4 we instead keep the volume constant. Note that since for each system we are performing two similar MD simulations, one to calculate an estimate of the T_g (section 2.2) and one to use for our dynamic descriptors (section 2.3.6) we give details on both protocols.

In the case of performing MD to estimate T_g, the first phase of the cooling involves cool the system from 1,000 K to 550 K over 20 ns. Then the second phase takes the system to 100 K over another 20 ns. Our testing showed that at the temperature of 100 K, all the molecules in our systems were frozen as there was not enough energy for the molecules to move, In fact, at any temperature below T_g the diffusion should be negligible. Since all our systems have negligible diffusion, we can be confident that 100 K is below the T_g of all the molecules in our dataset.

Our MD simulations to create an amorphous system for analysis follow a similar idea but are instead influenced by the molecule of interests computed T_g, for the rest of this section when we refer to the T_g of a molecule, we are talking about the computed T_g, not its actual T_g since this is unknown

in some cases. The first phase of cooling at a high pressure anneals the system temperature down to the half way point between 1000K and T_g - 50K . The second phase of annealing takes the system all the way down to T_g - 50K at atmospheric pressure. Both of these phases take 20ns each. By doing this, we can assume that our system is at a temperature below its true T_g even if the calculated T_g is somewhat incorrect. As shown in fig 2.1, based on the data for the T_g that we do have, this is a fairly safe assumption to make.

2.2.6 Equilibration 1

Most physical experiments in a lab take place using a constant number of particles, N, pressure, P, and temperature, T. We can emulate this computationally by utilising a common technique in MD simulations. A simulation where N,P and T is known as isobaric-isothermal ensemble simulation, or more succinctly, an NPT simulation. By calculating the dynamics of these constant NPT equilibration periods, the constant pressure and temperature allow the system to settle on a state of lowest potential energy. In an attempt to make our results as accurate as possible and mimic physical experiments, we perform NPT simulations after every change in system temperature. These NPT simulations are important for analysis, the data that we take from the MD simulations is always obtained at the end of these equilibration periods. The first occurrence of this is after the cooling phase described in section 2.2.5. It is important to perform NPT simulations for a long enough period of time to allow the system to reach a state of lowest potential energy, for this reason we simulate this ensemble for 10 ns.

To verify that the isobaric-isothermal ensemble is working as intended, in figure 2.2 we have included plots of temperature, pressure and density for Aceclofenac, a prototypical system in our dataset, during the first equilibration stage directly after the initial cooling. We notice that the temperature fluctuates between 240-226 K even though it is set to remain constant at 233 K, the same thing happens to the pressure albeit with much larger fluctuations between ± 1,500 bar. The reason for these fluctuations can be attributed to a number of factors such as finite size effects, the integration algorithm used, or the initial simulation conditions chosen. However, as we can see the rolling average value of temperature and pressure are close to the specified value of 233k and 1 bar respectively. Further investigation of this simulation shows that the temperature of the system has an average of 232.89k ± 0.57k, this is very close to the specified value of 233k. We notice that when we look at the average value for the pressure we get 29.88 bar ± 162.53 bar. Although this seems like a significant difference to the specified 1 bar, pressure typically has very large fluctuations in MD simulations.

To further verify that the fluctuations in pressure are not causing any issues, we can look at the density plot in fig 2.2.

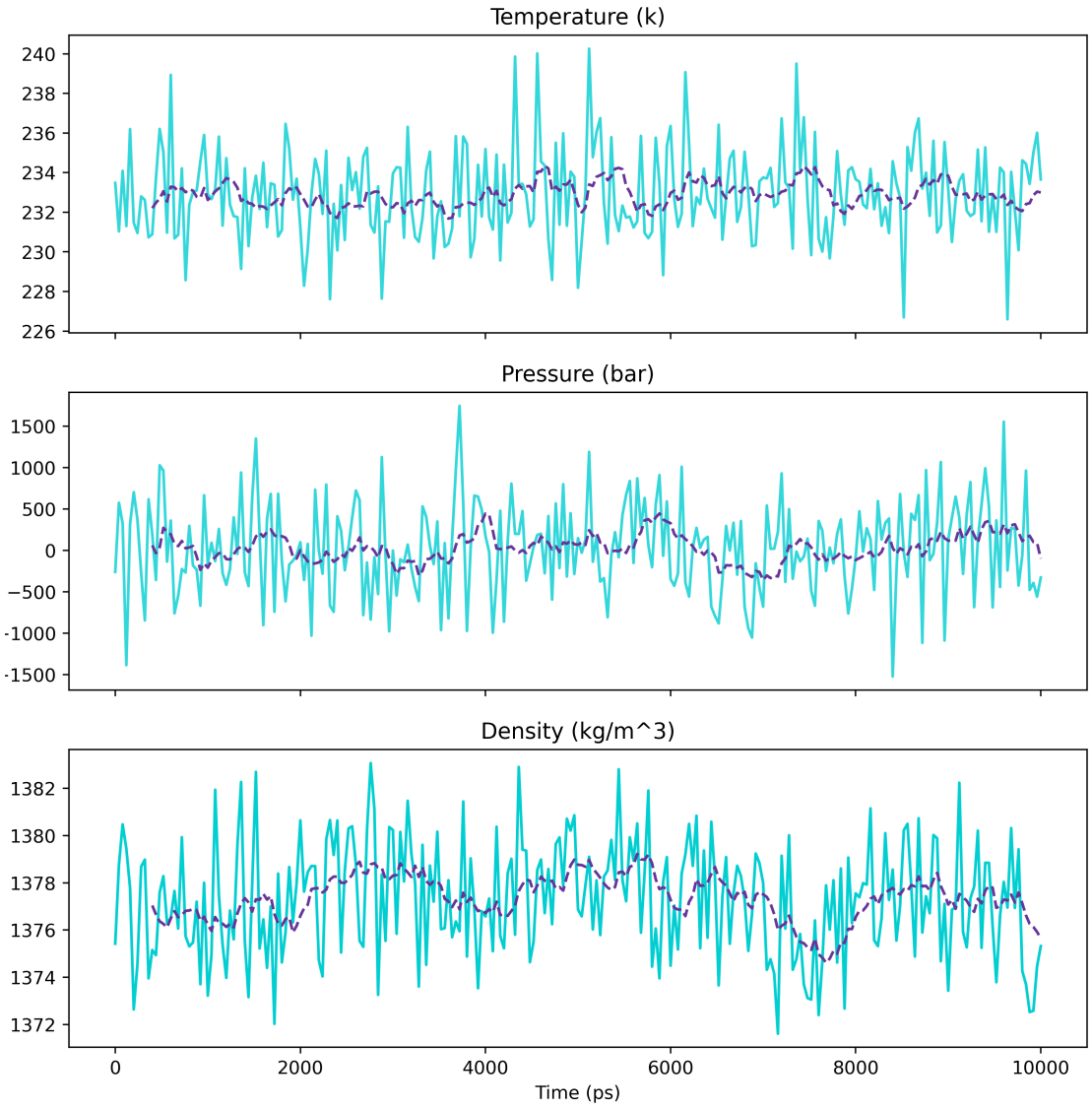

Figure 2.2: Temperature, pressure and density for Aceclofenac, a prototypical system in our dataset, during the first equilibration stage directly after the initial cooling. This molecule has a calculated T_g of 273k and therefore the system is set to keep the temperature at a constant 233k during this phase. The dashed purple line represents the 100ps running average.

2.2.7 Annealing the system

After the equilibration 1 phase described in section 2.2.6, we have a system that is at a very low temperature and total energy. We can also be fairly confident that this temperature is below the molecules true T_g, so the next thing to do is to heat up the system past the T_g so we can study the melt. This process is intentionally done in many different 'steps' in order to have multiple equilibrated configurations for a selection of temperatures. All of the following simulations are performed at 1 bar.

In the case of MD to calculate the T_g, we start at 100K , then gradually increase the temperature in steps of 40 K over 5 ns. After this, an NPT simulation is performed at 140 K to equilibrate the system. This process of increasing the temperature by 40 K and then equilibrating at the new temperature is repeated 9 times until a temperature of 460 K has been reached. Upon reaching this final temperature, the NPT simulation is run for 20 ns instead of 5 ns. The reason for the extended NPT is that a lot of descriptors can be calculated using the trajectories at this temperature, so having a longer ensemble of value will only lead to more accurate results.

For the MD to simulate an amorphous system, the process is almost identical, except we do not start at 100 K. For these simulations, we start at 50 K below the calculated T_g value obtained from the first simulations. We start at this temperature because we can say with a high level of certainty that 50K below the computed T_g is below the experimental T_g. Evidence for this claim is shown in fig 2.1 where we see the computed T_g plotted against the experimental T_g. For all molecules to the right of the red dotted line the following inequality holds true

$$\text{computed } T_g - 50K \leq \text{experimental } T_g$$

The only other difference to the MD simulation used to calculate an estimate for T_g, is instead of increasing the temperature by 40K 9 times, we only do this process 6 times. This is because we know we are starting much closer to the true T_g than in the previous simulation and therefore the temperature does not need to be increased by the same margin to reach the amorphous phase. The timescale for the simulations and the final, high temperature, NPT is identical to the calculating T_g simulation.

Finally, we now decrease the temperature back to the starting temperature. This is done by reversing the step up protocol. So where we stepped up the temperature 9 times by 40K , we instead now decrease it 9 times by 40K . Likewise when we increased the temperature 6 times by 40K , we decrease the temperature 6 times by 40K . Again, each change in temperature is followed by a 5ns NPT simulation to equilibrate the system.

There are a handful of reasons that we decrease the temperature in the simulation

after we have reached the desired amorphous phase. Firstly, this is necessary for when we are trying to calculate an estimate of the T_g since this estimate is based on the cooling and heating rate of the system, more information on this can be found in section 2.2. Secondly, our reason for decreasing the temperature is that it allows us to create statistically independant conformations of the system. Since we have such a small dataset, it may be useful to have multiple representations of the same molecule for the purposes of model training and data generation. This is particularly useful when dealing with our classification dataset due to the fact that we have a rather bad case of mismatching class representation, so having a way to equalise the number of datapoints in each class is necessary.

The process of creating these statistically independent conformations of the same molecule is fairly straightforward. Instead of beginning the step down process after the 20 ns equilibration simulation at high temperature, we can simply wait another 5 ns and begin the step down after 25 ns. Since the starting conformation is different, the trajectories for the following step down simulations will also be different. We can do this as many times as we want. By waiting an additional 5 ns and then doing the step down MD, we create a new trajectory for the same model each time.

An overview of the entire MD process for both the simulations to estimate the T_g and to simulate the amorphous phase is available in table 2.1, as well as a schematic available in fig 2.5 a) .

2.3 Descriptors

In this study, the molecular descriptors used can be classified into two distinct categories, single-molecule descriptors and solid-state descriptors. Single-molecule descriptors are derived solely from the molecular structure of an isolated drug molecule in a vacuum. This type of descriptor is widely used in the development of quantitative structure-activity relationship (QSAR) models. On the other hand, solid-state descriptors are derived from molecular dynamics (MD) simulations, which provide insights into the structural and dynamic properties of the actual material. The central premise of this study is that by incorporating solid-state descriptors derived from MD simulations, it is possible to augment and improve the existing portfolio of single-molecule descriptors used in QSAR modeling. This approach can potentially lead to more accurate and reliable predictions of the properties and behavior of amorphous drugs. By leveraging the information obtained from both types of descriptors, this study aims to provide a more comprehensive understanding of the molecular mechanisms underlying the behavior of amorphous drugs and pave the way for the development of more effective drug formulations.

Phase	Calculating T_g	Amorphous simulation
Compression	1,000K 1,000 bar 10 ns	1,000K 1,000 bar 10 ns
Cooling (phase 1)	1,000 K $\rightarrow 550K$ 1,000 bar 20 ns	1,000 K $\rightarrow 1000 + (T_g - 50)_{\frac{-}{2}}$ K 1,000 bar 20 ns
Cooling (phase 2)	550 K $\rightarrow 100K$ 1 bar 20 ns	$\frac{1000+(T_g-50)}{2}$ K $\rightarrow$ (T_g - 50) K 1 bar 20 ns
Equilibration 1	100 K 1 bar 20 ns	(T_g - 50) K 1 bar 20 ns
Step Up i	100 + 40i K for i $\in \{1,2,...,9\}$ 1 bar 5 ns	(T_g - 50) + 40i K for i $\in \{1,2,...,6\}$ 1 bar 5 ns
Step Up NPT i	100 + 40i K for i $\in \{1,2,...,9\}$ 1 bar 5 ns	(T_g - 50) + 40i K for i $\in \{1,2,...,6\}$ 1 bar 5 ns
Equilibration 2	460 K 1 bar 20 ns	(T_g + 190) K 1 bar 20 ns
Step Down i	460 - 40i K for i $\in \{1,2,...,9\}$ 1 bar 5 ns	(T_g + 190) - 40i K for i $\in \{1,2,...,6\}$ 1 bar 5 ns
Step Down NPT i	100 + 40i K for i $\in \{1,2,...,9\}$ 1 bar 5 ns	(T_g + 190) - 40i K for i $\in \{1,2,...,6\}$ 1 bar 5 ns

Table 2.1: Table showing an overview of the entire MD protocol for generating an estimate of the T_g, and then generating a simulation of the amorphous system using the previously calculated T_g.

2.3.1 Single-molecule descriptors

2.3.2 Standard descriptors

The most rudimentary descriptor we have used is an array of what we are going to label as 'standard' descriptors. These are physical properties that are easily calculable through many different software packages, such as the RDKit.Chem [81] Python package. This package gave us access to a set of 170 2D and 3D descriptors calculated from only the molecular SMILES. In order to make use of the 3D descriptors we had to generate 3D conformers of the

molecules. We deliberately used a very basic procedure to do this whereby we leveraged the ETKDG conformation generation protocol [82] followed by UFF forcefield optimisation [83]. These descriptors ranged from very basic properties such as molecular weight and number of hydrogen atoms, to more complicated ones such as the spherocity index [84]. Although a number of these parameters (such as the WHIM descriptor [85]) can be optimised, we have mimicked the minimal effort methods used in our previous work where we show that using a large number of descriptors is not as effective as using a few carefully selected descriptors [86].

2.3.3 Cliques

These descriptors are inspired by the work of Jin *et al.* [87], where the authors have decomposed a given molecular structure into sub-graphs ("cliques" in graph theory), thus providing a coarse-grained representation such as the one illustrated in Fig. 2.3 for the case of caffeine. Instead of connecting these components into a tree (as it was done Ref. 87), we have created a vocabulary of the unique cliques across the entire dataset of interest. Thus, different sets are typically characterised by cliques vocabularies of different length. Then, we index each of the cliques in the vocabulary via an integer $i = 0, 1, ..., N_{clq} - 1$, where N_{clq} is the total number of unique cliques in the vocabulary. Through one-hot encoding (see Fig. 2.3), each molecule in the dataset is converted into a vector of length N_{clq}: the value of the $i-th$ element of said vector is equal to the number of occurrences of the $i-th$ clique within that particular molecule.

In the context of natural language processing, we are thus treating the clique vocabulary as a "bag of words" to form sentences - i.e. molecules, in a similar fashion to the "bag of bonds" descriptor explored in e.g. Ref. 88. As the meaning of a given sentence may usually be determined to a good extent from its word content alone (i.e. without considering syntax), we are assuming that the presence of the cliques alone, without any information about the order by which they appear in a given molecular structure, would be enough to allow us to establish a structure-function relation between SMILES strings and the functional property of interest. It is thus reasonable to treat the cliques as a descriptor that is looking exclusively at the "chemistry" of the molecules, in that it highlights the presence or absence of specific molecular fragments and/or functional groups as opposed to the overall structure, albeit information about the size of the molecule is indirectly contained into the cliques vector. As we shall see in the Results section, this incredibly simple descriptor possesses a surprising predictive power, and it lends itself to feature selection in a very straightforward manner.

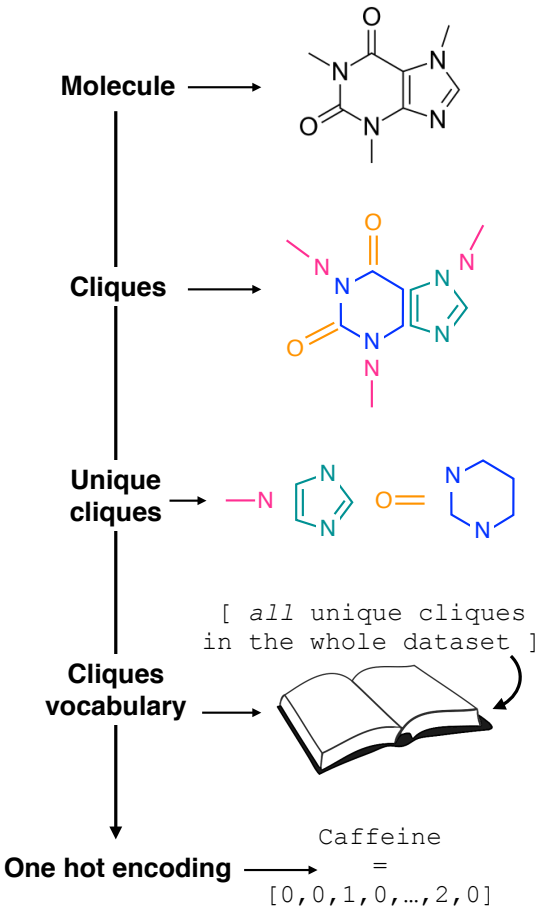

Figure 2.3: Constructing the molecular cliques descriptor. In line with the work of Jin *et al.* [87], a given molecular structure (we started from SMILES strings) is decomposed in molecular fragments known in graph theory as "cliques". All the N_{clq} unique cliques across the entire molecular dataset are then indexed and collected into a single cliques vocabulary. Each molecule in the dataset can thus be represented by means of one hot encoding as a N_{clq}-long vector with each $i-th$ element equal to the number of occurrences the $i-th$ clique appears in the molecule. Following an analogy with natural language processing, we are treating molecular fragments as words that we can combine together into sentences, i.e. molecules. Note the transparency of this descriptor, which requires as a starting point the molecular graph only and it does not include any information about the connectivity of the molecular fragments.

2.3.4 Histograms of Weighted Atom Centred Symmetry Functions

Atom-centred symmetry functions are popular three-dimensional descriptors in the context of ML-based interatomic potentials for molecular simulations (see e.g. Refs. [91, 92, 93]). While different flavours exist, they usually comprise sets of both radial and angular symmetry functions (SFs). In a nutshell, one sits on each atom i (see Fig. 2.4) and computes the value of (typically Gaussian) functions which depend on either $r_{ij} = |\bar{r}_j - \bar{r}_i|$ distances (radial SFs) or θ_{ijk} angles (angular SFs) between pairs or triplets of atoms up to a certain cutoff radius R_c. The interested reader can find a thorough introduction to SFs in Ref. [94]. Here, we have used as radial SFs:

$$G_i^{rad} = \sum_{j \neq i}^{N} e^{-\eta\left(r_{ij}-\mu\right)^2} f_{ij} \tag{2.5}$$

and as angular SFs:

$$\begin{aligned} G_i^{ang} = & \sum_{j \neq i}^{N} \sum_{k \neq i,j}^{N} \left(1 + \lambda \cos\theta_{ijk}\right) \\ & \times e^{-\eta\left(r_{ij}-\mu\right)^2} \times e^{-\eta\left(r_{ik}-\mu\right)^2} \times e^{-\eta\left(r_{jk}-\mu\right)^2} \\ & \times f_{ij} \times f_{ik} \times f_{jk} \end{aligned} \tag{2.6}$$

where μ and η represent the mean and width of the Gaussian respectively. The function f_{ij} is given by:

$$f_{ij} = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\pi r_{ij}}{R_c}\right) + 1\right] & \text{if } r_{ij} \leq R_c \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

Two sets of angular symmetry functions were calculated, one set with $\lambda = 1$, the other with $\lambda = -1$. Values for μ and η are determined by the number of SFs N used and the cutoff radius. For N SFs, the value of μ for function n is given by:

$$\mu = 0.5 + (n-1)\nabla r \tag{2.8}$$

where

$$\nabla r = \frac{R_c - 1}{N - 1} \tag{2.9}$$

and η is given by:

$$\eta = \frac{1}{2(\nabla r)^2}. \tag{2.10}$$

Crucially, the original formulation of SFs [89] required a distinct set of SFs for each combination of the different elements in a given molecule. While this is a perfectly sensible option in most materials science applications, where the number of elements involved is usually well below five (in fact, it is incredibly challenging to build ML-based interatomic potential for multi-component systems [91, 95, 96]), in the context of drug design and discovery a molecular dataset may very well contain more than ten elements, which leads to a huge number of SFs. Gastegger *et al.* have recently devised [90] a clever workaround to this issue by introducing so-called *weighted* SFs such as:

$$W_i^{rad} = \sum_{j\neq i}^{N} Z_i e^{-\eta\left(r_{ij}-\mu\right)^2} f_{ij} \tag{2.11}$$

$$\begin{aligned} W_i^{ang} = & \sum_{j\neq i}^{N}\sum_{k\neq i,j}^{N} Z_j Z_k \left(1+\lambda\cos\theta_{ijk}\right) \\ & \times e^{-\eta\left(r_{ij}-\mu\right)^2} \times e^{-\eta\left(r_{ik}-\mu\right)^2} \times e^{-\eta\left(r_{jk}-\mu\right)^2} \\ & \times f_{ij} \times f_{ik} \times f_{jk} \end{aligned} \tag{2.12}$$

where element-dependent weighting functions depending on Z_i (the atomic weight of atom i) are used to eliminate the need for separate sets of SFs for each combination of different elements, thus massively reducing the number of SFs needed as a whole.

Even weighted SFs, however, suffer from an issue of consistency, in that molecules with different elements and/or number of atoms are characterised by different numbers of SFs. As a result, the SFs vectors we would like to use as inputs for our ML algorithms are not of the same length. This problem may be circumvented in several ways, none of them optimal. As a start, if one seeks to predict a functional property that can be written as the sum of atomic contributions, the original approach of Behler and Parrinello [89] can be straightforwardly used. However, while one can think of some thermodynamic quantities such as energy or enthalpy as additive, functional properties or biomedical activities can often not be treated as such.

Here, we have decided to build histograms of weighted-SFs (H-wACSFs): by binning the values of all the weighted SFs for each molecule, we obtained a representation which is independent from the number of atoms in a given molecule. While the number of bins is one of the parameters we seek to optimise (see the following section), broadly speaking low and high numbers of bins provide more or less coarse-grained representations of the molecular structure. This interesting feature can be easily leveraged in the context of three-

dimensional models of crystalline or amorphous drugs - where we believe that materials science-inspired descriptors such as H-wACSFs could deliver important contributions.

As the starting point for our H-wACSFs sets we have arbitrarily chosen the following baseline parameter values: $N^{rad} = N^{ang} = 8$, $R_c^{Rad} = R_c^{Ang} = 20$ and $N_{H-bins} = 10$, where N^{Rad}, N^{Ang}, R_c^{Rad}, R_c^{Ang} and N_{H-bins} stand for the number of radial SFs, the number of angular SFs, the cutoff radius for the radial SFs, the cutoff radius for the angular SFs and the number of bins we have used to build the histograms, respectively.

2.3.5 Smooth Overlap of Atomic Positions

One descriptor that in many cases has been proven to be self-sufficient in offering an accurate representation of any given molecular structure is the Smooth Overlap of Atomic Potentials (SOAP) descriptor [97], even though its most commonly used form only encodes up to three-body correlations [98]. The SOAP descriptor has been gaining popularity lately given its impressive performance across a plethora of widely different classes of materials and problems ranging from hydrogen absorption of nano-clusters [99] to the development of bespoke interatomic potentials [100, 101]. The premise of the SOAP descriptor is that it offers a convenient method to describe atomic environments that are invariant to any form of rotation, translation, reflection or permutation of equivalent atomic species.

The SOAP descriptor formalism[97] is based on representing atomic environments by a scalar field centred on atom a, composed of Gaussian functions

$$\rho_a(\mathbf{r}) = \sum_{j \in a} \exp\left(-\frac{|\mathbf{r}-\mathbf{r}_j|^2}{2\sigma^2}\right) |s_j\rangle f_{\text{cut}}(r_j) \tag{2.13}$$

where the sum is performed over the neighbours j of atom a that are situated within a spatial cutoff. s_j denotes the atom species of atom j, forming specie-dependent basis functions $|s_j\rangle$, which allow distinction of different species within the atomic environment[102]. The cutoff function f_{cut} ensures that neighbouring atoms enter and leave the atomic environment in a smooth fashion. Summing the Gaussian functions representing the neighbouring atoms ensures permutational invariance to the atomic indices within the same species. The atomic density ρ is expanded in a basis formed of spherical harmonic Y_{lm}, and orthogonal radial basis functions $g_n(r)$

$$\rho_a(\mathbf{r}) = \sum_{s \in S} |s\rangle \sum_{n,l,m} c^a_{s,n,l,m} \cdot g_n(r) \cdot Y_{l,m}(\hat{\mathbf{r}}) \tag{2.14}$$

where the first sum is performed over the set of neighbouring species S_n. Invariant features

can be formed from the basis set coefficients by calculating the power spectrum

$$p^{a}_{s,s',n,n',l} = \sum_{m} \left(c^{a}_{s,n,l,m}\right)^{*} c^{a}_{s',n',l,m}$$

which can be shown to be invariant to rotations and reflections of the environment with respect to its central atom. Defined as such, each atomic environment is described by a single power spectrum. In addition, the formalism can be extended to describe molecules or condensed matter structures by averaging the representing density field across the constituent atoms. The basis set coefficients belonging to the same central atom species $\mathscr{S}$ are accumulated as:

$$c^{\mathscr{S}}_{s,n,l,m} = \sum_{s_a \in \mathscr{S}} c^{a}_{s',n',l,m} \tag{2.15}$$

which can be used to form a set of power spectrum components $\mathbf{p}^{\mathscr{S}}$ for each distinct atom species within the structure. In order to reduce the complexity of the descriptors, it is also possible to sum all individual coefficients regardless of the central atom species information, although this leads to a loss of information.

SOAPs are not the only way to generate rotationally invariant molecular descriptors [103, 104], however, some other rotationally invariant descriptors may be unsuitable to represent a heterogeneous dataset for ML purposes. For example, it is very challenging to perform ML with varying descriptor lengths without information being lost. When using SOAPs for ML it is possible to ensure the generated descriptor length does not scale with the number of atoms in the system leading to a uniform length descriptor vector for every element in the dataset. This is not the case for some other structural descriptors whereby they scale in length with the number of atoms in the molecule [90, 86]. The main drawback of SOAP descriptors is the potentially large computational cost, as the length $\mathscr{L}$ of their power spectrum can be written as $\mathscr{L} = \frac{1}{2} n_{max} S_n (n_{max} S_n + 1)(l_{max} + 1)$, where S_n is the number of neighbour species, n_{max} is the number of radial basis functions and l_{max} is the number of angular basis functions. In fitting ML models based on SOAP descriptors it is common to incur another factor proportional to S_n if a different model is defined for each centre species. This can lead, depending on the number of species used as centres and neighbors, to extremely large descriptor vectors which can be a challenge to compute due to the large amounts of computer memory required. As we show in Sec. 5 however, it is possible to compress these vectors with a relatively small decrease in predictive performance.

SOAPs work by using a series of orthonormal radial and angular basis functions to expand the local neighbourhood density around each atom. An individual expansion is used for each species of atom in the neighbourhood. In this paper we attempt to maximise

the predictive capabilities of SOAPs by optimising the following parameters that are stipulated when generating SOAPs:

- n_{max} - The number of radial basis functions g_n.
- l_{max} - The maximum degree of the spherical harmonics Y_{lm}.
- *cutoff* - The cutoff distance for the basis function (Å)
- *atom_sigma* - The Gaussian smearing width of atom density σ (Å).
- *centres* - The atomic species used as centres for the basis functions.
- *neighbours* - The atomic species used as neighbours for the basis function.

The optimisation of these parameters is no easy feat, particularly when dealing with heterogeneous datasets. It is not obvious which sets of parameters will work when working with datasets that contain diverse molecular structures or models characterised by a variety of atomic species or environments. Initially, it may seem intuitive to simply use trial-and-error or even an exhaustive grid search strategy to optimise these parameters, however due to the large computational costs of generating SOAP descriptors, these methods are rather inefficient. A number of approaches have been proposed in the last few years to optimise the performance of the SOAP descriptor [101, 105, 106].

In this work, we have leveraged genetic algorithms (GAs) [107, 108] in order to optimise the above mentioned SOAP parameters for one or multiple SOAP descriptors - given a certain choice of *centres* and *neighbours*. This idea is explored in significant detail in chapter 5

2.3.6 Solid-state descriptors

2.3.7 Calculated T_g

One of the most important descriptors we have generated in this work is the calculated T_g. Since we do not have data on the T_g for all molecules in our dataset, this descriptor has a practical use in our MD simulations and is an instrumental part of generating the amorphous phase used to calculate all of the dynamic descriptors in the following section. More information on the specifics of how and why this value is used in the simulations can be found in section 2.2.

To understand how the glass transition temperature is calculated using MD simulations it is first important to understand that, in most cases, when a system undergoes a

phase transition, the particles in the system fundamentally change their arrangement. This typically leads to a change in the volume of the system. For example, when a liquid transforms into a solid, the particles in the system generally become more ordered and densely packed, this results in an overall decrease in the volume of the system. There are exceptions to this, for example the process of water turning into ice is somewhat unique in this regard. This is because the hydrogen bonds between water molecules in the solid phase are arranged in a more open structure than in the liquid phase, this leads to a decrease in density and inversely an increase in volume. For the systems that we are concerned with though, a phase transition to the solid phase results in a decrease in volume. The arrangement of the particles in an amorphous solid lacks long-range order [109], and they are often packed closer together than in a crystalline solid [110]. As a result, when an amorphous phase transition takes place, the volume of the system typically decreases.

During our MD simulations, a barostat is used to enforce the pressure at each timestep. The formula for the barostat that we use is as follows:

$$P = \rho\, k_b\, T + \frac{1}{DV}\left\langle \sum_{i<j} \mathbf{f}(\mathbf{r_{ij}}) \cdot \mathbf{r_{ij}} \right\rangle \tag{2.16}$$

where P = pressure, ρ = density, K_b = Boltzmann's constant, T = temperature, D = dimensionality, V = volume, $\mathbf{f}(\mathbf{r_{ij}})$ = force between atoms i and j. Since we are using the isobaric-isothermal ensemble, the simulation attempts to keep the pressure constant. By equation 2.16 this can be done by adjusting V.

So when the system undergoes an amorphous phase transition by moving above/below the T_g on heating/cooling, the volume of the system has to change to accommodate the change in density. We can use this to estimate the T_g by finding the average temperature of where the phase transition appears to occur. This process is depicted in figure 2.5 e).

2.3.8 Pair correlation function

The pair correlation function (PCF) denoted as $g(r)$ is related to the probability of finding an atom at a given distance (r) from another atom in the system. This framework gives us an idea of the order of a structure over a given distance and can be used to describe both the short range and long range structure of a system.

For our data, the $g(r)$ was calculated using the following algorithm:

The algorithm for calculating the pair correlation function (PCF) involves the following steps. Firstly, the range of interest is defined as `R` and divided into sections of thickness `dr`. Each of these sections becomes the particular value of `r` that we will be analyzing. For

Algorithm 1 Pair Correlation Function $g(r)$

Parameters:

- `R` – The range you wish to search, in our case we set the range to be between 0 and half the system size.
- `r` – The particular distance to place the shell and search for atoms within.
- `N` – The number of atoms in the system.
- `dr` – The thickness of the 'shell'. For example, when $r = 1$ Å, and $dr = 0.25$ Å, we will be considering the atoms that are between 1-1.25 Å away from each other.

1: Divide `R` into sections of distance `dr`. This will give you `R/dr` sections and these sections become your `r` values.
2: **for** each value of `r` in the range `R` **do**
3: **for** each atom in the system **do**
4: Count the number of other atoms that are a distance d away where $\mathtt{r} \leq d \leq \mathtt{r}+\mathtt{dr}$
5: **end for**
6: Divide the total count by `N`
7: Normalise this value by dividing by the volume of the spherical shell, $4\pi\mathtt{r}^2\mathtt{dr}$
8: Normalise with respect to density by dividing by $\rho \times \frac{4}{3}\pi(\mathtt{r}+\mathtt{dr})^3 - \rho \times \frac{4}{3}\pi(\mathtt{r})^3$, where ρ is the number density of atoms in the system.
9: **end for**

each `r`, the number of atoms within a shell of thickness `dr` centered at `r` is counted. This is done for every atom in the system.

Once the count is complete, it is divided by the total number of atoms in the system, `N`. Next, the resulting value is normalized by dividing it by the volume of the spherical shell, $4\pi\mathtt{r}^2\mathtt{dr}$. Finally, to account for the density variation in different parts of the system, the value is further normalized by dividing it by $\rho \times \frac{4}{3}\pi(\mathtt{r}+\mathtt{dr})^3 - \rho \times \frac{4}{3}\pi(\mathtt{r})^3$, where ρ is the number density of atoms in the system. This gives the normalized PCF value at the distance `r`.

In amorphous drugs, the PCF provides insight into the extent of molecular ordering and packing in the system. For example, the presence of distinct peaks in the PCF indicates the presence of well-defined intermolecular distances and a degree of structural ordering. On the other hand, a featureless PCF suggests that the molecules are more randomly distributed and the system is less ordered. The stability of amorphous drugs is influenced by the degree of molecular ordering and packing, as disordered systems are more prone to relaxation and crystallization over time. Therefore, the PCF can be used as a tool to investigate the relationship between molecular structure and the stability of amorphous drugs, and to guide the development of more stable amorphous formulations. We note that in many cases

we look at the centre of mass to calculate descriptors such as the PCF.

After the MD simulation protocol outlined in this chapter has been completed, we are left with a huge amount of data that we can analyse to learn about how the molecules in our dataset behave in the amorphous phase. In this section we go through the various descriptors that we used to extract information from the output trajectories of these simulations.

2.3.9 Mean squared displacement

The mean squared displacement (MSD) is a measure of how far a particle has moved from its starting position over time. Although we do not use the MSD directly as a descriptor, we use it to calculate the diffusion coefficient (see section 2.3.10) as well as verifying that the simulations are behaving as expected.

The MSD is an ensemble average of displacements, with the MSD at time t being given by

$$MSD = \left\langle |r(t) - r(0)|^2 \right\rangle = \frac{1}{N} \sum_{i=1}^{N} |r_i(t) - r_i(0)|^2 \tag{2.17}$$

Where N is the number of particles and $r_i(t)$ is the position of particle i at time t. This is a relatively easy quantity to calculate and is computationally fairly cheap. It is important to note that the MSD can be affected by boundary conditions of the simulation.

In MD simulations, periodic boundary conditions are frequently employed to make a small system behave as though it were a component of a larger, repeating unit. This enables the simulation to simulate a small portion of an effectively infinite system. Particles that go past the simulation box's edges are wrapped around to the other side, creating the impression that they are still inside the simulation box. This generally does not cause any issues for a crystal given that it is identical to itself. However, this is not the case for amorphous systems, and as a result, this wrapping can cause problems when calculating the MSD. Therefore it is necessary to 'unwrap' the system. This unwrapping allows us to observe the particles as they would behave in an infinite system, free of the wrapping effects of the periodic boundary conditions. This is achieved by modifying the particles positions to represent their true positions outside the simulation box. This generally entails adding or subtracting a multiple of the simulation box dimensions to the coordinates of the particle.

2.3.10 Diffusion coefficient

Brownian motion is the random movement of particles in a fluid due to colliding with other particles or molecules. This is often modeled as a random walk whereby collisions cause

Algorithm 2 Ensemble Mean Squared Displacement

Parameters:

- `N` – The number of particles in the system.
- `M` – The number of time averages to be calculated.
- Δt – The time lags of the simulation. e.g a simulation with timesteps of 1ps would have time lags of 1ps, 2ps, 3ps, ...

```
for each time average do
    for each Δt you wish to calculate the MSD for do
        for each particle in the system do
            Record the position of the particle at time t = 0
            Calculate and then square the displacement of the particle at time t + Δt
            with respect to its position at t = 0
        end for
        Sum the squared displacement over all N particles in the system
    end for
    Average the summed squared displacement over all Δt
end for
```

particles to change direction. Diffusion coefficient is a measure of how quickly the particles in a fluid diffuse under this Brownian motion. In the general case, diffusion is defined as the process whereby particles that are suspended in a fluid will move around due to the thermal motion of the fluid

During his studies of Brownian motion, Einstein showed that for a particle following a random walk, the MSD of the particle grows linearly with the elapsed time. The diffusion coefficient D, is defined as the constant of proportionality between the MSD of a particle that exhibits brownian motion, and time and is given by the following equation.

$$\text{MSD}(t) = 2d\,D\,t \tag{2.18}$$

Where MSD(t) is the Mean squared displacement at time t, and d is the dimensionality (3 in our case).

If we treat the movement of particles in our simulation as following a random walk, we can quite easily use this equation to get a value for the diffusion coefficient. Taking the slope of the linear portion of the MSD when plotted against t then dividing by 6 ($2 * d$) gives a value for the diffusion coefficient. It is important that we only use the MSD values that grow linearly with time though, since these can be accurately represented as a random walk. At the beginning of the simulations, this is often not the case. Since particles initially are heading in a straight trajectory, it takes some time before they collide with other particles.

The particles colliding causes them to change direction and allow us to treat the MSD as a random walk, since there are less collisions during this time, we can not do this. This part of the MSD plot is often known as the ballistic regime, a schematic showing this ballistic regime is given in figure 2.6.

It is also important to note that at large timescales, towards the end of the simulation, the data can often be averaged poorly. Due to the fact the time lags are large, we therefore have less data and so our results lose accuracy when the average is calculated. In the context of calculating the MSD from MD simulations, the choice of time lag determines the number of displacement values that can be obtained for each molecule.

For example, suppose a 500 ps MD simulation is performed, and the MSD is calculated using a time lag of 500 ps. In this case, for each molecule, only one value for how far the particle has been displaced in 500 ps is obtained. However, if a time lag of 1 ps is used, then there are 499 displacement values available to track how far the particle has moved over 499 different time intervals.

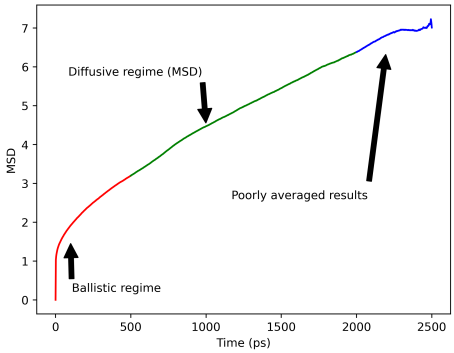

Figure 2.6: An example of a typical MSD plot. This was generated using the trajectories of Aceclofenac at 110K above the estimated T_g. The red portion of the MSD is the ballistic regime where the MSD does not scale linearly with time. The blue portion is the poorly averaged long lag time MSD. The green section is the linear portion that is used to calculate the gradient.

The choice of time lag affects the resolution of the MSD data, with smaller time lags providing more detailed information on the particle's movement over shorter time intervals. However, smaller time lags require longer simulation times to obtain statistically significant results. Therefore, the choice of time lag must be carefully balanced with the computational resources available and the desired level of resolution in the MSD data.

As we have non-linearity at the short and long timescales, we elect to ignore the first 500 ps and last 500 ps of the MSD when calculating the slope of the linear portion. These various sections of the MSD are illustrated in figure 2.6. Convergence of the MSD in MD simulations is determined by observing a plateau or steady-state behavior in the MSD vs. Time plot, as well as considering the variability of the MSD within that region. Longer simulation times and larger system

sizes often facilitate faster convergence.

The diffusion coefficient can be a useful tool to gather information about the T_g of a molecule as it gives us an insight into the transport properties and dynamics of the molecule. Since the molecular mobility decreases with temperature, so does D, and at temperatures below the glass transition temperature, D drops dramatically as the material begins to behave as a solid. In a heuristic sense, we are seeing how much the molecules in our system 'spread out' over a given time. The rationale behind using D as a descriptor is that it gives us some information about how much the system is moving about and thus how close it is to crystallising/forming a glass.

2.3.11 Van-Hove correlation function

Named after the American physicist Leland Van Hove, the Van Hove function describes the probability distribution of finding a particle at a certain distance away from its origin at a given time [111]. In other words, if a particle is at position $r(0)$ at time $t = 0$, then the VHF $G(r,t)$ gives us the probability that the particle is at position $r(0)+r$ at time t. Note that this is a more generalised form of the pair correlation function described in section 2.3.8.

Similarly to how we used the MSD to easily calculate other descriptors, we leverage the Van Hove function to more easily calculate other descriptors in this work.

The VHF can be split into two parts, the self part G_s, and the distinct part G_d

$$G(\mathbf{r},t) = G_s(\mathbf{r},t)+G_d(\mathbf{r},t) \tag{2.19}$$

$$G_s(\mathbf{r},t) = \frac{1}{N}\left\langle \sum_{i=1}^{N} \delta(\mathbf{r}+\mathbf{r}_i(0)-\mathbf{r}_i(t) \right\rangle \tag{2.20}$$

$$G_d(\mathbf{r},t) = \frac{1}{N}\left\langle \sum_{i\neq 1}^{N} \delta(\mathbf{r}+\mathbf{r}_j(0)-\mathbf{r}_j(t) \right\rangle \tag{2.21}$$

Where $t > 0$ and δ is the delta function defined as

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

By splitting the VHF into two parts, we now have the self part that shows us the probability density of a particle moving a distance r in time t, and the distinct part that gives us the probabilty that there is a particle a distance r at time t from a position where there was a different particle at time $t = 0$. Interestingly, the VHF can also be related to the MSD of a

system using the following equation

$$G(\mathbf{r},t)=\frac{1}{4\pi r^2}\frac{\partial}{\partial r}\mathrm{MSD}(t) \tag{2.22}$$

2.3.12 Intermediate scattering function and structural relaxation time

The Intermediate Scattering Function (ISF) is a function that describes how much radiation is scattered in a scattering experiment as a function of time. Although we do not use the ISF as a descriptor for ML, it is instrumental in our work in chapter 4, so we include a brief overview of it here.

The ISF measures the correlation of particle positions as a function of time and wave vector. In other words, it gives the probability that at a certain wave vector, $\mathbf{q}$, and time t, a particle will scatter. The ISF is defined as:

$$F(q,t)=\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\langle\exp\left(i\mathbf{q}\cdot[\mathbf{r}_j(t)-\mathbf{r}_k(0)]\right)\rangle \tag{2.23}$$

where $F(q,t)$ is the ISF, N is the total number of particles in the system, $\mathbf{r}_j(t)$ is the position of particle j at time t, $\mathbf{r}_k(0)$ is the position of particle k at time zero, $\mathbf{q}$ is the wave vector of the incident radiation or the scattering vector in the reciprocal space, and the brackets denote the ensemble average.

The ISF is related to some of the previously mentioned descriptors. It can be calculated by taking the Fourier transform of the Van Hove function, and it can also be used to calculate the self-diffusion coefficient and relaxation time.

The structural relaxation time τ is a measure of the timescale required for the structure of the system to evolve to a "significant" extent. In the context of the dynamics of strongly supercooled liquids and glasses, we can identify three separate relaxation timescales: 1. the ballistic regime, observed for very short timescales within which molecules do not have time to even collide with each other; 2. the cage-motion/rattling regime, which is absent in the hydrodynamic regime (it is a hallmark of supercooled liquids and to an even greater extent of glassy systems) and corresponds to the timescale involved with the "rattling" of the molecules within the "cages" formed by their neighbors. The relevant timescale is often indicated as the β-relaxation of the system; 3. the α-relaxation regime, which corresponds to a significant structural change of the system due to the molecules leaving their "cages". These three timescales can all be probed by means of the (self part of the) intermediate scattering function (equation 2.23)

We have explicitly verified this is the case by computing Eq. 2.23 by choosing $\mathbf{q}$ vec-

tors along each Cartesian direction. Note that the intermediate scattering function we obtained from our MD simulations can in principle be directly compared to experimental results from, e.g., inelastic neutron or X-ray scattering measurements. This is done in chapter 4.

As illustrated in Fig. 2.5f, for a liquid in its hydrodynamic regime (see e.g., the data re: T = T_g + 150 K in Fig. 2.5f), there is no β-relaxation regime, and $F_s(q,t)$ decays smoothly from one to zero via a single exponential decay. For a supercooled liquid, however, the emergence of the cage-rattling motion results in a characteristic plateau of the $F_s(q,t)$ (see e.g., the data re: T = T_g + 30 K in Fig. 2.5f), which only decays to zero from the onset of the α-relaxation regime. To extract a single metric that reflects the timescale associated with the onset of the α-relaxation regime, a common choice we have also adopted in this work is that of choosing as the structural relaxation time, τ, the time for which $F_s(q,t)$ is equal to $1/e$ = 0.368. Longer relaxation times are indicative of a dynamics, which in turn are characteristic of a lower propensity for the drug molecules in either the supercooled liquid or the glass to crystallise. τ must be related to D in some fashion, albeit τ is a much more robust quantity for a supercooled/glassy system, where not just one D exists. In fact, one can define a D characteristic to each of the time scales we have discussed. In this work, we have attempted to focus on the D associated with the α-relaxation regime, which should be directly correlated with τ. Note that, below and/or in proximity of T_g, τ might be longer than the extent of our MD simulations (see e.g., the data re: T = T_g -10 K in Fig. 2.5f).

2.3.13 Velocity autocorrelation

The velocity autocorrelation function (VACF) is not directly used to calculate any descriptors in this work, however, it is used in chapter 4 to generate representations of the TBOS frequencies

Velocity autocorrelation (VAC) is a mathematical tool used in the study of molecular dynamics to analyze the fluctuations of a particle's velocity over time. VAC can be used to determine the properties of a system by examining the correlation of a particle's velocity at different times. The VAC function is defined as the time-dependent correlation between the velocity of a particle at time t and its velocity at time $t+\tau$.

The VAC function can be expressed mathematically as follows:

$$C(\tau) = \frac{1}{N}\sum_{i=1}^{N}\langle v_i(t)v_i(t+\delta\tau)\rangle \tag{2.24}$$

where N is the number of particles, $v_i(t)$ is the velocity of the ith particle at time t,

and <...> denotes an ensemble average over all particles in the system. The VAC function is often plotted as a function of time lag ($\delta\tau$) and provides information about the dynamics of the system.

In the case of amorphous drugs, the VAC function can provide insight into the stability of the drug. Amorphous drugs are metastable and tend to undergo relaxation processes over time that can lead to degradation and loss of efficacy. The VAC function can be used to determine the time scale of these relaxation processes and to identify any changes in the dynamics of the system that may indicate degradation or instability.

2.4 Optimisation

Now that the descriptors used in this work have been established, in this section we talk about how we have attempted to improve the quality of predictions using the descriptors. We have used optimisation techniques as well as synthetic data generation in an attempt to get the best possible performance out of the descriptors.

2.4.1 Genetic algorithm

A genetic algorithm (GA) has been developed to optimise the hyperparameters of the SOAP descriptor (section 2.3.5). This descriptor has a large hyperparameter space and it is computationally expensive to compute, so it is essential that our optimisation technique converges quickly.

There are many different optimisation techniques but we decided to focus mainly on a genetic algorithm. The reason for this is twofold, firstly, a GA is highly customisable, we can easily change population sizes, convergence criteria, number of generations etc. to suit our needs. This is particularly useful when dealing with large systems where the SOAPS are very computationally expensive to generate. Secondly, the GA is easily parallelisable, again this is invaluable when computational expense is an issue.

Since the objective of this work is to create a transferable method of getting accurate predictions for properties related to recrystallisation for amorphous drugs, the GA was intended to work 'out the box' and be as general as possible without any prior knowledge of the molecules in the database. We have demonstrated that this was successful, as well as thoroughly explained how the GA works and the kind of results that can be expected from it in section 5.

2.4.2 Synthetic data generation

An issue that we faced from the beginning of this project was a lack of data. This is a common problem that is often tackled by generating synthetic data. This can be a fairly complicated process and there are many techniques available to complete this task such as variational autoencoders [112] and generative adversarial networks [113].

Generating synthetic data using these advanced techniques can have its pitfalls. Since the data is often generated using a neural network, it is hard to eliminate bias as a result of the underlying real data, and also it is a challenge to know whether poor final results are an artifact of poor data generation.

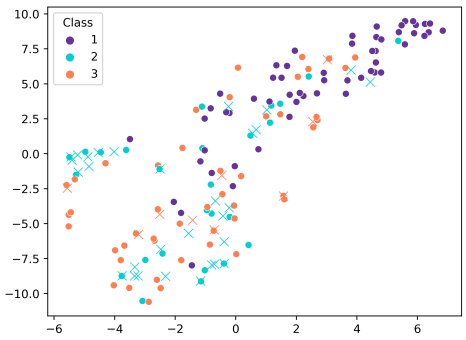

Figure 2.7: A representation of how SMOTE can be used to generate synthetic data. The descriptor used is a 'vanilla' symmetry function that has been visualised using t-SNE dimensionality reduction [114]. The dots represent real data in our dataset, while the crosses are datapoints generated using SMOTE. The units are arbitrary.

One prevalent challenge when working with molecules is determining whether synthetically generated molecules are physically viable and quantifying how challenging they would be to synthesise in a lab. Of course solutions exist for this problem too, such as the G-SchNet software [115], but since these methods are also based on learning the underlying distribution via neural networks it suffers many of the same problems.

It is clear that the process of generating reliable synthetic data using machine learning is a complex and arduous one. Thus, due to time constraints, we elected to use a simpler method that does not rely on learning any underlying distribution and is computationally inexpensive.

The Synthetic Minority Over-sampling Technique (SMOTE) is a method for generating synthetic data in a relatively simple way for classification datasets [116]. This method does not learn any underlying distribution and it does not verify that the generated data is physically viable, however, we argue that this is not an issue. This is because SMOTE is used on individual descriptors instead of the underlying dataset and it generates artificial descriptors as opposed to artificial molecules. For this reason it does not matter if a physically viable molecule can be

reconstructed using this descriptor. We simply have a descriptor for a molecule in a given class.

SMOTE works on a class by class basis. It chooses two descriptors from the same class and draws a line between those descriptors in feature space. A point along that line is then randomly selected and that point is mapped to a descriptor vector. This is a simple and straightforward approach to data generation, and although perhaps not as robust as other methods, it has been responsible for significant improvements in our ML results.

An illustration of how the synthetic data fits in with the real data is shown in figure 2.7. There are some caveats with using data generated with SMOTE. Firstly, SMOTE is only intended to equalise how many datapoints are in each class, it is not intended to create new datapoints. This is why no new class 1 descriptors were created in figure 2.7.

Secondly, data generated using SMOTE is obviously not independent of the underlying data. So we must be very careful when training models on this data. When it comes to machine learning it is recommended that a validation set of data is held aside and not used to generate new descriptors using SMOTE. If this is not done, it is possible that the training data can contain a significant amount of information about the test data leading to inaccurate results.

Finally, as we mentioned previously, SMOTE is not necessarily a very robust algorithm. SMOTE-generated datapoints, meant to alleviate class imbalance, have been created within the class 1 cluster but end up as outliers themselves. While it can be argued that predictive models should be robust enough to handle outliers in real-world data, this outcome raises concerns about the effectiveness of SMOTE.

This occurrence is not an isolated incident; it is a recurring challenge when employing SMOTE. The algorithm's tendency to produce synthetic instances that deviate significantly from the original data distribution can have adverse consequences. These artificially created outliers might mislead the learning process and hinder model generalization.

Researchers and practitioners should be acutely aware of these shortcomings when employing SMOTE and should carefully assess whether its use is appropriate for their specific dataset and problem domain. In some cases, alternative techniques or more robust oversampling methods may be more suitable to mitigate class imbalance without introducing such pronounced deviations from the underlying data distribution. In conclusion, while SMOTE can be a valuable tool, its limitations regarding outlier sensitivity and their impact on model performance must not be underestimated.

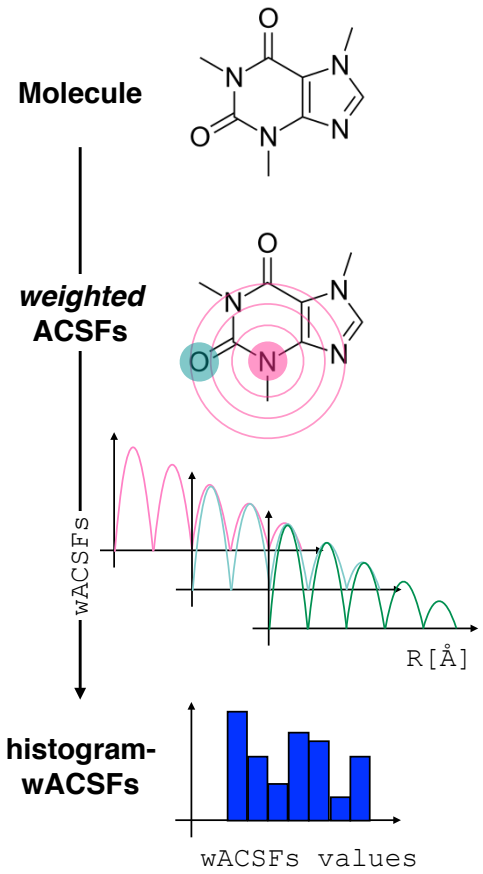

Figure 2.4: Constructing the H-wACSFs descriptor. A three-dimensional conformer (ideally, an ensemble of them) has to be generated for each molecule. Then, in line with the work of Behler [89], radial and angular symmetry functions are computed by sitting on each atom within the molecule and calculating the value of (usually Gaussian) functions that depends on either $r_{ij} = |\bar{r}_j - \bar{r}_i|$ distances (radial SFs) or θ_{ijk} angles (angular SFs) between pairs or triplets of atoms - up to a certain cutoff radius R_c. In principle, different sets of symmetry functions are needed for each combination of elements in a given molecule. Gastegger *et al.* have recently [90] introduced a weighting scheme that substantially reduces the number of functions needed to encode the structure of multi component systems such as drug-like molecules. As molecules with different number of atoms and or elements are characterised by different number of symmetry functions, we regularise these features by building histograms of weighted atomic symmetry functions. Each molecule can then be represented by a vector with as many elements as the bins chosen to build said histogram: low and higher number of bins thus provide more or less coarse-grained representations of the molecular structure. Note that this descriptor can straightforwardly applied to three-dimensional models of crystalline or amorphous drugs.

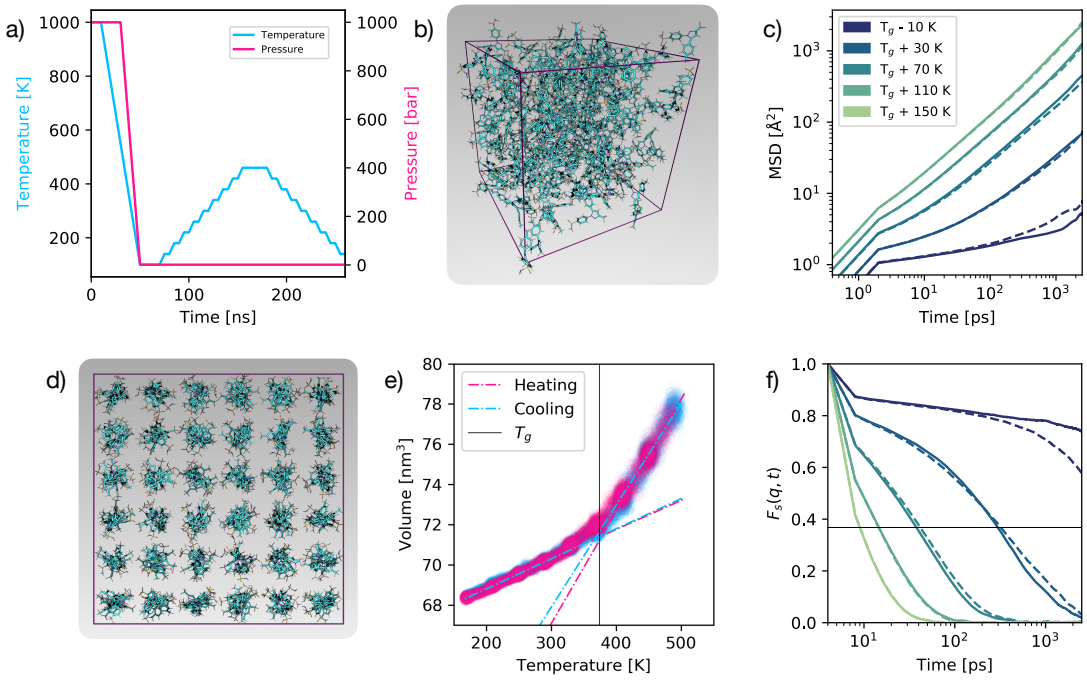

Figure 2.5: *Molecular dynamics simulations.* a) Computational protocol used to generate the amorphous phases of each molecule in the Amo-Reg dataset (see section 2.2). b) A representative snapshot of an amorphous phase (for celecoxib) from a MD trajectory at 100 K. c) Mean-squared displacement (MSD see text) for chrysin as a function of temperature - with respect to T_g. Solid and dashed lines refer to heating and cooling ramps, respectively. d) Initial configuration for the celecoxib system. e) Volume V as a function of temperature T for chrysin, reported along either the heating and cooling ramp. T_g can be estimated as the temperature at which the two $V(T)$ lines (obtained via linear regression, also shown) cross (see section 2.3.7) f) Self part of the intermediate scattering function (SISF, see section 2.3.12), for chrysin as a function of temperature - with respect to T_g. Solid and dashed lines refer to heating and cooling ramps, respectively. The intersection between each curve and the horizontal line (at $y = 1/e$) marks the structural relaxation time τ for the system.

Chapter 3

Machine Learning

3.1 Machine learning

By calculating the single-molecule and solid-state descriptors described in the previous chapter gives us several ways to numerically represent the physical properties as well as the behaviour of molecules in our datasets. In this chapter we outline the various machine learning techniques that we use to find patterns in these numerical representations that we can use to make predictions about the properties of interest.

There are many different flavours of ML algorithm that are available for us to use, and after careful consideration, the two types of model that we use are neural networks (NNs) and random forests (RFs) the justification for this choice is discussed in section 3.2.3.

The machine learning in this work was computed using Tensorflow 2.9.0 on Dell PowerEdge C6420 compute nodes each with 2 x Intel Xeon Platinum 826 (Cascade Lake) 2.9 GHz 24-core processors with 48 cores per node, 192 GB DDR4-2933 RAM per node, and 4 GB RAM per core.

3.2 An overview of NNs and RFs

3.2.1 Neural networks

Neural networks are arguably the most popular type of machine learning algorithm model. They are modeled after the structure of the human brain and composed of connected 'neurons' known as nodes that process and pass on information. Each node receives an input signal from other nodes and then processes that signal using an activation function. This activation function determines the output of the node that is passed on and propagated

through the network.

The nodes are connected to each other, much like how neurons are connected by synapses. These connections in a neuron are weighted, meaning that the connection between certain nodes can be adjusted to be stronger or weaker, making the output of that node more or less significant. The weight of these connections is optimised when the neural network is trained. In this work we are only concerned with a specific type of machine learning known as supervised machine learning, where the correct result for each input is known. In a supervised neural network, information is passed through the network with initially random weights and the output is compared against the true value. The error between the predicted and actual value is evaluated using a cost function, and the goal of the NN is to minimise this cost function. At each iteration of the neural network, the cost function is calculated and the error is propagated backwards throughout the network. The gradient of the cost function with respect to the weights of each neuron is calculated and used to update the weights to minimise the cost. This process is known as backpropagation. Backpropagation is repeated a number of times (known as epochs) until either a certain number of iterations have elapsed, or the cost function has converged.

There are many parameters of the neural network that can be fine-tuned such as the activation functions of nodes, the cost function, the method to update the weights of the nodes, and the number of nodes. In this chapter we explain how these parameters were chosen and the processes used to optimise them.

3.2.2 Random forests

In order to understand random forests, one must first have an understanding of what a decision tree is. A decision tree is a ML algorithm that builds a tree-like structure where each node represents a feature and each branch of the tree represents a possible value (or set of values) for that feature. At each node, the decision tree algorithm splits the data into subsets that are as 'pure' as possible. The purity can be determined using a number of different methods, the most common is the Gini coefficient (see equation 3.6. The process of splitting the data into subsets continues until some stopping criteria are met, such as a pre-determined maximum depth is achieved, or there are a minimum number of instances in each leaf node (a leaf node is a node without any branches).

Once the decision tree is built, a feature vector can be passed into it and it will traverse the tree based on its values. The value (or class) associated with the leaf node that it ends up in is the output of the tree.

A random forest is simply a collection of decision trees where the output is the av-

erage result from traversing various different trees. The trees in a random forest are trained using a process called bagging that is described in section 3.9.1.

3.2.3 The choice of ML algorithm

Neural networks and random forests are both suitable for regression and classification tasks. The reason we chose to use these two algorithms in particular is because the strengths and weaknesses of each algorithm can compliment each other, and combining them can result in a more effective and robust model.

Neural Networks strengths lie in their ability to learn complex nonlinear relationships in the data. This is because they are more flexible than random forests as RFs are limited by simple decision based rules. Neural networks also are more robust to noisy data than random forests. This is because NNs can learn to ignore noisy features as well as learning the underlying pattern of the noise in order to smooth it out. Random forests are susceptible to noise, especially if the noise is structured in such a way that it affects the decision boundaries of the algorithm.

NNs do have some weaknesses however, they can be computationally expensive to train. Random forests are easily parallelisable so they can be much quicker to train using multi-processing. Also, one of the biggest criticisms of neural networks is that they are a 'black box' in that it is given an input and produces an output but it is very unclear how it arrived at that output and which features had an effect on its decision. This is not the case with random forests as it is easy to see the decision boundaries for each feature and we can extract the importance of each feature based on how much it reduces the impurity of the decision trees.

3.3 Model parameters for NNs

The design and implementation of neural network models requires careful consideration of several parameters, including the number and size of the layers, the activation functions used in each layer, the learning rate, and the optimization algorithm used to train the network. Each of these parameters plays a critical role in determining the accuracy and performance of the neural network model, and must be chosen carefully to ensure optimal results.

In this section, we provide a comprehensive overview of each of these parameters, discussing their individual roles and how they interact with one another to affect the performance of the neural network model. We also provide examples and best practices for

selecting and tuning these parameters to achieve optimal performance.

3.3.1 Neural network architecture

The network architecture describes the number of nodes and hidden layers in the network. Unlike the previous parameters we have mentioned, we cannot simply choose a network architecture that reasonably works with all of our different types of features.

In order to determine which architecture to use, we must first understand the relationship between the number of layers/nodes and the performance of the neural network. The capacity of a neural network is a measure of how well the model can fit to a wide variety of functions. A high capacity will be able to fit to a large number of functions but will be at risk of overfitting to the training data. This is a cause of high variance and low bias. On the other side of the spectrum, if a model's capacity is too low, it will not fit the data very well and will suffer from underfitting. The capacity of a network is directly related to its architecture with a larger number of nodes/layers increasing the capacity.

We therefore seek to find a network architecture that is as simple as possible while also able to fit our data. There are many sophisticated techniques that can be used to accomplish this [117] but due to the large number of different features we are working with, we elect to go for a simple, easily generalised method.

It is shown in [118] that theoretically a two layer ReLU neural network with $2N_s + d$ nodes in each layer is capable of perfectly representing a dataset of N_s samples with feature dimensionality d. With this in mind we proceed with the assumption that the optimal number of nodes will between d and $2N_s+d$. To find the number of nodes in this interval that are sufficient for our use, we multiply $2N_s + d$ by a scaling factor k where $k \in \{0.2, 0.4, 0.6, 0.8, 1\}$

The impact of the scaling factor on each descriptor is shown in table 3.1. What we can see from this is that $0.2 < k < 0.8$ seems to give the best result for every descriptor. We also note that the descriptors with a larger dimensionality tend to perform better with a lower scaling factor whereas the lower dimensionality prefer a larger k. Intuitively this makes perfect sense.

A choice needs to be undertaken regarding whether to employ separate scaling factors for each descriptor or to utilize a universal variable k for all descriptors. While the variation in scaling factors doesn't notably alter the majority of descriptors, it's not insignificant. Furthermore, it significantly influences the efficacy of the relaxation time feature. So we conclude that it is sensible to use an individually selected scaling factor for each different descriptor network. This choice is made manually by inspecting table 3.1 and selecting the scaling factor that correlates to the lowest combines test and train MSE.

Descriptor	Nodes	Time per epoch (s)	Average epochs	Average time (s)	Test MSE	Train MSE
Std	191	0.128	176.4	23.467	1448.8	1037.3
	382	0.163	136.4	21.776	1346.7	1009.6
	573	0.228	135.4	30.497	1431.2	975.6
	764	0.572	126.8	58.876	1475.2	991.7
	955	0.818	126.8	84.279	1484.4	982.4
Cliques	27	0.093	284.0	31.306	1567.7	1123.2
	54	0.102	216.8	24.84	1632.7	1071.6
	82	0.104	149.2	19.298	1633.8	1081.4
	109	0.092	177.2	23.213	1398.7	1013.6
	137	0.135	198.4	26.235	1494.4	992.2
H-wACSF	50	0.217	231.6	26.271	1384.1	1076.1
	101	0.155	178.8	23.147	1485	1017.4
	152	0.132	148.6	18.173	1355	1012.6
	203	0.146	147.2	20.123	1513.7	1009.5
	254	0.12	153.6	22.158	1401.4	995.8
SOAP	128	0.132	181.0	21.601	1489.7	1013.9
	257	0.119	147.0	21.806	1381.9	1015.1
	385	0.166	137.6	22.31	1520.6	1021.6
	514	0.16	136.8	26.709	1424.3	965.5
	643	0.418	139.8	43.071	1453.3	983.3
DC	12	0.164	377.0	37.467	1453.3	1130
	24	0.167	233.6	24.248	1533.3	1181.4
	36	0.168	234.8	26.523	1398.9	1070.6
	48	0.165	184.2	20.823	1506.7	1050.1
	61	0.103	212.6	25.224	1539.2	1102.8
RT	12	0.135	326.6	31.808	1618.9	1262.1
	24	0.131	211.2	22.516	1945.6	1177.7
	36	0.189	201.6	22.917	1599.8	1138.7
	48	0.097	298.4	32.569	1323.2	1002.2
	61	0.092	185.0	21.562	1669	1090.5

Table 3.1: The effect the number of nodes has on a two layer ReLU network using the various descriptors at our disposal. DC and RT represent the results obtained for the diffusion coefficient and relaxation time descriptors respectivley. Note that in the case of descriptors with hyperparameters, namely SOAP and H-wACSF, we have used the vanilla parameters described in section 5.3.1 in this analysis

3.3.2 Activation function

The activation function is a function applied to each neuron of a neural network. This function introduces non-linearity to the model and is what gives neural networks their ability to

model complex relationships between the features and targets. The activation function we have used is the Rectified Linear Unit (ReLU) function. This activation function is defined as:

$$f(x) = max(0, x)$$

ReLU is a popular activation function that has the benefits of being more computationally efficient than other popular functions such as the sigmoid function. Since ReLU can return 0, not all neurons are necessarily activated, unlike with the sigmoid function which always returns a value greater than 0. Additionally, since the function is piecewise linear it is inherently easier to optimise than a non-linear function and also does not suffer from the vanishing gradient problem since the value of the node activations are proportional to the gradients [119]. For these reasons, we have used a ReLU activation function for all of the hidden layers in all the neural networks in this work.

When it comes to the output layer, we use a linear activation function for the regression tasks. This is standard practice because the linear activation function can output any value as it is unbounded. For the classification tasks, it is also standard practice to use the softmax function. The reason the softmax function is so ubiquitous for classification tasks is because it is capable of converting the outputs of the neural network into k probablities that sum up to one, where k is the number of classes. The softmax function is also differentiable everywhere which is a requirement for activation functions.

3.3.3 Optimiser

An optimiser is the algorithm that is used to minimise the loss function, this algorithm is responsible for updating the weights of the neural network. The choice of optimiser can have a fairly profound effect on the performance of a neural network. A good optimiser helps prevent overfitting while converging quickly. The optimisation algorithms can be divided into two categories, manual and adaptive. The manual algorithms require the user to set a learning rate which then has to be tuned. A learning rate that is too small will result in a large number of steps until convergence, whereas a learning rate that is too big may never converge. Adaptive algorithms automatically adjust the learning rate based on previous gradients.

For simplicity and to prevent us from having yet another parameter (the learning rate) to tune, we opt to use the Adaptive Moment Estimation (Adam) optimiser [120] since Adam is generally regarded as the best gradient decent optimisation algorithm to use [121]. An argument could be made that stochastic gradient descent (SGD) can generalise better than Adam [122], and while this may be true we show that it does not make any significant

difference

3.3.4 Loss function

A loss function is a mathematical function that measures the difference between predicted values and actual values of a given dataset. The loss function serves as a guide for the model to adjust its parameters in order to reduce the difference between the predicted output and the true output. The goal of the loss function is to minimize the error between the predicted and actual values, and thus, it is also known as an objective function or cost function.

The loss function is a crucial component of any machine learning algorithm, as it defines the optimization problem that the algorithm tries to solve. The choice of loss function depends on the type of problem being addressed and the desired output of the model.

The selection of an appropriate loss function is important because it affects the performance of the model. A poorly chosen loss function can lead to inaccurate predictions and poor performance of the model. Therefore, the choice of loss function should be made carefully, taking into account the specific requirements of the problem being addressed.

For regression, one might use the mean squared logarithmic error for values that are all positive with a long tail. A mean average error loss function can be used to try and mitigate the significance of outliers in the data. The distribution of our regression data is shown in figure 1.3 and by inspection we see that there are no significant outliers and the distribution is roughly normal. For this reason it makes sense to simply use the conventional mean squared error (MSE) as the loss function. The MSE is generally the default choice in the absence of any unusual trends in the data and is given by the formula:

$$\text{MSE} = \frac{1}{N}\sum_{i=0}^{N}\left(y_i - \hat{y}_i\right)^2 \tag{3.1}$$

where y is the models predicted value and $\hat{y}$ is the actual value.

When it comes to classification for neural networks, the categorical cross entropy is ubiquitous so we use this for all output layers in our classification neural networks. The formula for categorical cross entropy is:

$$H(y,\hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.2}$$

where y is a one-hot encoded vector representing the true label, $\hat{y}$ is a vector of probabilities outputted by the model for each class, and N is the number of classes.

3.4 Model parameters for RFs

Here we describe the parameters used to define the RF models. Much like the NN parameters, these can have a large impact on the overall performance of the model. Some examples of how much influence these parameters have can be found in table 3.2.

3.4.1 Number of Trees

The number of trees parameter in a Random Forest is the number of decision trees that are created during the training process. Increasing the number of trees generally improves the performance of the model, but at the cost of increased computational complexity and memory requirements. The optimal number of trees depends on the size of the dataset, the complexity of the problem being addressed, and the available computational resources.

There are several ways to determine the optimal number of trees for a Random Forest. One common approach is to use cross-validation to evaluate the performance of the model for different numbers of trees. Another approach is to monitor the out-of-bag error, which is the error rate of the model on the training data that was not used to build a particular tree. The out-of-bag error provides an estimate of the model's performance on new data and can be used to select the optimal number of trees.

3.4.2 Max depth

The max depth parameter in a Random Forest is a hyperparameter that specifies the maximum depth of each decision tree in the forest. The depth of a decision tree refers to the number of levels in the tree, where each level represents a decision based on one of the input features. A shallow tree has a low maximum depth, while a deeper tree has a higher maximum depth.

Setting the max depth too low can result in underfitting, where the model is too simple and cannot capture the complex relationships in the data. On the other hand, setting the max depth too high can result in overfitting, where the model becomes too specific to the training data and performs poorly on new data.

Choosing the optimal value for the max depth parameter is crucial for achieving good performance in a Random Forest model. In general, the optimal value for max depth depends on the complexity of the problem being addressed, the size of the dataset, and the available computational resources. In some cases, a larger max depth may be necessary to capture complex relationships in the data, while in other cases a smaller max depth may be sufficient.

3.4.3 Min split size

The min split size parameter in a Random Forest is an important hyperparameter that affects the complexity of the decision trees and the overall performance of the model.

During the construction of a decision tree in a Random Forest, the algorithm recursively splits the data based on the feature that provides the highest information gain. The splitting process continues until a stopping criterion is reached, such as when all the data points in a node belong to the same class or when the number of samples in a node falls below a certain threshold.

The min split size parameter specifies the minimum number of samples required in a node to perform a split. If the number of samples in a node falls below the specified threshold, the node is not split further, and the decision tree is considered to have reached its maximum depth. By setting a minimum threshold for the number of samples in a node, the min split size parameter helps to control the complexity of the decision trees and prevent the creation of small sub-trees that may only be effective at fitting noise in the data.

A low value of the min split size parameter allows the algorithm to split the data more aggressively, resulting in more complex decision trees. However, this can also lead to overfitting, where the model performs well on the training data but poorly on new data. On the other hand, a high value of the min split size parameter limits the complexity of the decision trees and can result in underfitting, where the model is too simple and unable to capture the underlying patterns in the data.

3.5 Preprocessing and forms of dimensionality reduction

3.5.1 Normalising the data

Normalising the data is an important preprocessing step in neural networks for a number of reasons:

1. **Prevents vanishing/exploding gradients:** We can prevent the gradients from becoming too small/large during backpropigation by normalising the data. Very small gradients can lead to slow convergence or getting stuck in a local minima, whereas large gradients can cause the model to be unable to converge.

2. **Reduces impact of scales:** Often the input data can have different scales, this is very common in our standard descriptors as they are all measuring different properties and not in the same units. Without normalisation, features that typically have larger values can have more influence over the training of the model. By normalising the

data before training we ensure that each feature has equal contribution to the model and it generalises to new data better.

There are different methods of normalising the most simple method to use min-max scaling so that each feature falls in the range (0,1) where the largest value is scaled to be 1 and the smallest is scaled to be 0. We found that the method of normalisation for the single-molecule descriptors does not have a significant impact on the results and since min-max scaling is robust, easy to interpret, and simple to implement we decided to use it. Note that one of the criticisms of min-max scaling is is that it is sensitive to outliers, so it was important that we checked each feature for outliers before applying the scaler.

In cases where the data spans a large range, min-max scaling is less appropriate, and it is standard practice to use logarithmic scaling. In this technique, a logarithmic function is applied to the data points, which compresses the range of values while preserving the order of magnitude. Logarithmic scaling works by transforming each feature by taking its logarithm.

In Machine Learning, logarithmic scaling is often used to transform skewed or heavily skewed data, as it can help to mitigate the effect of extreme values. Moreover, this technique can be particularly useful when dealing with data that has a wide range of values, as it can help to normalize the data and improve the accuracy of the model.

Parameters	MCC Train	MCC Test
n_est-80 md-4 mss-2	0.726 ± 0.019	0.151
n_est-80 md-4 mss-4	0.715 ± 0.02	0.176
n_est-80 md-32 mss-2	1.0 ± 0.0	0.279

Table 3.2: How the parameters of the random forest impact the results for the cliques descriptor for the Amo-Class dataset. n_est is the number of trees used (see section (3.4.1), md is the Max depth (see section 3.4.2), and mss is the Min split size (see section)

One of the key advantages of logarithmic scaling is that it can help to reduce the impact of outliers, which are data points that are significantly different from the rest of the data. By compressing the range of values, outliers are less likely to have a significant effect on the model, and the resulting predictions are typically more accurate.

3.6 Feature selection

Feature selection in machine learning is the process of selecting a subset of relevant features or variables from a larger set of available features or variables for use in model building. The goal of feature selection is to improve the performance of the machine learning algorithm by reducing the dimensionality of the dataset and removing irrelevant or redundant features.

Feature selection can help to improve the accuracy of a model by reducing overfitting and increasing its generalisation capabilities. It can also help to reduce the computational complexity and training time required for building the model.

In this section we highlight some common methods of feature selection and assess their strengths and weaknesses.

3.6.1 Variance threshold

The first method we used is a simple variance threshold. This can be used for both classification and regression. A variance threshold is the most simple method of feature selection and it removes features where their variance is below some threshold. Generally the threshold is set to 0 so that this simply removes features with constant value as they add no information to the model and only serve to add noise. It is not unusual to use a threshold greater than 0 but one must be careful to normalise the data before doing this if the values of the features measure different types of quantities. Since we seek to create a general workflow for feature selection we simply used a threshold of 0 so that we did not need to take into account the properties of each descriptor individually.

This method of feature selection is particularly useful when dealing with the H-wACSF descriptor. Since a large number of the histogram bins are empty for every molecule in the dataset, they contain no additional information. We can easily remove these redundant features using a variance threshold. With this being said, when we use a variance threshold on the H-wACSF descriptor to move features with no variance, it does not have any noticeable effect on the performance of the model. The changes in MSE for both the training and test set are negligible and much more likely to be attributed to the changes in network architecture based on the different input dimensionality. Nonetheless we still elect to use variance thresholds because even if they do not improve performance, at the very least, they improve computational efficiency.

The likely reason for this lack of improvement in the case of the H-wACSF descriptor is that since we are using the ReLU activation function, the nodes that handle the zero variance features are not being activated and therefore are having no impact on the model. This may not be the case for more complicated ensembles that we discuss in section 3.9 and

in general for ML it is not wise to include redundant features as they may just add noise. So, we elect to use a variance threshold on all of our descriptors. Another reason that we use a variance threshold is that it prevents unexpected results in the other feature selection methods that we use, for example if we have a feature of zero variance it is impossible to define the correlation of that feature with another which causes a problem with the removal of correlated features described in section 3.6.2.

3.6.2 Removal of correlated variables

Correlation is a measure of how changes in one variable relate to changes in the other variable. We can quantify exactly how closely related two variables are by using the Pearsons Correlation Coefficient (PCC). PCC measures the (linear) relationship between two variables, X and Y and is given by the equation:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.3}$$

For obvious reasons we seek to use features that are highly correlated with the target value. Since multicollinearity is a source of noise in regression it is important to identify features that are highly correlated with each other and eliminate this relationship as best as possible. A common technique for identifying correlations is by plotting a colour scaled correlation matrix as is shown in fig 3.1. This plot clearly shows some very strong positive correlations for some of the descriptors. The diffusion coefficient and relaxation time obviously have highly correlated features, this is to be expected. Disappointingly they seem to have almost no correlation with Tg. We also see areas of strong correlation with our standard descriptors, again this is to be expected as things like molecular weight/heavy atom count , and the number of hydrogen bond donors/number of hydrogen bond acceptors are strongly correlated. We also see areas of correlation within the SOAP descriptor and it is much harder to pinpoint exactly why this is the case.

To explore how these correlations can impact our model's performance we look at removing highly correlated features and observing how the model performs. This is done by finding all pairs of features where the absolute value of their PCC is above a given threshold and removing the feature that has the lowest correlation with the target. Since the standard descriptor has the highest number of correlated variables, we use this to illustrate how correlation based feature selection impact the model. The results of this are shown in table 3.3.

We can see from this testing that interestingly there is a drop in performance on the

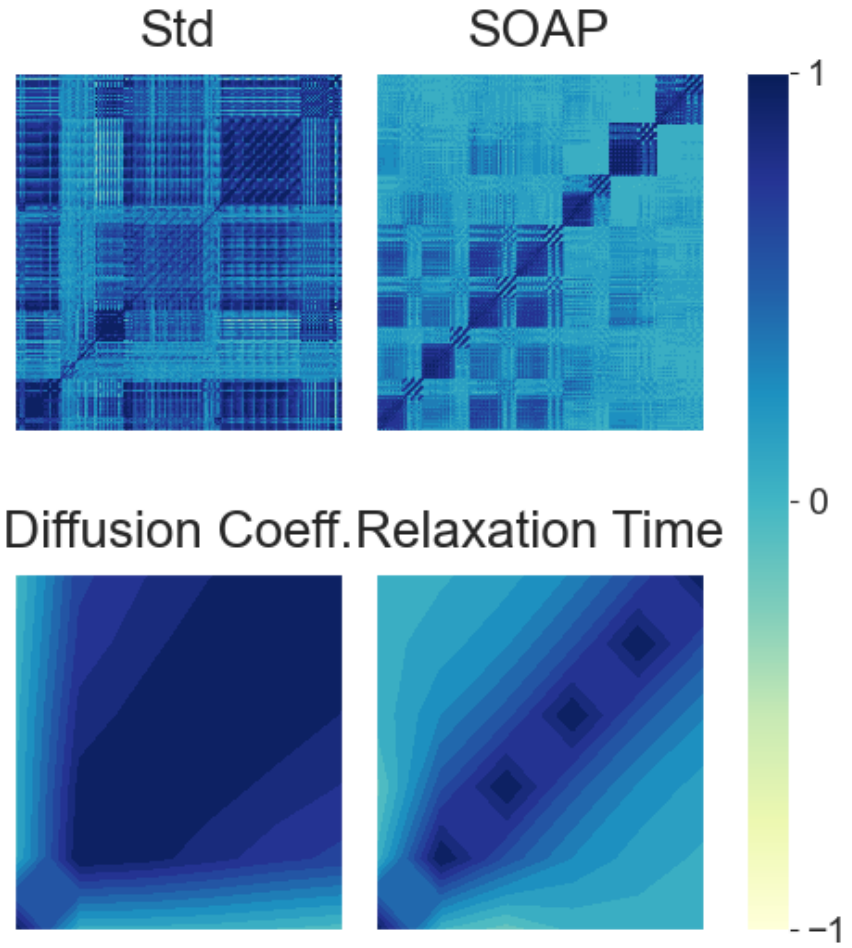

Figure 3.1: Heatmap showing the correlation of features with the Tg and with each other. The Tg is shown in the leftmost column and bottom row. The white space in the H-wACSF plot is an artefact of the histogram binning. Some histograms, particularly for the 'wider' symmetry functions, have bins that are empty for all molecules in the dataset thus resulting in a null value for the PCC. These features are of course removed during the variance threshold feature selection and are only shown in this plot for illustrative purposes.

test dataset when we use a threshold of 0.95 compared to when no features are removed. Using a lower threshold seems to increase the performance up to an optimal threshold of somewhere between 0.8 and 0.85. Decreasing the threshold below 0.8 appears to result in

a performance drop off. A point of interest about these results is the initial large drop in prediction quality when the threshold changes from 1 (no features removed) to 0.95. It is hard to know exactly why this is the case, but the most reasonable explanation is that there are some non-linear relationships between some of the features and the target. For example, if two features were highly correlated with each other and had little correlation with the target, it is likely that one of them would be dropped by the feature selection method. However, it is entirely possible for the NN to find some non-linear relationship between the two features that is correlated with the target. So by removing one of these variables in the feature selection process, we are hindering how well the model can learn to predict the target value.

In order to ensure that this is not happening, we adopt an intuitive approach to selecting the threshold to remove correlated variables. As previously mentioned, in the case of the Std descriptor for the Amo-Reg dataset, we choose a threshold of 0.825. This is the sensible choice since it has reduced the dimensionality of our descriptor vector by 82% while keeping the test MSE relativley similar when compared to no threshold. We perform a similar analysis on each of the descriptors and our results are given in table 3.4. We decided to impose a very limited threshold on the symmetry functions and SOAP descriptor. The reason for this is that since those descriptors are purely structural, we believe there may be some complex non-linear relationships between the variables and, especially in the case of H-wACSFs, there will be a high level of correlation between some of the features due to the nature of the descriptor.

Threshold	Test MSE	Train MSE	Dim.
1.00	1592.340	166.798	901
0.95	1713.828	86.965	423
0.90	1711.653	43.785	307
0.85	1511.507	38.109	220
0.80	1544.859	36.101	168
0.75	1894.682	53.535	129

Table 3.3: Table showing the relationship between the PCC threshold and the MSE when using standard descriptors. The Dim. column shows the dimensionality of the feature vector after features with a correlation above the threshold have been removed. A threshold of 1 means that no features have been removed.

Due to the fact that the H-wASCF values are binned, it is possible that the first bin will have the same number of atoms for almost every molecule, with only a small number of molecules having a different number. This is important information, however it would lead to a large correlation between these two features. For similar reasons we choose to use no threshold on the diffusion coefficient and relaxation time; these descriptors already

have a very low dimensionality and are intrinsically correlated, so it would not make sense to remove correlated variables.

3.6.3 Backwards Feature Elimination

Threshold	Descriptor
0.825	Std
0.9	Cliques
0.98	H-wACSF
0.98	SOAP
1	DC
1	RT

Table 3.4: Table showing the correlation thresholds when performing dimensionality reduction for each of the descriptors.

The next method we will discuss is the backward feature elimination approach. This methodology involves the utilization of all available features initially, and subsequently computes a measure of the individual feature's effectiveness in assessing the target variable. This evaluation is carried out using the MDI (see section 3.6.4), which is determined by the SKlearn RandomForest function. The feature with the lowest score is then eliminated iteratively until a certain stopping criterion is met. In the case of the regression tasks, we stop after 5 iterations with no improvement in the lowest MSE of the test set. For classification tasks we continue until 5 iterations have elapsed with no improvement in the best test set MCC.

It is important to note that this method necessitates the retraining of a new model each time a feature is removed, resulting in a significant computational expense, especially when implementing LOOCV. Despite this, the advantage of this approach lies in its ability to ensure the attainment of the most optimal outcomes due to the iterative nature of the algorithm.

3.6.4 Feature importance

We have explored the possibility of using the intrinsic ability of random forests (RFs) to provide a measure of importance for each individual feature that makes up the descriptor via a measure called the Mean Decrease in Impurity (MDI) [123]. An RF uses an impurity function $i(\tau)$ as a criterion for how to best split the dataset at each node τ such that similar target values will be in the same set [124]. In general, the impurity function for RF regression is the variance [124]; however, for illustrative purposes, we consider the simplest regression problem, one of binary classification, which utilises the Gini impurity function:

$$i(\tau) = 1 - p_1^2 - p_0^2, \tag{3.4}$$

where $p_k = \frac{n_k}{n}$ is the fraction of the n_k samples of class $k = \{0,1\}$ out of n samples at node τ, to measure how well a potential split at each node τ within the binary trees T will separate the data [125]. A decrease in $i(\tau)$ or Δi resulting from a split that sends a sample point to two sub-nodes, τ_l and τ_r, by a threshold t_θ on feature θ is defined as:

$$\Delta i(\tau) = i(\tau) - p_l\, i(\tau_l) - p_r\, i(\tau_r), \tag{3.5}$$

whereby the RF classifier considers a random subset of the features θ available at the node and all possible thresholds t_θ to determine the pair $\{\theta, t_\theta\}$ giving the maximal Δi, i.e. $\Delta i_\theta(\tau, T)$ [125]. This procedure is performed for all nodes τ in all trees T, to obtain the Gini importance for each θ:

$$I_G(\theta) = \sum_T \sum_\tau \Delta i_\theta(\tau, T), \tag{3.6}$$

when averaged by the total number of trees in the forest gives the MDI for feature θ, i.e. how relevant was its overall value [125, 126]. This framework may be generalized to more complex regression problems through using the total variance at each node τ in place of the Gini importance (see Refs. [124, 127]). Accordingly, the MDI is a direct by-product of training an RF model.

This strategy is easily implemented through the use of standard random forests algorithms. We have used the RandomForestRegressor model from the Scikit-learn [128] package. Once the MDI for each feature has been reliably assessed, we sort all the features in our descriptor vector according to their importance.

The most important feature is the molecular weight (MW) with a significance of 0.220. This was somewhat expected since it has been shown that there is a correlation between the glass transition temperature and the MW [129]. This is also fairly intuitive since molecules with a greater MW will naturally require more energy to increase their temperature and as shown in section 1.1.2 the glass transition temperature is determined by the rate of heating or cooling.

The second most significant feature is the total spatial autocorrelation of the relative atomic mass with a lag of 0. This descriptor has a feature importance of 0.00215. The equation for this descriptor is given by

$$ATS_k = \sum_{i=1}^{A} \sum_{j=1}^{A} w_i \cdot w_j \cdot \delta\left(k; d_{ij}\right) \quad k = 0,1,2,3,...d \tag{3.7}$$

where A is the number of atoms, w_n is the atomic mass of atom n, d_{ij} is the topological

distance between atoms i and j, and $\delta\left(k;d_{ij}\right)$ is a Dirac-delta defined as

$$\delta\left(k;d_{ij}\right)=\begin{cases}1 \text{ if } \delta_{ij}=k\\ 0 \text{ if } \delta_{ij}\neq k\end{cases} \tag{3.8}$$

Since we are looking at the case where $k=0$, this descriptor turns out to be the same as the molecular weight. The reason for the (minor) discrepancy in feature importance is due to the fact that the molecular weight is calculated using a slightly different method for each descriptor. The fact that our two most significant descriptors are describing the same property highlights the very significant need for the removal of highly correlated variables. By having a number of highly correlated variables we only add noise to the inputs of our model.

Finally, the feature with the third most importance is the Crippen Molar refractivity. The Crippen molar refractivity is a measure of the molecular size and polarisability of a compound, and it has been found to be relevant to the glass transition temperature of a material. Generally, compounds with higher molar refractivity have stronger intermolecular forces and higher packing density, leading to a higher glass transition temperature [130, 131].

3.6.5 Regularisation

The final type of feature selection we trialled was regularisation. Regularisation is particularly useful for complicated models that tend to overfit, such as NN's. In essence, regularisation is a way of desensitising the predicted value to small changes in the features. It is designed to reduce variance and increase bias.

Regularisation works by adding additional constraints or penalties to the model's optimisation process, discouraging it from fitting the noise or intricacies of the training data too closely. There are two main types of regularisation where their objective functions are given below:

$$\text{Objective(L1)} = \sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}x_{ij}\beta_j\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \tag{3.9}$$

$$\text{Objective(L2)} = \sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}x_{ij}\beta_j\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2 \tag{3.10}$$

In the objective functions above, y_i represents the observed value for the i-th sample, and x_{ij} represents the value of the j-th predictor for the i-th sample. The coefficient

for the j-th predictor is denoted as β_j. The parameter n denotes the total number of samples, and p denotes the number of predictors. The first term in each objective function represents the sum of squared errors between the observed values and the predictions made using the linear regression model. The second term in each objective function introduces a regularisation parameter λ, which controls the size of the coefficients.

Equation 3.9 and 3.10 are known as Lasso (L1) regularisation and Ridge (L2) regularisation respectively. Clearly they are very similar, the only difference is that L1 penalises the model by the sum of the absolute value of the weights, whereas L2 uses the sum of the squared value. Intuitively this can be interpreted as L1 regularisation seeking to estimate the median of the data where L2 tries to estimate the mean.

Both of these methods are used to reduce overfitting, although only L1 can be classified as a feature selection method. This is because it is possible for the optimal value of weights to be exactly 0 in L1 regularisation, whereas for L2 the weights get asymptotically close to 0 but never reach it. For this reason it is generally the case that L1 works better when not all features are relevant to the target, whereas L2 works where all features are relevant.

3.7 Early stopping

The process of early stopping is a technique used in machine learning to terminate the training process of a model when it is no longer improving on a particular metric. The early stopping technique aims to prevent overfitting, which occurs when a model becomes too complex and fits the training data too closely, leading to poor generalization to new, unseen data.

Early stopping involves monitoring a metric, in our case the loss of the test set, during the training process. If the metric fails to improve over a specified number of epochs, known as the "patience", the training process is halted. The number of epochs is a hyperparameter that can be tuned to achieve optimal performance on a given problem. For this work we set this hyperparameter to be 50 epochs.

The early stopping technique is especially useful when training deep neural networks or other complex models, as these models can require a large number of epochs to converge. By using early stopping, the training process can be halted before the model overfits to the training data, leading to better performance on new data.

3.8 Model selection and cross validation

When discussing performance of models, up until now we have simply been using the mean squared error given in equation 3.1. In this work we also use a few other scoring metrics to evaluate how certain models and techniques perform. The metric that we used with our genetic algorithm in chapter 5 is a combination between the MSE and the PCC given in 3.3. Finally, we also use the accuracy of predictions. This is the percentage of predictions that fall within 1 standard deviation of the true values. Although this is not particularly useful for the training of the models, it is useful when we assess how well the model is doing - being able to predict the target properties within one standard deviation is incredibly useful.

Now that we have set out the evaluation metrics it is important that we ensure they are as accurate as possible. NN's are sensitive to pertubations in initial conditions such as the subset of data used to train the model. In order to ensure that we are as confident of our results as possible, we use cross validation (CV).

The purpose of CV is to evaluate the performance of a model by seeing how well it will generalise to an independent set of data. CV uses various parts of the data, known as splits, as train and test sets for numerous iterations. This allows us to get a measure of uncertainty for our predictions by measuring the variance in predictions based on each different training set.

3.8.1 Cross Validation

Cross validation is sometimes referred to as k-fold cross validation, the method for performing k-fold cross validation is given in algorithm 3

When we are working with smaller datasets, such as the regression and classification datasets, it is preferable to use a special type of k-fold cross validation called leave-one-out cross validation (LOOCV). LOOCV is equivalent to choosing k equal to the number of points in the dataset. So in our case, each molecule will form a test set by itself and every other molecule will be used to train the model. This is done for each molecule in the dataset and the accuracy of the predictions can then be evaluated. The benefits of this are that it allows us to train on a larger training set, mitigating some of the pitfalls of using a small amount of data to train. The problem with LOOCV is the large computational cost - you have to train a model for each datapoint. Clearly this method is only suitable for smaller datasets, so it is used for our classification and regression tasks. When CV is performed on larger datasets it is much more common to use cross validation with a much lower number of folds since the amount of training data is less of a problem and computational cost can be limiting.

A benefit of using cross validation in general is that it allows us to plot error bars for

Algorithm 3 K-fold Cross Validation

Input:

- `D` – Dataset containing n samples, represented by feature vectors $\mathbf{x}1, \mathbf{x}2, ..., \mathbf{x}n$ and their corresponding labels $y1, y_2, ..., y_n$.
- `k` – Number of folds.

Output:

- `scores` – Array of length `k` containing evaluation scores for each fold.

1: Shuffle the dataset `D` randomly.
2: Divide the dataset `D` into `k` equally sized folds.
3: **for** each fold `i` in range `k` **do**
4: Take fold `i` as the validation set, and use the remaining folds as the training set.
5: Train the model on the training set.
6: Evaluate the model on the validation set and record the evaluation score.
7: **end for**
8: Calculate the average score across all folds.
9: Return the array of scores.

our predictions. These error bars give an insight into the bias and variance of the model. Bias and variance are always inversely correlated. Bias is the inability for the model being used to adapt to the data, for example trying to use a linear model to predict a non-linear dataset. Variance is defined in machine learning as the amount the objective function changes when trained on different data and is a symptom of overfitting.

Small error bars represent a high bias and low variance, while large error bars represent the opposite, a low bias and high variance. So we seek 'reasonable' error bars, although it is hard to quantify exactly what this means. To try and ensure sensible error bars, the results of the cross validation are manually inspected and the model is adjusted accordingly. A high bias means that we need to change either the architecture of the network by adding more nodes or hidden layers, or by training the data for longer by increasing the number of epochs. We can fix a high variance by using regularisation, simplifying the network architecture, or training for less time/using some sort of stopping criteria.

We can further increase our confidence in the outcome of the CV by repeating the process. Whenever we refer to repeated cross validation in this work, we have repeated the cross validation 5 times. With each repeat the data is shuffled so that the subset of data that makes up the training and test split for each fold is different. Obviously it does not make sense to repeat LOOCV since the training set will be the same each time.

3.9 Ensemble methods

Ensemble methods in machine learning are defined as the process of combining multiple models to improve the accuracy of predictions. The purpose of this is to reduce overfitting, improve how well the model generalises to new data, and reduce the significance of outliers. There are various ways in which this can be done that we explore in this work.

3.9.1 Bagging

Bootstrap Aggregating (Bagging) is a powerful and commonly used ensemble learning method that forms an integral part of the random forest model. The essence of Bagging is to generate multiple training sets from a single original training set using a technique called bootstrapping. Bootstrapping is performed by repeatedly sampling the original training dataset with replacement, until a new dataset of the same size is created. Due to the sampling with replacement, the new dataset may contain duplicate samples and some samples that were present in the original set may be missing.

By using each bootstrapped dataset, a decision tree is trained on each one, resulting in an ensemble of trees that are diverse and theoretically more robust. This is because each tree is trained on a slightly different dataset, leading to a different set of internal rules and decision boundaries.

Once the trees have been trained, the outcomes of each tree's predictions are aggregated to produce the final prediction. Specifically, a test sample is passed through every tree in the ensemble and the outcomes are averaged to produce the final prediction.

In modern machine learning techniques like AdaBoost [132], the aggregation process is further optimized by assigning weights to each tree's prediction based on its performance on the training data. A tree with better predictive capabilities will contribute more to the final aggregation than one with poor performance.

3.9.2 Max voting

Max voting, also known as majority voting, is an ensemble method that aggregates the results of multiple ML models. This method is a generalised term for the aggregation step of the bagging algorithm, as described in Section 3.9.1, and can be used with any type of ML model, not just limited to random forests. The process involves training multiple models and running the test set through each model. For regression tasks, the final prediction is the mean of all the models' outcomes, while for classification, the final prediction is the mode of all the outcomes. It is important to note that in the case of classification, when the vote

of each model is not weighted, an even number of models cannot be used. This is due to the fact that if the votes are tied, the final prediction would be ambiguous.

3.9.3 Ensemble of Neural Networks

This is a technique that we have leveraged heavily in this work. In our work we have numerous descriptors and we hypothesise that by combining these descriptors in a neural network we can outperform a neural network with only a single descriptor or even a max voting ensemble of individual descriptor models.

One might be tempted to simply concatenate each feature into one long feature before feeding that into a neural network. However, as we have shown in Table 3.2, each descriptor can be optimised by using different L2 regularisation rates and network architecture.

To make sure that each descriptor is performing optimally we can train a separate network for each descriptor. These networks are then concatenated in a concatenation layer which is then trained to produce an output. A schema of what this looks like is given in fig 6.1. What happens when using this architecture is that the first layer of networks learn some representation of the descriptor they are assigned. This representation is then fed into a layer where the concatenation layer is then trained as a regular neural network to output a prediction. The error at each epoch is backpropagated throughout the entire ensemble neural network.

Chapter 4

Using MD to understand the boson peak

4.1 Introduction

This chapter delves into a phenomenon that takes place during the formation of glasses. It also offers an additional application of the simulation protocol previously established and described in section 2.2. The findings demonstrate that the descriptors derived from the protocol have practical uses beyond their original purpose of predicting the stability of amorphous drugs.

Our journey into this intriguing realm is significantly informed by collaborative work with Gonzalez-Jiminez et al. [78], in which my primary contribution revolved around conducting MD simulations of the system. In the following sections, we provide contextual information to elucidate why MD simulations were essential for this research and underscore their significance beyond the confines of this thesis.

When a liquid is (super)cooled, the primary or α-relaxation rate—directly related to viscous flow—decreases and decreases dramatically somewhere near the Tg, where calorimetry measurements identify a rapid decrease in heat capacity[133]. Below the glass transition, secondary or β-relaxation processes can be observed due to faster dynamic processes that become uncoupled from the α-relaxation on vitrification and are the dominant relaxation channel in the glassy state[134].

The molecular origin of these secondary relaxations is still unclear[135]. Because secondary relaxations were first observed in polymer glasses, they were assumed to originate in the small angle orientational diffusion of the side chains or functional groups of the polymers and therefore of essentially intramolecular character. However, the observa-

tion of secondary relaxations in rigid polymer glasses and even metallic glasses [134] implies that at least some (now referred to as "Johari–Goldstein") secondary relaxations are of an intermolecular character. One suggestion is that they are caused by spatial inhomogeneities [136] giving rise to more loosely packed regions, distinct relaxing domains, defects[137, 138], or regions of different packing such as locally favoured vs. liquid-like structures[139]. Another approach starts from the liquid's potential energy landscape and views secondary relaxations as transitions between neighbouring potential energy minima while primary relaxation corresponds to the higher energy transitions between mega basins[133]. This picture explains why α relaxation is frozen out before β relaxations.

Another phenomenon often associated with the glass is the boson peak [133, 140], which is observed as an excess intensity of low-frequency modes around about 1 THz in spectroscopic studies or as an excess heat capacity signature. The boson peak represents a peak in the vibrational density of states [141] corresponding to an excess in the density of states over that expected from phonons in a perfect Debye crystal. This gives rise to anomalous behaviour of the low-temperature heat capacity C_p and a peak in C_p/T^3 at a few 10 s of K. The origin of the phenomenon has been assigned various interpretations, ranging from the occurrence of "two-level" excitations associated with broken bonds or other defects in glasses[142], to other mechanisms for creating an excess in the low-frequency vibrational density of states in the terahertz range over that expected for dispersive phonons in a perfect crystal, giving rise to an additional contribution to the very low-temperature heat capacity C_p leading to a peak in C_p/T^3[143]. The presence of the boson peak has been linked to fluctuating elastic constants within a structurally disordered amorphous matrix[144], quasi-localised soft potential defects[145, 146], localisation of transverse phonons associated with defective soft structures[147, 148] (quasi) localised vibrational modes of locally favoured structures [149, 150], a crystal-like van Hove singularity near the pseudo-Brillouin zone edge washed out by structural disorder[151], may be caused by diffusive damping rather than spatial disorder[152], and might not even contribute any extra heat capacity[153].

To gain insights into these intriguing phenomena, we utilise a combination of experimental techniques, including depolarized Raman spectroscopy, X-ray scattering, and calorimetry, alongside the powerful tool of MD simulations. The MD simulations allow us to probe the intermolecular structure and dynamics in supercooled and vitrified liquids, offering a unique perspective on the behavior of molecules at the atomic scale.

Our results from MD simulations reveal key changes in the coordination and network topology of the studied liquid, which are pivotal in preventing crystallisation as the material transitions to the glassy state. These structural transformations are further corroborated by experimental data, such as the evolution of Raman spectra and X-ray scattering

patterns. Together, these findings provide a comprehensive understanding of the physical processes at play during glass formation.

In summary, the integration of MD simulations into our investigation not only enhances our comprehension of these complex phenomena but also establishes a vital link between computational and experimental approaches. This multidisciplinary approach bridges the gap between atomic-scale insights and macroscopic observations, ultimately deepening our understanding of glass formation and the associated dynamic processes.

4.2 Results

Tetrabutyl orthosilicate (TBOS) is a viscous liquid that does not crystallise and has a glass transition temperature Tg as measured by calorimetry of 120 K.

Femtosecond optical Kerr-effect (OKE) spectroscopy[154, 155] was used to measure the Bose–Einstein corrected depolarised Raman spectrum using a time-resolved pump–probe technique and numerical Fourier deconvolution. In our set-up[156, 157, 158, 159] which has a time resolution of about 20 fs, the pump–probe delay can be as large as 4 ns resulting in spectra with a maximum spectral range from 125 MHz to 50 THz but is limited here to the range 10 GHz to 10 THz to maximise the signal to noise. The low-frequency depolarised Raman spectra of liquids typically contain (overlapping) contributions from orientational relaxation, translational relaxation, intermolecular cage rattling motions, librations, and vibrations[160, 161]. We have shown that—at least within the accessible frequency range >1 GHz—the orientational and translational relaxations in nearly all liquids follow the Stokes–Einstein–Debye and Stokes–Einstein laws tracking the macroscopic shear viscosity and are therefore representative of primary or α relaxations[157].

The amplitudes of orientational relaxation and librations in the spectra are proportional to the anisotropic molecular polarisability tensor, which vanishes in a molecule with tetrahedral, octahedral, or icosahedral symmetry. Quantum chemistry calculations were carried out on TBOS and stability calculations of dimers, trimers, and higher aggregates show that TBOS is expected to remain monomeric under normal conditions. The calculated infrared spectrum in the Si–O–C stretch region (around 1100cm^{-1}) matches the experimental one,

confirming that TBOS is monomeric and the silicon atom tetrahedrally coordinated as expected. Temperature-dependent 13°C NMR spectroscopy from 20°C to 50 °C shows four sharp bands as expected for a monomer. Calculation of the molecular polarisability tensor of 100 structures randomly picked from 31,500 low-energy conformers shows that the anisotropic polarisability remains an order of magnitude smaller than the isotropic one,

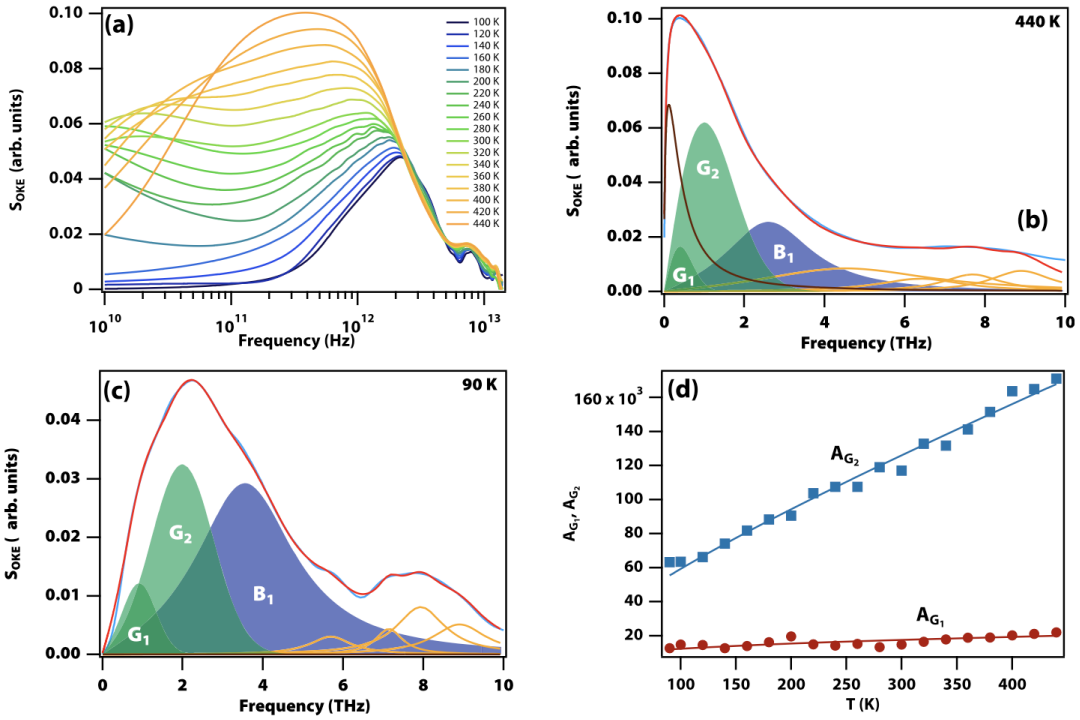

Figure 4.1: Optical Kerr-effect (OKE) spectra of supercooled and vitrified tetrabutyl orthosilicate (TBOS). a All data from 90 to 440 K. b, c Two representative temperatures and fits. The black line in (b) is the component due to diffusive relaxation fitted to a Havriliak–Negami function, which freezes out below the glass transition and is therefore absent in (c). The two green bands at low frequency are intermolecular modes fitted to two Gaussian functions. The blue band is an intramolecular vibration fitted to a Brownian oscillator function with constant amplitude. The yellow curves are additional intra-molecular vibrations. d Temperature-dependent amplitudes of the low-frequency (AG1, red disks) and high-frequency (AG2, blue squares) inter-molecular modes. The lines are guides to the eye. While these amplitudes change, the amplitude of the higher frequency intramolecular modes remain unchanged with temperature as expected.

demonstrating that the deviation from perfect tetrahedrality due to the flexibility of the butoxy side chains has a minimal effect. Therefore, the spectrum of TBOS should be greatly simplified due to symmetry, only showing translational relaxation, intermolecular cage rattling motions, and vibrations.

4.2.1 The OKE spectra

OKE spectra of TBOS were obtained over a temperature range from 90 to 440 K, as shown in Fig. 4.1. The spectra are largely temperature independent ≥ 3 THz as this region only features intramolecular modes. The low-frequency (≤ 3THz) part of the spectra has a strong temperature dependence with changes in shape as well as amplitude.

The spectra in this low-frequency range could be fitted consistently with four functions: a Havriliak–Negami function representing the diffusive α relaxation, two Gaussian functions representing the intermolecular modes, and a Brownian oscillator function for the lowest frequency intramolecular mode. As expected, the diffusive α relaxation is strongly temperaturedependent and freezes out below the glass transition.

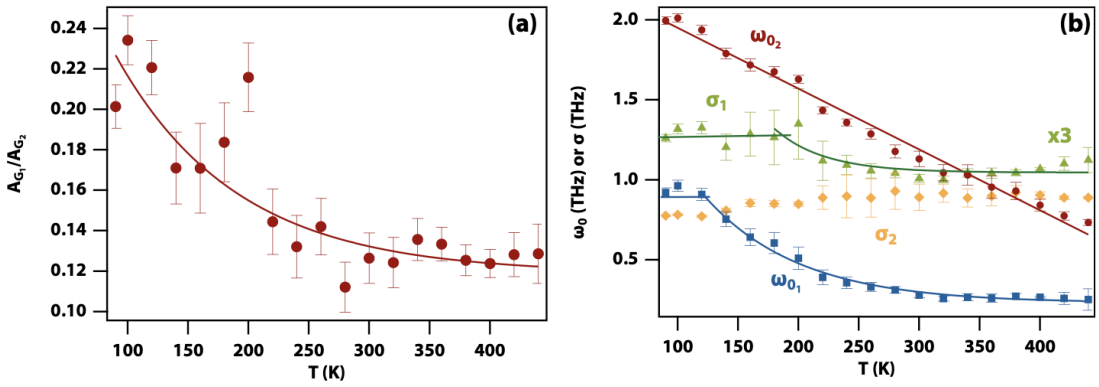

Figure 4.2: Temperature dependence of the fit parameters for the intermolecular modes. a) Ratio of the amplitude of the low-frequency intermolecular mode over that of the high-frequency one. The solid red line is an exponential fit to guide the eye (The data point at 200 K was omitted in this fit on account of the noise at low frequencies in the corresponding OKE data). b) Centre frequency ω_0 of the two intermolecular modes (blue squares and red disks, also shown are linear and exponential fits to guide the eye) and the corresponding widths σ (green triangles and yellow diamonds respectively, also shown are an exponential fit and a horizontal line to guide the eye).

The amplitudes of the two Gaussians (see Fig. 4.1) are proportional to temperature, showing that they are collision-induced intermolecular "cage rattling" modes. The spectra below the glass transition (e.g., at 90K in Fig. 4.1) clearly show the two intermolecular bands as a peak at 2 THz and a shoulder at 0.9 THz. As can be seen in Fig. 2, the two intermolecular bands evolve differently as a function of temperature. The ratio of the amplitudes of the low and high-frequency bands stays approximately constant at high temperature but doubles on cooling to the glass transition. The width of the high-frequency band is essen-

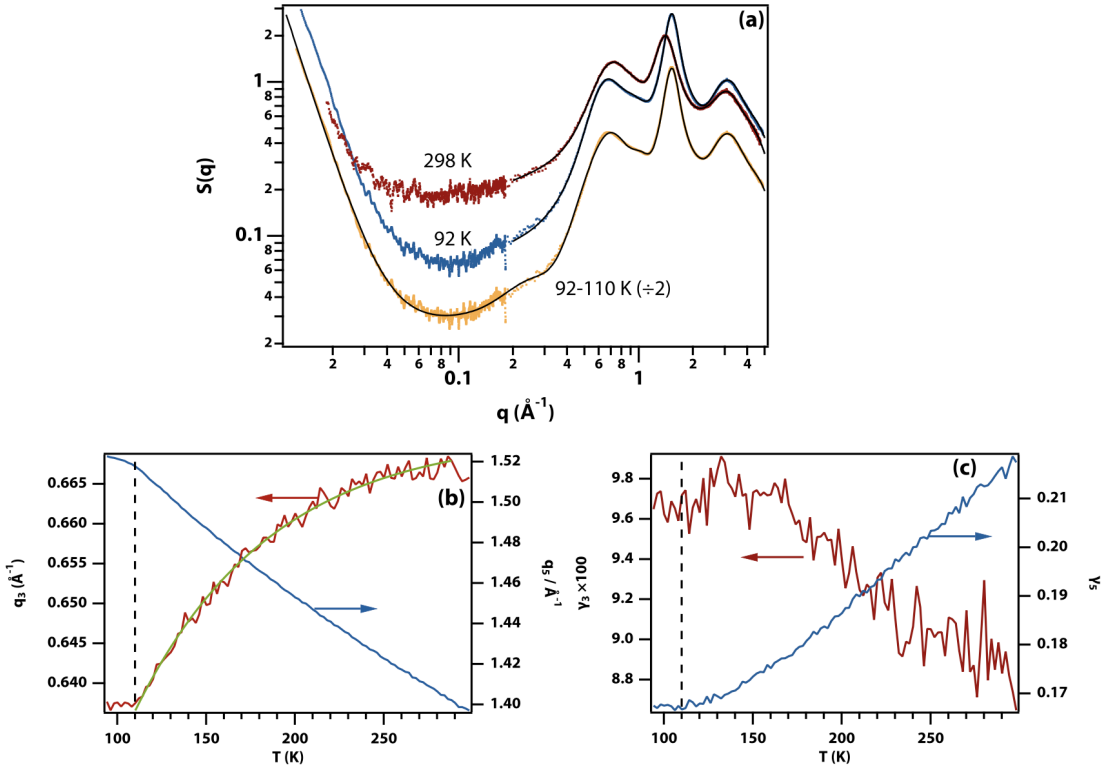

Figure 4.3: Analysis of temperature-dependent WAXS data. a) Experimental SAXS and WAXS data taken at 298 K (red) and 92 K (blue) and fit to four Gaussians and a Lorentzian (black). The average of the nine data sets at 110 K and below is shown (yellow) with a fit including an additional Gaussian to account for the prepeak (black). b) Variation of the peak of the first (q_3, red) and second (q_5, blue) sharp diffraction peaks in the WAXS data obtained from fits to a Gaussian and a Lorentzian, respectively. The green line is an exponential fit to guide the eye. c) Variation of the width of the first (γ_3, red) and second (γ_5, blue) sharp diffraction peaks.

tially temperature independent suggesting the corresponding inhomogeneity is constant. The width of the low-frequency band slightly increases on cooling and plateaus below 200 K but this is a minor effect. The centre frequency of both bands increases on cooling in a linear fashion for the high-frequency and a nonlinear fashion for the low-frequency band, the latter plateauing below the glass transition.

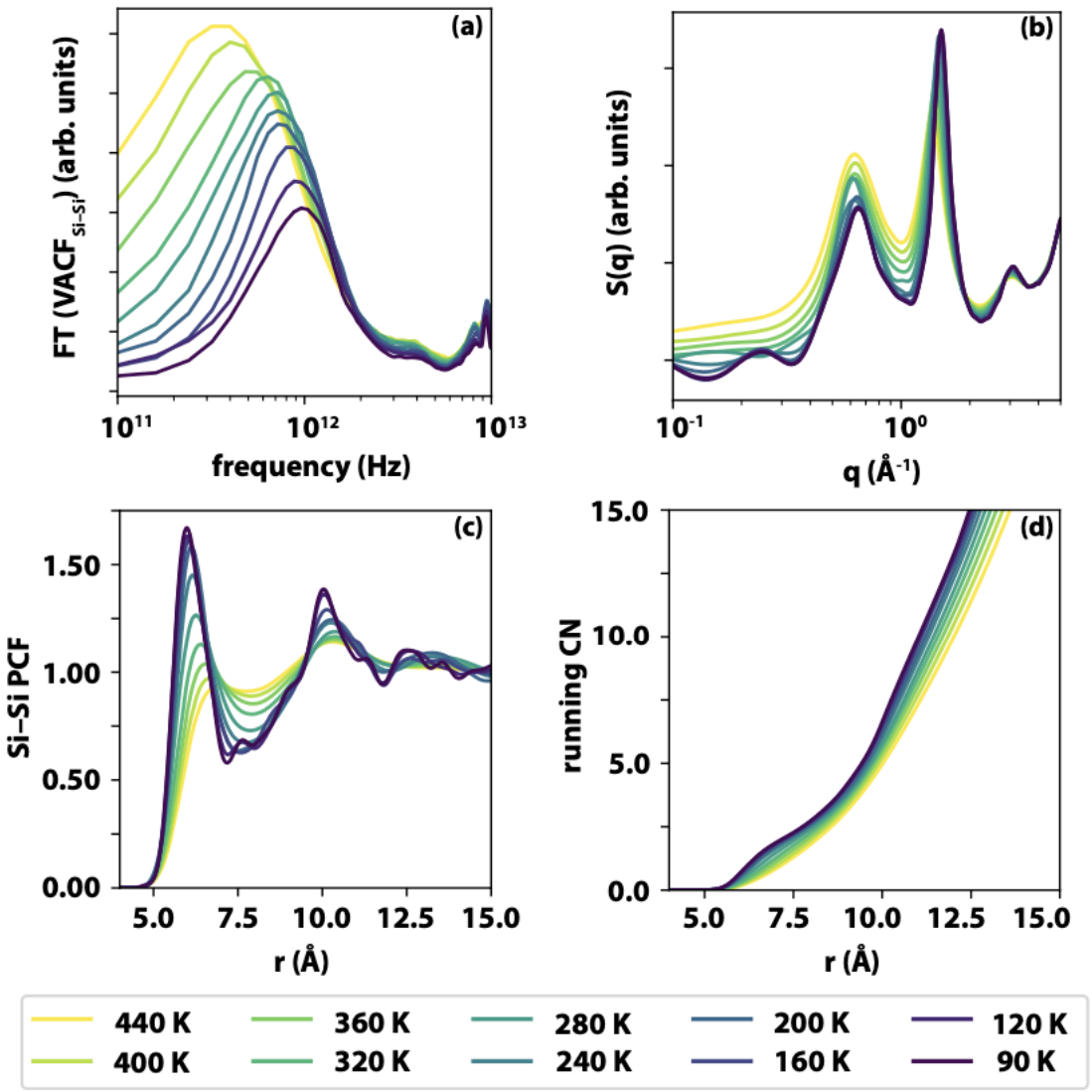

Figure 4.4: Results of the temperature-dependent molecular dynamics simulations of TBOS. a) Power spectra obtained from the Fourier transform of the Si–Si velocity–velocity autocorrelation functions. b) Static structure factor, including Si, O and C atoms. c) Si–Si pair correlation functions. d) running Si–Si coordination number

4.2.2 WACS and Raman experiments

Structural information can be obtained using small (SAXS) and wide-angle X-ray scattering (WAXS) experiments carried out over a similar temperature range as the OKE experiments with a 2 K step size (see Fig. 4.3). The first sharp diffraction peak at 0.65 Å^{-1} is consistent

with the Si–Si nearest neighbour distance of 10.4 Å calculated from the liquid density

The SAXS/WAXS intensity data were analysed by curve fitting, requiring a Gaussian function to fit the first sharp diffraction peak and a Lorentzian for the second The fits were used to calculate the radial distribution functions using analytical transformation, showing a reduction of the first and second solvation shell radius on cooling as expected.

Figure 4.4b), c) shows the evolution of the peak position and width of the first and second sharp diffraction peak as a function of temperature. The first sharp diffraction peak shifts in a nonlinear fashion to lower q on cooling, while the second peak linearly shifts to higher q. Both show a distinct change in their evolution at 110 K, which is slightly below the glass transition temperature as determined using calorimetry (124K). Temperature-dependent Raman spectra of the CH-stretch band show that, on lowering the temperature, the TBOS molecules reduce the number of gauche defects in the alkoxide side chains, consistent with the lowering of the peak position of the first sharp diffraction peak on cooling.

Close inspection of the data at the lowest temperatures shows the presence of a weak pre-peak at q 0.2 $Å^{-1}$, however, the signal-to-noise in the SAXS–WAXS transition region is insufficient for full temperature-dependent analysis of this feature. The nine lowest temperature data sets ($\leq$ 110K) were averaged for improved signal-to-noise revealing a clear pre-peak. This spectrum could be fitted with an additional Gaussian, allowing a determination of the pre-peak position as 0.234 $Å^{-1}$.

In conclusion, the analysis of SAXS/WAXS intensity data has provided valuable insights into the structural changes occurring in TBOS upon cooling. The curve fitting of the diffraction peaks using Gaussian and Lorentzian functions has enabled us to calculate radial distribution functions, which clearly indicate a reduction in the radii of the first and second solvation shells as the temperature decreases, in line with our expectations.

4.2.3 MD simulations

Atomistic MD simulations (see Sec 2.2 for details) were used to generate a 512-molecule model of liquid TBOS, quenched from 440 K to 90 K at a rate of 7.5 × 108 K/s into the glass. The resulting T_g is 249 ± 20 K, significantly higher than the experimental value (124 K) due to the much faster cooling rate. The Fourier transform of the Si–Si velocity–velocity autocorrelation function (Fig. 4.4a), which thanks to the molecular symmetry of TBOS should display the same features of the OKE spectra, indeed shows a decrease of signal intensity and a shift to higher frequencies on cooling. Computation of the total Si–Si (static) structure factor (Fig. 4.4b) shows the emergence of a low-q feature on cooling between 0.25 and 0.45 $^{-1}$, consistent with the pre-peak observed experimentally (Fig. 4.3a). The occurrence

of this structural feature was verified by generating three additional models of TBOS. The Si–Si pair correlation function (Fig. 4.4c) shows a significant increase in short-range order, particularly in the second coordination shell. The first coordination shell (up to 8 Å, see Fig. 4.4d) only contains, on average, about 2.5 molecules, showing a lack of tetrahedral order in both liquid and glassy TBOS. However, the second coordination shell (up to 12 Å) contains about 12 molecules. A Voronoi analysis using the Si atoms was performed (Fig. 4.5a) to gain further insight into these structural changes and to provide an effective coordination number. Overall, it is clear that the network of Si atoms is predominantly 12-coordinate at any given temperature. However, below T_g, the probability density of the volumes of the Voronoi polyhedra (VP) splits (Fig. 4.5b), indicating the emergence of specific structural features. These can be identified (Fig. 4.5c) as VP with 15 and 16 faces, which are unique to the glassy state. VP characteristic of ordered phases (see Fig. 4.5c) are very infrequently observed. VP characteristics of FCC order, such as the <0, 3, 6, 4 >VP, are found in the supercooled liquid but disappear below T_g, where the <0, 4, 4, 2 >VP, characteristic of HCP order, makes an appearance. This is consistent with both the local environment analysis and the bond-orientation order analysis, which show that TBOS displays a weak tendency toward HPC order—which however is frustrated by the overcoordinated, largely disordered network emerging in the proximity of T_g. In fact, the overcoordinated VP tend to form larger clusters as the liquid is cooled below T_g. The analysis of the orientation of TBOS molecules as a function of temperature, indicates that TBOS is perfectly isotropic even below T_g and that the degree of orientational order in the supercooled liquid is higher than that observed for the glass. The simulations also reproduce the reduction of the number of gauche defects in the alkoxide side chains on cooling . Below T_g the self-diffusion of the system is so slow that it is possible to relate the power spectrum of the system to specific polyhedra. It is found that the high-frequency region of the power spectra is mostly linked to the polyhedra with 16 faces (Fig. 4.5d) at 90 K. Thus, the increase in the local coordination may be part responsible for the blueshift of the OKE spectra on cooling.

4.3 Conclusion

Due to the nearly tetrahedral symmetry exhibited by TBOS, and consequently, its almost uniform molecular polarisability, the OKE spectra do not display any evidence of orientational relaxation or librational motions. It's important to note that this absence of observation does not imply that molecular orientation relaxation or librational processes are nonexistent; rather, it suggests that these particular phenomena are not discernible within the context of the OKE spectra. The OKE spectra primarily manifest the lowest-frequency re-

laxational component, which, to the extent that rotational and translational motions can be distinguished, arises predominantly from translational relaxation. This relaxational behavior becomes apparent in the spectra as a result of a collision-induced process [162, 156, 157].

However, it's worth highlighting that the lowest detectable frequency in these OKE experiments stands at 10 GHz. In contrast, β relaxations, which are typically observed around the glass transition temperature, tend to occur at significantly lower frequencies, roughly in the vicinity of 1 kHz[134, 135]. Consequently, the presence of β relaxations in TBOS cannot be conclusively ruled out. Similarly, the possibility of a deviation from the Stokes-Einstein law for translational relaxation, occurring much closer to the glass transition temperature, cannot be dismissed either.

Atomistic MD simulations provide further insights into TBOS's behavior, revealing the presence of a low-q peak at 0.25 $Å^{-1}$. Moreover, these simulations successfully replicate the essential characteristics observed in the OKE spectra and WAXS data. While establishing a precise quantitative link between these simulated structures and the supramolecular arrangements responsible for the boson peak identified in OKE measurements remains challenging (requiring even slower cooling rates and more extensive models), the MD simulations do demonstrate that these structures correspond to clusters of TBOS molecules with higher-than-usual coordination. If not for the extended side chains atop the tetrahedral silicon, these clusters would tend to exhibit a more crystalline structural order.

In summary, this chapter has underscored the versatility of our MD simulation protocol, demonstrating that it extends beyond the scope of this thesis. These simulations not only have the potential to predict functional properties of drug-like molecules but also serve as valuable tools for enhancing and validating physical experiments conducted on other molecular glasses.

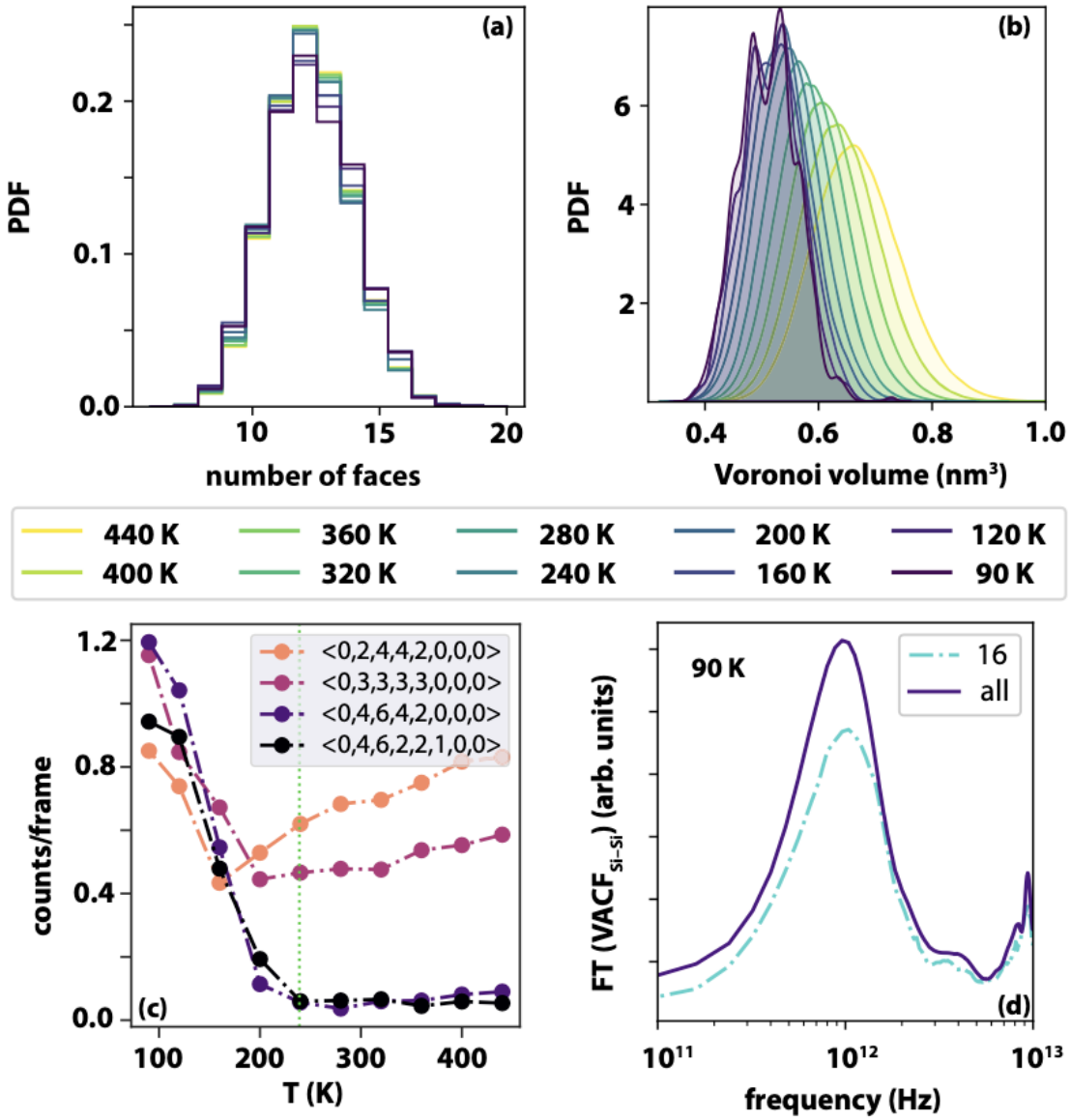

Figure 4.5: Temperature-dependent Voronoi analysis of TBOS models. a) Probability density function of the number of faces characterising the Voronoi polyhedra (VP) for each Si atom, averaged over 1000 frames across a 10 ns long MD trajectory. b) Probability density function of the volume of the VP. c) Frequency of the occurrence of selected VP as a function of temperature. green dotted line indicated the value of (computationally obtained) T_g. d) Power spectra obtained from the Fourier transform of the Si–Si velocity–velocity autocorrelation functions at 90 K. The purple and light blue lines refer to the result obtained considering all the Si atoms (same as Fig. 4.4a) and those Si atoms characterised by VP with 16 faces only, respectively.

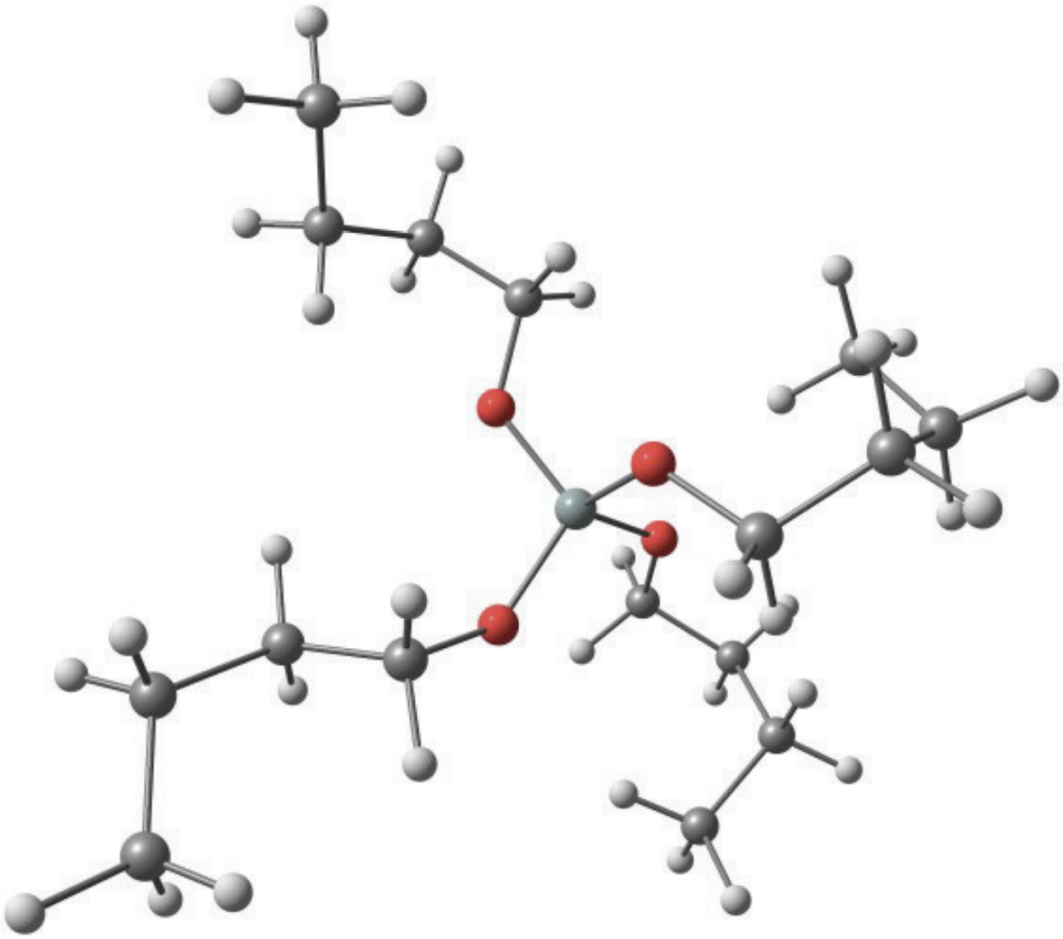

Figure 4.6: Model of TBOS. Structure obtained by wB97X-D3/ma-def2-SVP. Based on the molecular weight of TBOS of 320.54 g/mol and its density at 25°C of 0.899 kg/l (Sigma-Aldrich), one calculates a molecular volume of 592Å^3 or (assuming a spherical shape) a hydrodynamic radius of 5.2 Å.

Chapter 5

Genetic Algorithm for Optimisation

5.1 Overview

Within the scope of this chapter, we embark on an exploration of a hyperparameter optimisation approach, delving into the intricacies of this method. Through our investigation, we aim to demonstrate the efficacy of these techniques in fine-tuning the parameters associated with the SOAP descriptor. Importantly, we endeavor to illustrate that through the application of these novel optimization strategies, we can achieve parameter optimization in a manner that not only yields superior results but also significantly enhances computational efficiency when compared to conventional grid search methodologies.

Genetic Algorithms (GAs) are a metaheuristic optimisation method based on the principles of natural selection and evolution proposed by Charles Darwin [163, 108]. In essence, GAs are a relatively straightforward optimisation method and can be explained best by drawing analogies to natural selection. Let n be the number of parameters that require optimisation, we will refer to these parameters as genes. An individual is comprised of n genes and a population is comprised of N individuals where N is a hyperparameter that will affect the rate at which the algorithm converges and the parameter space that is explored. These individuals are then evaluated and assigned a 'fitness' score, which is a measure of how well they performed in the test. The fittest individuals and a handfull of randomly selected individuals are then paired together while the least fit individuals are discarded. The pairs of individuals then 'breed' together where a number of new individuals are created by taking a random selection of genes from either parent with a small chance that one of the genes will 'mutate' by transforming into a random value. This is repeated until a specified number of generations have occurred or some convergence criterion is met.

Algorithm 4 SOAP_GAS

Parameters:

- `popSize` The number of individuals in the population.
- `bestSample` The number of fittest individuals chosen for breeding.
- `luckyFew` The number of individuals that are randomly selected for breeding (This is done after `bestFew` have been selected.
- `numChildren` The number of children that each pair of individuals will produce. Note: `popSize` must remain constant at each generation, therefore the following equation must hold true:

$$\texttt{popSize} = \texttt{numChildren}^{\cdot}$$

`bestSample+luckyFew2` (5.1)

- `mutationChance` The probability that a gene will mutate to a random value after a new individual is created.
- `earlyStop` Tolerance criterion for any two generations to be considered equally accurate. In conjunction with textbfearlyNum (see below) it determines the early stopping criterion for the GA. E.g., earlyStop=0.04 implies that two generations which best score is within 4% of each other are to be considered as equally accurate.
- `earlyNum` Number of equally accurate generations (according to the `earlyStop` threshold, see above) that must be generated in order for the GA to stop. Note that the `earlyNum` do not need to be generated consecutively, but at any point along the GA instead.
- `numGenerations` The number of generations that the GA will continue for if the conditions for early stopping are not met.

1: Initialise `popSize` individuals with random values for each gene.
2: Second step
3: Evaluate the fitness of each individual.
4: Select the `bestSample` individuals with the highest fitness and `luckyFew` individuals from the remaining population. These individuals are then paired together randomly.
5: Each pair of individuals produces `numChildren` individuals by selecting each gene uniformly randomly from each parent. After every gene has been selected there is then a `mutationChance` chance that it randomly changes to a random value.
6: Go to step 3 and repeat until the early stop criteria are met or `numGenerations` have been completed
7: The optimised set of parameters are the genes for the individual with the best fitness score out of every generation.

5.2 Genetic Algorithms in the context of optimising SOAPs

Now that an understanding of how Genetic Algorithms operate has been established, we can discuss GAs in the context of SOAPs.

5.2.1 The SOAP_GAS algorithm

. In this section we describe the SOAP_GAS algorithm in detail, the work in this chapter uses exclusively the solubility dataset described in section 1.3. The exact steps performed in the SOAP_GAS are given in algorithm 4 and a schematic is provided in fig 5.1.

An initial population containing a certain number (`popSize`) of individuals is constructed. Each individual corresponds to a SOAP descriptor characterised by a fixed selection of atomic species as *centres* and *neighbours* as well as a randomly selected set of SOAP parameters (i.e., $n_{\max}$, $l_{\max}$, *cutoff* and *atom_sigma*), see section 2.3.5 for more information on these parameters. The user has the freedom to specify lower and upper boundaries (usually dictated by physical intuition and/or computational cost) for each of the four SOAP parameters.

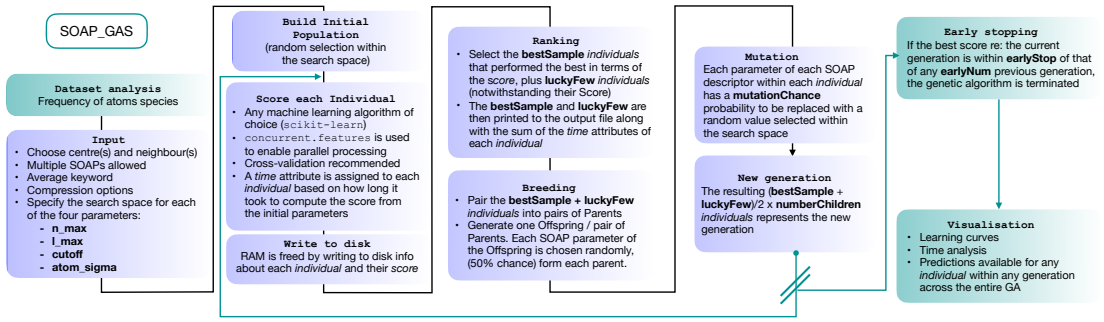

Figure 5.1: Schematics of the genetic algorithm framework implemented in the SOAP_GAS code.

The choice of which atomic species are to be selected as *centres* and *neighbours* for the SOAP descriptor is left to the user. The `average` keyword within the SOAP descriptor [164] implements the structure-wise SOAP descriptor described in Eq. 2.15, resulting in feature vectors of the same dimensionality across heterogeneous datasets containing different molecules or different number of molecules in a given structure. A simple script included in the SOAP_GAS package can be used to analyse a dataset of N molecular structures and gain information about the frequency by which a given atomic species is present within the dataset.

For each individual descriptor within the initial population we compute a score (or "fitness", as customary in the GA literature), i.e. a metric that quantifies the accuracy of the individual in predicting the functional property of interest - in this case, the solubility of a given molecular species. In particular, in our case we have chosen to combine two popular metrics, the mean squared error (MSE) and the Pearson correlation coefficient (PCC), for both the training and test sets, as follows:

$$\text{Fitness} = 2 \cdot MSE_{tr} \cdot (1 - PCC_{tr}) + MSE_{te} \cdot (1 - PCC_{te}), \tag{5.2}$$

where MSE_{tr} and MSE_{te} are the mean squared error of the training and test sets respectively, and PCC_{tr} and PCC_{te} are the Pearson correlation coefficients of the two sets.

This unusual score metric was used to balance the contributions of training and test sets for our relatively small dataset. We decided to include the PCC as opposed to just using MSE to fit the tail of the distribution where there is less data, but this score metric can be straightforwardly modified if necessary.

To obtain this fitness score, we have employed a straightforward random forests (RF) model. RF is an ensemble learning technique that averages the predictions from a collection of decision trees and may be utilized for both classification and regression [126]. Each decision tree is built on N data points that are bootstrapped, i.e., sampled with replacement, from the N-sized training data, with the results collected and averaged to obtain a single prediction, a procedure called bootstrap aggregation or "bagging". The split at each node is selected only from a subset of the features, with the feature that minimizes the error being selected. This framework randomizes the ensemble of decision trees, creating a set of independent predictions from weak learners that may not be as good individually but once aggregated and averaged, produces a better result. Furthermore, as bootstrapping creates multiple datasets that are distinct from the original to construct each decision tree, RFs are effective for modeling small datasets [165, 166]. In terms of the training/test split, we have used 33% of the dataset as the test set.

Once we have a fitness score for each of the `popSize` individuals, we select a certain number (`bestSample`) of them according to their fitness scores, plus a usually small number (`luckyFew`) of individuals regardless of their score. These selected individuals constitute the so-called "parents" of the next generation of SOAP parameters. At this point, we move onto the "breeding" procedure, where we randomly split the (even number of) parents into (`bestSample` + `luckyFew`)/2 pairs. Each pair of parents produces a "child", i.e. a new SOAP descriptor characterised by a new set of SOAP parameters - randomly picked with a 50% chance from either of the parents. We then proceed to apply "mutations":

each SOAP parameter within each child has a certain probability (`mutationChance`) to be changed into a randomly picked value (within the boundaries specified for that SOAP parameter). Note that the resulting population size, `popSize` = [(`bestSample`+`luckyFew`)/2) × `numberChildren`)] is identical to the size of the initial population.

Once we have obtained the new generation, we repeat the process until we reach the desired level of accuracy. The idea at the heart of SOAP_GAS and GA algorithms in general is that they allow to progressively explore the search space in an efficient, targeted fashion, while introducing mutations and other degrees of freedom (such as the number of `luckyFew`) to avoid getting stuck in local minima of the parameter space. SOAP_GAP also features an early-stopping criterion: if the current generation is within a certain threshold (in terms of fitness score) `earlyStop` of that of a specific number `earlyNum` of any previous generations, the algorithm is considered to be converged.

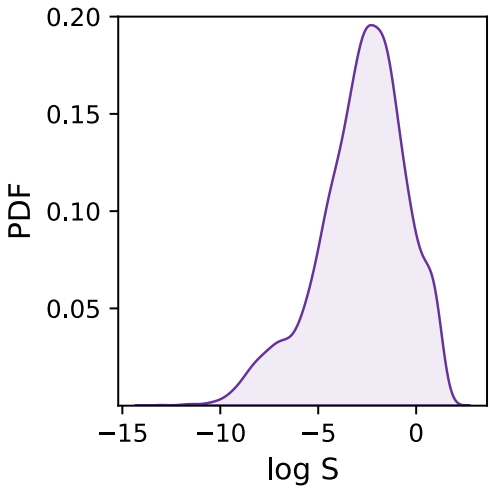

Figure 5.2: PDF of the solubility values (as log S) across the dataset.

Note that, depending on the size of the dataset, the choice of *centres* and *neighbours* as well as the choice of n_{max}, l_{max} and *cutoff*, the dimensionality if the SOAP vector can grow to the point of causing issues in terms of memory requirements. In fact, the usage of high-memory nodes is often necessary when working with SOAP. Aside from the compression strategy discussed in the next sections, in order to free memory the SOAP_GAS algorithm writes to disk the information about each individual in a bespoke class that contains the SOAP parameters, the target values of the whole database as well as the fitness score and each of the train/test splits used for the cross validation relative to that particular fitness score.

It is worth noting, however, that grid search approaches can trivially and effectively leverage parallel computing in that each grid point can be evaluated independently from the others. Conversely, SOAP_GAS is by its very nature a sequential algorithm, as the construction of given i—th generation of individuals depends on the accuracy of the individuals within the ($i-1$_th generation. However, we can still take advantage of parallel computing for the evaluation of different individuals within a given generation. To this end, we have

adopted the `concurrent.futures` Python module, which provides a simple platform to allocate different instances of the same task (in our case the evaluation of the different individuals using the very same RF model) to the available computing cores. A scaling test, demonstrating the quasi-linear scaling of SOAP_GAS with the number of CPU cores, can be found in figure 5.3.

The SOAP_GAS code is freely available on GitHub at `https://github.com/gcsosso/SOAP_GAS.git`.

5.2.2 Dataset utilised

Unlike the other chapters in this work, a different dataset has been utilised for testing the capabilities of our GA. The main limitation of the datasets used in the previous chapters is that they are very small. In order to assess the performance of the SOAP_GAS algorithm fairly and determine how well it will work on a larger range of molecular structures, we decided to use a significantly larger dataset.

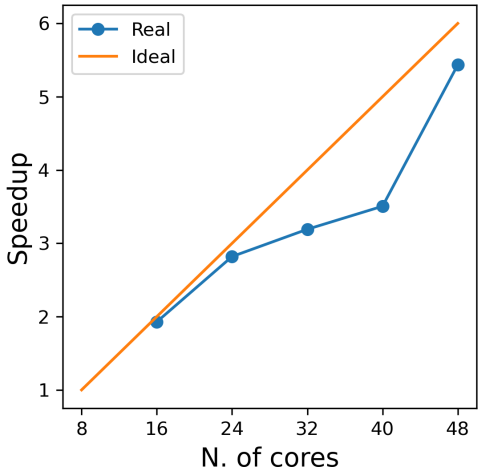

Figure 5.3: Scaling test for SOAP_GAS using `concurrent.futures` with a popSize of 48

The dataset we used is a prototypical dataset for drug design and discovery, which includes ~ 6000 small drug-like molecules and the values of their solubility in water as the target functional property. We stress that the aim of this chapter is not to advance the state-of-the-art with respect to this specific application of ML for drug design and discovery. Instead, we have picked this rather popular ML application so as to showcase the potential and general applicability of SOAP_GAS to any given molecular dataset.

We have found that, at least in the case of this particular "solubility" dataset, a number of significantly different combinations of SOAP parameters can result in similarly accurate models. Whilst some weak correlations exists between the different SOAP parameters, it may be concluded that pinpointing efficient combinations based on physical intuition alone is not an efficient strategy. Instead, SOAP_GAS offers a straightforward framework to identify these optimal combinations of SOAP parameters. It represents a solid alternative to the commonly used randomised grid search approach, which can prove rather inefficient

H **80119** (6068)	Cl **3319** (1270)	P **269** (246)	K **8** (5)	As **3** (3)
C **65389** (6119)	S **1383** (990)	I **114** (78)	Sn **5** (5)	Ge **2** (1)
O **14738** (4894)	F **699** (321)	Si **35** (16)	Hg **5** (5)	Ba **2** (2)
N **7302** (3289)	Br **373** (224)	Se **9** (6)	Zn **4** (4)	Ca **1** (1)
Mn **1** (1)	Cu **1** (1)	Sr **1** (1)	Ag **1** (1)	

Table 5.1: Frequency by which each atomic species appears in the dataset. The overall occurrences are reported in bold text, whilst the number of molecules containing a given atomic species are reported in parenthesis.

/ sub-optimal when dealing with the concurrent optimisation of the SOAP parameters of multiple SOAP vectors - which we have found to often result in more accurate descriptors when compared to concatenation of individually optimised SOAP vectors.

We have chosen to apply the SOAP_GAS algorithm to a dataset containing the SMILES [53] strings of 6,119 drug-like molecules and their solubility [62]. The solubility (S) i.e. the extent to which a chemical substance can dissolve in a solvent and form a homogeneous solution, is customarily represented using the base 10 logarithm as $\log S$, with S in moles per litre units [58, 59]. This dataset was curated by merging several sub-datasets containing solubility values characterised by an uncertainty inferior to 0.4 $\log S$ so as to maximise the reliability of the experimental data (a notorious issue when dealing with solubility measures) quality. This particular threshold in terms on uncertainty corresponds to the standard deviation relative to the sets of experimental measurements of S obtained for the same compounds by different research groups [62]. Prior to use, we discarded 35 compounds that were either inorganic (i.e. they contained no C atoms) or contained counter-ions.

To generate three-dimensional molecular models from the SMILES strings, we employed the `make3D()` method from Pybel, by performing 50 steps of geometry optimisation via the MMFF94 force field [167]. We note that this is not a sophisticated approach [168], particularly if compared to methods such as ensemble descriptors [169], where several different conformations are generated, optimised and evaluated for any given molecular structure. However, as previously stated, this work does not seek to improve on the current performance of ML methods in the context of predictive models for solubility. Instead, we are aiming to illustrate the potential of SOAP_GAS – and to that end, any realistic three-dimensional rendition of the SMILES strings will serve to illustrate the differences between optimised and non-optimised SOAP descriptors.

As shown in Figure 5.2, the dataset is characterised by a $\log S$ range between -13.2 and 1.58, and a mean of -2.78. Overall, the target values are distributed rather homogeneously, albeit one can notice a tail in the distribution corresponding to low solubility val-

ues (i.e. log$S$ < -6) which we expect to prove difficult to deal with in that they are under-represented within the dataset.

Table 5.1 reports instead the frequency by which each atomic species occurs within the dataset. Unsurprisingly, given the nature of the dataset, C, N, O and H are the most numerous, with a significant population of Cl and S as well. Several species such as K or Mn are only present within a handful of molecules, hence they will be omitted when constructing most SOAP vectors.

5.3 Results

5.3.1 Optimising individual SOAPs

As a first test of the SOAP_GAS framework, we have applied it to a number of different SOAP vectors characterised by different combinations of atomic species as *centres* and *neighbours*. In particular, we have built an "all-all" SOAP where the ten most abundant species (see Table 5.1) have been used as both *centres* and *neighbours* [164]. We have also built ten different SOAP vectors where each of the ten most abundant species has been used as $centre$ whilst all of the ten have been used as *neighbours*. Finally, we have also considered what we call a "double" SOAP, i.e. "all-all(double)", which consists of two all-all SOAP's using different values for the *cutoff* and $atom\_sigma$, allowing for short and long range structure to be described with different resolutions.

In terms of the search space for the SOAP_GAS algorithm, we have chosen the following, rather wide range: $2 < n_{max} < 10$, $2 < l_{max} < 10$, $5 < cutoff < 20$, and $0.1 < atom\_sigma < 1.5$. Note that these boundaries of the cutoff have been superseded by the following limits in the case of short/long SOAP in the context of the all-all(double) SOAP: $5 < cutoff < 12$ and $12 < cutoff < 20$ for short and long all-all SOAP, respectively.

The results are reported in Table 5.3 relative to the solubility discussed in 1.3. The "Vanilla" data refer to the results obtained via the following non-optimised set of SOAP parameters: *cutoff*=5, l_{max}=6, n_{max}=12 and *atom_sigma*=0.5 which have been taken off-the-shelf from the online documentation of the SOAP descriptor [164]. The "Vanilla" parameters are a set of sensible parameters that should work fairly well in most usecases. The "GA" data refer to the results obtained via the SOAP_GAS algorithm. We report the score metric described in Eq. 5.2 together with both the MSE and PCC (including the associated uncertainties as the standard deviation accumulated of a 5-fold cross validation). It is clear that applying the SOAP_GAS algorithm consistently results in SOAP vectors corresponding to more accurate models in all cases.

The greatest improvements in terms of accuracy can be appreciated for those SOAP vectors whose centres correspond to frequently occurring atomic species in the data set, such as C, H, and O. Conversely, the gains are only marginals for SOAP vectors with e.g., P or I as centres. This is expected, as the predictive power of those descriptors is bound to be rather weak given the only minimal occurrence of those species in the data set.

We note that the computational cost of dealing with the all-all SOAP stretched the capabilities of "regular" computing nodes. Dedicated high-memory computing nodes are often needed when dealing with SOAP vectors. Rather than adopting that approach, here we instead chose to take advantage of the compression scheme described in Ref. 170. Within this scheme the SOAP power spectrum is compressed through a combination of projecting the atomic neighbour density onto the surface of the unit sphere, which reduces the radially sensitive body order, and summing over the neighbour densities of different species, which reduces the element sensitive body order. Combining these operations in different ways leads to nine distinct options ranging from the full power spectrum, where the length scales as $\mathcal{O}(n_{max}^2 S^2 l_{max})$ to the most extreme compression which scales as $\mathcal{O}(l_{max})$.

μ	$\hat{\mu}$	ν	$\hat{\nu}$	Dim.	Score
0	2	0	0	8	3.756
2	0	0	0	22	2.845
1	1	0	0	15	2.587
0	1	0	1	71	0.739
0	0	0	2	386	0.584
0	1	1	0	141	0.445
1	0	1	0	281	0.442
0	0	2	0	1471	0.368
0	0	1	1	1401	0.366

Table 5.2: Compressing the SOAP vector allows to substantially reduce the dimensionality of the descriptor whilst retaining most of its predictive power. Dimensionality (Dim.) and Score (see text) for the all-all SOAP according to different choices of compression.

In Table 5.2, we report the dimensionality as well as the accuracy of the all-all SOAP obtained with these different levels of compression and denote them using the same notation as in Ref. 170; note that the μ=0, $\hat{\mu}$=0, ν=2 and $\hat{\nu}$=0 option corresponds to the original uncompressed SOAP vector. A schematic taken from Ref. 170 is shown in Fig.5.4 depicts how these compression parameters work. In light of the results reported in Table 5.2, we chose to apply the μ=0, $\hat{\mu}$=1, ν=1 and $\hat{\nu}$=0 option as it provides an excellent compromise between accuracy and compression. In particular, this combination retains much of the accuracy of the non-compressed SOAP vector (with a score of 0.445 against a score of 0.368) whilst drastically reducing the dimensionality from 1471 to 141 elements. The loss of accuracy is to be expected, as this level of compression does not preserve information, although it is interesting that accuracy is no worse than with μ=1, $\hat{\mu}$=0, ν=1 and $\hat{\nu}$=0, which, subject to certain conditions, is known to preserve information. All the results presented in

this section have been obtained using μ=0, $\hat{\mu}$=1, ν=1 and $\hat{\nu}$=0.

It is informative to look for correlations between the four SOAP parameters, as well as the resulting fitness score. To this end, we have chosen the all-all SOAP, where every atomic species within the dataset is used as both centre and neighbor - with the exception of atomic species occurring in less than fifty molecules across the entire dataset (see Table 5.1) so that $S = 10$. We have run 96, independent instances of SOAP_GAS, where the initial values of each set of SOAP parameters for each individual within the initial population have been randomly selected. The SOAP parameters that resulted in the best fitness score for each run are collected in Fig. 5.5. Overall, it is fair to say that there are no strong correlations between any of the SOAP parameters. This is quite interesting, as one might think that an increase in e.g., *cutoff* should be accompanied by a larger n_{max}, as the greater spatial extent of the local atomic environment might need a greater number of radial basis functions. However, this is not the case. In fact, none of the SOAP parameters seem to be strongly correlated with the fitness score. Again, this is somehow counter intuitive, as one might expect the SOAP vector to capture a greater deal of information about the molecular structure when increasing e.g., the number of basis functions. As such, it appears that physical intuition alone might not suffice to guide the choice of the SOAP parameters - hence the need for an optimisation strategy such as the one offered by SOAP_GAS.

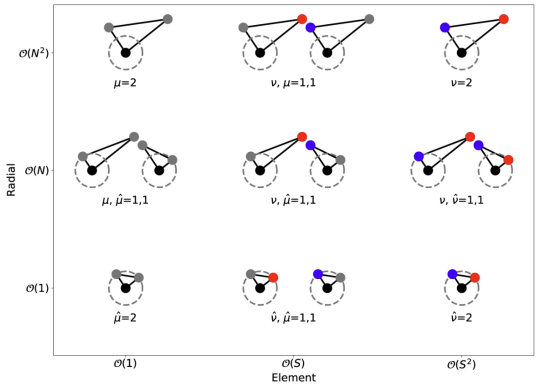

Figure 5.4: different combinations of μ, $\hat{\mu}$, ν and $\hat{\nu}$ values. In the case of 3-body terms, where $\mu+\hat{\mu}+\nu+\hat{\nu}$ equals 2, we exclude zero indices. The vertices highlighted in blue and red are responsive to specific elements, while the ones in grey are not. The dashed grey line represents the unit sphere. If the projection yields two separate triangles, both are displayed; otherwise, only one is presented.

The results reported in Fig. 5.5 also allow us to draw some conclusions in terms of the reproducibility of the SOAP_GAS results. Namely, there are specific combinations of SOAP parameters that tend to feature much more prominently than others, as illustrated by the histograms in Fig. 5.5. Whilst it is perfectly possible for the SOAP_GAS to yield very different combinations of SOAP parameters that ultimately offer the same accuracy, the $n_{max} = 9$, $l_{max} = 9$, *cutoff*=11 and *atom_sigma* = 0.8 scenario appears to be consistent in improving

the fitness score for the all-all SOAP vector.

The performance of SOAP_GAS itself might depend to an extent upon the choice of specific GA parameters. This aspect is investigated in the next section.

	Score	
	Vanilla	GA
all-all	0.483	0.309
all-all (double)	0.317	0.269
C-ten	0.573	0.341
H-ten	0.749	0.402
O-ten	2.66	1.963
Cl-ten	3.925	3.677
N-ten	6.047	5.428
S-ten	10.987	10.529
F-ten	11.688	11.369
Br-ten	12.412	12.114
P-ten	12.207	12.159
I-ten	14.361	14.267

	MSE			
	Vanilla		GA	
	Test	Train	Test	Train
all-all	1.43 ± 0.108	1.152 ± 0.018	1.154 ± 0.065	0.929 ± 0.015
all-all (double)	1.186 ± 0.095	0.924 ± 0.015	1.106 ± 0.069	0.855 ± 0.015
C-ten	1.569 ± 0.107	1.252 ± 0.016	1.209 ± 0.067	0.973 ± 0.013
H-ten	1.748 ± 0.105	1.427 ± 0.037	1.293 ± 0.076	1.074 ±0.02
O-ten	2.935 ± 0.235	2.683 ± 0.055	2.580 ± 0.202	2.325 ± 0.044
Cl-ten	3.394 ± 0.189	3.192 ± 0.037	3.265 ± 0.176	3.12 ± 0.38
N-ten	4.068 ± 0.178	3.83 ± 0.042	3.869 ± 0.17	3.676 ± 0.037
S-ten	4.891 ± 0.192	4.758 ± 0.044	4.812 ± 0.181	4.708 ± 0.04
F-ten	4.945 ± 0.229	4.841 ± 0.058	4.877 ± 0.238	4.823 ± 0.06
Br-ten	5.003 ± 0.176	4.915 ± 0.041	4.948 ± 0.176	4.902 ± 0.043
P-ten	4.968 ± 0.211	4.908 ± 0.05	4.955 ± 0.212	4.905 ± 0.05
I-ten	5.077 ± 0.208	5.057 ± 0.053	5.071 ± 0.208	5.057 ± 0.053

	PCC			
	Vanilla		GA	
	Test	Train	Test	Train
all-all	0.85 ± 0.008	0.884 ± 0.001	0.881 ± 0.003	0.907 ± 0.001
all-all (double)	0.877 ± 0.005	0.908 ± 0.001	0.887 ± 0.002	0.916 ± 0.001
C-ten	0.835 ± 0.006	0.875 ± 0.001	0.875 ± 0.002	0.902 ± 0.001
H-ten	0.812 ± 0.006	0.853 ± 0.003	0.867 ± 0.002	0.893 ± 0.002
O-ten	0.654 ± 0.019	0.694 ± 0.005	0.705 ± 0.013	0.742 ± 0.004
Cl-ten	0.577 ± 0.016	0.61 ± 0.002	0.598 ± 0.014	0.621 ± 0.003
N-ten	0.45± 0.022	0.503 ± 0.005	0.49 ± 0.023	0.53 ± 0.005
S-ten	0.194 ± 0.019	0.259 ± 0.004	0.231 ± 0.011	0.275 ± 0.004
F-ten	0.166 ± 0.021	0.219 ± 0.006	0.202 ± 0.027	0.225 ± 0.006
Br-ten	0.124 ± 0.023	0.183 ± 0.005	0.161 ± 0.023	0.188 ± 0.005
P-ten	0.151 ± 0.021	0.186 ± 0.003	0.157 ± 0.018	0.186 ± 0.003
I-ten	0.03 ± 0.015	0.067 ± 0.004	0.047 ± 0.003	0.067 ± 0.053

Table 5.3: *SOAP_GAS improves the accuracy of any given SOAP vector.* Search space: $2<n_{max}<10$, $2<l_{max}<10$, $5<cutoff<20$, and $0.1 < atom\_sigma < 1.5$. We report the score metric described in Eq. 5.2 together with both the MSE and PCC (including the associated uncertainties as the standard deviation accumulated of a 5-fold cross validation). The μ=0, $\hat{\mu}$=1, ν=1 and $\hat{\nu}$=0 combination has been used in terms of compression (see text).

5.3.2 SOAP_GAS: performance tuning

As opposed to brute-force optimisation approaches such as grid searches, genetic algorithms are characterised by a number of parameters that allow for the fine tuning of the algorithm. In this section, we explore the impact of these parameters on both the accuracy of the resulting SOAP descriptors as well as the computational time needed, on average, to converge the GA to a satisfactory accuracy. In particular, we have:

- The mutation chance, `mutationChance`.
- The population size, `popSize`.
- The early stopping criteria `earlyStop` and `earlyNum`
- The ratio between the number of children `numChildren` and the number of individuals `luckyFew` that are randomly selected as parents notwithstanding their fitness score.

We begin with Fig. 5.6 a), where we report the evolution of the fitness score with the number of generations for a set of GA characterised by different `mutationChance`. It is clear that introducing a sufficient level of mutation is key for the GA to avoid getting stuck into a particular region of the phase space (`mutationChance` $= 0.1$ in Fig. 5.6 a)). Moving from a specific GA to the result reported in Fig. 5.6 c), which has been averaged over 48 different GA, we conclude that a `mutationChance` greater or equal to 0.20 introduces, for this specific dataset at least, a sufficient degree of flexibility into the GA. Broadly speaking, we envisage the occurence of a compromise between accuracy and computational effort, as lower and higher values results in terms of `mutationChance` might result in inferior accuracy and substantially higher number of generations, respectively. However, it is important to note that - exception made for very low `mutationChance` - the extent of mutation has a relatively minor impact on the performance of SOAP_GAS in this case.

A similar compromise in terms of accuracy vs computational effort can be observed when varying `popSize`, i.e. the size of the GA population. As illustrated in Fig. 5.6 b), increasing `popSize` results in an improvement in terms of the fitness score. As a larger population size allows the GA to explore the search space more effectively, the number of generation needed to converge SOAP_GAS tends to shrink as `popSize` increases. However, in terms of computational time we need to consider that the total number of machine learning models evaluated within the GA is equal to the number of generations multiplied by `popSize`. As such, increasing `popSize` appears to be a feasible strategy to improve the

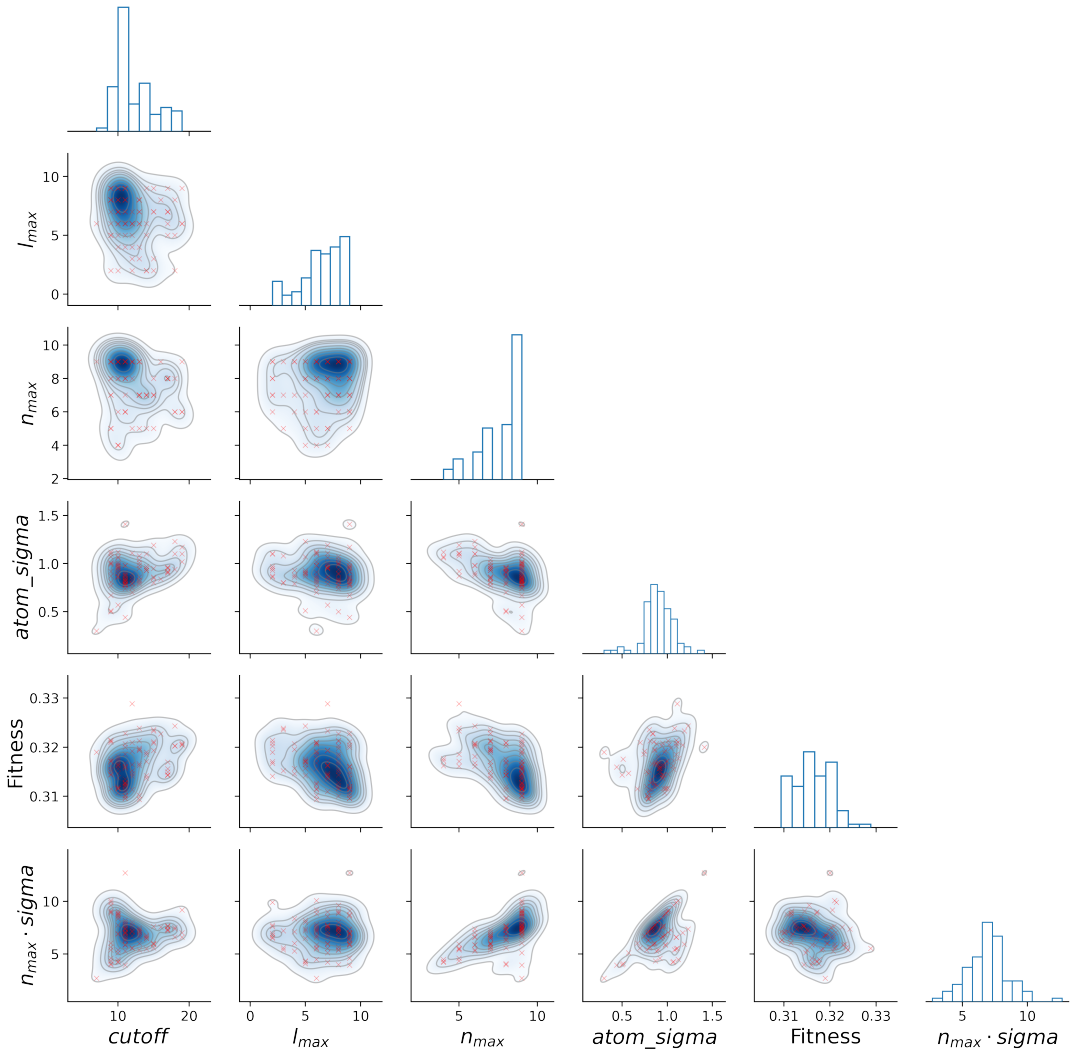

Figure 5.5: *There are no simple (cor)relations between the different SOAP parameters.* Correlations between the SOAP parameters for the all-all SOAP, as well as the fitness score. The scatter plots refer to the best set of SOAP parameters obtained for 96 statistically independent GA.

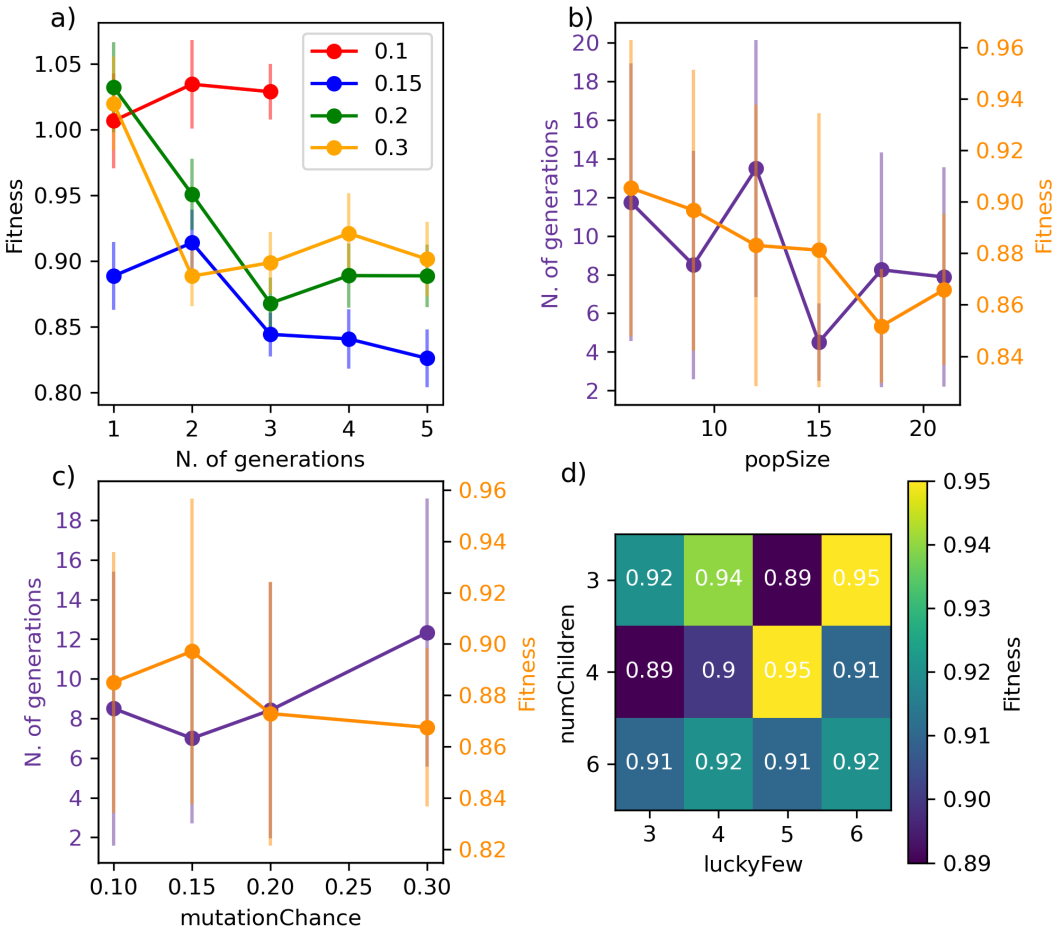

Figure 5.6: *SOAP_GAS offers a reliable performance notwithstanding the choice of parameters concerning the underlying GA.* Impact of the different SOAP_GAS parameters on both accuracy (fitness score) and computational cost (n. of generations). a) Learning curves for different mutation chances for a specific GA. b) Impact of `popSize` c) Impact of `mutationChance` d) Impact of ratio between `numChildren` and `luckyFew`. The data in panels b)/c) and d) have been averaged over 48 and 96 statistically independent GAs respectively.

overall accuracy of the SOAP descriptors - bearing in mind the associated increase in computational effort.

Interestingly, we have found that the early stopping criteria (i.e. `earlyStop` and `earlyNum`) have a negligible impact on the performance of SOAP_GAS. This is encouraging, as it serves to highlight the robustness of the algorithm. On the other hand, the ratio between `numChildren` and `luckyFew` can have a significant impact as illustrated in the heat map reported in Fig. 5.6 d). In principle, increasing the fraction of `luckyFew` relative to `numChildren` equates to increase the flexibility of the GA, by introducing an element of randomness that should be akin to the effect of the mutation process. However, we have found that this particular parameter behaves much more erratically than `mutationChance`, in that specific `numChildren` / `luckyFew` ratios seems to lead to fairly different accuracy. Note that as many as 96 statistically independent GA (i.e. started with different random combinations of SOAP parameters for each individual within the initial population) have been used to obtain the heat map in Fig. 5.6 d). Thus, while the impact of this parameter is not drastic, it is certainly worth exploring different choices.

In light of these findings, we can conclude that SOAP_GAS is rather robust with respect to the choice of the GA parameters.

5.3.3 SOAP_GAS timing accuracy: comparison with grid search

Having established that SOAP_GAS provides robust results, notwithstanding the specific choice of the GA parameters discussed in the previous section, we can now attempt to make a comparison between SOAP_GAS and what is potentially the most straightforward alternative approach to optimise SOAP parameters, namely a randomised grid search (RSCV). The latter simply involve a trial-and-error procedure whereby a certain number of SOAP characterised by randomly chosen SOAP parameters are evaluated and scored.

To offer a fair comparison, we have used the same search space for both RSCV and SOAP_GAS (see Sec. 5.3.1). In terms of centres and neighbors, we have used C atoms exclusively, and we have not applied any compression. The results are summarised in Fig. 5.7, where we report the fitness score for both RSCV and SOAP_GAS as a function of the number of individuals. In the case of SOAP_GAS, the number of individuals corresponds to the total number of SOAP descriptors evaluated across all the generations needed to converge the algorithm. As the time needed to obtain the fitness score for a given SOAP vector (which is the computational intensive step for both approaches) is exactly the same for either RSCV or SOAP_GAS, we can compare directly the interplay between accuracy and computational effort for the two methodologies. We have accumulated 84 and 96 statistically indepen-

dent instances of RSCV and SOAP_GAS respectively to obtain a robust statistics. Whilst the SOAP_GAS results are reported as a scatter plot (as different GA require different number of individuals to converge), the nature of the RSCV results allows us to assign (min-max, in this particular case) error bars to the accuracy corresponding to specific numbers of individuals.

The results for both RSCV and SOAP_GAS show a substantial degree of variability, thus strengthening the concept that multiple combinations of potentially even quite dissimilar SOAP parameters can provide similar results in terms of the accuracy of the SOAP vector. The number of individuals required to achieve a significant improvement in terms of accuracy is similar for RSCV and SOAP_GAS, albeit we found that in some cases SOAP_GAS manages to outperform the RSCV results, given the same number of individuals. This suggests that, overall, SOAP_GAS provides a framework which is equally efficient to the RSCV approach, with the potential to identify combinations of SOAP parameters which lead to greater accuracy.

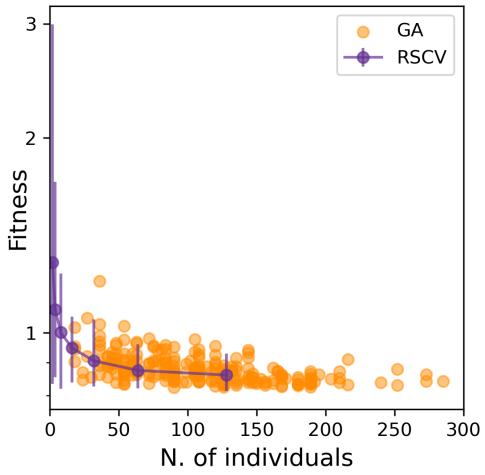

Figure 5.7: *The performance of SOAP_GAS is similar to that of a randomised grid search approach when dealing with the optimisation of a single SOAP descriptor.* Accuracy (fitness) as a function of the number of individuals taken into account for either the RSCV (purple markers and line) and the SOAP_GAS algorithm (orange markers). The error bars relative to the RSCV data are min-max error bars.

5.3.4 Working with multiple SOAPs

We would expect the accuracy of multiple SOAPs optimised *at the same time* (Conc. in Table 5.4) to be higher than that of multiple SOAPs optimised individually and *subsequently* concatenated (Ind. in Table 5.4). This is indeed the case, as illustrated by the results summarised in Table 5.4. In particular, we have chosen to focus on 5 different combinations of the ten $\mathscr{X}$-ten SOAP vectors (where $\mathscr{X}$ stand for one of the most abundant atomic species in the dataset, see Table 5.1). The gains in terms of accuracy are small but suggestive of the possibility that the set of the "best" SOAP parameters for different SOAP vectors considered as part of concatenated descriptor does depend to an extent on whether or not the individual SOAP vectors are considered in

Performance						
	Ind.			Conc.		
	Score	Time (s)	Gens.	Score	Time (s)	Gens
[C,H]	0.297	6061	8	0.291	1302	5.3
[C, H, O]	0.287	7201	11	0.29	1305	5
[C, H, O, Cl]	0.289	7966	14	0.281	1452	5.2
[C, H, O, Cl, N]	0.282	9019	17	0.271	1454	5.4
[Br, C, Cl, F, H, I, N, O, P, S]	0.283	12121	32	0.268	3655	4

MSE				
	Ind.		Conc.	
	Test	Train	Test	Train
[C,H]	1.144 ± 0.067	0.905 ± 0.014	1.14 ± 0.07	0.898 ± 0.013
[C, H, O]	1.138 ± 0.061	0.883 ± 0.011	1.141 ± 0.062	0.89 ± 0.011
[C, H, O, Cl]	1.139 ± 0.058	0.889 ± 0.01	1.165 ± 0.070	0.926 ± 0.013
[C, H, O, Cl, N]	1.125 ± 0.051	0.880 ± 0.011	1.105 ± 0.07	0.866 ± 0.011
[Br, C, Cl, F, H, I, N, O, P, S]	1.125 ± 0.051	0.88 ± 0.011	1.113 ± 0.067	0.872 ± 0.016

PCC				
	Ind.		Conc.	
	Test	Train	Test	Train
[C,H]	0.883 ± 0.002	0.91 ± 0.001	0.884 ± 0.002	0.911 ± 0.001
[C, H, O]	0.884 ± 0.002	0.913 ± 0.001	0.883 ± 0.001	0.912 ± 0.001
[C, H, O, Cl]	0.883 ± 0.001	0.912 ± 0.001	0.885 ± 0.002	0.913 ± 0.001
[C, H, O, Cl, N]	0.885 ± 0.002	0.913 ± 0.001	0.888 ± 0.003	0.915 ± 0.001
[Br, C, Cl, F, H, I, N, O, P, S]	0.885 ± 0.002	0.913 ± 0.001	0.887 ± 0.003	0.915 ± 0.002

Table 5.4: *SOAP_GAS improves on both the accuracy and the computational cost for combinations of SOAP descriptors, optimised concurrently as opposed to individually - and subsequently concatenated.* Performance (Score, MSE and PCC, see main text) of individually optimised (and subsequently concatenated) combinations of SOAP (Ind.) compared with that of concurrently optimised combinations of SOAP (Conc.). The "Gens" columns report the number of generations needed to converge the SOAP_GAS algorithm. In the case of Ind., this number is the sum of the number of generations needed to converge the SOAP_GAS for each individual SOAP, whilst for Conc. this number is the average (taken over ten different runs) number of generations needed to converge the SOAP_GAS algorithm for the concurrent optimisation of the SOAP combinations. The "Time" columns refer to the total time needed to converge the SOAP_GAS algorithm. In the case of Ind., this number is the sum of the time needed to converge the SOAP_GAS for each individual SOAP, whilst for Conc. this number is the average (taken over ten different runs) time needed to converge the SOAP_GAS algorithm for the concurrent optimisation of the SOAP combinations.

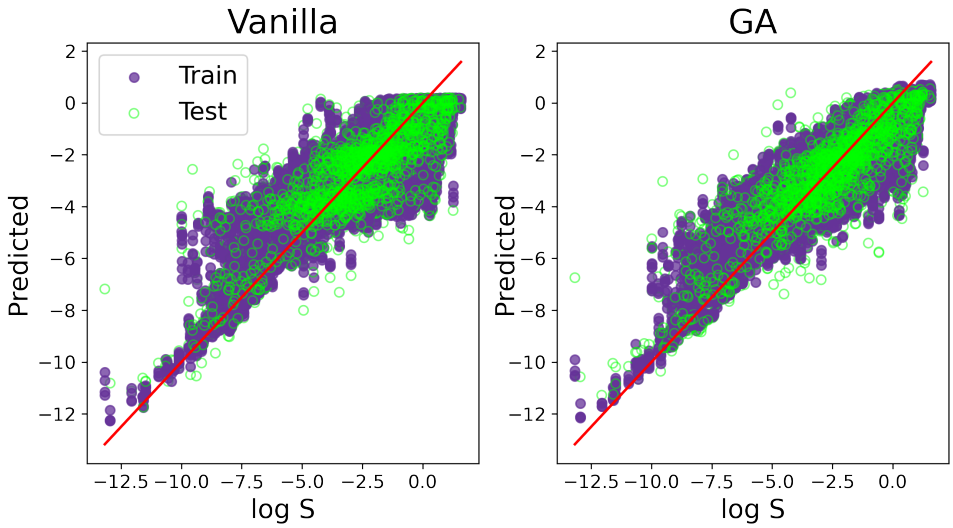

Figure 5.8: *A visual comparison of the accuracy improvement obtained via the SOAP_GAS algorithm applied to a specific SOAP descriptor.* "Vanilla" refers to the results obtained without optimising the SOAP parameters, whilst "GA" refers to the results obtained by applying the SOAP_GAS algorithm. Purple and green markers correspond to train and test prediction (obtained over a 5-fold cross-validation.). The results refer to the C-ten SOAP, with compression: μ=0, $\hat{\mu}$=1, ν=1 and $\hat{\nu}$=0. The SOAP parameters for the Vanilla results are: *cutoff*=5, l_{max}=6, n_{max}=12 and *atom_sigma*=0.5, which upon optimisation (GA) changed to *cutoff*=12, l_{max}=8, n_{max}=5 and *atom_sigma*=1.1.

isolation. To strengthen this claim, it appears that the accuracy gains increase when considering longer combinations of SOAP vectors - which in turn offer a larger parameter space to be optimised as a whole.

Importantly, the optimisation of multiple SOAP vectors at the same time is a situation where SOAP_GAS outperforms the RSCV approach, as illustrated in Fig. 5.9. Not only the results of SOAP_GAS are consistently more accurate than the RSCV ones, but it turns out that in this case the RSCV can barely manage to improve on the accuracy of the individual (concatenated) SOAP (obtained via RSCV), which corresponds to the blue, dashed line in Fig. 5.9. Conversely, SOAP_GAS pushes well below the red dashed line corresponding to the accuracy of the individual (concatenated) SOAP obtained via SOAP_GAS.

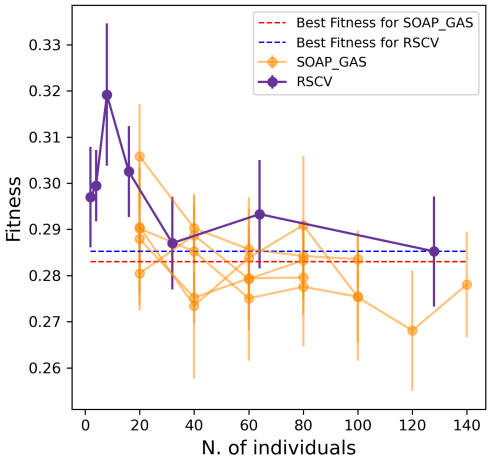

Figure 5.9: SOAP_GAS outperforms the randomised grid search approach when dealing with the concurrent optimisation of multiple SOAP descriptors. Interplay between accuracy and timing (as in, n. of ML models generated) for the concurrent optimisation of multiple SOAPs at the same time, RSCV vs GAs. The blue and red lines correspond to the best score of the same SOAP combinations where the different SOAP vectors have been optimised individually and subsequently concatenated. The performance of the SOAP_GAS algorithm was evaluated 5 times to ensure consistent results.

This finding is not entirely surprising, as when attempting to optimise multiple SOAPs at the same time via RSCV the number of grid points needed to converge increases massively. On the other hand, SOAP_GAS is inherently better equipped to explore the search space in a more clever fashion, steering the SOAP parameters toward different local minima without wasting time in probing regions of the phase space that result in very low accuracy.

In addition, we have found that - when using SOAP_GAS - optimising multiple SOAP vectors at the same time is less computationally expensive then optimising each SOAP vector individually. This is illustrated in Table 5.4, where we compare the number of generations ("Gens." column - or, equivalently, the time) needed for the SOAP_GAS to converge in these two distinct scenarios. Crucially, the more SOAP vectors we include, the larger the divide in terms of com-

putational cost between individual and concurrent optimisations. For instance, a total of 32 generations are needed to converge the [Br,C,Cl,F,H,I,N,O,P,S] SOAP vectors individually, whilst only 4 generations are - on average - required to converge this combination of SOAP vectors concurrently. Note that the same number of individuals per generation has been used to craft this comparison. This is important, as it demonstrates that the SOAP_GAS framework can be used to efficiently optimise combinations of different SOAP vectors at the same time, which, as we have seen, also typically leads to further (if rather small) gains in terms of the overall accuracy of the descriptor as well.

We conclude our discussion by offering a visual comparison of the improvement, in terms of accuracy, obtained by applying SOAP_GAS to this particular dataset. The result in Fig. 5.8 have been obtained with the C-ten SOAP, $\mu = 0$, $\hat{\mu} = 1$, $\nu = 1$ and $\hat{\nu} = 0$ compression. The initial SOAP parameters were $cutoff = 8$, $l_{max} = 6$, $n_{max} = 2$ and $atom\_sigma = 0.5$, which upon optimisation changed to $cutoff = 12$, $l_{max} = 8$, $n_{max} = 5$ and $atom\_sigma = 1.1$. The MSE(PCC) for the Test set improved from 1.515(0.839) to 1.209(0.875).

Solubility measurements found in literature are usually characterised by experimental uncertainties of the order of $\pm 0.5 - 0.7$ log S [171]. On account of that, ML models for predicting solubility having MSEs (for the test set) in the range of $0.5-1.2$ are generally considered to be good [61]. Additionally, recent solubility models leveraging ML algorithms, including random forest regression, have produced PCCs (again, for the test set) in the range of $0.81-0.95$ [172, 58, 173, 174]. As shown in Table 5.4, the SOAP_GAS solubility model resulted in MSE and PCC values that lie in those ranges for both the training and test sets, this is not always the case for the vanilla parameters. Although our model MSE values may fall in the upper range of what is favourable, we highlight that most of the models cited used comparatively much smaller test sets (less than 100 molecules, to be compared with ~ 2000 in our case) than the ones we have considered in this work.

To conclude, our research endeavors have culminated in a compelling demonstration of the tangible benefits derived from the integration of GAs into the optimisation process. These advancements have notably and substantially enhanced the overall performance and efficacy of the SOAP descriptor. Moreover, we have introduced and employed our specialized SOAP_GAS package, which has proven to be a remarkably efficient and user-friendly tool for the systematic optimisation of SOAP descriptor parameters. Importantly, our investigations have extended to encompass a diverse spectrum of SOAP variants, encompassing not only simple single SOAP implementations but also more complex configurations, such as double concatenated SOAPs and SOAPs with single species centres.

Chapter 6

A Materials Science-inspired Paradigm to Predict the Physical Stability of Amorphous Drugs

6.1 A Materials Science-inspired Paradigm to Predict the Physical Stability of Amorphous Drugs

The present chapter aims to explicate the outcomes derived from our predictive models concerning the Amo-Reg and Amo-Class datasets. Specifically, we shall delve into the performance of our machine learning models and highlight their inherent strengths and limitations.

To commence, we shall provide a detailed description of the methods employed in constructing, improving, and evaluating the models. Furthermore, we shall furnish a case study that elucidates the significance of feature selection in the modeling process.

Moreover, this chapter encompasses an exhaustive analysis of the ensemble methods utilized in our study, and we shall assess their efficacy in comparison to the individual models. We shall also present information on how the number of models included in the ensemble influences the results obtained.

In conclusion, we shall propose areas of future research that can improve the models' performance. We shall accentuate the constraints of our study and offer recommendations on how to overcome these limitations in subsequent research endeavors. In essence, this chapter offers a comprehensive overview of the machine learning analysis results and provides a fundamental basis for further deliberation and interpretation of our research

findings.

6.2 Machine learning workflow

In this section we outline the workflow we used for creating, optimising, and testing the models used for the Amo-Tg and Amo-Class datasets.

6.2.1 Preprocessing

For both the Amo-Reg and Amo-Class datasets, we follow the same preprocessing steps for all the single-molecule descriptors. We begin by normalising the data to values between 0 and 1 using min-max scaling as detailed in section 3.5.1. However, in the case of solid-state descriptors we must take care with the normalisation. Firstly, in the case of the computed T_g we do not normalise the data. This is because the descriptor is only a single feature and is the same magnitude as the target. There were also no outliers in the data, so in theory, normalisation be superfluous. Then, for the DC and RT descriptors, we have a range of values in the features that spans from very small to very large. This is especially prevelant in the case of relaxation time where we have molecules that do not reach the $1/e$ threshold mentioned in section 2.3.12. In these cases, we set the relaxation time to simply be a very large number, in our case 9,999. We acknowledge that this is not ideal, however it is challenging to deal with missing data elegantly in these sort of cases, especially when data is so limited that it is not feasible to drop null data. Now, due to the large range that each feature may take, using min-max scaling would be ineffective. We instead use logarithmic scaling as described in section 3.5.1.

Next, we eliminate features with zero variance as described in section 3.6.1. For most descriptors, this does not remove any features since most have some degree of variance. However, as previously mentioned, in the case of the H-wACSF descriptor, features with zero variance are relatively prevalent.

6.2.2 Parameter optimisation

Once the preprocessing is completed, we must now select the best parameters to use for our models. As shown in section 3.3 and 3.2, the parameters we choose for our NNs and RFs can have a profound effect on the overall results.

Since this is such an important step, we invest a significant amount of computational resources to find the optimal parameters. The methodology for this involves a comprehen-

sive grid search of a number of parameter sets.

Neural Networks have two parameter values we need to optimise, namely the regularisation parameter λ (see section 3.6.5) that can take values in the set [0.01, 0.001, 0.0001] and the scaling factor (see section 3.3.1) which dictates the network architecture and can take values in the set [0.2, 0.4, 0.6, 0.8, 1].
Random Forests have three parameters that we seek to optimise, the number of estimators (see section 3.4.1) which can be any of the values in the set [80, 100, 120, 140]. Max depth (see section 3.4.2) which takes values in the set [4, 8, 16, 32, None], and finally the minimum split size (see section 3.2) which takes values in the set [2, 4, 6, 8].

For both NNs and RFs, models are trained using every permutation of values for each parameter. Clearly this results in a lot of models being trained, in the case of RFs we have to train 100 models to find the optimal parameters. Furthermore, since these models are validated using LOOCV, that means for each set of parameters we have N test sets where N is the number of molecules in the dataset. So in the case of RFs for the Amo-Reg dataset we need to train 13,600 models to optimise a single descriptor. Admittedly, with some thought and exploratory analysis of each descriptor, it may be possible to manually choose a smaller set of parameters that would be suitable for each descriptor. However due to the scope of this work, it was more efficient to simply train a large number of models for each descriptor.

Once the optimal set of parameters have been found, the results from the LOOCV are reported in tables 6.1 - 6.4.

6.2.3 Genetic algorithm

To further improve our results, we use a GA on appropriate descriptors that require hyper-parameter optimisation, namely the SOAP and H-wACSF descriptors. In depth details of the GA, as well as a case study on their effectiveness with different datasets can be found in chapter 5. The GA for the results in this chapter is performed in the same way as chapter 5 with a key difference. The set of hyperparameters is evaluated using the parameter optimisation method outlined in section 6.2.2. This means that for each individual created in the GA, we are computing training up to 100 models and evaluating them using LOOCV. Obviously this leads to a very computationally expensive algorithm, but unfortunately it is unavoidable. If we did not use the parameter optimisastion or CV method then we can not be sure that the GA is going to result in the best overall result. For example, if we used 5-fold CV instead of LOOCV, the computational time would be much better, but we would not be able to fairly compare it to the models for which we used LOOCV as LOOCV has a signif-

icantly larger training set. The results of the GA optimsiation on the SOAP and H-wACSF descriptor for both RF and NN models can be found in tables 6.4 - 6.7.

6.2.4 Feature Selection

In order to reduce noise and remove redundant parameters that have not been removed with the variance threshold, we use backwards feature elimination as outlined in section 3.6.3. Again, each time a feature is removed, we must train models for every possible set of parameters in order to remain consistent with our results.

The results of the backwards feature elimination are given in tables 6.4 - 6.7 for NN and RF models. We do note that the feature selection appeared to perform in a rather underwhelming way for these datasets, it often failed to improve the results when compared to the descriptors with no features removed. This was especially prevelant in the optimised descriptors. We attribute this to the fact that since those parameters are optimised using a GA, by removing features we are actually losing information and thus the performance does not improve. However, we include a closer look at feature selection in section 6.3 where we make an argument for the importance of feature selection.

6.2.5 Ensemble methods

We use ensemble methods to further improve the predictive capabilities of our models. At this stage, each descriptor has two optimised models, one random forest and one neural network, that provide the best score for their type of model. By combining a selection of these models we can produce predictions that outperform that of the individual descriptors by themselves.

- **Max voting** An outline of the protocol for max voting can be found in section 3.9.2. We use this max voting strategy for both the Amo-Res and Amo-Class datasets with NN and RF models. We found that by combining the predictions from our two best performing models, the quality of predictions improved. The results of this max voting framework are shown in Fig. 6.7

- **Ensemble neural networks** As described in section 3.9.3, we utilise an ensemble of NN's to see if multiple, concatenated NN's using different descriptors can learn to produce more accurate predictions than individual NN's using a single descriptor. The way that this has been evaluated is by choosing every possible combination of two or more descriptors. Once these descriptors have been chosen, a complex NN is then created using the parameters that yielded the best results for the respective descriptor.

These NNs are concatenated in a concatenation layer which is then linked through a hidden layer to the output node/s. A schema for an example of this NN structure can be found in 6.1.

This model assumes that the optimised values for the individual descriptors NNs also represent the optimal values for their respective portions of the complex NN ensemble. We understand that this may not be the case, however we argue that on the basis that they were the best parameters for the individual NNs, they are highly unlikely to be below average parameters when used in a more complex, deep learning approach. Furthermore, since the number of combinations of descriptors is very high, performing optimisation techniques on these parameters would be exceptionally computationally expensive and in reality offer little benefit to the final result. With this being said, we do seek to optimise the parameters between the concatenation layer and the output layer. Using a similar methodology to optimising the individual NN models, we seek to optimise the parameters of the network going between the concatenation layer and the final hidden layer. This is done by again performing a grid search of the regularisation rate and scaling factor parameters for the NNs..

For this process we have 7 descriptors. This means that there are 120 possible combinations of descriptor $\left(2^7-\binom{7}{1}-\binom{7}{0}\right)$. For each of these combinations we perform a grid search that has 20 possible combinations. This gives us a total of 2,400 possible NN models that need to be trained. Again, these models are validated using LOOCV which multiplies the number of models required by 136 and 124 for the Amo-Reg and Amo-Class datasets respectively.

6.3 On the importance of feature selection

As reported throughout this chapter, we see that the gains in performance from FS are relatively negligible, in this section we provide results on different datasets that highlight the importance of FS.

Feature selection is often an overlooked part of developing machine learning models, in particular neural networks. It is tempting, given the availability of so many different molecular descriptors, to leverage as many of them as possible: for instance, the DRAGON software [175] can calculate more than 4800 descriptors [176]. As such, this approach is not only incredibly simple these days, but it may also enhance the flexibility of the ML algorithm of choice, in that the more descriptors we add into the mix, the higher the chances to include those features that are actually of relevance to improve the predictive capabilities

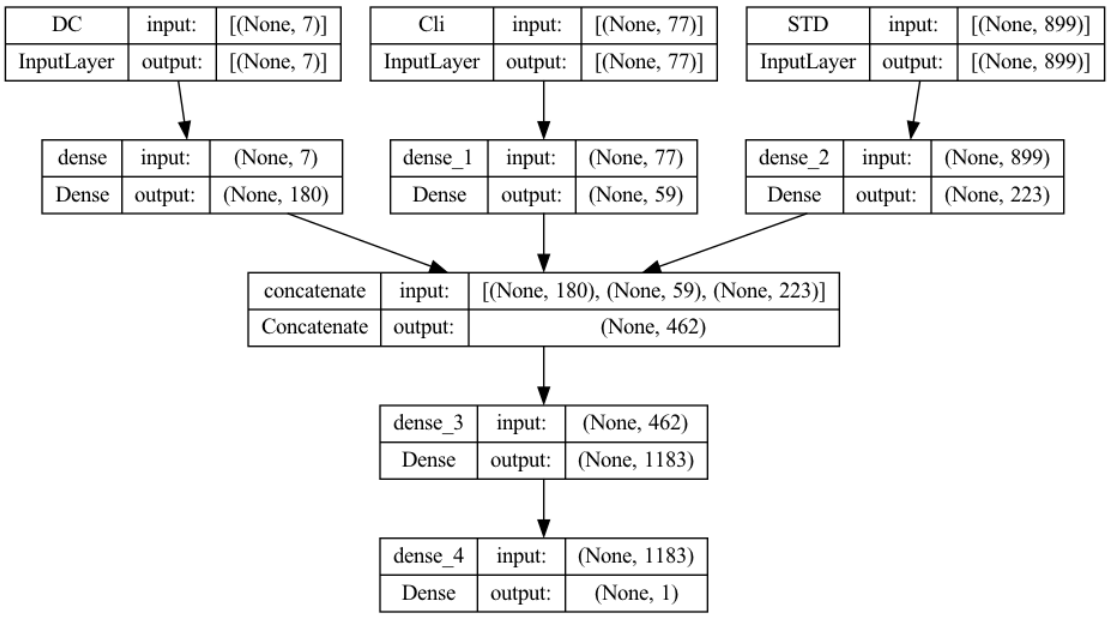

Figure 6.1: Ensemble neural network architecture example

of the framework [177]. However, this strategy suffers from at least two major issues: (1.) redundancy/correlation: the more descriptors we choose to use, the higher the chance they will feed similar if not identical information to the ML algorithm [178], with the risk of introducing artificial noise that can be detrimental to both the accuracy and the reliability of the predictive framework; (2.) lack of transparency [179, 180]: it becomes quite challenging to pinpoint the structural features that have the largest impact on the functional properties of interest. While from a purely practical perspective one may not care about this pitfall, understanding the structure-function relation is key to achieve the truly rational design of the novel generation of drugs [181].

Both redundancy and lack of transparency can be mitigated by using feature selection [28] and/or by optimising the parameters that often enter the formulation of advanced molecular descriptors. In this section, we compare how well Cliques and H-wACSFs compare to standard descriptors obtained from RDKit. Note that we do not consider SOAPs here as the purpose of this section is simply to highlight the importance of FS and calculating SOAPs for the large datasets we are using here would be computationally very expensive.

We make this comparison on the Hepa, Lipo and Amo-Reg datasets, and we show that, in some cases, utilising just a handful (10-20) of carefully designed molecular descriptors may yield results comparable - or even better - than those obtained by using a large

number ($\sim$ 100) of standard descriptors. We find that this is especially true when dealing with small datasets containing 100-500 molecular structures, where the number of STD descriptors that we may want to use can get dangerously close to the number of data points we intend to feed into our ML framework - an obviously over-determined problem.

We choose in particular to probe the efficiency of feature selection on cliques and H-wACSFs because the nature of cliques and H-wACSFs makes them perfectly suitable to exploit feature selection and optimisation, respectively. We find that a surprisingly small set of tailored descriptors, as obtained upon either feature selection (cliques) and optimisation (H-wACSFS), can provide results comparable, if not of better quality, than those we have obtained by employing large numbers of STD descriptors. While an analysis of the most relevant cliques obtained upon feature selection allows us to draw interesting conclusions about the influence of specific functional groups on biomedical activities of pharmaceutical interest such as human hepatocytes intrinsic clearance [57], the H-wACSFS offer a very convenient opportunity to bridge the ML gap from a single molecule in vacuum to 3D models of e.g. amorphous drugs. While an ongoing effort within our research group is probing the benefits of bringing together "chemistry and structure" by combining these two classes of descriptors, we have made available via a public GitHub repository [182] the entirety of our ML framework, in an effort to promote transparency and cross-fertilisation between different groups.

6.3.1 Methodology

In terms of the specific ML algorithm, we have been experimenting with multiple options, including neural networks, Gaussian processes and random forests. We have found that the choice of the ML algorithm has very little impact on our results. The numbers reported in the Results section have been obtained by using feed-forward neural networks, built using the Keras API [183] with Tensorflow [184] as a backend. The descriptors and the target properties for each dataset (Lipo, Hepa and Amo, see above) have been pre-processed by scaling them between zero and one and by removing the mean and scaling to unit variance, respectively. In terms of the neural networks optimisation, a simple parameter space grid search optimisation has been employed, taking into consideration different neural networks architectures (in terms of number of layers and nodes), different activation functions, and different solvers for the optimisation of the weights. Further details are included in the MSDE_Sosso_alpha GitHub repository [182].

As many as 300 epochs have been accumulated for each combinations of these parameters. The “optimal“ number of epochs was determined according to an early stopping

criterion based on the mean square error relative to the test set. 80% and 20% of the datasets have been randomly selected as training and test data, respectively, according to a k-fold cross validation [185] procedure with k=5 which allowed us to reliably assess the average performance of each neural network architecture. The "best" model was then selected and used to compute the results reported in Section 6.4. We note that while it is certainly possible to leverage more advanced techniques (e.g. some form of ensemble learning [186]) to improve the accuracy of these algorithms, we have focused in here to provide a rather unbiased picture of the performance of the different descriptors under consideration. As a result, the numerical quality of our results is on average not very impressive, albeit we envisage that both the Hepa and Amo-Reg datasets will probably provide a tough challenge in terms of accuracy for more advanced ML frameworks as well.

6.4 Results

The overall performance of the three classes of descriptors discussed in the previous section is summarised in Table 6.1: STD, Cliques and H-wACSFs refers to the $\sim$ 100 "standard" RDKit descriptors, the vocabularies of molecular cliques and the histograms of weighted atom-centred symmetry functions, respectively. We report the mean squared error (MSE) and the Pearson correlation coefficient (PCC) [187] for both the training and the test sets; averages and uncertainties (included as $\pm\frac{\sigma}{2}$) have been obtained according to the cross-validation procedure detailed in Section 3.8.1.

Concerning the Lipo dataset, STD outperform both cliques and H-wACSFs. The latter are clearly not very suitable to deal with this particular dataset. As discussed in further detail below, this was expected, given the nature of the target property to be predicted. On the other hand, by using the full set of cliques (i.e. without feature selection) one can achieve results of similar quality to those obtained via the STD - quite impressive, considering how basic the cliques descriptors are. Upon feature selection, specifically utilising only 15 cliques (out of 246), the performance of the cliques degrades further; however, being able to retain some predictive capabilities using 15 molecular fragments is indicative of the potential of this descriptor.

In fact, the cliques consistently outperform the STD in the case of both the Hepa and the Amo-Reg dataset: we remind the reader that while the Lipo dataset is relatively

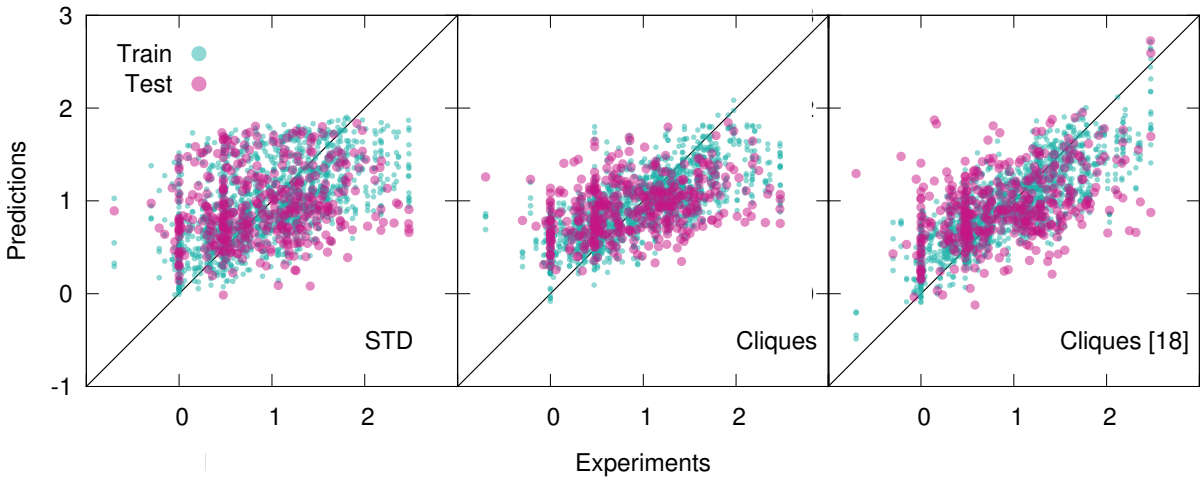

Figure 6.2: Scatter plots of the predicted *vs* experimental values of human hepatocytes intrinsic clearance for the Hepatocytes dataset, using ∼ 100 "standard" RDKit descriptors (STD), the full vocabulary of molecular cliques (Cliques), and just 18 out of 132 cliques (Cliques [18]) according to the outcomes of the feature selection procedure discussed in Section 3.8.1. The results obtained for five different training-test splits are plotted on the same graph, which thus contains 406x5 = 2030 points. Note the improvement of the predictions upon the cliques feature selection.

large (∼ 4000 molecules), the Hepa and particularly the Amo dataset are quite small (∼ 400 and ∼ 150 molecules, respectively). Interestingly, in the case of the Hepa dataset, using just the most relevant (according to the MDI-based feature selection procedure discussed in Sec. 3.6) 18 cliques (out of the 132 contained in the full set) results in even better outcomes compared to what we have obtained for the full set of cliques, as illustrated in Tabl. 6.2. This is an impressive result: just 18 molecular components appear to capture some of the structure-function relation at the heart of a complex biomedical activity such as human hepatocytes intrinsic clearance. As detailed in Table 6.2, these 18 cliques are characterised by an MDI about one order of magnitude larger compared to that of the least important cliques. We also note that the RF-based feature selection procedure we have used is capable to assign MDIs characterised by very small uncertainties, thus making the selection process quite reliable indeed. Amongst these 18 cliques we find molecular components such as CC, C=O, C1CCCCC1 (cyclohexane) and C1=CC=CC=C1 (benzene) which are ubiquitous in small drug-like molecules: in fact, they possess quite high MDI scores for the Lipo and Amo datasets as well. On the contrary, we also find cliques whose role in the context of human hepatocytes intrinsic clearance is perhaps not immediately obvious: CF, CS and C1=CSCN1/C1=NCCS1 (2,3/4,5-dihydrothiazole).

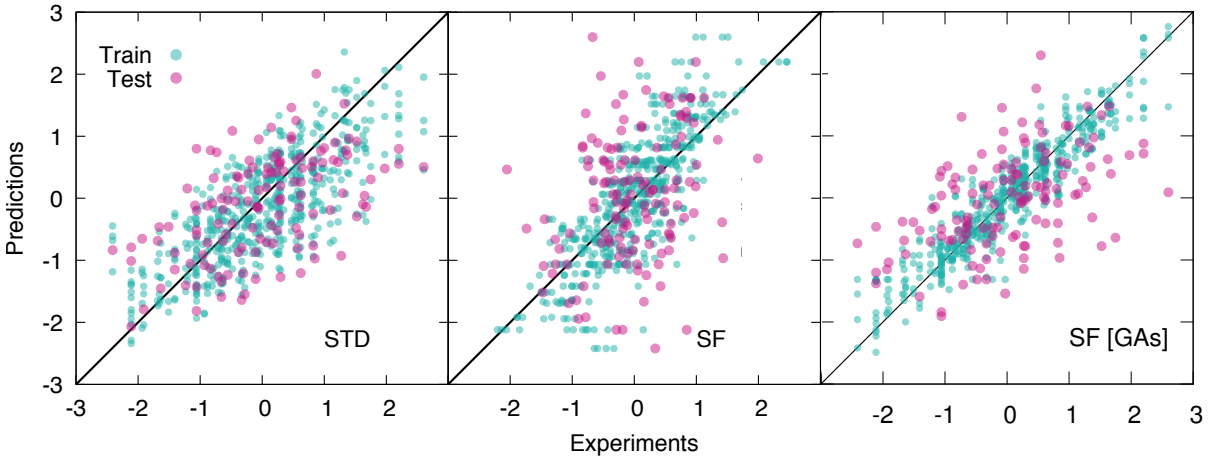

Figure 6.3: Scatter plots of the predicted *vs* experimental values of the glass transition temperature T_g for the Amorphous dataset, using $\sim$ 100 "standard" RDKit descriptors (STD), H-wACSFs (SF), and H-wACSFs optimised according to the genetic algorithms-based procedure describe in Chapter 5 (SF [GAs]). The results obtained for five different training-test splits are plotted on the same graph, which thus contain 132x5 = 660 points. Note the improvement of the predictions upon the H-wACSFs optimisation.

The situation is slightly different in the case of the Amo-Reg dataset: while using the full set of cliques results in a substantial improvement with respect to the STD outcomes, using 13 out of 87 cliques (according to the results of feature selection) worsens the numerical accuracy of our prediction. Nonetheless, this small set of cliques provides predictive capabilities of the same quality of STD - i.e. using 13 molecular components gives similar results to those obtained by using $\sim$ 100 different descriptors. Appropriately, our findings suggest that molecular cliques may represent, despite their simplicity, an interesting way forward to identify structural patterns of interest in the context of drug design and discovery.

As opposed to cliques, which captures the main elements of the chemistry of the system, H-wACSFs provide information about the whole molecular structure. Thus, it is reasonable to expect them to perform their best when deployed to predict target properties largely determined by structure as opposed to chemistry. Indeed, we find that H-wACSFs score best when applied on the Amo-Reg dataset, where the property we seek to predict is the T_g of amorphous drugs. Using the non-optimised values of the H-wACSFs parameters N^{Rad}, N^{Ang}, R_c^{Rad}, R_c^{Ang} and B (see Table 6.3), we obtain a marginal improvement in the

MSE when compared to the STD results (see Table 6.1), but also a significantly worse value for the PCC, as evident from Fig. 6.2. However, upon the optimisation of the above mentioned parameters via the genetic algorithms, we obtain a significant improvement of our predictions across all metrics, as illustrated in Fig. 6.2. It is interesting to note that the optimised parameters obtained for the three different datasets (see Table 6.3) vary significantly, with no robust trends emerging - the potential benefits of introducing constraints within our genetic algorithms would be addressed in future work.

For the Hepa and Amo-Reg datasets, where the H-wACSFs have outperformed STD, the genetic algorithms seem to have emphasised the connectivity of the molecular network as opposed to geometry of the specific conformers, in that $N^{Rad} \sim 2N^{Ang}$. As discussed in Section 6.3.1, the procedure we have used to generate said conformers is very basic, and as such, we expect the angular contributions to H-wACSFs to feature more prominently for ensembles of thoroughly (e.g. from first principles) optimised molecular conformers, and even more so in the case of actual three-dimensional models of either crystalline or amorphous drugs. Further support to this hypothesis comes from the fact that H-wACSFs did not perform especially well in the case of the Hepa dataset, where upon optimisation, we obtained results of similar, but not better quality when compared to the STD descriptors. In contrast to the Amo-Reg dataset, the Hepa dataset - and in fact, the Lipo dataset as well - seeks to predict a target property which may very well be ruled chiefly by chemistry as opposed to structure. While the distinction between cliques and H-wACSFs is not absolute in this respect (the cliques hold some structural information, and the H-wACSFs indirectly contains information about all cliques), we believe there is scope to bring the two classes of descriptors together, perhaps by using some sort of clique centred symmetry function, thus combining chemistry and structure - within a reasonably small number of descriptors, as opposed to harnessing the whole array of STD currently available.

Overall, our results are suggestive of the fact that while for relatively large datasets there might be value in trying to take advantage of the many descriptors readily available via open source computational packages, for small datasets containing hundreds of molecular structures, one might very well obtain better results by using just a handful of carefully crafted descriptors. In this work, we focused on cliques and H-wACSFs, but countless other options are obviously available. Despite the still limited scope of our investigation, we feel confident in saying that feature selection and optimisation should be treated as a necessary step of any ML algorithm for drug design and discovery, much as data pre-processing - as opposed to be considered as optional possibilities. We also note that many datasets of interest to the pharmaceutical companies are very limited in size: the Hepa dataset considered in here is just one example, but broadly speaking it is still challenging, despite the speed

with which the field is progressing, to collect large amounts of experimental measurements of complex biomedical activities. While it should be very clear at this point in time that no universal recipe exists that can provide a general-purpose framework to treat any given dataset, we believe this is yet another reason to be selective with respect to the choice of molecular descriptors.

6.4.1 One-molecule descriptors

As a first attempt to predict either the T_g or the crystallisation class of the drug molecules in the Amo-Reg and Amo-Class datasets, respectively, we have used different one-molecule descriptors (see Sec, 2.3.1).

In Table 6.4 and Table 6.5, we report the results we have obtained regarding the prediction of T_g (Amo-Reg dataset) with the NN and RF models. In particular, we report the MSE for both the training and the test sets (MSE - Train and MSE - Test, respectively). Please note that, due to the unique nature of our LOOCV approach (see Sec. 3.8.1), the MSE calculated for our test set may appear unusually large. This is because we derive the MSE by comparing experimental and predicted values of T_g for individual molecules or models, with each prediction coming from a model trained on slightly different data due to LOOCV.

These seemingly high MSE values in the test set should not be taken as a direct indicator of poor model performance or substantial prediction errors. Instead, they primarily serve as a relative measure of the variation in our predictions when different descriptors are used in our models. In essence, these values reflect how much our predictions can differ due to the LOOCV process, rather than conveying the absolute accuracy or uncertainty of our results.

We also report the Pearson correlation coefficient (PCC) for both the training and the test sets (PCC - Train and PCC - Test, respectively). The lack of an error bar regarding PCC - Test is due to the above mentioned nature of the LOOCV.

The Std descriptor is the most accurate overall, which is to be expected as it is in fact a collection of 170 descriptors. Surprisingly, the Cliques descriptor alone, despite its simplicity, shows similar accuracy. This is important, as the Cliques descriptor is a very transparent feature that can be straightforwardly leveraged to, e.g., identify specific functional groups that have an impact on, in this case, the value of T_g. The usage of our GA is effective in improving the accuracy of the SOAP descriptor, which is consistent with our recent results [188], but only marginally effective in doing the same for H-wACSFs. FS further serves to improve the accuracy of some - but not all - descriptors. Overall, Std (with FS), Cliques and SOAPs (with FS) provide similarly accurate predictions. We will discuss how exactly our

results compare to the state-of-the-art (particularly Ref. 28) in Sec. 6.5.

We also note that the RF seems to perform significantly better than NNs for Amo-Reg dataset. Although they do tend to have a much smaller error for the training set which indicates that there may be some overfitting of the data going on. Overall we see similar relative levels of performance for each descriptor in both the RF and NN models. With this being said, the NNs seem to be more receptive to feature selection, we did not include the FS results for descriptors where the score saw no improvement over the descriptor with no feature selection. For RFs the only descriptor that saw an improvement due to FS was the Std descriptor, whereas NNs with FS improved the score of the SOAP and H-wASCF descriptor as well as the Std. Both RF and NN models saw no improvement with FS on the GA optimised descriptors.

The results we have obtained regarding the prediction of the crystallisation class for the drug molecules in the Amo-Class dataset are reported in Table 6.6 and 6.7. In this case, we have chosen the Matthew's correlation coefficient (MCC [189]) to quantify the accuracy of our ML models. The MCC is much more robust than, say, the traditional definition of classification accuracy, namely the ratio between the number of correctly classified samples and the overall number of samples, and it especially well-suited to deal with unbalanced datasets such as the Amo-Class dataset [190]. Again, the absence of error bars regarding the MCC for the test set is due to the nature of the LOOCV we have used. Clearly, the usage of the SMOTE oversampling technique (see Sec. 2.4.2) substantially improves the predicting capabilities of every descriptor we have considered, nearing perfect accuracy regarding the training set and leading to very robust results for the test sets as well. Interestingly, the performance of each individual descriptor appears to be more uniform across the Amo-Class dataset then what we have observed for the Amo-Reg dataset (where specific descriptors performed significantly better or worse). Our best result for the Amo-Class dataset has been obtained by applying SMOTE, GA and FS on H-wACSFs - but the resulting accuracy is not drastically superior to that of any other descriptor we have considered. This is encouraging, as the prediction of the crystallisation class is the most useful aspect for practical considerations related to the physical stability of amorphous drugs. Again, we will discuss how exactly our results compare to the state-of-the-art (particularly Ref. 47) in Sec. 6.5.

6.4.1.1 Solid state descriptors

Before we analyse the accuracy of the ML models we have built by using solid state descriptors, it is instructive to discuss the dynamical properties we have obtained across our datasets. It is worth pointing out that, to our knowledge, this is the first, consistent col-

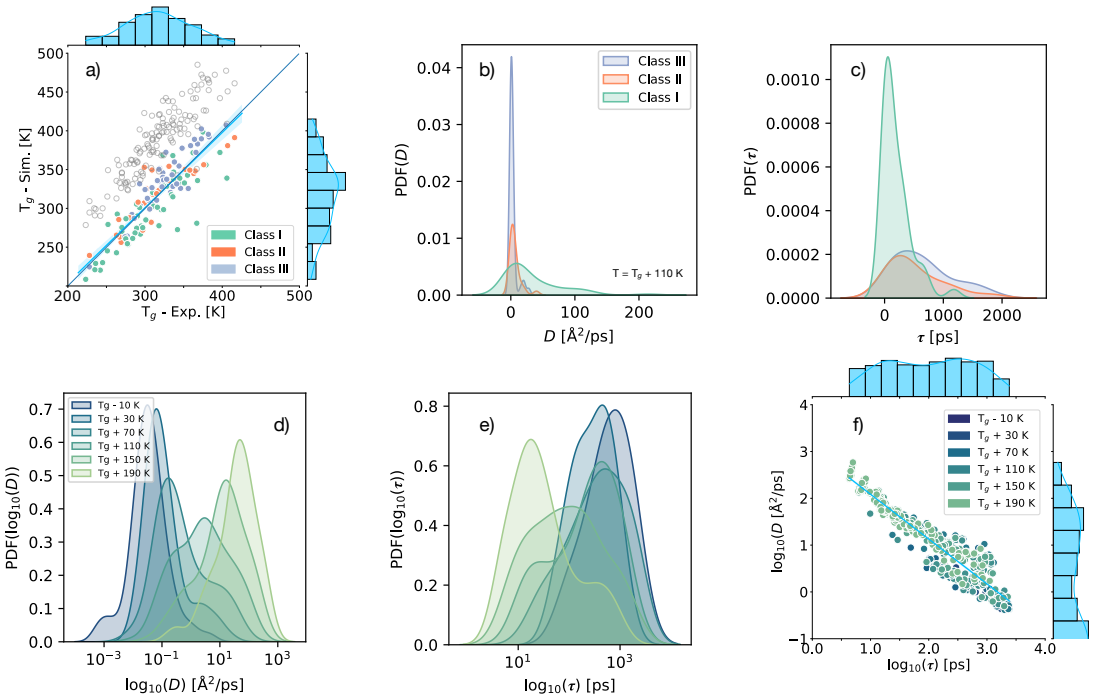

Figure 6.4: *Solid state descriptors.* a) Correlation between the experimental values of T_g (T_g - Exp.) and the estimate of T_g obtained via our MD simulations (T_g - Sim.). The raw MD results correspond to the empty, grey circles. The points colored according to the crystallisation class of the corresponding molecule have been obtained by shifting the raw MD results by -70 K (see text). b) PDF of the self-diffusion coefficient D of the supercooled liquids, labelled by crystallisation class, at $T = T_g + 110$ K for each molecular specie in the Amo-Reg dataset. c) PDF of the structural relaxation time τ (see text) of the supercooled liquids, labelled by crystallisation class, at $T = T_g + 110$ K for each molecular specie in the Amo-Reg dataset. d) PDF of the self-diffusion coefficient for each molecular specie in the Amo-Reg dataset at different temperatures. For visualisation purposes, we have taken the logarithm (base 10) of the values of D. e) PDF of the structural relaxation time for each molecular specie in the Amo-Reg dataset at different temperatures. For visualisation purposes, we have taken the logarithm (base 10) of the values of τ. f) Correlation between the (logarithm, base 10, of the) diffusion coefficient and the (logarithm, base 10, of the) structural relaxation time, color-coded according to different temperatures.

lection of dynamical properties computed via MD simulations for 100+ molecular glasses, which required almost 30 μs of simulation time (roughly 3 million CPU hours).

We begin by comparing the estimates of T_g we have obtained from our MD simulations (see Sec. 2.2) with the experimental data. The results are summarised in Fig. 6.4a. The raw data (grey circles in Fig. 6.4a) is very strongly correlated (Pearson Correlation Coefficient, PCC = 0.96) with the experimental data, albeit it systematically overestimates the latter. This is to be expected, as the cooling rates we are forced to use in MD simulations are orders or magnitude faster than the rates achievable experimentally. However, this discrepancy appears to be entirely systematic in nature: in fact, upon shifting the raw simulation data by -70 K, we recover a very good *quantitative* agreement with the experimental data, as illustrated in Fig. 6.4a (coloured points). This result demonstrates the robustness of our MD protocol and suggests that MD can indeed be used to estimate the T_g of molecular glasses with good accuracy - once the systematic error due to the nonphysically fast cooling rates characteristic of typical MD timescale has been corrected for.

Next, we move onto the self-diffusion coefficient D. As we now have an estimate of the T_g from our MD simulations, we can measure D for the different drugs utilising their different T_g as our reference. For instance, with "$T_g + 30$ K" we label the result obtained for a MD simulations at a temperature 30 K above the T_g of each specific drug. The distribution of D across the whole Amo-Reg dataset is reported in Fig. 6.4d at different temperatures. Note that we have taken the logarithm (base 10) of the actual values for visualisation purposes. As expected, the higher the temperature, the higher the diffusion coefficient, which spans almost three orders of magnitude. The opposite trend can be observed for the structural relaxation time τ, also reported as a function of temperature in Fig. 6.4e. In fact, D decays exponentially as τ increases, as shown by the log-log plot of D as a function of τ reported in Fig. 6.4f. Whilst intuitive, we are not necessarily aware of a universal relation between D and τ for molecular glasses - an interesting finding in itself.

The crucial question is whether any of these dynamical quantities is related to either T_g and/or the crystallisation class. Interestingly, we observe no simple relationship between either D or τ and T_g. However, there is a strong correlation between D or τ and the crystallisation class, as illustrated in Fig. 6.4b and Fig. 6.4c, respectively. In particular, the diffusion coefficient of Class I molecules is on average much higher than Class II or Class III molecules. The distributions reported in Fig. 6.4b refer to the data obtained at a specific temperature above T_g (T = T_g + 110 K), but for the diffusion coefficient this trend is present to some extent for each temperature we have considered. Note that there is no point in considering temperatures below T = T_g - 10 K, as the vast majority of the glasses has impossibly low(long) diffusion coefficients(structural relaxation times). In fact, one can easily discard

any drug with a $D > 50$ Å^2/ps at T = $T_g + 110$ K for the practical purposes of determining whether it might be suitable as an amorphous formulation. The same can be said for τ, albeit we note that the correlation between τ and the crystallisation class is less strong - in that it holds to different extents according to the specific temperature chosen. These findings are of great practical relevance, as they can offer a rather inexpensive route to probe whether novel candidates for amorphous drugs would fall in Class I.

Regression [T_g]: We have used these three solid state descriptors (T_g, D and τ) to predict the experimental T_g. The results are summarised in Table 6.8. Unsurprisingly, given the strong correlation between simulated and experimental T_g that we have discussed above, the simulated T_g gives very accurate results: the MSE for both training and test set is an order of magnitude lower than that obtained for any other descriptor we have used (one-molecule descriptors included, see Sec. 2.3.1).

For D and τ, we have not just one value, but multiple values for each drug molecule, as we have computed these dynamical quantities at several temperatures below and above their corresponding T_g. In fact, the descriptor vector for both D or τ relative to each molecule contains five elements for $T_g - 10$ K, $T_g + 30$ K, $T_g + 70$ K, $T_g + 110$ K, $T_g + 150$ K. This allow us to use information about the dynamical behavior of each system as a function of temperature. Whilst we do have data at lower/higher temperatures as well (down/up to $T_g - 50$ K, $T_g + 190$ K), virtually every system behaves in exactly the same fashion - the dynamics is either impossibly slow at $T_g - 50$ K or similarly fast at $T_g + 190$ K. As such, including D or τ for those temperatures does not increase the accuracy of our models - if anything, it introduces noise.

The results for the diffusion coefficient are as accurate as our best results we have obtained for the one-molecule descriptors upon both optimisation (GAs) and feature selection - which is quite impressive. Interestingly, τ performs very poorly in terms of predicting the experimental T_g. This is not entirely unexpected, as D varies much more smoothly than τ as a function of temperature. For low/high temperatures, τ tends to be either beyond the timescale that we have simulated, or equally short notwithstanding the molecular species.

Upon examining the performance of the NN and RF models, it is apparent that their results on the test sets are strikingly alike. However, a noticeable contrast exists between the two models in terms of their Train sets. Specifically, the RF models, once again, display a propensity to overfit, indicating that they may have learned the noise present in the data rather than the underlying patterns. Nevertheless, it is challenging to determine if overfitting has indeed occurred without a validation set. It is noteworthy that a low MSE on the training set does not necessarily imply the presence of overfitting. This underscores the importance of utilizing a validation set to assess the generalisability of the models.

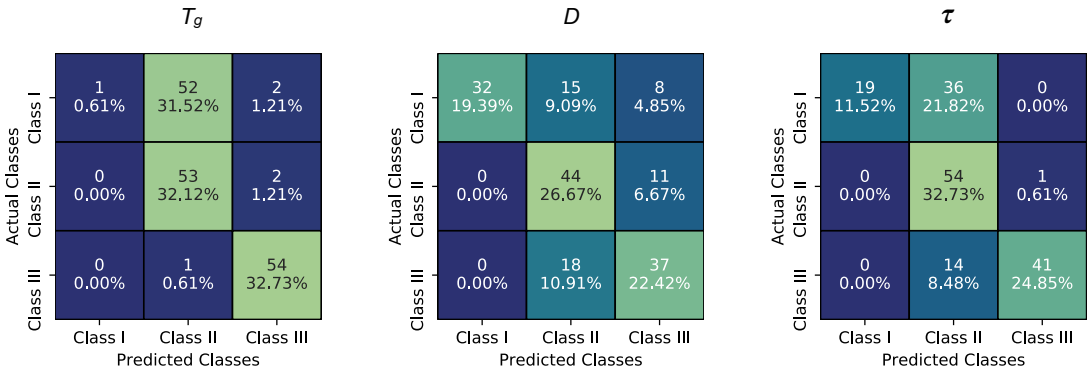

Figure 6.5: *Confusion matrices for the solid-state descriptors.* Confusion matrices relative to our prediction of the crystallisation class regarding the Amo-Class dataset, via different solid state descriptors. These results have been obtained upon applying the SMOTE technique and refer to the test sets only - as we adopted a LOOCV approach(see text).

Classification [Crystallisation Class]: We now move onto the results we have obtained with the three solid state descriptors (T_g, D and τ) in terms of the prediction of the crystallisation class of the drug molecules in the Amo-Class dataset. The results are summarised in Table 6.6, Table 6.7 and Fig. 6.5. Similarly to what we have observed in the case of the one-molecule descriptors, the usage of the SMOTE technique consistently improves the accuracy of our results. In terms of numerical accuracy, the results we have obtained via our solid state descriptors perform slightly worse than the one-molecule descriptors. To be specific, our best result for the one-molecule descriptors (H-wACSFs, upon FS and SMOTE, see Table 6.6 and 6.7) is a MCC - for the test set - of 0.757, whilst our best result for the solid state descriptors (τ, upon SMOTE, see Table 6.6, Table 6.7) is 0.605.

In stark contrast to one-molecule descriptors, however, the solid state descriptors are characterised by a very low dimensionality - just one for Tg and five for both D and τ, whilst descriptors such as SOAPs can easily count 10^3 elements per SOAP vector. As such, the fact that the solid state descriptors can achieve a respectable accuracy in terms of our ML models is quite encouraging indeed. In addition to this, the solid state descriptors are very much transparent - in that they are representative of well-defined dynamical properties of the system. Again, this is substantial advantage with respect to most one-molecule descriptors: in our case, the Cliques descriptor alone can be considered as a transparent descriptor.

Thus, it is instructive in the case of solid state descriptors, to inspect the confusion matrices relative to our predictions. These are reported in Fig. 6.5, and refer to the results

we have obtained upon applying SMOTE - for the test set only. We report a single confusion matrix per descriptor as these have been obtained via LOOCV (see Sec. 6.5 for discussion of these results).

We begin with T_g, which appears to be able to classify correctly the vast majority of Class II and Class III molecules, whilst it almost entirely mislabels Class I molecules as Class II molecules. By looking at the PDFs of the T_g for each crystallisation class (see Fig. 1.3e; the PDFs for the *simulated* T_g are very similar, given the strong correlation between the two, see Fig. 6.4a), the T_g of Class II and Class III molecules is very rarely lower than ~ 250 K. This helps in explaining why Class II and Class III are never predicted as Class I molecules in our T_g ML model, but it does not explain the close-to-perfect distinction between Class II and Class III, nor the mislabeling of almost every Class I molecule as Class II molecules.

The accuracy we have obtained when using the diffusion coefficient D as our descriptor is very similar (in terms of MCC) to that of T_g. However, as illustrated in Fig. 6.5, the model sacrifices some accuracy in telling apart Class II from Class III molecules to improve on the labelling of Class I molecules. Similarly to what we have observed for T_g, however, no Class II or Class III molecules are ever predicted as Class I molecules.

The relaxation time τ gave us our best result in terms of accuracy. This was somehow expected, as τ is a rather "binary" measure, in our case, of whether the system is behaving like a supercooled liquid (for which we observe a τ well within the timescale of our MD simulations) or a glass (in which case, we simply assign a blanket value of 9999 to the descriptor at that given temperature, as we have no way to probe τ across the relevant timescale). On the contrary, τ performed rather poorly as a descriptor for the regression problem of T_g, particularly compared to D (see Table 6.4 and Table 6.5). The confusion matrix we have obtained for τ (see Fig. 6.5) is more balanced across the different Classes. However, the two distinct features of all our classification models we have obtained via solid state descriptors persist, namely: 1. a substantial mislabeling of Class I molecules as Class II molecules and 2. no Class II or Class III molecules ever classified as Class I molecules.

As it stands, our models seem to be able to predict with incredible accuracy whether a drug molecule belongs to Class II or Class III - but not Class I. This is a practically important aspect. However, ideally one would be able to either ”filter out” Class I molecules with great accuracy or to classify correctly Class III molecules - which are the most suitable candidates in terms of amorphous drugs formulations. For now, these models cannot achieve the former and only partially meet the expectations of the latter. We shall put our results into context regarding the state-of-the-art in the next section, where we will combine our descriptors into an ensemble learning framework.

6.4.2 Ensemble methods

With our optimised descriptors we now attempt to further improve their performance using the ensemble methods of max voting and ensemble NNs. These methods combine multiple models and therefore should be less prone to overfitting as well as providing more accurate predictions. These ensemble methods were all performed using the computational framework explained in Sec 3.1.

6.4.2.1 Max voting:

The first ensemble method we assess is the max voting ensemble described in section 3.9.2. In summary, we have chosen to perform max voting so by training (and crucially, optimising) models for each descriptor in isolation and combine the different predictions. The results of this ensemble approach are summarised in Fig. 6.7. For discussion on these results, please see section 6.5

Regression [T_g]: We begin by discussing the results with respect to the prediction of the experimental T_g. As illustrated in Fig. 6.7a, averaging our predictions over multiple descriptors does indeed lead to an improvement in the overall accuracy of our models. It is important to distinguish between two sets of results: ensemble models that did not include the simulated T_g as a descriptor ("no Tg" label in Fig. 6.7a) and models that did include the simulated T_g as a descriptor ("Tg" label in Fig. 6.7a). In the latter scenario, we are effectively using a computational estimate of the target property as a descriptor - which clearly leads to much more accurate predictions.

It is interesting to look at representative scatter plots (Fig. 6.7b) of predicted versus experimental T_g for our best regression models. The "no Tg" model has been obtained by combing Std descriptors and the diffusion coefficient D. The "Tg" model has been obtained by combining the simulated T_g and the diffusion coefficient D. From these outcomes, it is evident that the diffusion coefficient brings important information about the dynamical properties of the system into the model. This is key, as it demonstrate the potential of the solid state descriptors, obtained via MD simulations, which we are putting forward in this work. Interestingly, it appears that our models tend to underestimate the experimental T_g when the latter is higher than 350 K. This is somehow counter intuitive in that MD simulations systematically *over*-estimate the values of T_g (see Fig. 6.4a) due to the non-physically rapid quenching rates of the supercooled liquid into the glass.

We can now attempt to put our results into context with respect to the state-of-the-art, particularly Ref. 28. In that work, the authors considered a smaller dataset - 71 molecules, to be compared with the 136 molecules in our Amo-reg dataset (see Sec. 1.3 for

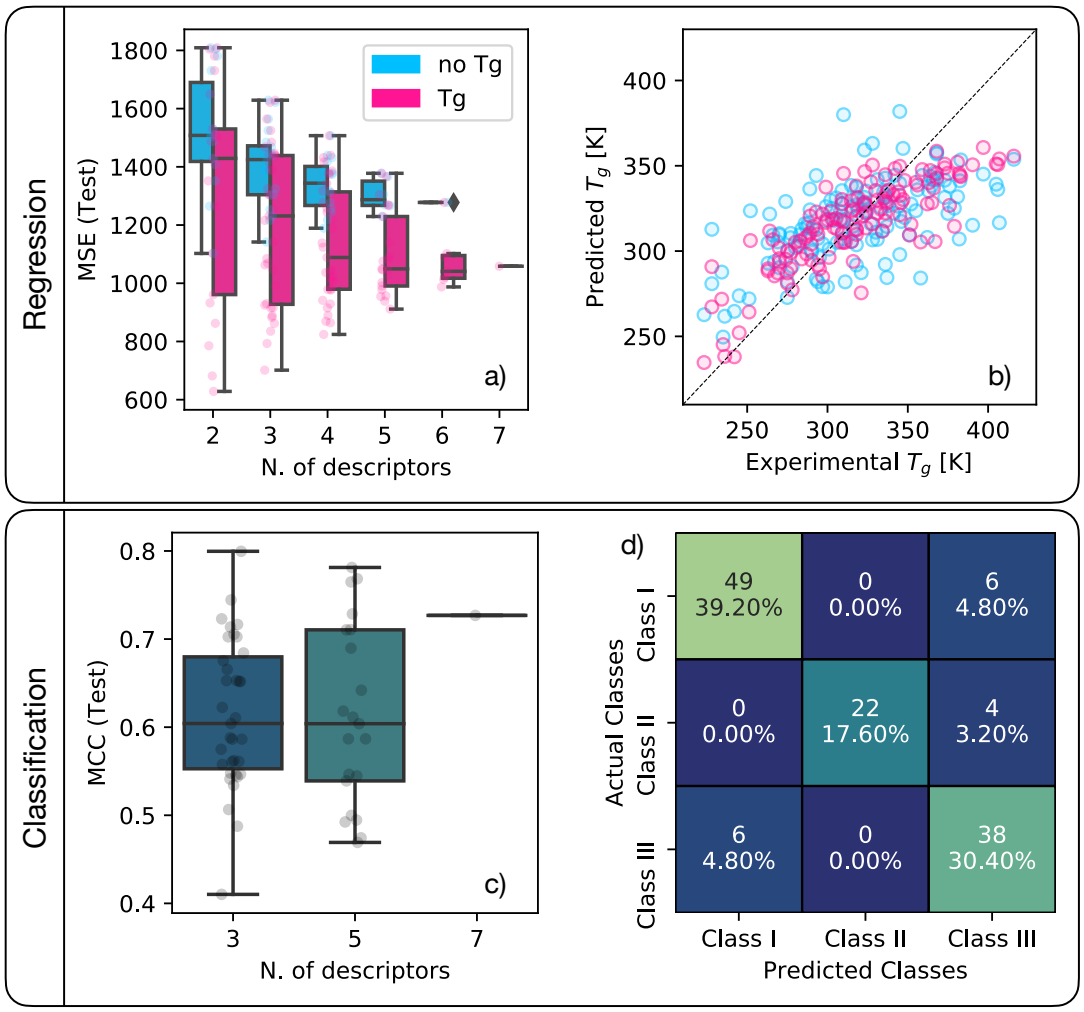

Figure 6.6: *Ensemble learning.* a) MSE relative to the test set for our predictions of the experimental T_g (Amo-reg dataset), as a function of the number of descriptors we have combined via ensemble learning (see text) using NN model. The simulated T_g is included as a descriptor in the "Tg" results, but it has not been included in the "no Tg" results. b) Scatter plot of the predicted versus the experimental T_g for our best regression models. The color code is the same as in panel a). c) MCC relative to the test set for our predictions of the crystallisation class (Amo-class dataset), as a function of the number of descriptors we have combined via ensemble learning. d) Confusion matrix for our best classification model.

further details). The authors have reported their results regarding a single, specific test set of 24 molecules (which implies that they have used a single, specific training set containing 47 molecules). We believe that this approach is not ideal, in that - particularly given the small size of the dataset - the variability associated with the choice of a specific test set is bound to be very large indeed. In particular, the best result reported in Ref. 28 might have been obtained with an especially "unlucky" or even especially "lucky" test set, which prevents us from a direct comparison with our results. In contrast, by adopting the LOOCV approach we have used in this work, we are confident that our results are independent on the choice of a specific test set.

With this in mind, the MSE regarding the test set obtained in Ref. 28 is 686.44. This number refers to a model leveraging NNs in a similar fashion to our work, using a set of "molecular descriptors" - none of which have been obtained from the outcomes of MD simulations. As such, the descriptor we used which is closer to the set of descriptors used in Ref. 28 is the Std descriptor (see Sec. 6.4.1). The PCC relative to the very same model in Ref. 28 is 0.78. In comparison, our best regression model (see Fig. 6.7b) including the simulated T_g as a descriptor gave a MSE for the test set of 628.47, and a PCC of 0.83. Our best regression model (see Fig. 6.7b) obtained without including the simulated T_g as a descriptor gave a MSE for the test set of 1102.50, and a PCC of 0.60. In Ref. 28 we can also find a more accurate result (MSE = 349.69 and PCC = 0.88) obtained via support vector regression (SVR). At this stage we do not have an explanation as to why SVR would perform significantly better than NNs, but we intend to explore alternative ML algorithms in the future.

Overall, our results in terms of regression are probably comparable to that of Ref. 28. Our combination of one-molecule and solid state descriptors gave only slightly more accurate results than the set of molecular descriptors used in Ref. 28 - if we are to compare these in the context of NNs only. However, the results obtained in Ref.Ref. 28 via SVR appear to be significantly more accurate than ours, albeit it is difficult to make direct comparisons given both the different datasets and the choice of using a single, specific test set.

Classification [Crystallisation Class]: We have seen in Sec. 6.4.1.1 that the relaxation time τ showed significant potential as a descriptor to identify the crystallisation class. As such, it comes as no surprise that our best classification ensemble models, obtained via the straightforward max-voting approach described in Sec. 3.9.2, would include τ. The confusion matrix relative to our best classification model, which has been obtained by combining τ, wACSFs, and Std, is reported in Fig. 6.7d. Note that, overall, combining multiple descriptors together did consistently improve the accuracy of our models (see Fig. 6.7c). In contrast to the results obtained with the solid state descriptors in isolation (see Fig. 6.5, Class II molecules appear to be the most problematic ones to label correctly. This is expected - as

we discussed in section 1.3.

We can now compare our results with those of Ref. 47, where the dataset utilised (131 molecules) is almost identical to the Amo-Class dataset (124 molecules). There are two main differences between the approach of Ref. 47 and our work. The first one is that, in a similar fashion to Ref. 28, the authors have opted for single, specific test set, which in this case encompasses 31 molecules (whilst the training set includes 100 molecules). Again, we do not believe this approach to be ideal. The second difference is that in Ref. 47, the authors do not take into account Class I molecules at all for the purposes of their classification models. This has been justified on the basis of a fixed threshold regarding the molecular weights of the molecules. Specifically, the authors argue that drug molecules characterised by a molecular weight (MW) < 200 g/mol can be considered as Class I molecules. Whilst it is definitely true that a strong connection between MW and crystallisation class exists (see Fig. 1.3d), only 20 molecules belonging to Class I have a MW < 200 g/mol. The remaining 35 Class I molecules are all characterised by MW > 200 g/mol. As such, we believe that it is not fair to exclude Class I molecules from a classification model, as the MW criterion is not robust enough to provide a practical indication in terms of choosing a given drug-like molecule as a potential candidate for an amorphous formulation.

In Ref. 47, the authors have built a model (based on decision trees) that distinguish between Class II and Class III molecules (as we discussed, Class I molecules have not been taken into account) with an accuracy of 69% relative to the single, fixed test test discussed above. Given than Class II is severally under-populated with respect to Class III, we argue that the usage of a metric such the MCC we have employed here gives a better representation of the accuracy of the model. Nevertheless, for the purposes of comparing the results of our best model with that of Ref. 47, the model which confusion matrix is reported in Fig. 6.7d predicts Class I, II and III molecules with an accuracy of 89%, 84% and 86%, respectively. Not only these numbers indicate a substantially more accurate model, but - crucially - our classification model does include Class I molecules as well.

Overall, our results in terms of classification are very encouraging - albeit it needs to be said that in order to achieve a truly predictive model, additional experimental data are probably needed. The fact that our model identifies Class I molecules with an accuracy of almost 90% is especially intriguing for practical purposes - as we shall discuss in greater detail in the conclusion.

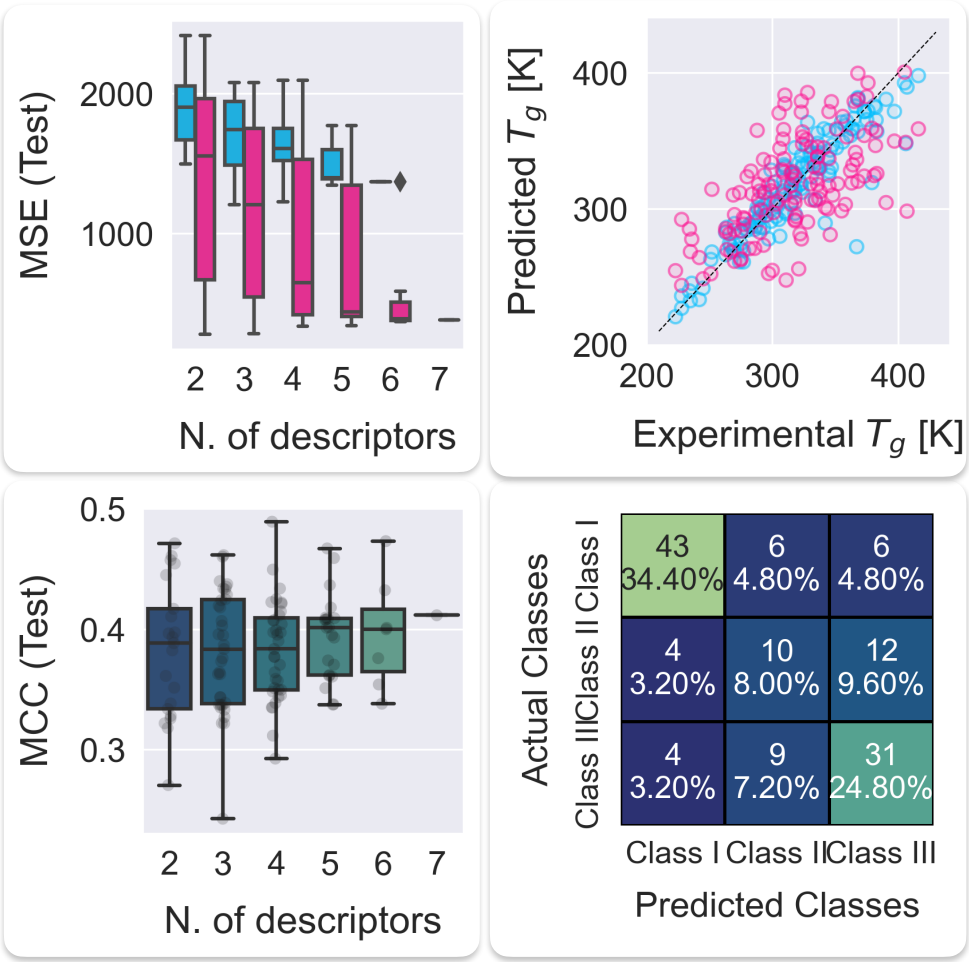

Figure 6.7: *Ensemble learning.* a) MSE relative to the test set for our predictions of the experimental T_g (Amo-reg dataset), as a function of the number of descriptors we have combined via ensemble learning (see text) using ensemble NN. The simulated T_g is included as a descriptor in the "Tg" results, but it has not been included in the "no Tg" results. b) Scatter plot of the predicted versus the experimental T_g for our best ensemble NN regression models. The color code is the same as in panel a). c) MCC relative to the test set for our predictions of the crystallisation class (Amo-class dataset), as a function of the number of descriptors we have combined via ensemble learning. d) Confusion matrix for our best classification model.

6.4.2.2 NN ensemble

When we analyse the results of the ensemble NN model (see section 3.9.3) we observe that the max voting ensemble performs better for both the Amo-Reg and Amo-Class dataset. The reason for this is likely due to the fact that it is very challenging to optimise the model. The descriptors perform optimally individually due to the rigorous parameter tuning, however, this is not possible when combining NNs as shown in figure 6.1. For example, if we have a different optimal regularisation parameter for two different descriptors, in the max voting ensemble, we are able to get predictions for each descriptor using their optimal parameter. In the ensemble NN case it is not possible to train the different portions of the network with a different regularisation rate and as a result, the quality of predictions is not as good.

6.5 Discussion and Conclusions

Packaging drug molecules as amorphous solids represents an intriguing possibility for the pharmaceutical industry to circumvent the long-standing issue of the low solubility of traditional crystalline formulations. One of the major hurdles in implementing this approach, however, is the physical stability of the amorphous phase - i.e., the timescale required for it to transition into the crystal. Clearly, in order for an amorphous formulation to be marketable, the amorphous phase needs to be stable for the entire shelf life of the product - which is very difficult to predict *a priori*.

Herein lies the motivation for our research: to harness the power of machine learning and molecular simulations to enhance our ability to forecast the stability of drug molecules in their amorphous state. While we have made significant progress in classifying these molecules into different crystallisation classes, we recognize that this is just the beginning. Our journey has underscored the importance of expanding both our dataset and our modeling techniques to unlock the full potential of amorphous solid formulations. As we stand at the intersection of computational innovation and experimental exploration, our work highlights the need for a collaborative effort that combines the strengths of both domains to revolutionise the pharmaceutical landscape.

Machine learning can help in this context, by developing models capable to predict the stability of a given drug molecules in its amorphous phase. However, the limited data in our possession only enable us to devise classification models aimed at predicting the so-called "crystallisation class" relative to a given molecule. Class I, II and III correspond, loosely, to classes of drug-like molecules with very low, intermediate and rather high physical stability. In this work, we build on the datasets and the previous results of Alhalaweh,

Mahlin, Bergström *et al.* (particularly Ref. 28 and Ref. 47) to deliver regression and particularly classification models that represent a step forward regarding the state-of-the-art in terms of both accuracy and reliability. In particular, we adopt an approach that leverages the outcomes of molecular dynamics simulations to build bespoke descriptors that can be used to complement the picture offered by the traditional, "one-molecule" descriptors commonly used in QSAR models.

We combine these "solid state" descriptors with an array of optimisation strategies, including genetic algorithms, feature selection, over-sampling and ensemble learning, to craft a portfolio of classification models that - despite the very limited size of the dataset at our disposal - can correctly label drug-like molecules as Class I, II and III with accuracies of ~85%. The ability of our models to "filter out" Class I molecules - which are unsuitable as candidates for amorphous formulations - is especially intriguing from a practical standpoint. The outcomes of our work demonstrate the usefulness of combining molecular simulations with traditional machine learning approaches, not only to increase the predictive power of the latter, but also to enable the usage of more transparent descriptors that can effectively be used to build genuine structure-function relationships between molecular structure and functional properties. Much work remains to be done in the context of machine learning to predict the physical stability of amorphous drugs. As already mentioned, the limited size of the datasets available severely limits the accuracy of our models. We do hope that the encouraging results we have obtained here will motivate further experimental measurements aimed at increasing the size of said datasets.

It is also worth noting that many amorphous drugs are packaged as amorphous solid dispersions: these are heterogeneous systems including amorphous drugs dispersed in (typically) polymeric matrices. Expanding the scope of our models to take into account such systems will only be possible given a substantial amount of experimental data that, to our knowledge, is largely missing from the current literature.

In summary, we have advanced the state-of-the-art by bringing molecular simulations into the mix: at this stage, whilst further computational improvement are certainly possible, we believe that any decisive step forward in the field can only be achieved in conjunction with bespoke experimental efforts.

		MSE	
	STD	Cliques	Cliques [FS]
Lipo	0.198 ± 0.098	0.412 ± 0.016	0.690 ± 0.005
	(0.682 ± 0.023)	(0.950 ± 0.019)	(1.032 ± 0.040) [15]
Hepa	0.253 ± 0.063	0.176 ± 0.007	0.125 ± 0.005
	(0.413 ± 0.059)	(0.317 ± 0.029)	(0.304 ± 0.028) [18]
Amo	0.460 ± 0.127	0.130 ± 0.009	0.497 ± 0.029
	(0.806 ± 0.171)	(0.950 ± 0.360)	(0.994 ± 0.167) [13]
		H-wACSFs	H-wACSFs [GAs]
Lipo		0.889 ± 0.020	0.746 ± 0.019
		(0.939 ± 0.022)	(0.920 ± 0.031)
Hepa		0.590 ± 0.055	0.314 ± 0.010
		(1.238 ± 0.171)	(0.350 ± 0.037)
Amo		0.362 ± 0.041	0.124 ± 0.019
		(1.348 ± 0.465)	(0.838 ± 0.084)
		PCC	
	STD	Cliques	Cliques [FS]
Lipo	0.933 ± 0.003	0.859 ± 0.003	0.727 ± 0.003
	(0.737 ± 0.019)	(0.623 ± 0.010)	(0.554 ± 0.020) [15]
Hepa	0.687 ± 0.043	0.731 ± 0.012	0.826 ± 0.007
	(0.295 ± 0.031)	(0.359 ± 0.054)	(0.450 ± 0.041) [18]
Amo	0.873 ± 0.008	0.935 ± 0.007	0.733 ± 0.015
	(0.637 ± 0.058)	(0.400 ± 0.218)	(0.349 ± 0.111) [13]
		H-wACSFs	H-wACSFs [GAs]
Lipo		0.336 ± 0.011	0.503 ± 0.020
		(0.273 ± 0.020)	(0.0327 ± 0.013)
Hepa		0.641 ± 0.035	0.417 ± 0.037
		(0.148 ± 0.033)	(0.136 ± 0.077)
Amo		0.802 ± 0.028	0.936 ± 0.009
		(0.261 ± 0.101)	(0.497 ± 0.134)

Table 6.1: Comparing the performance of three classes of descriptors: $\sim$ 100 "standard" RDKit descriptors (STD), molecular cliques (Cliques) and histograms of weighted atom-centred symmetry functions (H-wACSFs). For each dataset: Lipophilicty (Lipo), Hepatocytes (Hepa) and Amorphous (Amo) we report the mean square error (MSE) and the Pearson correlation coefficient (PCC) for both the training and, in brackets, the test sets. All the numbers have been averaged according to the cross validation procedure discussed in Section 3.8.1. Uncertainties are included as $\pm\frac{\sigma}{2}$. Cliques [FS] and H-wACSFs [GAs] refer to the results obtained for cliques upon feature selection (the numbers in square brackets specify the number of selected cliques) and H-wACSFs upon optimisation, respectively. See text for further details about both datasets and descriptors.

Feature selection - Cliques *Hepatocytes dataset*		
Smiles	MDI (mean)	MDI (σ)
CC	0.082263	0.002642
CO	0.069692	0.002545
CN	0.069352	0.001979
C	0.054925	0.002775
C1=CC=CC=C1	0.052196	0.002532
C=O	0.032964	0.001487
CF	0.031491	0.002122
C1=CN=CCC1	0.030531	0.005510
C1=COC=CC1	0.028628	0.003882
C1COCCN1	0.027860	0.002575
C1CCNCC1	0.025989	0.002891
CCl	0.025489	0.001000
C1=CSC=C1	0.024680	0.003132
C1CCCCC1	0.021090	0.002438
CS	0.017693	0.001977
C1CNCCN1	0.017380	0.002165
C1=CSCN1	0.017038	0.002653
C1=NCCS1	0.013932	0.001524
C1CNSC1	0.015341	0.003452
C#N	0.013333	0.001248
[...]		
C1=CCOCC1	0.005135	0.000685
C1CNC1	0.005111	0.001257
C1CNCN1	0.004771	0.000744
C1=CCNC=C1	0.004578	0.000439
C1=CCCCC1	0.004489	0.000649

Table 6.2: Feature selection for the cliques descriptor in the case of the Hepatocytes dataset. The full cliques vocabulary contains in this case 132 cliques. For the 18 most important cliques (bold font) as well as for the 5 least important cliques we report the corresponding MDI (mean and standard deviation σ), computed as discussed in Section 3.6. Note that the most and least important cliques are characterised by values of the MDI that differ roughly by an order of magnitude.

Optimisation - H-wACSFs				
	Non-optimised	Lipo	Hepa	Amo-Reg
N_{Radial}	8	3	14	22
$N_{Angular}$	16	14	8	10
$R_{c,Radial}$ (Å)	20	2	21	7
$R_{c_{Angular}}$ (Å)	20	21	12	2
N_{H-bins}	10	16	19	12

Table 6.3: Parameters of the H-wACSFs before and after optimisation via the genetic algorithms-based procedure described in Chapter 5. N^{Rad}, N^{Ang}, R_c^{Rad}, R_c^{Ang} and B stand for the number of radial symmetry functions (SFs), the number of angular SFs, the cutoff radius for the radial SFs, the cutoff radius for the angular SFs and the number of bins we have used to build the histograms, respectively. Results for the three datasets: Lipophilicty (Lipo), Hepatocytes (Hepa) and Amorphous (Amo) are shown. Note the absence of any solid trend for any of the SFs parameters across the different datasets.

Descriptor	MSE - Train (K^2)	MSE - Test (K^2)	PCC - Train	PCC - Test
Std	1455.899 ± 333.474	1625.971 ± 2637.659	0.442 ± 0.096	0.459
Cliques	1967.728 ± 608.089	1995.273 ± 2690.114	0.266 ± 0.212	0.393
H-wACSFs	1916.213 ± 508.731	2594.003 ± 6422.134	0.326 ± 0.121	0.157
H-wACSFs - GA	2203.201 ± 359.928	2356.687 ± 4482.981	0.19 ± 0.105	0.166
SOAPs	2101.346 ± 796.46	3001.488 ± 13712.585	0.353 ± 0.079	0.285
SOAPs - GA	2063.968 ± 321.171	2237.609 ± 3135.72	0.293 ± 0.061	0.316
Feature selection				
Std	1331.376 ± 244.329	1458.926 ± 2095.038	0.495 ± 0.066	0.511
H-wACSFs	1818.283 ± 428.084	2446.945 ± 4816.482	0.351 ± 0.109	0.186
SOAPs	1706.709 ± 516.041	2105.706 ± 6532.711	0.407 ± 0.109	0.359

Table 6.4: *ML predictions regarding the Amo-Reg dataset: one-molecule descriptors.* Accuracy of our ML models in predicting the T_g regarding the Amo-Reg dataset, via different one-molecule descriptors using NNs (see text).For a discussion on how this compares to the literature, see section 6.5. We note that the Std descriptors do not perform particularly well in this regime and we see that the model struggles to learn from the training set.

Descriptor	MSE - Train (K^2)	MSE-Test (K^2)	PCC-Train	PCC-Test
Std	198.025 ± 7.056	1315.096 ± 1930.636	0.969 ± 0.002	0.49
Cliques	624.233 ± 15.543	1645.365 ± 2261.237	0.883 ± 0.005	0.261
H-wACSF	659.248 ± 17.181	1656.536 ± 2249.296	0.865 ± 0.007	0.256
H-wACSFs - GA	256.874 ± 6.78	1310.202 ± 1804.481	0.962 ± 0.002	0.494
SOAP	246.778 ± 5.737	1518.32 ± 2234.844	0.968 ± 0.002	0.358
SOAP - GA	287.72 ± 7.881	1393.164 ± 2065.139	0.95 ± 0.002	0.444
Feature selection				
Std	1245.319 ± 368.329	1230.619 ± 2420.128	0.459 ± 0.086	0.535

Table 6.5: *ML predictions regarding the Amo-Reg dataset: one-molecule descriptors.* Accuracy of our ML models in predicting the T_g regarding the Amo-Reg dataset, via different one-molecule descriptors using RFs (see text). We note that again, the Std descriptor does not perform well and we see some overfitting.

Descriptor	MCC - Train	MCC-Test	MCC-Train SMOTE	MCC - Test SMOTE
Std	0.837 ± 0.101	0.573	0.999 ± 0.003	0.746
Cliques	0.982 ± 0.016	0.282	0.99 ± 0.011	0.655
H-wACSF	0.957 ± 0.06	0.405	0.421 ± 0.023	0.579
H-wACSF - GA	0.421 ± 0.104	0.726	0.401 ± 0.111	0.601
SOAP	0.543 ± 0.055	0.432	0.955 ± 0.057	0.659
SOAP - GA	0.349 ± 0.104	0.59	0.979 ± 0.048	0.694
Feature selection				
STD - FS	0.993 ± 0.014	0.544	0.999 ± 0.04	0.747
H-wACSF - FS	0.687 ± 0.133	0.49	0.991 ± 0.013	0.673
H-wACSF - GA - FS	0.738 ± 0.175	0.464	1 ± 0.02	0.757
SOAP - FS	0.693 ± 0.154	0.399	1 ± 0	0.746
SOAP - GA - FS	0.918 pm 0.093	0.418	-	-

Table 6.6: *ML predictions regarding the Amo-Class dataset: one-molecule descriptors.* Accuracy of our ML models in predicting the crystallisation class regarding the Amo-Class dataset, via different one-molecule descriptors (see text) for the NN descriptor.

Descriptor	MCC - Train	MCC-Test	MCC-Train SMOTE	MCC - Test SMOTE
Std	1.0 ± 0.0	0.572	1.0 ± 0.0	0.719
Cliques	1 ± 0	0.279	0.978 ± 0.005	0.555
H-wACSF	0.958 ± 0.012	0.414	1.0 ± 0.0	0.665
H-wACSF - GA	0.99 ± 0.008	0.532	1.0 ± 0.0	0.691
SOAP	0.967 ± 0.016	0.357	1.0 ± 0.0	0.675
SOAP - GA	1.0 ± 0.0	0.503	0.993 ± 0.001	0.693
Feature selection				
STD - FS	-	-	1.0 ± 0.002	0.728
H-wACSF - FS	0.687 ± 0.133	0.49	0.96 ± 0.011	0.585
H-wACSF - GA - FS	-	-	1.0 ± 0.0	0.701
SOAP - FS	0.693 ± 0.154	0.399	0.997 ± 0.004	0.7

Table 6.7: *ML predictions regarding the Amo-Class dataset: one-molecule descriptors.* Accuracy of our ML models in predicting the crystallisation class regarding the Amo-Class dataset, via different one-molecule descriptors (see text) for the RF descriptor.

Descriptor	MSE - Train	MSE - Test	PCC - Train	PCC - Test
T_g	421.759 ± 103.799	432.236 ± 902.112	0.859 ± 0.22	0.866
D	1404.177 ± 502.664	1444.206 ± 1886.394	0.487 ± 0.106	0.441
τ	1690.3 ± 233.864	1864.22 ± 2522.985	0.14 ± 0.151	0.058

Table 6.8: *ML predictions regarding the Amo-Reg dataset: solid state descriptors.* Accuracy of our ML models in predicting the T_g regarding the Amo-Reg dataset, via different solid state descriptors with a NN model (see text).

Descriptor	MSE - Train	MSE - Test	PCC - Train	PCC - Test
T_g	200.842 ± 4.009	483.013 ± 913.996	0.941 ± 0.001	0.851
D	279.16 ± 7.411	1586.628 ± 2148.29	0.953 ± 0.002	0.336
τ	603.565 ± 12.349	1826.099 ± 2291.228	0.856 ± 0.005	0.192

Table 6.9: *ML predictions regarding the Amo-Reg dataset: solid state descriptors.* Accuracy of our ML models in predicting the T_g regarding the Amo-Reg dataset, via different solid state descriptors with a RF model (see text).

Chapter 7

Conclusion

7.1 Conclusion

In this thesis, we present an investigation of how molecular dynamics (MD) simulations and novel descriptors can be used to predict important characteristics of amorphous drugs that affect their stability, such as the glass transition temperature and the crystallization class. The use of computational methods to predict these properties is becoming increasingly popular due to the significant cost and time savings they offer in comparison to traditional experimental methods.

We focus on predicting the glass transition temperature and the crystallization class of amorphous drugs, as these are key factors that impact the stability of these drugs. Our results demonstrate that the combination of solid-state descriptors derived from MD simulations with machine learning algorithms can provide accurate predictions of these properties, indicating that it has the potential to be a valuable tool in the pharmaceutical industry. Our approach provides an efficient way to screen large numbers of potential drug formulations and identify those that are likely to have good stability properties.

While the accuracy of our models for directly predicting the stability of amorphous drugs cannot be evaluated due to the limited availability of data, we hope that this work serves as a foundation for future research. We anticipate that as more data becomes available, our approach can be further developed and optimised to improve its predictive capabilities.

The key deliverables of this work are as follows:

- **Advancing Classification:** This work contributes to the field of class classification, specifically in the context of predicting important characteristics of amorphous drugs, such as the glass transition temperature and crystallisation class.

- **Application of Computational Methods:** We demonstrated the application of computational methods, including MD simulations and novel descriptors, to predict crucial properties of amorphous drugs.
- **Cost and Time Savings:** We highlighted the cost and time-saving benefits of using computational approaches compared to traditional experimental methods in the pharmaceutical industry.
- **Predictive Models:** Our work results in the development of predictive models that combine solid-state descriptors from MD simulations with machine learning algorithms. These models offer accurate predictions of T_g and crystallisation class, making them valuable tools for pharmaceutical formulation screening.
- **Foundation for Future Research:** Despite data limitations, our thesis serves as a foundation for future research in this area, with the potential for further improvements in predictive accuracy as more data becomes available.
- **Optimisation of Descriptors:** In Chapter 5, we optimised the SOAP descriptor, enhancing its efficacy in machine learning applications and materials science.
- **Potential for Industry Impact:** Our research holds potential for significant advancements in the pharmaceutical industry by improving the development of stable amorphous drug formulations.

In **chapter 4** we give a special example of how MD simulations can be useful in other similar fields of study. In this chapter, we aimed to investigate the microscopic origins of the boson peak, which is a common feature observed in glasses as an excess in heat capacity or an additional peak in the terahertz vibrational spectrum. To achieve this, we studied liquids that are made up of highly symmetric molecules and used depolarised Raman scattering to isolate and observe the boson peak from the liquid state into the glass state. We observed that the boson peak becomes more intense as the temperature is lowered, and we also detected the simultaneous appearance of a pre-peak due to molecular clusters consisting of about 20 molecules through wide-angle x-ray scattering. By conducting atomistic molecular dynamics simulations, we were able to identify that these clusters were caused by over-coordinated molecules. Our findings represent an essential contribution towards understanding the physics of vitrification.

In **chapter 5** we focused on the optimisation of the Smooth Overlap of Atomic Positions (SOAP) descriptor, which is a relatively new descriptor gaining popularity in machine

learning applications. Specifically, we utilised a genetic algorithm to optimise the SOAP descriptor for a large dataset, and we demonstrated that it outperforms a random grid search. Our genetic algorithm is capable of searching the parameter space for optimal SOAP descriptor parameters in a more efficient manner, as compared to conventional random grid searches.

In addition to improving the performance of the SOAP descriptor, our work has important implications for the field of machine learning and materials science. The SOAP descriptor has been shown to be highly effective in describing the structural and chemical properties of materials, and is therefore an important tool for developing accurate machine learning models. By optimising the SOAP descriptor using a genetic algorithm, we have enhanced the predictive power of machine learning models that rely on the SOAP descriptor.

Overall, we believe that the results of this chapter will help to accelerate the adoption of the SOAP descriptor in machine learning applications. Our genetic algorithm provides a more efficient way of optimising the SOAP descriptor, which can improve the accuracy of predictive models, reduce computational costs, and enhance the efficiency of materials design. This nature of our SOAP_GAS package also allows researchers to use the GA for other descriptors, and we show that for H-wACSF the GA provides a substantial improvement in the accuracy of predictions.

In **chapter 6**, we present the results of our study, which demonstrate the potential of descriptors derived from molecular simulations in predicting two important characteristics of amorphous drug formulations - T_g and crystallisation class. Our approach involved the use of molecular dynamics simulations to generate bespoke descriptors that capture the structural and thermodynamic properties of amorphous drug molecules.

Our results indicate that the combination of molecular simulations with machine learning techniques can lead to highly accurate predictions of T_g and crystallisation class. We compare our results with the current state-of-the-art in this area and find that in some cases, our approach outperforms existing methods. This suggests that our approach has the potential to significantly improve the development of amorphous drug formulations.

Furthermore, we believe that there is still room for further improvement. For instance, more advanced machine learning techniques and optimisation strategies could potentially be employed to enhance the accuracy of our predictions. Additionally, the use of more sophisticated molecular dynamics simulations may also lead to the generation of more informative descriptors that can better capture the behavior of amorphous drug molecules under different conditions.

This study advances beyond the prevailing state of the art by not only enhancing the predictive accuracy in the domains of crystallisation class classification and glass tran-

sition temperature prediction but also by introducing a novel array of molecular descriptors. The framework established herein not only optimises these descriptors effectively through a GA but also substantiates their efficacy on our specific dataset. This innovation potentially paves the way for broader applicability, as these descriptors may find utility in diverse datasets in future research endeavors by others.

Overall, our study highlights the potential of combining molecular simulations with machine learning techniques for the development of amorphous drug formulations. The results of this work provide a foundation for further research in this area and demonstrate the potential for significant advancements in the field of pharmaceutical science.

7.2 Further work

One of the key limitations of this work has been the lack of available data for training and validating our models. As we have seen, with the limited data available, our models were able to predict the physical stability of amorphous drugs with reasonable accuracy. However, there is a clear need for more data to improve the accuracy and reliability of these models. If we had additional time and resources, it would have potentially been very beneficial to increase the size of our dataset using computationally generated molecules, this is definitley something that could be done in future work.

In this context, generative machine learning methods represent a promising avenue for future research. Specifically, generative adversarial networks (GANs) and other generative AI techniques could be used to create computationally generated molecules that mimics the characteristics of our real-world data. This computationally generated data could then be used to train and validate machine learning models, providing a larger and more diverse dataset than is currently available. As far as we are aware, there is currently no ongoing research using computationally generated data to predict crystallisation class.

One potential advantage of generative methods is that they can be used to create data that reflects the underlying distribution of the real data, rather than simply replicating the patterns present in the limited existing dataset. This could help to overcome the issue of data imbalance, which can be a problem when training machine learning models on small datasets.

Of course, there are also potential challenges and limitations associated with using generative methods for this purpose. For example, it may be difficult to ensure that the synthetic data accurately reflects the real-world data distribution, or to generate data that is sufficiently diverse to capture all relevant features of the data.

Additionally, further exploration of advanced machine learning techniques and op-

timization strategies is warranted. By continuously refining and evolving our modeling approaches, we can expect enhanced predictive accuracy and more robust models. Moreover, the incorporation of state-of-the-art molecular dynamics simulations may unlock the potential for generating even more informative descriptors. This, in turn, could enable us to capture and comprehend the nuanced behavior of amorphous drug molecules under an extensive range of conditions.

Also, reinforcement learning (RL) holds intriguing potential within the context of pharmaceutical science and predictive modeling. Specifically, RL algorithms could be employed to optimise and automate the process of molecular simulation and descriptor generation. By setting up a reinforcement learning framework, researchers can develop agents that learn to navigate the vast parameter space of molecular dynamics simulations, autonomously selecting simulation conditions that maximise the informativeness of generated data. Furthermore, RL can facilitate the dynamic adaptation of simulation parameters based on the evolving needs of predictive models, ensuring that descriptors are continuously refined for enhanced accuracy. This not only expedites the descriptor optimisation process but also empowers researchers to harness the full potential of computational resources efficiently. Additionally, RL can play a pivotal role in automating the selection of appropriate machine learning algorithms and hyperparameters, streamlining the model-building process and paving the way for more comprehensive predictive models in the realm of pharmaceutical science.

In conclusion, the completion of this thesis marks a significant milestone in the pursuit of predictive modeling for amorphous drug formulations. However, the road ahead is paved with opportunities for further research and innovation. By addressing data limitations through generative methods, exploring advanced modeling techniques, and refining molecular simulations, we can collectively contribute to the continued advancement of pharmaceutical science, ultimately benefiting the industry and society at large.

Bibliography

[1] E. Hadjittofis, M. A. Isbell, V. Karde, S. Varghese, C. Ghoroi and J. Y. Heng, *Pharmaceutical Research*, 2018, **35**, 1–22.

[2] S. Datta and D. J. Grant, *Nature Reviews Drug Discovery*, 2004, **3**, 42–57.

[3] S. Kalepu and V. Nekkanti, *Acta Pharmaceutica Sinica B*, 2015, **5**, 442–453.

[4] T. Loftsson and M. E. Brewster, *Journal of pharmacy and pharmacology*, 2010, **62**, 1607–1621.

[5] R. Laitinen, K. Löbmann, C. J. Strachan, H. Grohganz and T. Rades, *International journal of pharmaceutics*, 2013, **453**, 65–79.

[6] S. B. Murdande, M. J. Pikal, R. M. Shanker and R. H. Bogner, *Journal of pharmaceutical sciences*, 2011, **100**, 4349–4356.

[7] G. Chawla and A. K. Bansal, *European journal of pharmaceutical sciences*, 2007, **32**, 45–57.

[8] B. C. Hancock and M. Parks, *Pharmaceutical research*, 2000, **17**, 397–404.

[9] C. Leuner and J. Dressman, *European journal of Pharmaceutics and Biopharmaceutics*, 2000, **50**, 47–60.

[10] T. Vasconcelos, S. Marques, J. das Neves and B. Sarmento, *Advanced drug delivery reviews*, 2016, **100**, 85–101.

[11] D. Q. Craig, P. G. Royall, V. L. Kett and M. L. Hopton, *International journal of pharmaceutics*, 1999, **179**, 179–207.

[12] E. O. Kissi, H. Grohganz, K. Lobmann, M. T. Ruggiero, J. A. Zeitler and T. Rades, *The Journal of Physical Chemistry B*, 2018, **122**, 2803–2808.

[13] A. D. Phan, K. Wakabayashi, M. Paluch and V. D. Lam, *RSC advances*, 2019, **9**, 40214–40221.

[14] S. Huang and R. O. Williams, *AAPS PharmSciTech*, 2018, **19**, 1971–1984.

[15] A. T. Serajuddin, *Journal of pharmaceutical sciences*, 1999, **88**, 1058–1066.

[16] C. Bhugra, R. Shmeis and M. J. Pikal, *Journal of pharmaceutical sciences*, 2008, **97**, 4446–4458.

[17] X. Ma and R. O. Williams III, *Journal of Drug Delivery Science and Technology*, 2019, **50**, 113–124.

[18] A. Singh and G. Van den Mooter, *Advanced drug delivery reviews*, 2016, **100**, 27–50.

[19] M. Myślińska, M. W. Stocker, S. Ferguson and A. M. Healy, *Journal of Pharmaceutical Sciences*, 2023.

[20] A. H. Ibrahim, H. M. Ibrahim, H. R. Ismael and A. M. Samy, *Pharmaceutical Development and Technology*, 2018, **23**, 358–369.

[21] A. N. Bristol, M. S. Lamm and Y. Li, *Molecular Pharmaceutics*, 2021, **18**, 4299–4309.

[22] X. Lin, Y. Hu, L. Liu, L. Su, N. Li, J. Yu, B. Tang and Z. Yang, *Pharmaceutical research*, 2018, **35**, 1–18.

[23] A. Schittny, J. Huwyler and M. Puchkov, *Drug Delivery*, 2020, **27**, 110–127.

[24] S. V. Bhujbal, B. Mitra, U. Jain, Y. Gong, A. Agrawal, S. Karki, L. S. Taylor, S. Kumar and Q. T. Zhou, *Acta Pharmaceutica Sinica B*, 2021, **11**, 2505–2536.

[25] S. Greco, J.-R. Authelin, C. Leveder and A. Segalini, *Pharmaceutical research*, 2012, **29**, 2792–2805.

[26] B. Cassel and B. Twombly, *American laboratory (Fairfield)*, 1998, **30**, 24–27.

[27] J. E. Patterson, M. B. James, A. H. Forster, R. W. Lancaster, J. M. Butler and T. Rades, *Journal of pharmaceutical sciences*, 2005, **94**, 1998–2012.

[28] A. Alzghoul, A. Alhalaweh, D. Mahlin and C. A. Bergström, *Journal of Chemical Information and Modeling*, 2014, **54**, 3396–3403.

[29] D. Mahlin and C. A. Bergström, *European Journal of Pharmaceutical Sciences*, 2013, **49**, 323–332.

[30] R. Fulchiron, I. Belyamani, J. U. Otaigbe and V. Bounor-Legaré, *Scientific reports*, 2015, **5**, 8369.

[31] R. Han, H. Xiong, Z. Ye, Y. Yang, T. Huang, Q. Jing, J. Lu, H. Pan, F. Ren and D. Ouyang, *Journal of Controlled Release*, 2019, **311**, 16–25.

[32] H. Lee, J. Kim, S. Kim, J. Yoo, G. J. Choi and Y.-S. Jeong, *Journal of Chemistry*, 2022.

[33] K. Nurzynska, J. Booth, C. J. Roberts, J. McCabe, I. Dryden and P. M. Fischer, *Molecular pharmaceutics*, 2015, **12**, 3389–3398.

[34] S. Ekins, *Pharmaceutical research*, 2016, **33**, 2594–2603.

[35] Y. Yang, Z. Ye, Y. Su, Q. Zhao, X. Li and D. Ouyang, *Acta pharmaceutica sinica B*, 2019, **9**, 177–185.

[36] S. A. Kumar, T. D. Ananda Kumar, N. M. Beeraka, G. V. Pujar, M. Singh, H. S. Narayana Akshatha and M. Bhagyalalitha, *Future Medicinal Chemistry*, 2022, **14**, 245–270.

[37] Y. Xu, K. Lin, S. Wang, L. Wang, C. Cai, C. Song, L. Lai and J. Pei, *Future medicinal chemistry*, 2019, **11**, 567–597.

[38] F. Raimundo, L. Meng-Papaxanthos, C. Vallot and J.-P. Vert, *Current Opinion in Systems Biology*, 2021, **26**, 64–71.

[39] E. H. Weissler, T. Naumann, T. Andersson, R. Ranganath, O. Elemento, Y. Luo, D. F. Freitag, J. Benoit, M. C. Hughes, F. Khan *et al.*, *Trials*, 2021, **22**, 1–15.

[40] P. Shah, F. Kendall, S. Khozin, R. Goosen, J. Hu, J. Laramie, M. Ringel and N. Schork, *NPJ digital medicine*, 2019, **2**, 69.

[41] G. Szarvas, R. Farkas and R. Busa-Fekete, *Journal of the American Medical Informatics Association*, 2007, **14**, 574–580.

[42] X. Zeng, S. Zhu, W. Lu, Z. Liu, J. Huang, Y. Zhou, J. Fang, Y. Huang, H. Guo, L. Li *et al.*, *Chemical Science*, 2020, **11**, 1775–1797.

[43] F. Yang, Q. Zhang, X. Ji, Y. Zhang, W. Li, S. Peng and F. Xue, *Interdisciplinary Sciences: Computational Life Sciences*, 2022, **14**, 15–21.

[44] M. I. Miller, L. C. Shih and V. B. Kolachalama, *Neurotherapeutics*, 2023, 1–15.

[45] E. Lin, C.-H. Lin and H.-Y. Lane, *Molecules*, 2020, **25**, 3250.

[46] J. C. Mauro, Y. Yue, A. J. Ellison, P. K. Gupta and D. C. Allan, *Proceedings of the National Academy of Sciences*, 2009, **106**, 19780–19784.

[47] A. Alhalaweh, A. Alzghoul, W. Kaialy, D. Mahlin and C. A. Bergström, *Molecular Pharmaceutics*, 2014, **11**, 3123–3132.

[48] M. Rams-Baron, *Amorphous drugs: benefits and challenges*, Springer Berlin Heidelberg, New York, NY, 2018.

[49] M. Chanda, *Introduction to polymer science and chemistry: a problem-solving approach*, CRC Press, 2006.

[50] A. Alhalaweh, A. Alzghoul, D. Mahlin and C. A. Bergström, *International journal of pharmaceutics*, 2015, **495**, 312–317.

[51] S. Baghel, H. Cathcart and N. J. O'Reilly, *Journal of pharmaceutical sciences*, 2016, **105**, 2527–2544.

[52] J. A. Baird, B. Van Eerdenbrugh and L. S. Taylor, *Journal of pharmaceutical sciences*, 2010, **99**, 3787–3806.

[53] D. Weininger, *Journal of chemical information and computer sciences*, 1988, **28**, 31–36.

[54] *Datasets*, `http://moleculenet.ai/datasets-1`.

[55] E. Anderson, G. Veith and D. Weininger, *Environmental Research Laboratory-Duluth. Report No. EPA/600/M-87/021*, 1987.

[56] E. Rutkowska, K. Pajak and K. Jóźwiak, *Acta Pol Pharm*, 2013, **70**, 3–18.

[57] C. Lu, P. Li, R. Gallegos, V. Uttamsingh, C. Q. Xia, G. T. Miwa, S. K. Balani and L.-S. Gan, *Drug Metab Dispos*, 2006, **34**, 1600–1605.

[58] S. Boobier, A. Osbourn and J. B. Mitchell, *Journal of cheminformatics*, 2017, **9**, 1–14.

[59] C. Saal and A. Nair, *Solubility in pharmaceutical chemistry*, Walter de Gruyter GmbH & Co KG, 2020.

[60] A. Llinàs, R. C. Glen and J. M. Goodman, *J. Chem. Inf. Model.*, 2008, **48**, 1289–1303.

[61] A. Llinas, I. Oprisiu and A. Avdeef, *Journal of chemical information and modeling*, 2020, **60**, 4791–4803.

[62] M. C. Sorkun, J. V. A. Koelman and S. Er, *Iscience*, 2021, **24**, 101961.

[63] L. C. Blum and J.-L. Reymond, *J. Am. Chem. Soc.*, 2009, **131**, 8732.

[64] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller and O. A. v. Lilienfeld, *New J. Phys.*, 2013, **15**, 095003.

[65] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, *Journal of Cheminformatics*, 2011, **3**, 33.

[66] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov *et al.*, *Journal of computational chemistry*, 2010, **31**, 671–690.

[67] W. Yu, X. He, K. Vanommeslaeghe and A. D. MacKerell Jr, *Journal of computational chemistry*, 2012, **33**, 2451–2468.

[68] K. Vanommeslaeghe and A. D. MacKerell Jr, *Journal of chemical information and modeling*, 2012, **52**, 3144–3154.

[69] K. Vanommeslaeghe, E. P. Raman and A. D. MacKerell Jr, *Journal of chemical information and modeling*, 2012, **52**, 3155–3168.

[70] Q. Ke, X. Gong, S. Liao, C. Duan and L. Li, *Journal of Molecular Liquids*, 2022, **365**, 120116.

[71] J. R. Perilla, B. C. Goh, C. K. Cassidy, B. Liu, R. C. Bernardi, T. Rudack, H. Yu, Z. Wu and K. Schulten, *Current opinion in structural biology*, 2015, **31**, 64–74.

[72] J. Norberg and L. Nilsson, *Quarterly Reviews of Biophysics*, 2003, **36**, 257–306.

[73] F. Bachtiger, T. R. Congdon, C. Stubbs, M. I. Gibson and G. C. Sosso, *Nature Communications*, 2021, **12**, 1323.

[74] C. A. Stevens, F. Bachtiger, X.-D. Kong, L. A. Abriata, G. C. Sosso, M. I. Gibson and H.-A. Klok, *Nat Commun*, 2021, **12**, 2675.

[75] C. M. Miles, P.-C. Hsu, A. M. Dixon, S. Khalid and G. C. Sosso, *Phys. Chem. Chem. Phys.*, 2022, **24**, 6476–6491.

[76] M. T. Warren, I. Galpin, F. Bachtiger, M. I. Gibson and G. C. Sosso, *J. Phys. Chem. Lett.*, 2022, **13**, 2237–2244.

[77] G. C. Sosso, P. Sudera, A. T. Backes, T. F. Whale, J. Fröhlich-Nowoisky, M. Bonn, A. Michaelides and E. H. G. Backus, *Chem. Sci.*, 2022, **13**, 5014–5026.

[78] M. González-Jiménez, T. Barnard, B. A. Russell, N. V. Tukachev, U. Javornik, L.-A. Hayes, A. J. Farrell, S. Guinane, H. M. Senn, A. J. Smith, M. Wilding, G. Mali, M. Nakano, Y. Miyazaki, P. McMillan, G. C. Sosso and K. Wynne, *Nat Commun*, 2023, **14**, 215.

[79] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen, *The Journal of chemical physics*, 1995, **103**, 8577–8593.

[80] L. Martínez, R. Andrade, E. G. Birgin and J. M. Martínez, *Journal of computational chemistry*, 2009, **30**, 2157–2164.

[81] G. Landrum *et al.*, 2016.

[82] S. Riniker and G. A. Landrum, *J. Chem. Inf. Model.*, 2015, **55**, 2562–2574.

[83] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard and W. M. Skiff, *J. Am. Chem. Soc.*, 1992, **114**, 10024–10035.

[84] R. Todeschini and V. Consonni, *Handbook of Chemoinformatics: From Data to Knowledge in 4 Volumes*, 2003, 1004–1033.

[85] P. Gramatica, *QSAR & Combinatorial Science*, 2006, **25**, 327–332.

[86] T. Barnard, H. Hagan, S. Tseng and G. C. Sosso, *Molecular Systems Design & Engineering*, 2020, **5**, 317–329.

[87] W. Jin, R. Barzilay and T. Jaakkola, *arXiv:1802.04364 [cs, stat]*, 2018.

[88] C. R. Collins, G. J. Gordon, O. A. von Lilienfeld and D. J. Yaron, *The Journal of Chemical Physics*, 2018, **148**, 241718.

[89] J. Behler and M. Parrinello, *Phys. Rev. Lett.*, 2007, **98**, 146401.

[90] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi and P. Marquetand, *The Journal of Chemical Physics*, 2018, **148**, 241709.

[91] G. C. Sosso, V. L. Deringer, S. R. Elliott and G. Csányi, *Molecular Simulation*, 2018, **44**, 866–880.

[92] A. Singraber, T. Morawietz, J. Behler and C. Dellago, *J. Phys.: Condens. Matter*, 2018, **30**, 254005.

[93] J. Li, K. Song and J. Behler, *Physical Chemistry Chemical Physics*, 2019, **21**, 9672–9682.

[94] J. Behler, *The Journal of Chemical Physics*, 2011, **134**, 074106.

[95] F. C. Mocanu, K. Konstantinou, T. H. Lee, N. Bernstein, V. L. Deringer, G. Csányi and S. R. Elliott, *J. Phys. Chem. B*, 2018, **122**, 8998–9006.

[96] V. Quaranta, J. Behler and M. Hellström, *J. Phys. Chem. C*, 2019, **123**, 1293–1304.

[97] A. P. Bartók, R. Kondor and G. Csányi, *Physical Review B*, 2013, **87**, 184115.

[98] S. N. Pozdnyakov, M. J. Willatt, A. P. Bartók, C. Ortner, G. Csányi and M. Ceriotti, *Phys. Rev. Lett.*, 2020, **125**, 166001.

[99] M. O. Jäger, E. V. Morooka, F. Federici Canova, L. Himanen and A. S. Foster, *npj Computational Materials*, 2018, **4**, 1–8.

[100] J. L. Priedeman, C. W. Rosenbrock, O. K. Johnson and E. R. Homer, *Acta Materialia*, 2018, **161**, 431–443.

[101] M. A. Caro, *Physical Review B*, 2019, **100**, 024112.

[102] S. De, A. P. Bartók, G. Csányi and M. Ceriotti, *Physical Chemistry Chemical Physics*, 2016, **18**, 13754.

[103] R. Todeschini and P. Gramatica, *3D QSAR in drug design*, Springer, 2002, pp. 355–380.

[104] V. Zaverkin and J. Kästner, *Journal of Chemical Theory and Computation*, 2020, **16**, 5410–5421.

[105] A. Goscinski, F. Musil, S. Pozdnyakov, J. Nigam and M. Ceriotti, *The Journal of Chemical Physics*, 2021, 1–12.

[106] M. O. J. Jäger, E. V. Morooka, F. F. Canova, L. Himanen and A. S. Foster, *npj Computational Materials*, 2018, 1–8.

[107] K. De Jong, *Machine learning*, Elsevier, 1990, pp. 611–638.

[108] J. J. Grefenstette, 1993, 3–4.

[109] P. K. Gupta, *Journal of Non-Crystalline Solids*, 1996, **195**, 158–164.

[110] Y.-T. Cheng and W. L. Johnson, *Science*, 1987, **235**, 997–1002.

[111] L. Van Hove, *Physical Review*, 1954, **95**, 249.

[112] D. P. Kingma and M. Welling, *arXiv preprint arXiv:1312.6114*, 2013.

[113] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Communications of the ACM*, 2020, **63**, 139–144.

[114] A. Gisbrecht, A. Schulz and B. Hammer, *Neurocomputing*, 2015, **147**, 71–82.

[115] N. Gebauer, M. Gastegger and K. Schütt, *Advances in neural information processing systems*, 2019, **32**, year.

[116] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, *Journal of artificial intelligence research*, 2002, **16**, 321–357.

[117] C. Guo, G. Pleiss, Y. Sun and K. Q. Weinberger, 2017, 1321–1330.

[118] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, *Communications of the ACM*, 2021, **64**, 107–115.

[119] X. Glorot, A. Bordes and Y. Bengio, 2011, 315–323.

[120] D. P. Kingma and J. Ba, *arXiv preprint arXiv:1412.6980*, 2014.

[121] S. Ruder, *arXiv preprint arXiv:1609.04747*, 2016.

[122] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi *et al.*, *Advances in Neural Information Processing Systems*, 2020, **33**, 21285–21296.

[123] L. Breiman, 2004.

[124] G. Biau and E. Scornet, *Test*, 2016, **25**, 197–227.

[125] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich and F. A. Hamprecht, *BMC Bioinformatics*, 2009, **10**, 213 – 213.

[126] L. Breiman, *Machine learning*, 2001, **45**, 5–32.

[127] G. Louppe, *arXiv preprint arXiv:1407.7502*, 2014.

[128] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

[129] C. Asgreen, M. M. Knopp, J. Skytte and K. Löbmann, *Pharmaceutics*, 2020, **12**, 483.

[130] B. M. Spowage, C. L. Bruce and J. D. Hirst, *Journal of cheminformatics*, 2009, **1**, 1–13.

[131] S. A. Wildman and G. M. Crippen, *Journal of chemical information and computer sciences*, 1999, **39**, 868–873.

[132] R. E. Schapire, *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, 2013, 37–52.

[133] C. A. Angell, K. L. Ngai, G. B. McKenna, P. F. McMillan and S. W. Martin, *Journal of applied physics*, 2000, **88**, 3113–3157.

[134] H. B. Yu, W. H. Wang, H. Y. Bai and K. Samwer, *National Science Review*, 2014, **1**, 429–461.

[135] R. Boyer, *Polymer Engineering & Science*, 1968, **8**, 161–185.

[136] K. Ueno and C. A. Angell, *The Journal of Physical Chemistry B*, 2011, **115**, 13994–13999.

[137] G. Johari, *The Journal of Physical Chemistry B*, 2019, **123**, 3010–3023.

[138] G. Johari, *Journal of non-crystalline solids*, 2002, **307**, 317–325.

[139] H. Tanaka, *Physical Review E*, 2004, **69**, 021502.

[140] M. A. Ramos, *Low-Temperature Thermal and Vibrational Properties of Disordered Solids: A Half-Century of Universal "Anomalies" of Glasses*, World Scientific, 2023, pp. 1–20.

[141] T. S. Grigera, V. Martín-Mayor, G. Parisi and P. Verrocchio, *Nature*, 2003, **422**, 289–292.

[142] D. Parshin, H. Schober and V. Gurevich, *Physical Review B*, 2007, **76**, 064206.

[143] C. Alvarez-Ney, J. Labarga, M. Moratalla, J. Castilla and M. Ramos, *Journal of Low Temperature Physics*, 2017, **187**, 182–191.

[144] W. Schirmacher, G. Ruocco and T. Scopigno, *Physical review letters*, 2007, **98**, 025501.

[145] S. Taraskin, Y. Loh, G. Natarajan and S. Elliott, *Physical review letters*, 2001, **86**, 1255.

[146] E. Lerner and E. Bouchbinder, *The Journal of chemical physics*, 2021, **155**, 200901.

[147] H. Shintani and H. Tanaka, *Nature materials*, 2008, **7**, 870–877.

[148] Y.-C. Hu and H. Tanaka, *Nature Physics*, 2022, **18**, 669–677.

[149] H. Tanaka, *The Journal of chemical physics*, 1999, **111**, 3163–3174.

[150] E. Lerner and E. Bouchbinder, *arXiv preprint arXiv:2210.10326*, 2022.

[151] A. Chumakov, G. Monaco, A. Monaco, W. Crichton, A. Bosak, R. Rüffer, A. Meyer, F. Kargl, L. Comez, D. Fioretto *et al.*, *Physical Review Letters*, 2011, **106**, 225501.

[152] M. Baggioli and A. Zaccone, *Physical review letters*, 2019, **122**, 145501.

[153] A. I. Chumakov, G. Monaco, A. Fontana, A. Bosak, R. P. Hermann, D. Bessas, B. Wehinger, W. A. Crichton, M. Krisch, R. Rüffer *et al.*, *Physical review letters*, 2014, **112**, 025502.

[154] J. S. Bender, M. Zhi and M. T. Cicerone, *Soft Matter*, 2020, **16**, 5588–5598.

[155] H. Cang, J. Li, H. C. Andersen and M. Fayer, *The Journal of chemical physics*, 2005, **123**, 064508.

[156] J. Reichenbach, S. A. Ruddell, M. Gonzalez-Jimenez, J. Lemes, D. A. Turton, D. J. France and K. Wynne, *Journal of the American Chemical Society*, 2017, **139**, 7160–7163.

[157] A. J. Farrell, M. Gonzalez-Jimenez, G. Ramakrishnan and K. Wynne, *The Journal of Physical Chemistry B*, 2020, **124**, 7611–7624.

[158] F. Walton, J. Bolling, A. Farrell, J. MacEwen, C. D. Syme, M. G. Jiménez, H. M. Senn, C. Wilson, G. Cinque and K. Wynne, *Journal of the American Chemical Society*, 2020, **142**, 7591–7597.

[159] M. González-Jiménez, G. Ramakrishnan, N. V. Tukachev, H. M. Senn and K. Wynne, *Physical Chemistry Chemical Physics*, 2021, **23**, 13250–13260.

[160] A. Farrell, M. G. Jiménez, N. Tukachev, D. A. Turton, B. A. Russell, S. Guinane, H. M. Senn and K. Wynne, 2021.

[161] P. Lunkenheimer, U. Schneider, R. Brand and A. Loid, *Contemporary Physics*, 2000, **41**, 15–36.

[162] D. A. Turton and K. Wynne, *The Journal of chemical physics*, 2009, **131**, 201101.

[163] C. Darwin, *The origin of species*, 1909.

[164] B. N. Bartok, Albert and J. Kermode, *GAP and SOAP documentation - GAP documentation*.

[165] M. Olson, A. Wyner and R. Berk, *Advances in Neural Information Processing Systems*, 2018, **31**, year.

[166] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins and N. Khovanova, *Biomedical Signal Processing and Control*, 2019, **52**, 456–462.

[167] N. M. O'Boyle, C. Morley and G. R. Hutchison, *Chemistry Central Journal*, 2008, **2**, 1–7.

[168] K. Chen, C. Kunkel, K. Reuter and J. T. Margraf, *Digital Discovery*, 2022, **1**, 147–157.

[169] S. Axelrod and R. Gomez-Bombarelli, *Molecular machine learning with conformer ensembles*, 2021, http://arxiv.org/abs/2012.08452, Number: arXiv:2012.08452 arXiv:2012.08452 [physics].

[170] J. P. Darby, J. R. Kermode and G. Csányi, *arXiv preprint arXiv:2112.13055*, 2021.

[171] D. S. Palmer and J. B. Mitchell, *Molecular Pharmaceutics*, 2014, **11**, 2962–2972.

[172] S. Boobier, D. R. Hose, A. J. Blacker and B. N. Nguyen, *Nature communications*, 2020, **11**, 1–10.

[173] A. Avdeef, *ADMET and DMPK*, 2020, **8**, 29–77.

[174] M. Lovrić, K. Pavlović, P. Žuvela, A. Spataru, B. Lučić, R. Kern and M. W. Wong, *Journal of Chemometrics*, 2021, **35**, e3349.

[175] A. Mauri, V. Consonni, M. Pavan and R. Todeschini, *Match*, 2006, **56**, 237–248.

[176] I. Olier, N. Sadawi, G. R. Bickerton, J. Vanschoren, C. Grosan, L. Soldatova and R. D. King, *Mach Learn*, 2018, **107**, 285–311.

[177] A. Bender, J. L. Jenkins, J. Scheiber, S. C. K. Sukuru, M. Glick and J. W. Davies, *J. Chem. Inf. Model.*, 2009, **49**, 108–119.

[178] M. Dehmer, F. Emmert-Streib and S. Tripathi, *PLOS ONE*, 2013, **8**, e83956.

[179] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek and K.-R. Müller, *Nat Commun*, 2019, **10**, 1–8.

[180] D. Castelvecchi, *Nature News*, 2016, **538**, 20.

[181] J. Drews, *Science*, 2000, **287**, 1960–1964.

[182] G. C. Sosso, *Less may be more: an informed reflection on molecular descriptors for drug design and discovery: gcsosso/MSDE_Sosso_alpha*, 2019, https://github.com/gcsosso/MSDE_Sosso_alpha, original-date: 2019-08-16T09:23:18Z.

[183] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

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

[185] K. A. Ross, *Encyclopedia of Database Systems*, Springer US, Boston, MA, 2009, pp. 301–304.

[186] S. Raschka, *Python Machine Learning*, Packt Publishing, 2015.

[187] K. Pearson, *Proceedings of the Royal Society of London*, 1895, **58**, 240–242.

[188] T. Barnard, S. Steng, J. Darby, A. P. Bartók, A. Broo and G. C. Sosso, *Mol. Syst. Des. Eng.*, 2022.

[189] B. W. Matthews, *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 1975, **405**, 442–451.

[190] D. Chicco and G. Jurman, *BMC Genomics*, 2020, **21**, 6.