Application Of Neural Network Algorithms To Chemical Problems

by
Silvia Amabilino

Department of Chemistry
University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctor of Philosophy in the Faculty of Science.

September 2019

Word count: 41481

Abstract

In this thesis, neural networks are applied to fitting potential energy surfaces (PES) and to *de novo* drug design, to study the current suitability and effectiveness of these algorithms to different chemical problems.

The first goal was to fit the PES of a long hydrocarbon chain (30 carbons) reacting with a cyano radical (CN). The size of this system makes creating a training set computationally expensive and time consuming. Consequently, a 'fragment-learning' approach was employed. The training data set was constructed using hydrocarbons no larger than hexane reacting with CN, as this would reduce the time required to both generate the data and training the neural network. Thanks to the software developed during this project, the fitted PES showed mean absolute errors within $10\,\mathrm{kJ\,mol^{-1}}$ compared to the reference data. In addition, the prediction times were a couple of orders of magnitude faster than the reference electronic structure calculations. This result is encouraging because it shows the transferability of neural networks potentials of reactive systems.

The second goal was to study the ability of recurrent neural networks (RNNs) to generate new drug candidates. Initially, multiple techniques described in the literature, such as fine-tuning and reinforcement learning, were used to designing new Kinase inhibitors. From this first exploratory phase it became clear that the quality of the fine-tuning data set has a heavy impact on the results. Consequently, a more deep investigation of the process of fine-tuning RNNs for medicinal chemistry projects was carried out. The results suggest that RNNs should not be fine-tuned with fewer than 250-300 samples, although more are needed if the molecules in the data set are very diverse. This means that in their current form, RNNs may not be the best tool for the early stages of *de novo* drug design projects and further development is needed.

Dedication & Acknowledgements

The work presented in this thesis would not have been possible without the help and contributions of many people.

First of all, I would like to thank the EPSRC Theory and Modelling in the Chemical Sciences (TMCS) Centre for Doctoral Training for providing funding and a first year of training, without which I would not have been able to carry out this project in the allocated time. Thank you to my TMCS cohort for both academic and moral support.

Thank you to my supervisor Dr. David Glowacki for providing additional funding, resources and opportunities that enriched my learning experience.

The members of the Glowacki research group also helped considerably. The contributions of the following people were particularly invaluable. Dr. Lars Bratholm taught me most of what I know about applying machine learning to chemical systems. Without him, I would probably still be using the Coulomb matrix as a molecular descriptor. Dr. Mike O'Connor initially helped with the computer science aspect of the project and was incredibly patient. Dr. Simon Bennie contributed electronic structure knowledge and spent considerable time helping me setting up the VR to generate datasets. Rob(plotlib) Arbon was extremely supportive and got me through the difficult times of my PhD. They are also a gold mine of learning resources, ideas and suggestions, which have greatly contributed to my scientific development.

Thank you also to Dr. Natalie Fey, for her academic and personal support in the last few months of my PhD.

From outside the University of Bristol, I would like to thank Dr. Michael Mazanetz for giving me the opportunity to learn about recurrent neural networks and presenting my work while working for his company.

Many thanks also to the people at GlaxoSmithKline for giving me the chance to work in their computational chemistry group. I really enjoyed my time in Stevenage and learnt a lot. In particular, thank you to Dr. Peter Pogány, Dr. Darren Green and Dr. Stephen Pickett.

Thank you to Mirko Franchini, for painstakingly reading through my thesis with a critical eye and providing invaluable feedback.

Finally, I would like to thank my family for their encouragement and support.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

Contents

List of Figures

List of Tables

List of Abbreviations

ACSF	Atom Centred Symmetry Functions
ATP	Adenosine TriPhosphate
CC	Coupled Cluster
CFDA	Combined Function Derivative Approximation
CMD	Constrained Molecular Dynamics
CN	Cyano radical
CPU	Central Processing Unit
DFT	Density Functional Theory
DOGS	Design Of Genuine Structures
FDA	Food and Drug Administration
FFNN	Feed Forward Neural Network
GAN	Generative Adversarial Network
GPU	Graphic Processing Unit
GRU	Gated Recurrent Unit
GSK	GlaxoSmithKline
IMD-VR	Interactive Molecular Dynamics in Virtual Reality
IMLS	Interpolating moving least squares
JAK	Janus Kinases
KL	Kullback-Leibler
KRR	Kernel Ridge Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MD	Molecular Dynamics
MS-EVB	Multi-State Empirical Valence Bond theory
NLP	Natural Language Processing
NN	Neural Network
PAINS	Pan Assay INterference compoundS
PCA	Principal Component Analysis
PES	Potential Energy Surface
QSAR	Quantitative Structure-Activity Relationship
RECAP	Retrosynthetic Combinatorial Analysis Procedure
REOS	Rapid Elimination Of Swill
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SLATM	Spectrum of London and Axilrod-Teller-Muto
SMILES	Simplified Molecular-Input Line-Entry System
VAE	Variational Auto Encoder

Part I

Fitting potential energy surfaces with neural networks

Chapter 1

Introduction to fitting potential energy surfaces

A global potential energy surface is a function that gives the potential energy of a molecular system as a function of the internal coordinates. [1] It is a high-dimensional analytic function.

They are generally constructed by fitting a large number of electronic structure calculations. In molecular dynamics simulations, potential energy surfaces can be used to obtain the energy of a system and the forces acting on each atom at every time step. This is useful, because the energies and the forces obtained will be approximately at the level of the electronic structure method used to generate the training set, but the speed of their evaluation should be higher than performing a high level electronic structure calculation.

The first fitted potential energy surfaces were obtained for systems where an atom reacts with a diatomic molecule, such as $F + H_2 \longrightarrow H + HF$ and $H + H_2 \longrightarrow H_2 + H$. [1] Additional systems whose potential energy surfaces have been fitted are reviewed in the next few sections. As the size of the molecules increases, the following steps become more complex:

1. How to generate the data set of electronic structure calculations.
2. How to represent the coordinates of the molecules in the system.
3. What algorithms to use to fit the data.

The next few sections will treat each of these steps in more detail and discuss what approaches have been presented in the literature to tackle them.

1.1 Generating a data set

There are two aspects of data set generation: how to efficiently sample the high dimensional configuration space and which method to use to compute the actual reference data. These two aspects are interrelated, since using a computationally expensive method to generate the reference data puts a constraint on the number of data points that one can realistically sample.

1.1.1 Sampling methods

One way of sampling a surface is to use a fine grid of points which systematically covers all the chemically plausible values of the degrees of freedom. This gives control over which regions of the potential energy surface are sampled and enables a homogeneous sampling of the different regions. However, grid sampling is unfeasible for all but the smallest systems. [2] For complex systems with many degrees of freedom, a variety of techniques are available. These include: [3]

1. *Molecular dynamics*: this samples the most probable and most easily accessible regions of a potential energy surface. However, energy barriers between stable states can make transitions rare to observe. [4] Pukrittayakamee et al. adjusted the time interval between sampling each configuration as a function of the average atomic acceleration in a system. In this way, when they studied the reaction between H and HBr, the regions of configuration space where the system is close to equilibrium were not sampled as often. [5] This is useful because the resulting data set has a more homogeneous density of sampling, which facilitates fitting surfaces. [6] However, to obtain a satisfactory amount of sampling in non-equilibrium regions requires running simulations for a very long time. This makes it an inefficient way of constructing a data set for fitting potential energy surfaces.

2. *Constrained molecular dynamics*: here the user selects a grid of points along the most important degrees of freedom. Then, molecular dynamics simulations are carried out with the specified degrees of freedom constrained to the grid values. [7, 3] For example, constrained molecular dynamics has been used to study the potential of mean force between the $Na^+ + Cl^-$ ion pair in solution. [7] However, this technique is not often used with more complex systems, because running the dynamic simulations with constraints can lead to molecules rearranging in unexpected ways. [3]

3. *Enhanced sampling molecular dynamics*: this is an extension of molecular dynamics that attempts to increase the sampling of fluctuation-driven processes which occur infrequently. [4] There are

a variety of methods to do enhanced sampling, such as replica exchange molecular dynamics, [8] meta-dynamics, [9] boxed molecular dynamics, [10] etc. These generally work by biasing the potential along a reaction coordinate or by raising the temperature. This alleviates the rare event problem, so that transitions between stable states can be sampled more often. [11] Meta-dynamics and boxed molecular dynamics are based on 'collective-variable' biasing. The collective variables are multidimensional functions of the Cartesian coordinates of atoms in a molecule. [12] In order to successfully sample a transition between different metastable states, one has to pick the right collective variable, where distinct regions in the collective variable space correspond to the different metastable states of the system. Finding these collective variables is not always straight-forward, as for example when studying protein folding, which often involves complex paths in conformational space. [9] Instead of methods based on collective variables biasing, one can use methods based on tempering. These rely on increasing the temperature of the system so that it becomes more likely to overcome energy barriers. [12] One disadvantage of these methods is that considerable time is spent sampling non-physical situations at high temperature, and directing which regions of configuration space will be explored is difficult. [12] Most enhanced sampling methods are non-trivial to use and require careful setting up. It is most likely for this reason that enhanced sampling methods are used less than expected for this application, with meta-dynamics being the most commonly used approach among them. [13, 14]

4. *Adaptive sampling schemes*: these usually start by selecting a set of points along a coordinate and obtaining their energies. Then, a functional form is fitted to these points and a small number of molecular dynamics trajectories are run using the potential energy surface obtained to evaluate energies and forces. New configurations are picked from these trajectories in the regions that have not been sampled in the initial data set. This procedure is continued iteratively until the surface is sufficiently accurate globally, i.e. the statistics are converged. [15] Due to its iterative nature, this method can take a considerable time to carry out. In addition, it requires having software that enables running molecular dynamics with the newly fitted potential.

5. *Interactive molecular dynamics*: this permits to manipulate molecules directly in molecular dynamics simulations, as a user can apply real-time biasing forces to specific atoms in a simulation. [16] This method enables to use human chemical intuition to bias a system to explore different regions of configurational space. In the past, the main drawback of this approach was that molecular systems are inherently 3D, which makes it difficult for a user to interact with a simulation through a 2D screen. However, Virtual Reality (VR) has recently become more affordable and its 3D technology enables to interact with an artificial three-dimensional environment. To exploit this functionality, an interactive molecular dynamics framework in virtual reality has recently

been developed. [17] This framework, called 'Narupa', relies on a client/server model. An HTC Vive VR headset is connected to a VR client in charge of rendering a 3D view. The VR client is connected to a compute server (hosted either on a local computer cluster or on a cloud supercomputer) on which a user-specified force engine performs the molecular dynamics simulation and streams the results in real-time to the VR client. Various compute engines (such as PM6 or DFTB) are available on the server side to calculate the energies and the forces acting on atoms in the simulation.

1.1.2 Accuracy of reference data

In addition to the sampling method, the level of theory at which the energies and forces are evaluated also needs to be considered. If chemical reactions have to be sampled, electronic structure methods tend to be the best choice.

These include a range of 'ab-initio' methods, which attempt to solve approximately the Schrödinger equation with increasingly complex approximations. The most accurate and computationally expensive method is Full Configuration Interaction [18], which provides the exact solution of the time-independent, non-relativistic Schrödinger equation, within the limits of the Born-Oppenheimer approximation for a given basis set. However, this technique is unfeasible for all but the smallest systems, as it scales exponentially with the number of electrons. [19] Several more approximate but computationally more efficient approaches have been devised. Among the most popular, mentioned here in order of decreasing accuracy and increasing computational speed, we have truncated configuration interaction [20] methods, coupled cluster [21], Moller-Plesset perturbation theory [22], all the way down to the Hartree-Fock method [20]. For each of the methods mentioned above, there are also different levels of approximation. For example, for truncated configuration interaction methods, the computational scaling depends on which excited states are truncated. If only singly and doubly excited states are included, then the method scales as $O(N^6)$ (where N is the number of basis functions). [19] Similarly, for coupled cluster methods if singly and doubly excited states are included, then it also scales as $O(N^6)$. The commonly used Moller-Plesset perturbation theory method MP2 scales as $O(N^5)$, while Hartree-Fock scales formally as $O(N^4)$, but can be reduced to as $O(N^2)$ depending on its implementation. [23]

Depending on the accuracy required for a particular application, different methods are chosen. For example, Glowacki et al. [24] decided to use coupled cluster to study the reaction of cyano radicals with hexane, because methods based on a lower level of theory did not give satisfactory results. On the other hand, Le et al. used Moller-Plesset perturbation theory to calculate the energies of BeH reacting with

H_2 as it was enough to reach the accuracy they were after. [25] However, the most common method encountered in the litterature to generate data for potential energy surfaces is Density Functional Theory (DFT). [26, 13, 27]

DFT is a particularly popular method due to its good balance between accuracy and computational efficiency (it scales roughly as $O(N^3)$ [28]). DFT is based on the principle that the energy of a molecule is a functional of the electron density ρ. [29] The electron density is a function of only 3 variables, compared to the wave function which is a function of 3N variables, where N is the number of electrons in the system. The energy functional $E[\rho]$ can be decomposed into three parts: the kinetic energy $T[\rho]$, the interaction energy of the electrons with the nucleus $V_{\text{ext}}[\rho]$ and the interaction energy between the electrons $V_{\text{ee}}[\rho]$.

$$E[\rho] = T[\rho] + V_{\text{ext}}[\rho] + V_{\text{ee}}[\rho] \tag{1.1}$$

For a given electron density, the interaction energy of the electrons and the nucleus ($V_{\text{ext}}[\rho]$) is known, while the others are not. Kohn and Sham reformulated this problem in a more practically useful way. They introduced a fictitious reference system with the same electron density as the original one, except in this system the electrons do not interact with each other. The kinetic energy of the non-interacting electrons ($T_{\text{s}}[\rho]$) can easily be evaluated, and a large portion of the electron-electron interaction consists of the Coulomb interactions ($V_{\text{coul}}[\rho]$). Consequently, only the non-classical interaction between electrons and the difference between the kinetic energy of interacting and non-interacting electrons remain unknown. [30] Hence, the functional can be re-written as:

$$E[\rho] = T_{\text{s}}[\rho] + V_{\text{ext}}[\rho] + E_{\text{xc}}[\rho] \tag{1.2}$$

where all the unknown terms are grouped together in $E_{\text{xc}}[\rho]$:

$$E_{\text{xc}}[\rho] = (T[\rho] - T_{\text{s}}[\rho]) + (V_{\text{ee}}[\rho] - V_{\text{coul}}[\rho]) \tag{1.3}$$

A variety of functionals of increasing complexity have been developed to approximate $E_{\text{xc}}[\rho]$. The simplest class of functionals are based on the 'Local Density Approximation' (LDA), [31] where the energy functional depends only on the value of the electron density at a point in space. A more advanced approximation is the 'generalised gradient approximation' (GGA), [32] where the energy functional depends not only on the electron density, but also on its gradient. Another common type of functionals are 'hybrid functionals', [33, 34] which include linear combinations of Hartree-Fock

exact exchange and other approximated exchange-correlation functionals. One of the most commonly used functionals is B3LYP, [33, 34] which is a combination Hartree-Fock exchange, LDA and GGA functionals. In addition to the functionals, the basis sets also affect the quality of the calculations. A basis set is the set of atomic orbitals that are used to build molecular orbitals. The atomic orbitals are usually described by basis functions or linear combinations of basis functions, which are most commonly Gaussians. [35] In minimal basis sets such as STO-3G, [36] the minimum number of basis functions are used to represent the atomic orbitals. In larger basis sets, such as double- or triple-zeta basis sets, 2 or 3 basis functions are used to describe each atomic orbital. [37]. Another way of improving the basis sets is to add 'polarisation', which means adding orbitals with higher angular momentum compared to what is required for the ground state description of each atom. [37]

Although there are several issues with approximated functionals, [38] DFT can reach an accuracy comparable to that of much more expensive wave-function methods. However, despite the development of efficient DFT codes, [39] there is still need for faster and more efficient methods that allow studying large systems. [40] For example, using hybrid functionals such as PBEh-3c, simulating systems with more than 1000 atoms is still impractical. [41]

Semi-empirical quantum-chemical methods [42] are a class of more approximate and computationally cheaper methods, which bridge the gap between quantum mechanical and force fields methods. [43] Semi-empirical quantum-chemical methods take inspiration from either Hartree-Fock or Kohn-Sham DFT. They typically consider only valence electrons described by a minimal basis set combined with a self-consistent field method. The integrals evaluated in the self-consistent calculation are drastically approximated, which results in a speed up of at least 2 orders of magnitude compared to ab-initio quantum methods. [43] To compensate for the approximations, empirical parameters are introduced and are calibrated against reliable reference data. [42] Terms describing hydrogen bonds and dispersion interactions are often included in such methods. Examples of semi-empirical quantum-chemical methods are PM6, PM7 [44] and GFN-xTB, [43] which cover a wide range of the periodic table and can be applied to very large systems, in the thousands of atoms. [44]

A similar method is Density Functional based Tight Binding (DFTB), which is an approximate method based on DFT and tight-binding methods. The main idea behind it is to describe the Hamiltonian eigenstates with linear combinations of atomic-like orbitals and replace the Hamiltonian with a parametrised one that only depends on internuclear distances. [45] The DFTB model is parametrized against DFT calculations. [46]

1.2 Algorithms for interpolating and fitting

When constructing potential energy surfaces, the reference data can either be fitted or interpolated. When fitting, one assumes that there is some error in the reference data and therefore there is no need for the fitted curve to pass exactly through the reference data points. On the other hand, with interpolation one assumes that the reference data points are 'exact', and the interpolated curve will pass exactly through them. [47]

There are a variety of methods that have been used for interpolating/fitting the results of electronic structure calculations. These include:

1. *Permutationally invariant fitting*: [48] This is a technique used to fit potential energy surfaces for systems with up to 10 atoms and 10^4 to 10^5 electronic structure data points, developed by Braams and Bowman. A potential energy surface needs to be invariant with respect to the permutations of identical atoms. In systems with more than a few atoms, it is unfeasible to produce a surface that is numerically invariant simply by replicating permutationally equivalent configurations. Hence, the authors realised that the permutational invariance had to be built directly into the surface by using a fitting basis that is permutationally invariant. They use polynomials of the interatomic distances as the bases for fitting. The disadvantage of this approach is that it is complex to derive the invariant polynomials for systems larger than 10-15 atoms. For example, systems for which a potential energy surface was constructed using this method include $CH_5{}^+$, a water dimer and CH_3CHO. [48]

2. *Cubic splines*: These are piece-wise functions where each of the 'pieces' is a third degree polynomial of form $s(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i$, where a_i, b_i, c_i and d_i are coefficients to be optimised and x_i is the i^{th} data point. The cubic spline and its first and second derivative need to be continuous. [49] The disadvantage of cubic splines for fitting potential energy surfaces is that the error grows with increasing dimensions, and for systems with more than four degrees of freedom it becomes impractical. [50] Consequently, this technique has been used only on small systems such as $He + H_2{}^+$ and $D + HCl$. [51]

3. *Modified Shepard Interpolation*: This technique was developed by Collins and collaborators. [52] The potential energy surface is constructed by taking a weighted average of Taylor series. Each of the Taylor series is expanded around one of the points present in the data set. The Taylor expansions make use of the energy and both its first and second derivative with respect to the internal coordinates. This means that this method is robust, but computationally very expensive. [53] Consequently, it has also only been applied to small systems, such as $H + CH_4$, H + CCl4,

$H + NH_3$ and $Cl + NH_3$. [53]

4. *Interpolating moving least squares (IMLS)*: The fitted potential is expressed as a linear combination of polynomial basis functions. It is based on the least squares and the weighted least squares methods. In least squares, there is a 'cost function' expressed as:

$$E_x(p) = \sum_{i=0}^{N} [p(x_i) - f_i]^2 \tag{1.4}$$

where (x_i, f_i) are the N data points used in the fitting and $p(x) = \sum_{i=0}^{M} a_i x^i$. The expression of the potential is obtained by finding the coefficients a_i in the polynomials p that minimise E_x. In weighted least squares, each term of the sum in equation 1.4 is multiplied by a weight function, which gives a greater weight to the data points x_i that are closest to x. In IMLS, the weight functions are modified to attempt solving the problem of discontinuities and the first derivatives equal to zero at the data points. [54] The disadvantage of this method is related to computational time. The value of the energy at a given point is found by solving a matrix equation, and the size of the matrices scales with the number of data points and the degree of the polynomials included. These two quantities increase as the number of degrees of freedom in a molecular system increases. This can quickly become too expensive. [55] It has been used to construct the potential energy surface of HOOH, [55] of $H_2CN \longrightarrow H + HCN$, [56] and the cis/trans isomerisation of nitrous acid. [55]

5. *Multi-State Empirical Valence Bond theory (MS-EVB)*: With MS-EVB, the molecular system is represented as a superposition of covalent and/or ionic resonance forms. The potential energy surface for the isolated system (system in the gas phase, without environment) is obtained by mixing the resonance forms. This is done by defining a Hamiltonian where the diagonal elements are the energies of the resonance forms and the off-diagonal elements are the energies of the interactions between them. Then, it is assumed that the Hamiltonian in the solution phase can be obtained from the Hamiltonian in the gas phase by modifying the elements corresponding to ionic resonance forms with a solution term. Then, the off-diagonal elements that depend on those diagonal elements are modified accordingly. The energies of the ground and excited states can then be obtained by diagonalising the new Hamiltonian. The disadvantage of this method is that it is not always evident which resonance forms should be used and the form of the off-diagonal elements in the Hamiltonian needs to be parametrised for each system. [57, 58] This method has recently been used to construct potential energy surface of the $F + CD_3CN \longrightarrow DF + CD_2CN$ reaction in CD_3CN solvent, [58] as well as that of a cyano radical reacting with propane [24] and cyclohexane [59].

6. *Kernel Ridge Regression (KRR)*: KRR is a technique related to least-squares linear regression and to ridge regression. [60] KRR generalises linear ridge regression to non-linear functions. The inputs are mapped to a different input space called 'feature space'. The input in feature space should have a linear relation to the output. Choosing the mapping function is not always easy. These functions can be very complex and result in infinite feature spaces. However, only the dot product between the input vectors in feature space is needed. So, 'kernels' are introduced, which are the dot product of two input vectors in feature space:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{1.5}$$

where $\phi(x)$ is a mapping function and $k(x, x')$ is a kernel function. This enables one to leave the mapping function and the feature space completely implicit. Consequently, one just needs to choose a kernel function instead of a mapping function. Now, the model is:

$$f(\mathbf{x}) = \sum_{i=1}^{N_{samples}} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \tag{1.6}$$

where α_i is the coefficient for a particular data point and needs to be optimised. In KRR there is usually a quadratic constraint on the norm of the parameters α_i. The minimisation problem then has a closed form solution that is:

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y} \tag{1.7}$$

where $\mathbf{K}$ is the kernel matrix, $\mathbf{I}$ is the identity matrix and $\mathbf{y}$ is the vector of outputs for all the data points. One main drawback of kernel based methods is that they are not suited to large data sets. [27] One of the key problems is to compute and store the kernel matrix, which take $\Theta(n^3)$ time and $\Theta(n^2)$ space, where n is the number of training samples. [61] KRR has attracted considerable attention in recent years. In 2017, Chmiela et al. used KRR to construct potential energy surfaces for benzene, toluene, naphthalene, ethanol, uracil, and aspirin. They used a limited number of ab initio molecular dynamics trajectories for training. They addressed the challenge of obtaining an energy-conserving force field, as normally there can be small energy and forces inconsistencies that can yield artefacts in the potential energy surface. They achieved this result by training directly on the forces and adding energy conservation constraints. [62] In the same year, Dral et al. presented a 'self-correcting' approach. They used a first KRR model to fit ab-initio energies of methyl chloride and then a second KRR model to fit the difference

between the reference ab-initio energies and the predictions of the first model. The predictions of the second model were added to the predictions of the first model as a 'correction'. [63]

7. *Artificial neural networks*: A detailed discussion of the technical aspects of artificial neural networks is presented in section 1.3 while a discussion of the recent applications of neural networks to the construction of potential energy surfaces is presented in section 1.5.

1.3 Artificial neural networks

Artificial neural networks, most commonly referred to just as 'neural networks', are a class algorithms inspired by the nervous system of animals. [64] Neural networks belong to the broader class of algorithms and computational techniques called 'machine learning'.

Given a set of labelled data such as $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$, the neural network learns the mapping between the inputs $\mathbf{x}^{(i)}$ and the outputs $y^{(i)}$, which can be non-linear. In order to be trained, both input and output data need to be available, which makes this technique part of the 'supervised learning' algorithms class. [65] The goal is to get the neural network to predict the values of $y^{(i)}$ from values of $\mathbf{x}^{(i)}$ that were not part of the data on which it was trained. Once trained, the neural network is a model of the data collected and can be used for predictive purposes. In the following sections, the neural networks will often be referred to simply as 'a model'.

Neural networks are composed of layers of units called 'neurons'. The neuron takes a vector $\mathbf{x}$ as an input and gives a scalar $h_{\mathbf{w}}$ as output (Fig. 1.1).

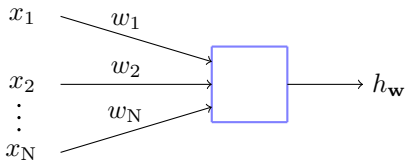


Figure 1.1: Diagram of a single neuron, with N input features.

The input vector to each neuron is multiplied by some parameters $\mathbf{w}$ that are referred to as 'weights' and added to another parameter called 'bias'. The resulting vector is processed by an 'activation function' (Eq. 1.8):

$$h_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{w}^T \cdot \mathbf{x}) = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{1.8}$$

where $\mathbf{x} = [x_1, x_2, ..., x_N, 1]$ is the input vector, $\mathbf{w} = [w_1, w_2, ..., w_N, b]$ is the vector containing the weights and the bias b for this particular neuron. There are a variety of possibilities for the activation function f, and the best one depends on the data to be fitted. Some common choices are the sigmoid function (Eq. 1.9) and the hyperbolic tangent (Eq. 1.10). [66]

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.9}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.10}$$

The neurons are organised in layers, as can be seen in Fig. 1.2. The first and the last layer are called input and output layer respectively, while the middle ones are referred to as 'hidden layers'. Each neuron has a vector of weights associated with it. Consequently, each weight is referred to by 3 indices: one for the layer in which the neuron is located, one to specify which neuron in that layer and one to specify which weight in the vector weights for that particular neuron. More explicitly, one can denote the weight that connects neuron j in layer l to neuron i in layer $l + 1$ with $w_{ij}^{(l)}$. An example is shown in Fig. 1.2, where the weights connecting layer 2 and 3 are explicitly shown. On the other hand, the biases only need 2 indices: one for the layer and one to identify the neuron in the layer. Consequently, the bias for unit i in layer l would be denoted with $b_i^{(l)}$.

The size of the input layer is fixed, which means that all data samples need to have the same dimension. In 'feed forward neural networks', each neuron in a layer takes as input the output of all the neurons in the previous layer.

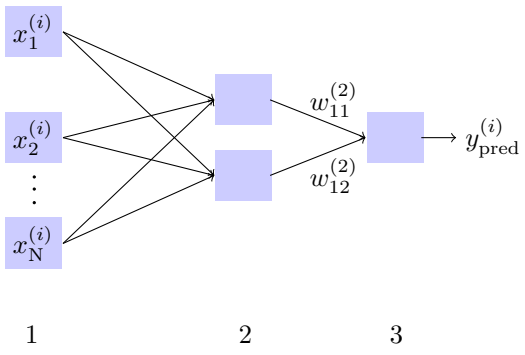


Figure 1.2: Diagram of a neural network with input layer on the left, one hidden layer in the middle and the output layer on the right. Layers are also numbered from 1 to 3 and the weights between layer 2 and 3 are shown explicitly.

The neural network 'learns' the mapping between input and output data by optimising the weights

and bias of all neurons. To quantify the similarity between the neural network prediction $y^{(i)}_{\text{pred}}$ and the reference data $y^{(i)}$, a 'cost function' is introduced. A common choice of cost function is the 'squared error cost function'. For a single data sample the cost function $J(\mathbf{w})$ is:

$$J(\mathbf{w}) = \frac{1}{2}\left(y^{(i)}_{\text{pred}} - y^{(i)}\right)^2 \tag{1.11}$$

Where $y^{(i)}$ is the reference value and $y^{(i)}_{\text{pred}}$ is the prediction of the neural network. In order to 'train' the neural network, one needs to minimise the the average cost function for all the points in the training set with respect to the weights and biases in each neuron. This will give the values of the parameters that minimise the error of the neural network predictions.

$$\min_{\mathbf{w}} \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} J(\mathbf{w}, x^{(i)}, y^{(i)}) \tag{1.12}$$

where N_{samples} is the number of samples in the training set. Neural networks can learn very complicated relationships between inputs and their outputs. However, if too many features are present in the data, the model may become overly specific to the training data and not generalise well to samples that it has never seen. This is generally referred to as over-fitting. [67] To alleviate this problem, one can add a 'regularisation' term to the cost function. This is a penalty that is added to the weights, so that one has a degree of control over their magnitude. This means that all the features in the data are kept, but each of them contributes a small amount. One type of regularisation is called L2 regularisation and the 'penalty' term is the sum of the squared magnitude of the weights: [68]

$$J_{\text{Reg}}(\mathbf{w}) = \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} J(\mathbf{w}; x^{(i)}, y^{(i)}) + \frac{1}{2}\lambda \sum_{ij}\sum_{l} (w^{(l)}_{ij})^2 \tag{1.13}$$

The first term of equation 1.13 is an average of the cost over all the samples, the second term is a sum of all the squared weights, where the sum index l is over the layers in the network, i is over the neurons and j is over all the weights for a neuron. λ is the regularisation parameter that will control the magnitude of the penalty on the weights. A high value of λ will make the features contribute a small amount and if it is too large it will cause under-fitting of the data. Another commonly used type of regularisation is L1 regularisation, where instead of the square of the weights, one uses the absolute value. [69]

The process of 'training' is done with an algorithm called 'back-propagation'. [70] At the beginning of training, the network is initialised with small random weights. Then, the input data is propagated

forward through the network and the output value is compared with the target value, giving a value for the cost function. This output value is then differentiated with respect to all the weights and biases in the network. Once the gradient with respect to each weight w and bias b is obtained, these are adjusted as follows:

$$w = w - \alpha * \frac{\partial J}{\partial w} \tag{1.14}$$

$$b = b - \alpha * \frac{\partial J}{\partial b} \tag{1.15}$$

Where weights w and biases b after the update are the modified by a term including α, which is the learning rate, and $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$, which are the gradients of the cost function with respect to the weights and biases respectively. [71, 72] The learning rate is a parameter that adjusts how much the weights and biases are modified at each time step.

To speed up training, during each iteration the training set is divided into batches. The average cost function J is calculated for the first batch and its gradient with respect to the weight and the biases are obtained. These are then used to update the weights and the biases and then the process is repeated on the next batch. Consequently, if a data set is divided into M batches, during each training iteration the weights and the biases will be updated M times.

This has the advantage that the model will be updated more quickly and usually will reach convergence faster. It also avoids needing to have all data in memory, which makes for a more efficient algorithm. The disadvantage is that there is an additional hyper-parameter that has to be picked (the batch size). [73]

1.3.1 Atomic neural networks

A common type of networks used to fit potential energy surfaces is called 'atomic neural network'. [14, 74] These were introduced by Behler and Parinello to overcome the issues associated with feed forward neural networks when fitting potential energy surfaces. [75]

One of the problems is related to the symmetry of the neural network. If two atoms with the same chemical identity are swapped, the atomic configuration should still have the same potential energy, since the chemical structure has not changed. [14] This property can be taught to the network by providing more training data where the positions of identical atoms are swapped. In this way the network

can learn that multiple different representations of a molecule map to the same energy. However, this is impractical as it requires huge amounts of data for any chemically interesting system.

Another issue is transferability to systems with different size (i.e. different number of atoms) compared to that in the training set. Since the input layer of a feed forward neural network has a fixed size, the input vectors cannot change dimensions. More explicitly, once a feed forward neural network has been trained on input data represented by vectors of length N, it cannot be used to predict the properties of a molecule represented by a vector of length M.

Atomic neural networks overcome these issues. [14] The idea is to decompose the potential energy E_{tot} into atomic energy contributions (E_i):

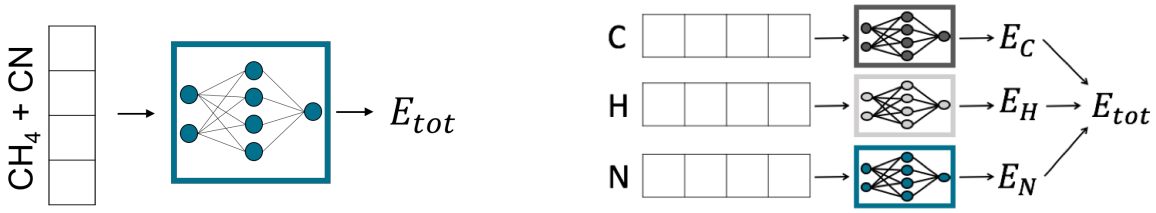
$$E_{\text{tot}} = \sum_{i=1}^{N_{\text{atoms}}} E_i \tag{1.16}$$

The atomic energies depend on the local environment around a particular atom and each one will be predicted by a feed forward neural network. The overall neural network is now composed of many feed forward networks, generally one for each type of chemical element present in the system. Each one of them takes as input a fixed-size vector describing the local environment around a certain atom. All the atomic energies are then summed together to give the total energy of the molecule E_{tot}. The cost function remains identical to that in feed forward neural networks (eq. 1.13), i.e. one does not need to have atomic energies in the training set, just total energies.

An example of an atomic neural network is shown schematically in Fig. 1.3b for a system that contains only carbon, hydrogen and nitrogen atoms. With this architecture, it does not matter if the order of the atoms is swapped in the initial configuration file, as all the atomic contributions are summed together. This ensures permutation invariance. Furthermore, such neural networks can be trained on and predict systems of different sizes. The transferability of atomic neural networks will be the main theme of chapter 4.

1.4 Representing molecules

Cartesian coordinates are a simple and unambiguous representation of the atomic positions in a molecule. However, they are not suitable to make comparisons between structures, because the Cartesian coordinates of two systems can be different even if the two systems can be mapped onto each other by a rotation/translation, [76] i.e. they are not rotation and translation invariant.



(a) Feed Forward Neural Network

(b) Atomic Neural Network

Figure 1.3: Diagrams to compare the architecture of a feed forward neural network (Fig. 1.3a) and an atomic neural network (Fig. 1.3b). (a) A configuration is represented by one global representation and the network learns the energy of the whole system. (b) The input vectors that describe the local environment around a certain atom are shown on the left. In the middle are the feed forward neural networks for each atom type. On the right, the atomic energies that come out of each network are summed to give the total energy of the system.

Consequently, other representations are used when fitting potential energy surfaces. A representation should not only be rotation and translation invariant, but also computationally inexpensive to calculate from the Cartesian coordinates. Furthermore, if the forces acting on all atoms in a system are needed in addition to the energy, then the representation should be differentiable with respect to atomic positions. [77]

One of the first representations used for neural networks was a vector of reaction coordinates values. For example, in the system where a CO molecule reacts with a Nickel surface, the representation would be a vector containing the lateral position x of CO along a line between two sites on the surface, and the angle θ of the molecular axis relative to the surface normal. [78] However, this is not easily generalizable and has the problem that different representations, such as $\{x, \theta\}$ and $\{x, \theta + 2\pi\}$, correspond to the same configurations. [79]

The Z matrix (also referred to as 'internal coordinates') is a rotation invariant representation that is often used to describe the geometry of entire molecules. It is based on bond lengths, bond angles and dihedral angles. This ensures translation and rotation invariance, but it is not invariant to the permutation of the atoms. [80]

Another possibility is to use the distance matrix M, i.e. an $N \times N$ matrix (where N is the number of atoms in the system) where each matrix element m_{ij} is the Euclidean distance between atom i and atom j. The inverse distance matrix is more commonly used (each matrix element is the inverse distance between two atoms), because if the distance between two atoms i and j approaches infinity, then the matrix element m_{ij} approaches zero. This is intuitively more useful for chemistry applications, as one expects the interaction between atoms to be less important if these are far apart. A representation based on the inverse distance matrix is the Coulomb matrix. [81] The Coulomb matrix

is very commonly used because of its simplicity and it will be discussed further in section 1.4.1.

More sophisticated representations include the atom centred symmetry functions (ACSFs), which describe the local environment around each atom in the system. These are possibly the most commonly used representations in the recent literature [3, 26, 13] and will be discussed in detail in section 1.4.2. A reproach that is often made to ACSFs is that they require fine tuning of internal parameters. A similar representation to ACSFs is the Spectrum of London and Axilrod-Teller-Muto (SLATM), which will be discussed in section 1.4.3. SLATM has been recently used in the literature, but mostly for kernel ridge regression models. [82]

Broadly speaking, there are two classes of molecular representations that are used with neural networks: 'global representations', used with standard feed forward neural networks and 'local representations', used with atomic neural networks. In global representations, one vector encodes the information of the full configuration of a system. The Coulomb matrix, described in section 1.4.1 is an example of a global descriptor. In local representations, for each atom in the system there is a vector encoding information about the local environment of that atom. An example of local representation is the Atom Centred Symmetry Functions (ACSFs). They are discussed in depth in section 1.4.2. The SLATM representation, which has both a local and global version, will be briefly discussed at the end of this section.

1.4.1 The Coulomb matrix

The Coulomb matrix was introduced by Rupp et al. as a molecular representation for predicting the atomisation energies of molecules with Kernel Ridge Regression (KRR). [83] This representation is based only on atomic positions and nuclear charges and is defined as:

$$M_{ij} = \begin{cases} 0.5\, Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i\, Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \text{for } i \neq j \end{cases}$$

where Z_i is the nuclear charge on atom i and $\mathbf{R}_i$ is its Cartesian coordinate.

The idea of Rupp et al. was to treat molecules as fully connected undirected graphs, where the nodes are the atoms and the edges are weighted by the Coulombic interaction between the atom pairs. This is a first approximation to the structure of a molecule. [84]

The diagonal elements of the Coulomb matrix are designed to effectively encode the atomic identity of each atom. [81] The expression $0.5 Z_i^{2.4}$ is an approximation of a free atom energy. [74] The off-diagonal

elements represent the Coulomb repulsion between the atom pairs ij.

The Coulomb matrix needs to be flattened to a vector to be input into the neural network. Since it is a symmetric matrix, the number of unique matrix elements (and therefore the length of the input vector) is $N(N+1)/2$, where N is the number of atoms in the system. The size of this vector scales quadratically with the number of atoms, making this representation impractical for large systems.

Since the Coulomb matrix for a molecular system encodes information about the whole system, it is referred to as a 'global representation'. Global representations are used with feed forward neural networks to predict molecular properties. This is in contrast with representations such as 'Atom Centred Symmetry functions' (ACSFs) which are 'local representations' and are used with atomic neural networks. ACSFs are discussed in section 1.4.2.

With the exception of enantiomers, distinct molecules (and distinct conformation of the same molecule) will have distinct Coulomb matrices, i.e. the Coulomb matrix is a unique representation.

One problem with the Coulomb matrix representation is that a molecule can generate different Coulomb matrices depending on the order of the atoms, i.e. it is not permutation invariant. To avoid this problem, Hansen et al. suggested working with the sorted eigen spectrum, i.e. the sorted set of eigenvalues. [27] The eigen spectrum is invariant upon atom permutations, but it contains only a subset of the information of the whole Coulomb matrix. [74] Hansen et al. also suggested another way of solving this problem. They introduced the 'sorted Coulomb matrix', where the rows and the columns of the Coulomb Matrix are sorted in descending order of their norm. [27] A final way of dealing with the problem of atom indexing that they introduced is to use the 'random Coulomb matrix'. Here, for each data point in the data set, multiple random Coulomb matrices are randomly sorted. [27] While the random Coulomb matrix was reported to improve the results for the prediction of atomisation energies using KRR, [27] this increases the sizes of the data sets used and considerably slows down training.

1.4.2 Atom Centred Symmetry Functions

As mentioned earlier, ACSFs are a 'local representation', which means that each atom in the molecule is represented by one fixed-size vector. The ACSFs representation of an atom is composed by a two-body and three-body term. The two-body term has the following form [85]:

$$G_i^2 = \sum_{\substack{j \neq i \\ Z_j = Z}} e^{-\eta(R_{ij} - R_s)^2} \, f_c(R_{ij}) \tag{1.17}$$

where G_i^2 is the two-body term for atom i, the sum over j runs over all the neighbouring atoms of i of a certain element type Z, η and R_s are parameters that need to be picked, R_{ij} is the distance between atom i and j and $f_c(R_{ij})$ is the cut-off function:

$$f_c(R_{ij}) = \begin{cases} 0.5 \times \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{if } R_{ij} \leq R_c \\ 0 & \text{if } R_{ij} > R_c \end{cases} \tag{1.18}$$

where R_c is the cut-off radius. The cut-off is shown in Fig. 1.4.

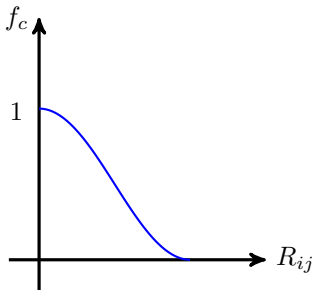


Figure 1.4: The form of the cut off function (f_c) used in Atom Centred Symmetry functions, where R_{ij} is the distance between atom i and j.

The two-body term of the ACSFs is constructed by making a vector of all the G_i^2 for neighbours of different atom types. The process is repeated for different values of R_s and the vectors obtained are all concatenated together. The shape of the two-body terms is shown in Fig. 1.5 for different values of R_s.

The three-body term of the ACSFs can have a variety of functional forms. [14] In this work, the expression in equation 1.19 will be used, and a plot of what the three-body term looks like as a function of different angle values is shown in Fig. 1.6.

$$G_i^3 = 2^{1-\zeta} \sum_{\substack{j \neq i, j \neq k \\ Z_j = Z_1 \\ Z_k = Z_2}} (1 + \cos(\theta_{ijk} - \theta_s))^\zeta e^{-\eta\left(\frac{R_{ij}+R_{ik}}{2} - R_s\right)^2} f_c(R_{ij}) f_c(R_{ik}) \tag{1.19}$$

where ζ, θ_s, η, R_s are parameters that need to be picked, θ_{ijk} is the angle between the atoms i, j and k. For each atom, the three-body term is generated by obtaining the values of G_i^3 for different atom-pair neighbours. Then, this procedure is repeated using different values of R_s and θ_s and the vectors obtained are concatenated. The sum is over the neighbouring atom pairs of same atom type.

This formulation of the symmetry function was suggested by Smith et al. and was shown to give

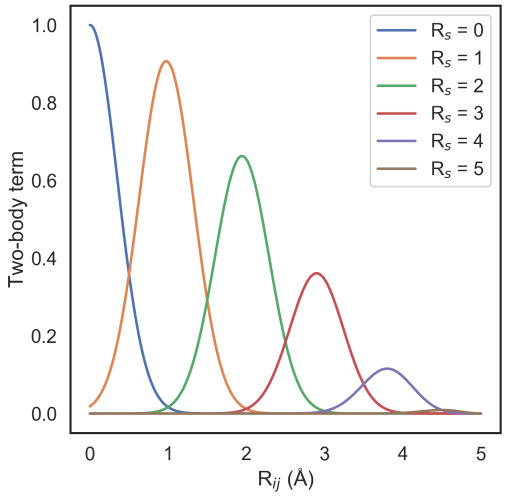


Figure 1.5: Two-body term of the ACSF using $\eta = 4$, $R_s = [0, 1, 2, 3, 4, 5]$. R_{ij} is the distance between two atoms i and j.

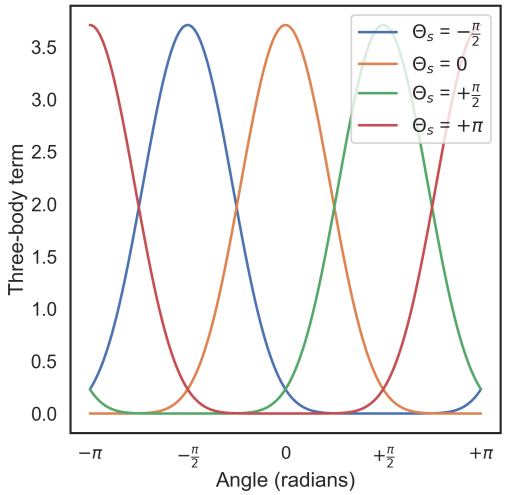


Figure 1.6: Three-body term of the ACSF using $\eta = 4$, $\zeta = 4$, $R_s = 1$ and $\theta_s = [-\pi/2, 0, \pi/2, \pi]$. A fixed value of $R_{ij} = R_{ik} = 1.5$ were used.

good results for learning the potential energy of organic molecules containing the atoms H, C, N, and O. [13, 26] The additional θ_s parameter, which was not present in previous formulations, [85] makes them better suited to probe the angular environment around each atom compared to the original formulation. [26]

The parameters η and ζ effectively control the width of the radial and angular functions respectively, while R_s and θ_s control the position of their centres. If the values of η and ζ are too large, the Gaussian functions will be too narrow and will not overlap enough. On the other hand, if the values of η and ζ are too small, the Gaussians will be too large and will overlap too much.

Once the two- and three-body terms are obtained, the two are concatenated to give the ACSFs for each atom. This representation has both rotational and translational invariance, because the vectors $\mathbf{G}_i$ only depend on the distance and angles between the atoms. Permutation invariance is ensured by the architecture of the atomic neural network, as discussed in section 1.3.1. [85] Since it can be difficult to understand how to construct ACSFs in practice, an example for a toy system is presented in Appendix A.

1.4.3 SLATM

The Spectrum of London and Axilrod-Teller-Muto (SLATM) descriptor [82] has both a local and global version. The local version, referred to as 'atomic SLATM', is constructed in a similar way to

the ACSFs, but the two and three-body terms have different functional forms. Unfortunately, there are still ambiguities about the formulation of these functions as the publication that introduced them only partly revealed their formulation. The two-body term is reported to be: [82]

$$S_2 = \frac{1}{2}\sum_{j\neq i} Z_j \frac{1}{\sigma\sqrt{2\pi}} e^{-(R-R_{ij})^2} g(R) \tag{1.20}$$

where $g(R)$ is a distance dependent scaling function, Z_j is the nuclear charge of atom j, σ is a parameter. It is not clearly stated what R and R_{ij} are, therefore it is assumed that R is the distance between atom i and j and R_{ij} is a parameter analogous to R_s in the ACSFs. $g(R)$ is also not explicitly defined. The three-body term is reported to be:

$$S_3 = \frac{1}{3}\sum_{j\neq k\neq i} Z_j Z_k \frac{e^{-(\theta-\theta_{ijk})^2}}{\sigma\sqrt{2\pi}} \frac{1+\cos\theta\cos\theta_{jki}\cos\theta_{kij}}{(R_{ij}R_{ik}R_{jk})^3} \tag{1.21}$$

where θ is the angle between atoms i, j and k, θ_{jki} and θ_{kij} are functions of θ, but their form is not reported. Similarly, R_{kj} is a function that depends on the angle between atoms i, j and k and R_{ij}, R_{ik}, but the form of this function is not reported. The SLATM descriptor can be made into a global descriptor too, but it is not clear how to construct it. The ambiguities related to the SLATM formulation make it a non-ideal candidate for fitting potential energy surfaces in the long run. However, SLATM had the advantage that a Fortran implementation with Python interface was available. This means that it could be used as a black box to generate representations that could be used to test our implementation of a neural network framework. For this reason, in this work SLATM was only used as a temporary representation while an ACSFs implementation was being developed.

1.5 Fitting potential energy surfaces with neural networks

Using neural networks to fit potential energy surfaces is not a new idea. One of the first potential energy surfaces that was fitted with a neural network involved a CO molecule chemisorbed on a Nickel(111) surface and was published in 1995. [78] However, in recent years the molecular sciences have seen a surge in popularity of neural networks for a variety of applications, from designing new drug molecules [86] to planning synthetic chemistry strategies. [87] In particular, multiple research groups have been applying neural networks to the prediction of molecular energies and forces with the goal of accelerating molecular dynamics (MD) simulations.

In 2007, Behler and Parinello introduced atomic neural networks and Atom Centred Symmetry Func-

tions (ACSFs). [75] They used them to fit the potential energy surface of bulk silicon, and showed how simulations using the fitted potential could accurately reproduce the DFT radial distribution of melted silicon at 3000 K. In 2012, Artrith and Behler used atomic neural networks with ACSFs to study copper surfaces. [88] Their method was implemented in a closed source software called RuNNer. [89] The combination of atomic neural networks with ACSFs started becoming more popular after the paper by Behler titled 'Constructing High-Dimensional Neural Network Potentials: A Tutorial Review'. [14] In the years following this publication, multiple groups started working on their own implementation of these neural network models. [90, 91, 92]

In 2017, Smith et al. used atomic neural networks with atom centred symmetry functions to fit quantum mechanical DFT calculations in order to learn accurate and transferable potentials for organic molecules. [26] The data set used to train their neural network was obtained using Normal Mode Sampling. This method samples a set of data points around a structure that represents an energy minimum of the potential energy surface. This structure is modified by applying random displacements to the atoms along the normal modes coordinates. All the new structures with energy below a certain threshold are kept. These neural networks were trained on a dataset called 'ANI-1' containing around 17.2 million geometries of organic molecules with up to 8 heavy atoms. Then, they were able to predict the energies of molecules with up to 53 atoms. [26] The Roitberg group developed a Python package based on PyTorch [93] in order to obtain the results described by Smith et al. The first release on Github of this package, called 'TorchANI', happened in April 2019. [90, 91, 92]

Kun et al. developed another architecture of neural networks which could not only predict energies, but also forces and dipole moments. This architecture has two components. It has an atomic neural network (with the atom centred symmetry functions as the molecular representation) to calculate one term of the energy. The differentiation of this term yields the forces. Then, there is a second neural network which calculates dipole moments. The dipole moments are used to calculate an additional term of the energy that takes into account the long range electrostatic interactions not taken into account in the other energy term. In order to train their model, the authors used a two step procedure. They trained the network that predicts the dipole moments first. Then, they trained the energy network using the contribution from the already trained dipole network (which is kept frozen). For training this neural network, Kun et al. generated a data set containing 15000 organic molecules in different geometries. They used metadynamics to obtain a total of 3 million geometries. They reported predicting the energy of molecules outside of the dataset within chemical accuracy. [13] The Parkhill group has an open repository [91] with the code used to generate the results in the publication by Kun et al.

In 2017, Schütt et al. introduced an architecture of neural networks called 'deep tensor neural network'.

[94] Deep tensor neural networks are based on the same principle of atomic neural networks, where a property is obtained by calculating the atomic contribution from each atom (eq. 1.16). The difference from previous work is that this architecture takes as input a vector of nuclear charges and a distance matrix. The network then learns a local representation from the data. The representation is then used to predict the desired property. The following year, Schütt et al. published a new paper about their implementation of an improved version of deep tensor neural networks which they called 'SchNet'. In SchNet, convolutions are used in the part of the network that learns the representations of the molecules. This implementation was made open source and available under the name 'SchNetPack'. [92, 95] Schütt et al. tested the prediction of potential energy surfaces and force fields for a collection of 8 small organic molecules. [92] They compared their method to a potential energy surface fitted with Kernel Ridge Regression [62] and their own deep tensor neural networks. SchNet outperformed both, with energies and forces errors of $0.50\,\mathrm{kJ\,mol^{-1}}$ and $1.38\,\mathrm{kJ\,mol^{-1}\,\AA^{-1}}$.

Multiple reactive potential energy surfaces for small systems have been also been constructed over the years.

For example, in 2006 Lorenz et al. created a neural network potential energy surface for the dissociation of diatomic molecules on the clean and the sulfur covered Pd(100) surfaces. [96]

In 2009, Pukrittayakamee et al. introduced the combined function derivative approximation (CFDA), which is a different way of training neural networks compared to what had previously been done. The novelty of this method is that the model is trained on both the energy and the forces of each atom in the system. In this way, the derivative of the neural network will match the gradient of the potential energy surface. They used the reaction of H + HBr as an example of their method. [97] Another example of reactive surfaces is that of Le et al., who interpolated MP2 calculations for the reaction of $BeH + H_2 \longrightarrow BeH_2 + H$. [25] They showed that it is possible to reach errors around $0.4\,\mathrm{kJ\,mol^{-1}}$ for the energies and $2.5\,\mathrm{kJ\,mol^{-1}\,\AA^{-1}}$ for the forces.

1.6 Cyano radical reacting with hydrocarbons

To the best of my knowledge, neural networks have not been applied to fit the potential energy surface of reactive systems larger than those mentioned in the previous section (section 1.5). [97, 25] Consequently, one of the aims of this thesis is to attempt to fit the potential energy surface of larger reactive systems.

This project attempts to build on what has been learnt from small systems to fit the potential energy surface of the reaction of squalane ($C_{30}H_{62}$) with a cyano radical (Fig. 1.7). The cyano radical is

known to perform a hydrogen abstraction from the hydrocarbon to form HCN.

Hydrogen abstractions by CN radicals have been extensively studied both computationally and experimentally at the University of Bristol. [98, 24, 59] The first studies of the reactive dynamics of CN with polyatomic organic species involved propane [24] and cyclohexane. [59] The reactions were studied both in the gas phase and in solution, with CH_2Cl_2 as the solvent. Ab-initio dynamic simulations were performed by first obtaining a potential energy surface for the system under study. The dynamics simulations showed that the HCN produced by the hydrogen abstraction is vibrationally excited along the CH stretching and the HCN bending coordinates. This non-equilibrium energy distribution may persist for hundreds of picoseconds. [24] Later experimental studies showed that relaxation to the ground-state HCN mostly results from vibrational relaxation via coupling to the solvent on a slower time scale. [59] These studies showed the importance of the dynamics simulations using fitted potential energy surfaces for the mechanistic study of reactions. In light of this success, similar methods were used to study the reaction of CN with tetrahydrofuran (THF). [98]

Studies of hydrogen abstractions by cyano radicals are now also being studied in collaboration with the group of Dr M. L. Costen at the Heriot-Watt University. They are focusing on the reaction between surfaces of squalane ($C_{30}H_{62}$) and cyano radicals to shed light on processes happening at gas-liquid interfaces. Interactions between gas molecules and liquids are important in a wide variety of biological, atmospheric and industrial processes. [99] The reactions between hydrocarbons and CN have been extensively studied in the gas phase [100, 101] and in the liquid phase, [58] but not at the gas-liquid interface. Consequently, this represents an interesting system to understand the mechanism of radical reactions at gas-liquid interfaces.

In order to aid the study of the CN + squalane reaction, Dr Glowacki's group began studying these reactions computationally. Since obtaining high quality reference data for a system as large as a surface of squalane is computationally challenging and expensive, it was decided to investigate alternative approaches. Since neural networks have been shown to give transferable potential energy surfaces for small organic molecules, it was decided to investigate whether this is possible also for reactive radical systems. This would facilitate fitting a potential energy surface for squalane reacting with CN, as only reference data for smaller systems would have to be gathered.

The goal of this part of the thesis is to investigate whether atomic neural networks can learn to predict the energy of squalane reacting with CN radicals when being trained only using smaller hydrocarbons. In order to do this, numerous steps had to be taken. First of all, a software framework had to be developed to be able to train neural networks on systems of different sizes. While now there are multiple software frameworks available to perform this sort of potential energy surface fitting, at the

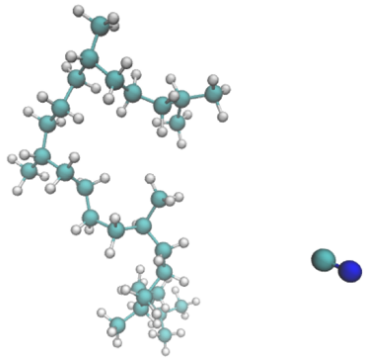


Figure 1.7: Ball and stick representation of a squalane molecule and a cyano radical.

beginning of this work none was available. Consequently, atomic neural networks and atom centred symmetry functions were implemented and tested first on methane and isopentane reacting with CN. The results for these test systems are shown in chapters 2 and 3 respectively.

Then, ways of generating the training data for the neural networks had to be chosen. Data has to be generated for all studied species with samples covering the relevant parts of configurational space as uniformly as possible. Then, the energies of the geometries sampled should be refined as much as practical with accurate electronic structure methods. This aspect is present in all chapters 2-4.

Finally, it was important to understand what is the minimum size of the hydrocarbons that should be included in the data set in order to still be able to accurately predict the energies of squalane reacting with CN. This is discussed in chapter 4.

Chapter 2

Reaction of cyano radical and methane

The first system studied in this work is methane reacting with a cyano radical. Methane is the smallest hydrocarbon that can be investigated and is therefore a good starting point for exploring different aspects of fitting potential energy surfaces with neural networks. Specifically, this chapter will discuss:

- The generation of a training set with the aid of Interactive Molecular Dynamics in Virtual Reality (iMD-VR).
- Technical details of the neural network models and the frameworks used.
- The performance of various global and local molecular representations in combination with feed forward and and atomic neural networks.

2.1 Method

2.1.1 Generating the data set

The data set was generated using the recently developed Narupa framework [102] for interactive molecular dynamics. This multi-user VR-enabled interactive molecular dynamics framework combines rigorous real-time atomistic physics simulations with VR hardware. [3] It has recently been made open-sourced and is available on Gitlab. [103]

In iMD-VR, users can interact in real-time with molecular dynamics simulations in a virtual reality environment. Users are able to reach into the simulation and apply an external force on individual atoms using VR controllers, effectively steering the MD simulations (Fig. 2.1). This framework enables rapid and intuitive sampling of configurations of molecular systems. [3]

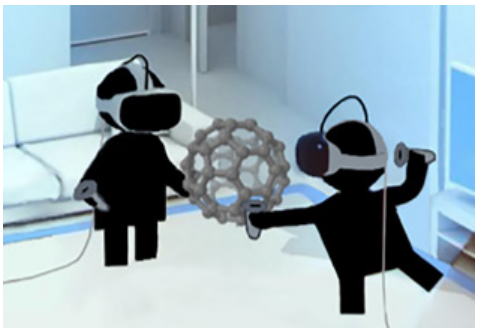


Figure 2.1: Two users in iMD-VR interacting with a molecular system using the Narupa framework. [102]

Trajectories for methane reacting with the cyano radical were thus generated by manipulating the cyano radical and methane into close proximity, so that they could react. Once the hydrogen was transferred from the methane molecule to the cyano radical, the two products were then brought closer again so that the reverse reaction could happen. This process was repeated numerous times in order to thoroughly sample the pathways.

The iMD-VR was run using the Velocity Verlet integrator [104] with a time step of 0.5 fs and the Berendsen thermostat [105] was used to maintain the temperature at 300 K (with a collision frequency of $10\,\mathrm{ps}^{-1}$). The engine used to calculate the energies and the forces was DFTB+ with the mio parameter set. [106] For the interaction of the user with the atoms, a spring potential with a force constant of $1000\,\mathrm{kJ\,mol}^{-1}\,\mathrm{Da}^{-1}$ was used. After each time a user interacts with an atom, a velocity re-initialisation procedure rapidly re-equilibrates the system between interactions with the user. [102] This removes the momentum of the atoms introduced by the users when manipulating them. This procedure is described in detail in the paper by O'Connor et al. [102]

The iMD-VR trajectories of methane reacting with the cyano radical were then pruned by removing high-energy configurations (energy greater than $300\,\mathrm{kJ\,mol}^{-1}$ than the lowest energy configuration in the data set), which left 47733 data points. These high energy structures occurred when users applied too much force to the atoms in the simulation, resulting in atoms getting too close to each other or the molecule being atomised. The energy of the remaining structures was recalculated at a higher level of theory than DFTB. The method used was DFT with the PBE (GGA functional) [107] with the minimal basis set STO-3G [36]. This level of theory was chosen because it is very fast (table 2.1),

but is an improvement in terms of accuracy compared to DFTB. After the refinement of the energies, the data set was pruned again to reduce its size further: only the structures with lowest energies were kept, giving a data set with 20698 samples. The energies of these structures were then re-calculated with DFT, but at a higher level of theory: B3LYP (hybrid functional) with the augmented correlation-consistent basis set aug-cc-pVTZ [108]. For the final round of pruning, a subset of 17756 structures that had the lowest energy was selected. The energy of these configurations was recalculated at CCSD(T)-F12b [109] level of theory using the aug-cc-pVTZ basis set. All the electronic structure calculations in this chapter were performed by Dr Simon Bennie. The average timings of the calculations performed with the different methods are reported in table 2.1.

Table 2.1: Average timings of electronic structure calculations of methane reacting with CN with different methods.

Method	**Average time** (s)
DFT (PBE/STO-3G)	10
DFT (B3LYP/aug-cc-pVTZ)	200
CCSD(T)-F12b	10000

The obtained data set was divided into a training set, a test set and a test trajectory containing 120 data points.

The test trajectory of the cyano radical abstracting a hydrogen from methane was kept separate from the other test set and the data points were kept in order (Fig. 2.2), so as to be able to visualise the neural network prediction on a full trajectory. The trajectory is not a minimum-energy abstraction trajectory (Fig. 2.2), so there are oscillations in the energies both for the reactants and the products. The transition between the reactant and the product is very clear as the reaction energy appears to be about $100\,\mathrm{kJ\,mol^{-1}}$. In Fig. 2.2, the energies are relative to an arbitrary reference value of $-349\,716.6\,\mathrm{kJ\,mol^{-1}}$. This energy was selected by taking a random reactant geometry from the data set.

Of the remaining data points, 10% were removed and kept for testing (1764 data points). These were just randomly picked and did not constitute full trajectories.

This left 15872 data samples for training.

2.1.2 Software details

Due to the wide range of fields in which neural networks-based algorithms have been applied, several generic and highly optimized libraries for building and training them are available. Prominent examples

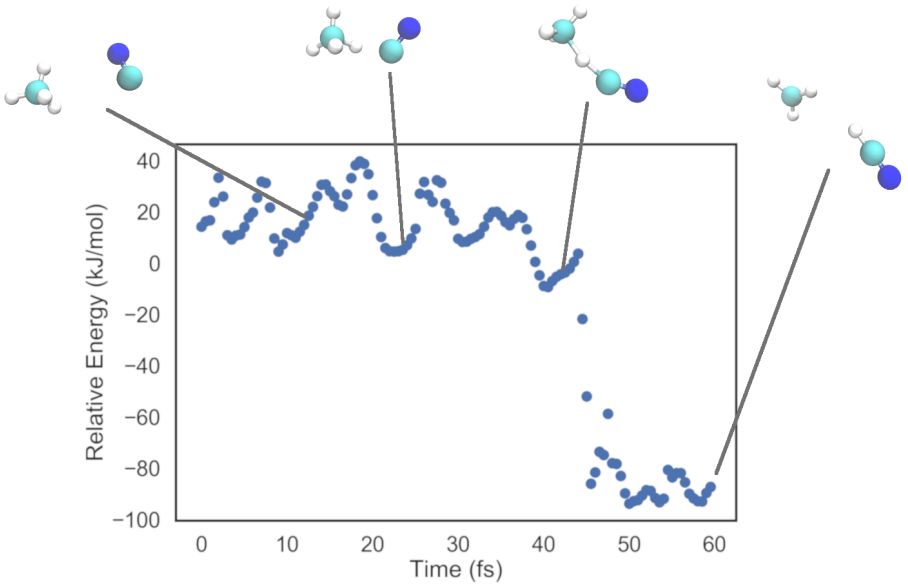


Figure 2.2: Trajectory of a cyano radical abstracting a hydrogen from methane. The energies are relative to an arbitrary value of $-349\,716.6\,\mathrm{kJ\,mol^{-1}}$.

are TensorFlow, [110] developed by Google, and PyTorch, [93] developed by Facebook. In order to use these neural network libraries for a specific application domain (in this case, fitting potential energy surfaces), it is generally necessary to build a software framework for handling the domain specific data and for interfacing to the library itself. As of late 2019, a variety of such software frameworks for applications in chemistry are available. For example, there are Schnet, [92] TensorMol, [13] ANI, [26] AMP [111] and RuNNer. [85] However, most of these have been released in the last two years and were either not readily available [85] or did not have Graphics Processing Unit (GPU) support when this project was started. GPUs help speed up the training because they enable to perform the same operations on multiple pieces of data in parallel. CPUs cannot offer the same extent of parallelisation. Consequently, an in-house software framework for training neural network in the context of molecular modelling was developed. It took me over a year to implement this code, write the documentation and the tests to make it usable and reliable. Dr Lars Bratholm improved the interface of the code to make it more user-friendly. All code developed in this project was later merged with the already existing open-source QML package and is now available on Github. [112]

The neural network component of QML uses the TensorFlow library [110] to build and train the neural networks. TensorFlow turns algorithms into graphs of connected operations that can be executed on GPUs. Each operation in the graph gets assigned to a computational device and is executed asynchronously and in parallel once all the tensors and their data become available. Automatic differentiation is also available, i.e. once the computational graph is specified, the program can calculate

the derivatives automatically. This avoids having to implement manually the procedure of calculating the gradients of the cost function with respect to the weights of the neural network, which is required to train the networks (this procedure is called back-propagation).

The developed framework includes implementations of both feed forward neural networks (using global molecular representation) and atomic neural networks (using local molecular representation). When implementing the two neural network architectures, the Scikit-learn interface was used as a reference. [113] The code was developed in a modular manner to facilitate future extension and maintenance. In addition, in collaboration with Dr Lars Bratholm we developed a wrapper to make the neural networks compatible with Osprey, a package for automating hyper-parameter optimisation. [114]

The models used in this chapter require only the molecular representations and the energies as the training data, i.e. no forces were used for training.

When training neural networks, there are other parameters in addition to the weights and biases whose value affect the performance of the network, but they are not optimised during back-propagation. These parameters are referred to as 'hyper-parameters' and they include, among others, the number of hidden layers, the number of neurons in each layer, the regularisation parameters, the batch size and the learning rate of the optimisation algorithm, the number of training iterations, etc. The procedure to obtain these parameters is described in section 2.1.4.

2.1.3 Representing molecules

Initially, only the Coulomb matrix was used as the representation for this project, since it is very easy to implement. However, other descriptors like the Atom Centred Symmetry Functions (ACSFs) and Spectrum of London and Axilrod-Teller-Muto (SLATM) were shown in the literature to give good results. [26, 13, 85] When this project was started, there was no available implementation of the ACSFs, so the global and local SLATM were used while the ACSFs implementation was being developed. This is because the atomic SLATM should be similar to the ACSFs. In the long run it is preferable to use ACSF due to the larger amount of documentation available on it compared to SLATM. In this section, the performance of the Coulomb matrix, SLATM, atomic SLATM and ACSFs is compared.

The Coulomb matrix and the ACSFs were implemented in TensorFlow, while the SLATM and atomic SLATM were taken from the Quantum Machine Learning (QML) Python package, where they are implemented in Fortran. [112] Having the representations implemented in TensorFlow means that the gradients of the representation with respect to the Cartesian coordinates can be obtained without extra effort thanks to the automatic differentiation. This is useful if the forces have to be calculated

as well as the energies. However, since in this chapter the focus is only on obtaining the energies and not their gradients, the fact that not all representations were implemented in TensorFlow was not a problem.

The Coulomb matrix and SLATM were used with the feed forward neural network, while atomic SLATM and ACSFs were used with the atomic neural network. The Coulomb matrix for the system with methane and cyano radical (7 atoms) has 28 features, while SLATM and ACSFs have a variable number of features depending on the parameters chosen. The formulation of the ACSFs was the one described by Smith et al. [26]

Another advantage of the TensorFlow implementation of the ACSF and the Coulomb matrix is that some of the operations involved in their calculation can be performed on the GPU. For example, Fig 2.3 shows the computational graph for the ACSFs, where all the operations are grouped together based on what is being calculated. These groups include calculating the two-body term (radial part) and the three-body term (angular part) and then summing together the terms from atoms of same atom type. The parts of the graph coloured in green are executed on a GPU, while those in blue are on the CPU.

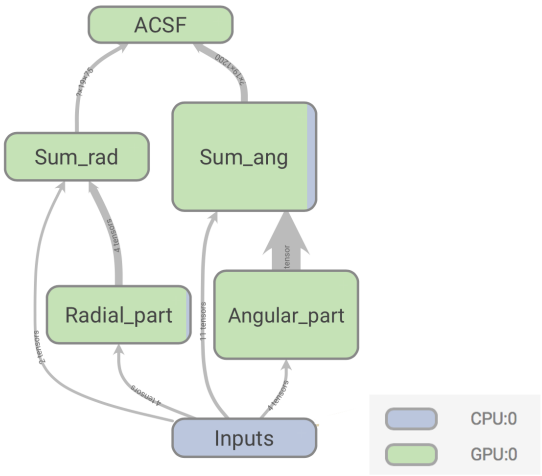


Figure 2.3: Visualisation of the TensorFlow computational graph for the ACSFs. The parts of the graphs in green are executed on a GPU, while the blue parts on a CPU. 'Radial_part' is the calculation of the two-body terms and 'Sum_rad' is the sum of the two-body terms corresponding to atoms of same type. 'Angular_part' and 'Sum_ang' are the equivalent for the three-body terms. The 'ACSF' operation concatenates the two- and three-body terms

2.1.4 Hyper-parameter optimisation

The hyper-parameters that were optimised for the models used in this chapter were: the learning rate, the batch size, the number of neurons in the first and second hidden layers, the L1 and L2 regularisation parameters and the number of iterations. The number of layers was limited to 2, as increasing to 3 did not show improved performance. All the hyper-parameters were optimised with a random search implemented in the package Osprey. This is done by specifying a range for each hyper-parameter that needs to be optimised and then a random value in this range will be picked for each parameter.

Random search was used for the hyper-parameter optimisation. Random search has been shown to be more effective for finding better hyper-parameters compared to manual and grid search. [115]

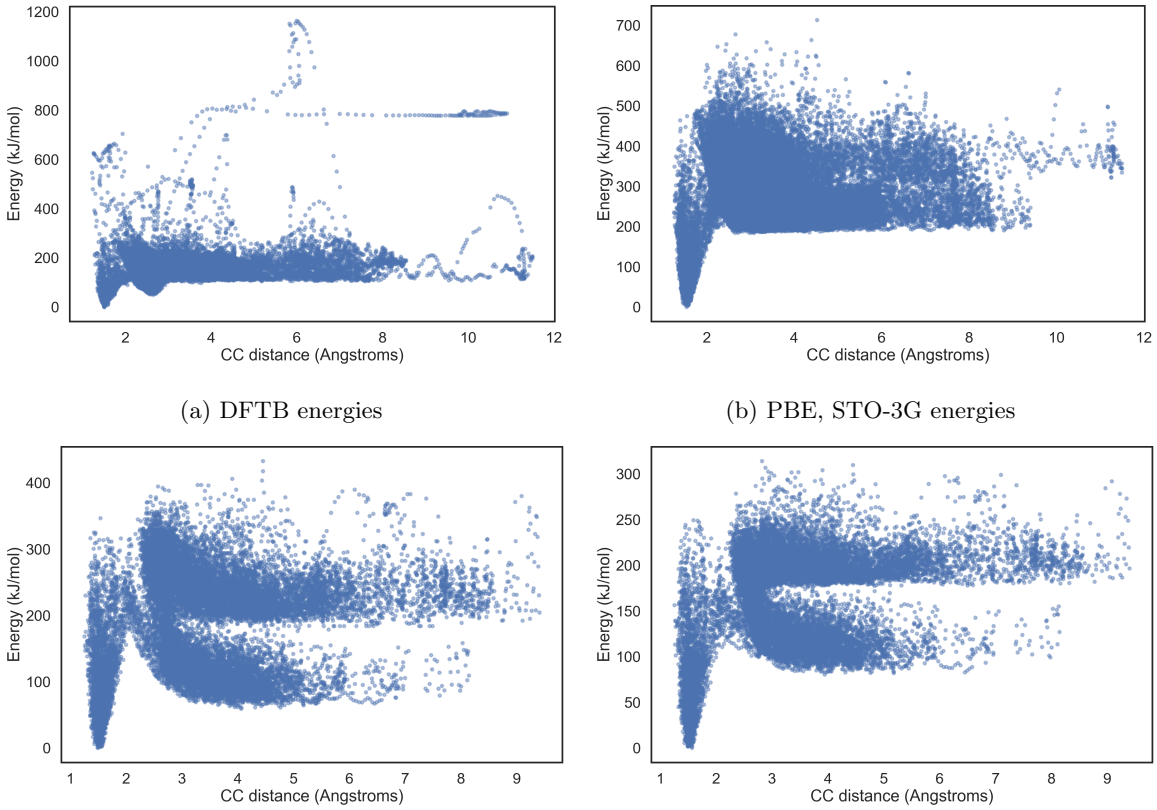
The hyper-parameters were chosen by training using 3-fold cross validation. During 3-fold cross validation, the training set is divided into 3 subsets, each containing a third of data points. The network is trained on two folds (i.e. two thirds of the data, which corresponds to 10581 data points in this case) of the data at a time and tested on the remaining fold (5291 data points). This process is repeated three times alternating the training and test folds. The scores obtained on each of the folds are then averaged to give the final score for that particular set of hyper-parameters. [116]

The values of the best scoring hyper-parameters are reported in Appendix C.

2.2 Results and discussion

2.2.1 Data set

The raw data sampled with iMD-VR is shown in Fig. 2.4a. There are two features to notice from this figure. The first one is the presence of high-energy structures. These are usually generated when the user applies too much force to an atom and causes it to dissociate or to get too close to another atom. The second feature is the presence of two energy minima: one at C-C distance of 1.8 Å and a second one at C-C distance of 2.5 Å. The minimum at higher carbon-carbon distance corresponds to structures where the cyano radical is in close proximity to a methane hydrogen, while the minimum at lower C-C distance corresponds to structures where the HCN carbon is bonded to the methyl radical carbon (structure on the top left of Fig. 2.5). Fig. 2.4c shows that when the energies are refined further, there appear to be two parts to the potential energy surface. The structures were analysed to understand what these two parts represent. In the following discussion, the geometry with the lowest energy in the data set was taken as the reference. With respect to this reference, the structures with B3LYP



(a) DFTB energies

(b) PBE, STO-3G energies

(c) B3LYP, aug-cc-pVTZ energies

(d) CCSD(T)-F12b, aug-cc-pVTZ energies

Figure 2.4: Comparison of the raw data obtained from iMD-VR (a) for the $CH_4 + CN$ trajectories, and the data after the first round of pruning (b), after the second round of pruning (c) and after the third round of pruning (d). Energies have been scaled relative to the lowest energy structure present in the data set.

energy larger than $200\,\mathrm{kJ\,mol^{-1}}$ and carbon-carbon distance larger than $3\,\text{Å}$ are the reactants: the methane and the cyano radical. On the other hand, all the structures with energy below $200\,\mathrm{kJ\,mol^{-1}}$ and carbon-carbon distance larger than $3\,\text{Å}$ are the products: HCN and the methyl radical. The structures with energy below $200\,\mathrm{kJ\,mol^{-1}}$ and Carbon-Carbon distance smaller than $2\,\text{Å}$ appear to be structures where the carbon of HCN and methyl radical are bonded together and the radical is now on the nitrogen.

The data also includes some hydrogen abstractions by the nitrogen of the cyano radical, rather than by the carbon. In iMD-VR, this product was produced quite easily. However, most studies of CN reacting with saturated hydrocarbons do not report observing the formation of this product. [117, 118, 119] The trajectories with abstractions from the nitrogen atom were kept to see how they affected the learning of the neural networks.

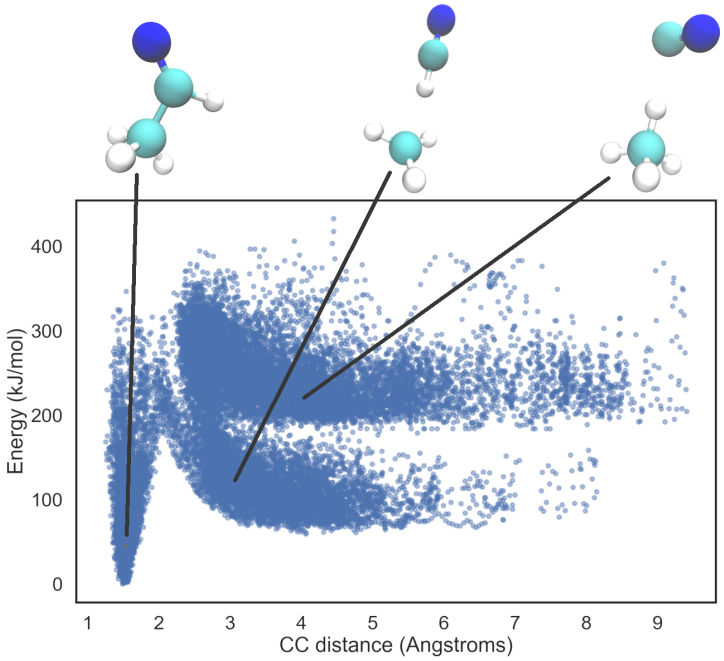


Figure 2.5: Visualisation of the different regions of the B3LYP potential energy surface. The three most common structures present in the data set are shown and they are connected to the region of the potential energy surface to which they correspond.

2.2.2 Neural network models

The hyper-parameters of the neural network were optimised using the procedure described in the Methods section. The optimisation process was carried out using each of the following descriptors: the Coulomb matrix, SLATM, atomic SLATM and the ACSFs, so four sets of hyper-parameters were obtained. The hyper-parameters of the representations themselves, such as η and R_s in equation 1.17 for ACSFs, were kept to the default values in the QML package for SLATM, atomic SLATM and ACSFs. The cross-validation mean absolute errors (MAEs) using each descriptor are reported in Table 2.2. As can be seen from Table 2.2, it appears that the ACSFs give the best result after 3-fold cross validation, with the two SLATM descriptors following closely.

The set of hyper-parameters giving the lowest MAE for each molecular representation was selected and then used to train the models on 15872 data points from the training set. The models obtained were then used to predict the energies of the 1764 data points in the test set and of the abstraction trajectory. The error for the cross-validation MAE is the standard deviation between the three MAEs from each fold of the data. For the test set and for the trajectory, the MAE is the standard deviation of the errors obtained for all data points and are shown in Table 2.2.

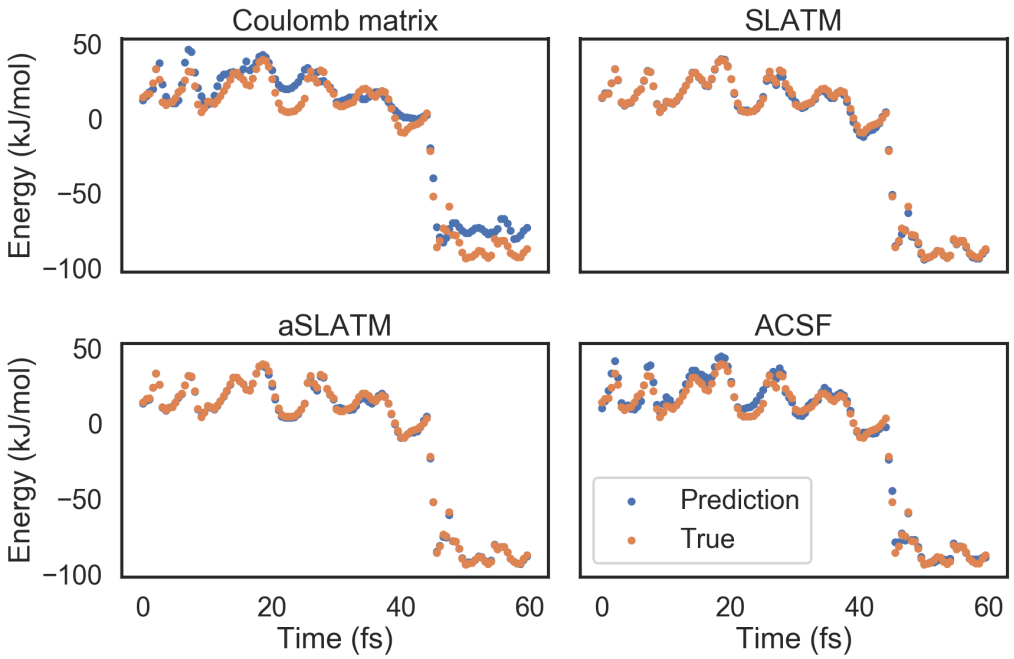


Figure 2.6: Plots showing the energies obtained with coupled cluster and the energies predicted by the neural network with the Coulomb matrix, the SLATM, aSLATM and ACSF as the representations (unoptimised hyper-parameters). The energies are relative to an arbitrary value of −133.2 Ha. The results of ACSFs with optimised hyper-parameters is shown in Fig. 2.7.

Table 2.2: Mean Absolute Error (MAE) from the cross-validation (CV) and the test set and the single H-abstraction trajectory. All are given in $\mathrm{kJ\,mol^{-1}}$. Two values are given for the ACSFs: 1. with hyper-parameters not optimised, 2. with optimised hyper-parameters.

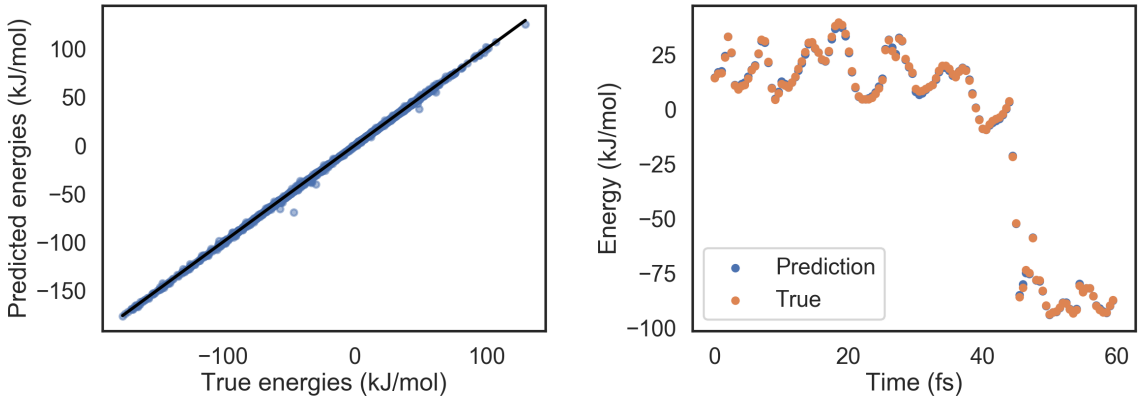
Representation	**CV**	**test set**	**trajectory**
Coulomb matrix	4.85 ± 0.25	9.42 ± 12.8	7.83 ± 7.35
SLATM	1.92 ± 0.05	1.52 ± 2.85	0.89 ± 1.15
Atomic SLATM	2.07 ± 0.20	1.48 ± 2.78	0.72 ± 0.98
ACSF 1	0.97 ± 0.04	4.56 ± 6.82	3.12 ± 3.42
ACSF 2	0.92 ± 0.11	0.61 ± 1.02	0.52 ± 0.65

As can be seen from Fig. 2.6 and Table 2.2, the Coulomb Matrix gives the worst results out of all the descriptors, but these results are still qualitatively good considering its simplicity. The model can still predict a decrease in energy between the reactants and the products, but the details of the potential energy surface related to the molecules vibrating and moving before and after the reaction are not well reproduced. This is probably due to the fact that the Coulomb matrix takes into account only 2-body interactions, while all the other descriptors take into account also the 3-body interactions. For both the Coulomb matrix and the ACSFs, the errors obtained during the cross-validation procedure and for the test set are quite different. This could be due to the fact that during cross-validation the models are trained on 10581 data points, and then the best scoring hyper-parameters are used to train a model on the full training set (15872 data points). For both the Coulomb matrix and the ACSFs, it seems that the highest scoring hyper-parameters are better suited for the smaller training set and do not transfer well to the larger one.

Table 2.3: Timings for the generation of different representations of 17756 data points from the CH_4 + CN data set.

Representation	**Generation time** (s)
Coulomb matrix	1
SLATM	13
Atomic SLATM	63
ACSF	7

The two SLATM descriptors give less than $2\,\mathrm{kJ\,mol^{-1}}$ errors, and reproduce almost perfectly all the features of the hydrogen abstraction trajectory (Fig. 2.6). The main issue with this descriptor is the time taken to generate it and the fact that its form is not clearly stated in the literature. Table 2.3 shows how much longer it takes to generate the atomic SLATM descriptor compared to ACSFs. This means that it would be impractical to use it for training on larger systems. Finally, the ACSFs give much better results compared to the Coulomb matrix, but not as good as the SLATM descriptors. This is most likely due to the fact that the default hyper-parameters in the SLATM representation



(a) Test set predictions

(b) Trajectory predictions

Figure 2.7: Predictions of the NN trained using the ACSF as the molecular representation, where also the hyper-parameters of the ACSFs have been optimised. The R^2 for the correlation plot is 0.9996.

happened to be better compared to those in the ACSF. However, even without optimising the ACSFs hyper-parameters, the error on the test set is of $4.56 \pm 6.82\,\mathrm{kJ\,mol^{-1}}$.

In order to see whether the results improve, the hyper-parameters were re-optimised for the model trained with the ACSFs. This time, the hyper-parameters of the ACSFs were also optimised. These include η, ζ, R_s and θ_s (equations 1.20 and 1.21). The accuracy of the predictions improved, with a MAE of $0.61 \pm 1.02\,\mathrm{kJ\,mol^{-1}}$ for the test set and $0.52 \pm 0.65\,\mathrm{kJ\,mol^{-1}}$ for the test trajectory (Fig. 2.7, table 2.2). Consequently, ACSFs appear to be the best option as the molecular representation for the next chapters.

2.3 Conclusion

In conclusion, this chapter investigated multiple aspects of fitting a potential energy surface with NNs. First of all, a new way of generating the data set was used. This involved the iMD-VR framework Narupa, recently developed in the Glowacki research group. This was the first time that a quantum mechanical simulation was steered using virtual reality to generate the data set for a neural network. The structures sampled in iMD-VR included the reactant, the transition state and the products of the reaction of $CN + CH_4$. However, it was realised that the DFTB+ force engine was not the best choice for this system, as it over stabilises the transition state of the hydrogen abstraction (as can be seen from Fig. 2.4a, where the energy minimum at CC distance of 2.5 Å corresponds to structures where the cyano radical is in close proximity to a methane hydrogen). The transition state appears to be lower in energy than the products of the reaction, which is not the case with more accurate electronic-

structure methods. Consequently, in the next chapters the PM6 force engine will be investigated instead. Secondly, feed forward NNs with global descriptors and atomic NNs with local descriptors were used for fitting a potential energy surface and their performance was analysed. It appeared that the Coulomb matrix could not give a satisfactory descriptions of the molecular configurations and hence the fitting was of lower quality compared to the other descriptors. However, while NNs with both the global and local SLATM gave excellent fits of the potential energy surface, the time required to generate the molecular representations makes SLATM unusable for larger systems. The ACSFs appear to be the best option currently available. Once their hyper-parameters are optimised, NNs with the ACSFs can give errors as low as $0.61 \pm 1.02\,\mathrm{kJ\,mol^{-1}}$.

Chapter 3

Reaction of cyano radical and isopentane

In this chapter, the system under study is an isopentane molecule reacting with a cyano radical (Fig. 3.1). Isopentane is the smallest hydrocarbon with primary, secondary and tertiary hydrogens. In the literature, as was discussed in section 1.5, NNs have been used to fit potential energy surfaces of reactive radical systems with fewer than 5 atoms. [97, 25, 96, 5, 120, 121] Consequently, isopentane and CN represent the largest reactive radical systems whose potential energy surface has been fitted using NNs. [3]

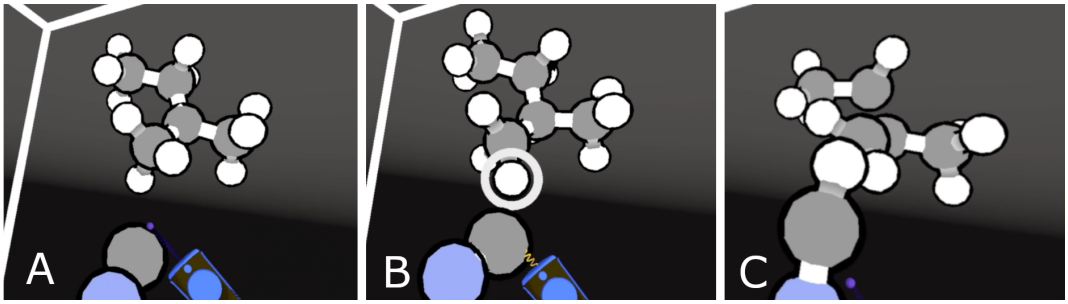


Figure 3.1: Reaction of isopentane and cyano radical sampled in iMD-VR. The reactants (A) are brought in close proximity (B) to form the products (C). A movie of the abstraction can be seen online. [122]

In order to assess and analyse the quality of data sets generated using iMD-VR, in this chapter the performance of two atomic neural networks trained on different data sets is compared. One data set was generated using iMD-VR, and the other using the more traditional constrained molecular dynamics (CMD) method.

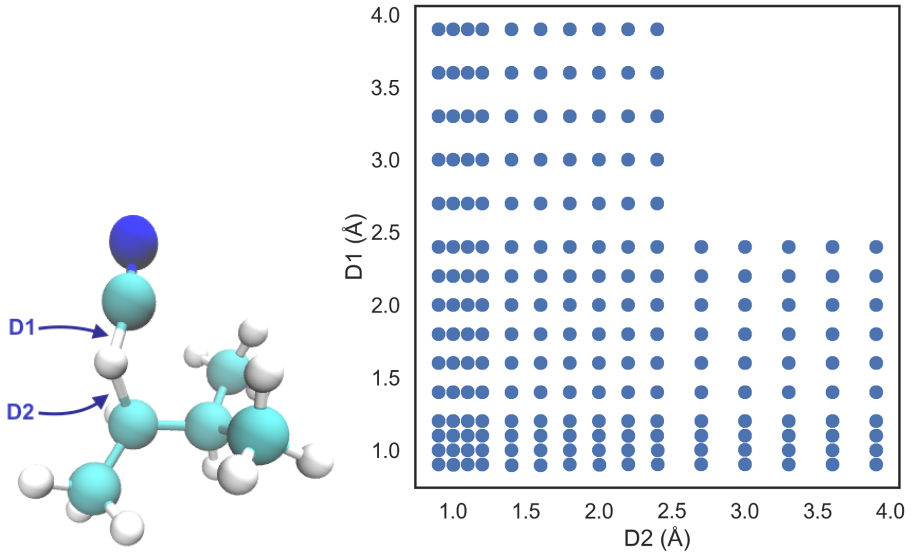
3.1 Method

3.1.1 Generating the data sets

Two data sets were generated by sampling the hydrogen abstractions from different isopentane sites. The first data set was obtained using interactive molecular dynamics in virtual reality (iMD-VR). Here, the sampling of the reaction between the cyano radical and isopentane was performed by loading a starting structure of each of the reactants in XYZ format into the Narupa environment. They were spawned in random (non-overlapping) positions within a cubic box with length 30 Å. In order to sample hydrogen abstractions, the reactants were brought in proximity to each other by the user, to enable the reaction to take place (Fig. 3.1). Once the products were formed they were pulled away from each other. After each reaction, the system was re-initialised to a random configuration of the reactants. It was found that separating each trajectory in this way made the data processing considerably easier, because it was possible to keep track of which trajectory each configuration came from.

The molecular dynamics simulations were run using a Velocity Verlet integrator with a time step of 0.5 fs. The Andersen thermostat was used to maintain the system temperature at 300 K, with a collision frequency of $10\,\mathrm{ps}^{-1}$. The system was constrained to stay within the box via velocity inversion. For the interaction of the user with the atoms, a spring potential with a force constant of $1000\,\mathrm{kJ\,mol}^{-1}\,\mathrm{Da}^{-1}$ was used. A velocity re-initialization procedure was used to rapidly re-equilibrate the system between interactions with the user. Unlike in chapter 2, the semi-empirical method PM6 [123] was used to evaluate energies and forces in the molecular dynamics simulation. It was decided to switch from DFTB+ (with the mio parameter set) to PM6 because, as described in the previous chapter, DFTB+ over-stabilised the transition state of the hydrogen abstraction. The implementation of PM6 is from the SCINE sparrow package, developed by Reiher and co-workers. [124, 125] This package includes implementations of tight-binding engines like DFTB alongside a suite of other semi-empirical methods.

The second data set was generated by Dr Lars Bratholm using constrained molecular dynamics (CMD). This data set was generated in order to compare iMD-VR to a more conventional enhanced sampling method. He chose the CH distance on the isopentane and on the cyano radical as the degrees of freedom to constrain. The molecular dynamic simulations were then run with each of the 12 isopentane hydrogens constrained in turn. The constrained simulations were performed using PM6 in the CP2K package. [126] The simulations were run in the NVT ensemble at 300 K with the CSVR thermostat, [127] using a 20 Å simulation box. The time step was 1 fs and a total of 5000 steps were carried out, with a structure being saved to an XYZ file every 500 steps. The values of the constraints which were



(a) Constrained degrees of freedom.

(b) Values of the constraints

Figure 3.2: Constrains used to generate the data set with constrained molecular dynamics.

used are shown in Fig. 3.2. The spacing between the constrained distances is not uniform: smaller spacing was used near the equilibrium distances of the reactant and product (CH distances between 0.9 Å and 1.2 Å) in order to get higher resolution along the minimum energy path. Larger spacing was used for larger CH distances. [3]

To refine the PM6 energies of the structures sampled with both iMD-VR and CMD, electronic structure calculations using MOLPRO [128] were performed. The Coulomb fitted [129] unrestricted PBE [107] functional with the Def2-TZVP[130] basis set were used. In the rest of the report this method is referred to as CF-uPBE/TZVP. This method was chosen because Dr Lars Bratholm calculated the reaction enthalpy for propane on a primary and secondary hydrogen. He evaluated the energy of the reactants and product structures optimised with CF-uPBE0/SVP. [131] With these he calculated the reaction enthalpies ($-101.9\,\mathrm{kJ\,mol^{-1}}$ and $-117.4\,\mathrm{kJ\,mol^{-1}}$ for primary and secondary hydrogen abstractions respectively) and he found these to reproduce well the experimental values. [132] The experimental values are $-108.8\,\mathrm{kJ\,mol^{-1}}$ and $-121.3\,\mathrm{kJ\,mol^{-1}}$ for the primary and secondary hydrogen abstractions respectively. This lower level of theory seemed adequate enough for a first test of fitting potential energy surfaces of a 19 atom system using atomic neural networks. Generating a data set at the CCSD(T)-F12b woud have been considerably more time consuming and unnecessary at this point.

3.1.2 Implementation details

In this chapter, only atomic neural networks with ACSFs were used to fit the potential energy surface of the isopentane molecule reacting with the cyano radical. The formulation of the ACSFs used was that described by Smith et al. [26] The number of correlated hyper-parameters used was reduced. This was done by following the procedure described in Appendix D. [3] QML was used again, but additional ways of generating the representations were implemented and the methods in the neural network class were modified. These modifications are described below.

Implementation of neural networks in QML

Each network has a '`generate_representation`' method. Once the data is input into the model, this method can be called to transform the data from Cartesian coordinates and element types to the representation of choice. Depending on whether a feed forward neural network or an atomic neural network are used, a choice of global and local representations are available. Once the representation has been generated, it is stored and then used for fitting. TensorFlow has an in-built class to deal with processing of large data sets (`tf.data.Dataset`). An object from the `tf.data.Dataset` class can be instantiated and the Cartesian coordinates and the nuclear charges can be stored in it. Then, the two alternatives for generating the representations were implemented:

1. *Calculating the representations in batches*: a dataset of 3000 data points was split into batches of size 1, 10, 100, 500. Then, an object of the `tf.data.Iterator` class was instantiated. This has a method `get_next` that enables to iterate over the data set one batch at a time. The representations were generated for each batch and then they were concatenated together to give the full descriptor.

2. *Calculating the representations one at a time*: the same data set of 3000 data points was used. Here, the method `map` from the `tf.data.Dataset` class is used. This method applies a user defined function to all the samples in the data set. Therefore, the `map` function is used to generate the representation for each molecule.

After the representations are generated, they are used to train models. The fitting is done through the '`fit`' method. This method constructs the TensorFlow computational graph. The operations involved are multiplications and additions between the parameters and the input data, as well as non-linear functions such as the sigmoid function or the hyperbolic tangent.

Once the model has been fitted to the data, it can be used to predict the properties of new molecules. This can be done using the method '`predict`'. This method takes some Cartesian coordinates and

transforms them into the molecular representation that was used when fitting the model. Then, the molecular representations are input into the neural network and the predicted properties are obtained.

After training, the fitted model can be saved for later re-use. When saving a TensorFlow model, two binary files are generated. One binary file needs to contain the structure of the graph, i.e. the operations that are present. The other binary file contains any saved variables. These are, for example, the values of the weights and biases after training the neural network. The values of the variables have to be saved separately because they are not part of the graph, but they are stored in the TensorFlow session object.

3.1.3 Hyper-parameter optimisation

The hyper-parameter optimisation procedure in this chapter was performed by Dr Lars Bratholm. The method used is more sophisticated than that in the previous chapter. This procedure involves fitting a Gaussian process [133] to the MAE of the NN as a function of the hyper-parameters and using it to select which hyper-parameters to explore. This is a more efficient way to explore hyper-parameter space compared to the random search, as it is meant to find better hyper-parameters in a smaller amount of time. It is however more complex to set up compared to a random search.

The way the optimisation is done is explained below: [3]

1. First, 10 iterations of random search are performed. Then, a Gaussian process is fit to the MAE obtained in these first 10 iterations with respect to the hyper-parameters. The Gaussian process has a variance associated with it which indicates how confidently it can predict the MAE as a function of hyper-parameters.

2. Then, a new set of hyper-parameters to explore is picked. The choice is a balance between regions of hyper-parameter space where there is large uncertainty and regions that are predicted to give low MAEs. The Gaussian process is then used to pick a value of the hyper-parameters which minimises the log(MAE) minus one standard deviation. This enables to explore regions that give low errors, but where there is uncertainty as well. Once enough hyper-parameter combinations have been explored, the 'best scoring hyper-parameters' are chosen by minimising the log(MAE) *plus* a standard deviation. In this way, regions of the hyper-parameter surface which give good hyper-parameters with high degree of confidence are selected.

The Osprey [114] implementation of the Gaussian process [133] was used. Separate optimisations were performed for the data set generated with iMD-VR and CMD. The hyper parameters optmised

were: the learning rate, the batch size, the number of neurons in the first, second and third hidden layer, the L1 and L2 regularisation parameters and the number of iterations, as well as the ACSF hyper-parameters.

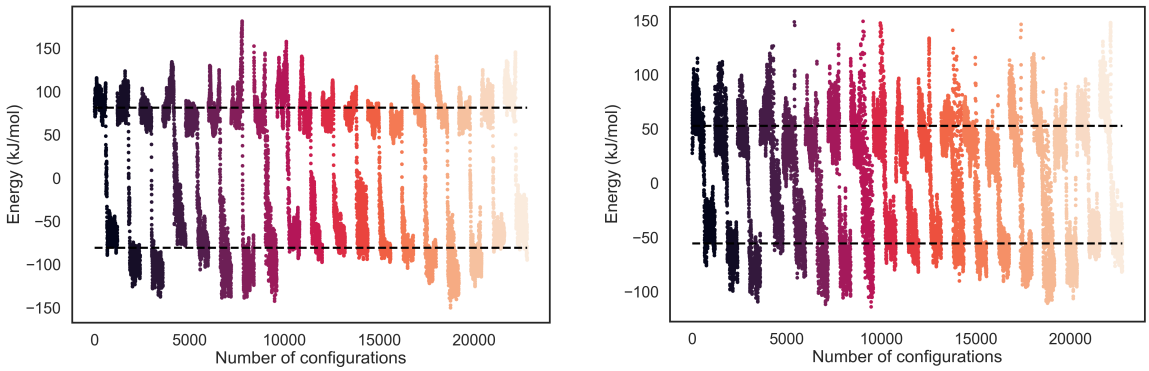
In addition, the Gaussian process optimisation was combined with a variation of the standard K-fold cross-validation procedure. Since the data used to train the model consists of molecular dynamics trajectories, each configuration is highly correlated with the one from the next time step. Consequently, if data points from one trajectory are split across the training and test set, this can give too optimistic error values for the model. This can be avoided by never splitting data from a particular trajectory across the training and the test set. Therefore, during cross-validation, configurations from the same trajectory always end up being part of the same fold of data. This then gives more realistic predictions of the MAE for a particular model.

3.2 Results and discussion

3.2.1 Data sets

Fig. 3.3a shows the pruned trajectories obtained with iMD-VR. There are in total 19 trajectories of the cyano radical abstracting a hydrogen from isopentane. Initially, around 25 trajectories were generated with iMD-VR which took approximately 1 h. Then, any trajectory corresponding to HNC formation, breaking of the isopentane molecule in smaller fragments or formation of H_2 were removed. The remaining trajectories included 11 primary abstractions, 3 secondary and 5 tertiary. These trajectories were then further pruned by keeping 600 configurations before and after each trajectory reaches a reference energy which is approximately half way between that of the stable reactants and products (-290.175 Ha). The energies of the remaining configurations were recalculated with CF-uPBE/TZVP. The reference energy was then subtracted and any structure with energy larger than 150 kJ mol^{-1} was removed. Overall, this left 22756 data points.

For the CMD data set, often trajectories lead to the molecules breaking apart due to the constraints. For example, several simulations resulted in a non-constrained hydrogen being abstracted or formation of H_2. These trajectories were removed from the data set. All structures with energy larger than 150 kJ mol^{-1} above the same reference energy used for the iMD-VR data set were also removed. After pruning the unwanted structures only 7621 structures were left out of the possible 24000 (10 snapshots for 12 hydrogens and 200 constraints). The energy of these structures was then refined at the CF-uPBE/TZVP level.



(a) PM6 (b) CF-uPBE/TZVP

Figure 3.3: Trajectories obtained in VR of the CN radical reacting with isopentane. The configurations from each trajectory are coloured differently. The dotted lines indicate the average energy of the first and the last 400 frames of each trajectory, corresponding to the average energy of the reactants and products, respectively.

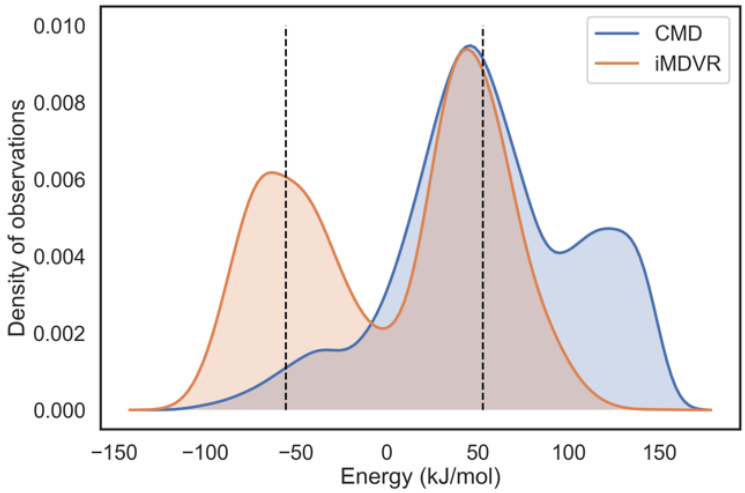


Figure 3.4: Kernel density estimate of the DFT energies of the configurations sampled using the iMD-VR approach (orange) and the constrained MD approach (blue). The dotted lines show the average energy of the reactants and the products as shown in Fig. 3.3.

Fig. 3.4 shows a kernel density estimate [134] for how frequently configurations with a certain energy were sampled at specific energies using the iMD-VR and CMD. The kernel density estimate is constructed by placing a Gaussian on each observation and then summing all the Gaussians to obtain a smooth curve representin how often each value occurs. The orange curve shows that iMD-VR samples the equilibrium structures more often than the transition regions as there are two peaks clearly corresponding to the reactant and the product energies. The peak corresponding to the product energies

is slightly wider than the reactant peak. This is because the energy of the products resulting from primary, secondary and tertiary abstractions are all slightly different, while the reactant energies are only the energies from the isopentane molecule and a cyano radical. On the other hand, the sampling of CMD is considerably different from that of iMD-VR. As can be seen from the blue curve (Fig. 3.4), the CMD data set contains a more significant fraction of higher-energy configurations, with values between $100\,\mathrm{kJ\,mol^{-1}}$ and $150\,\mathrm{kJ\,mol^{-1}}$. While the iMD-VR structures are usually close to minimum energy path, the CMD data set samples structures in the vicinity of the transition state.

3.2.2 Learning Curves

A learning curve is a plot of the MAE as a function of the number of training data points. [135] Here, a learning curve was done for both the iMD-VR and the CMD data set. For the iMD-VR data set, one trajectory for a primary hydrogen abstraction was removed from the data set, so that it could be used as a test of the models performance. Of the remaining 21563 data points of the iMD-VR data set, random subsets of 100, 300, 1000, 3000 and 10000 data points were selected. For each subset, 3-fold cross validation was performed, meaning that a neural network was trained on 2/3 of the data and then tested on 1/3 of the data and the process was repeated 3 times. This means that the neural network is trained on 67, 200, 667, 2000, 6667 and 14375 data points for each of the subsets (Fig. 3.5). The hyper-parameters used are those optimised for the subset with 14375 data points. This is due to the fact that optimising hyper-parameters is a lengthy process and can take weeks. Each network that was trained on a different fold of each data set was tested on the primary abstraction trajectory that was kept separate. The MAE was averaged between the 3 folds of each data set. The resulting learning curve is shown in Fig. 3.5. The learning curve shows that past 6000 data points there is no large improvement of the neural network performance.

The CMD data set only contained 7621 data points, so the learning curve could not be run as far as for the iMD-VR data set. A set of configurations where a primary hydrogen is being abstracted were kept separate to test the performance of the models. Consequently, only 6939 data points were used to train the models for the learning curve. The hyper-parameters used were those optimised for the data set with 6939 data points. As can be seen from Fig. 3.5, for the same number of data points, the neural network does not reach values of the errors as good as for the iMD-VR data set. This is most likely due to the presence of a more significant fraction of higher-energy configurations in the CMD data set. The high energy regions of the potential energy surface vary more quickly with respect to coordinates. [136] Consequently, more data would be required to fit these regions to the same level of accuracy as the lower energy ones.

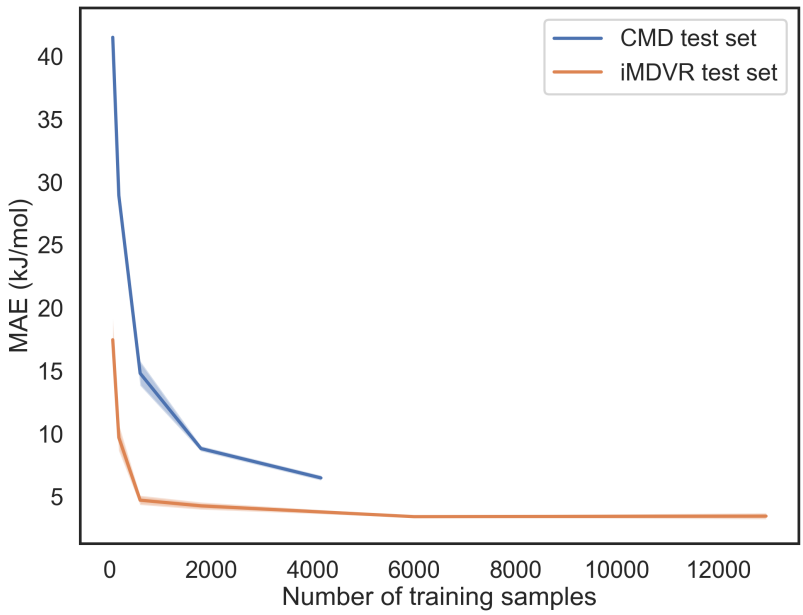


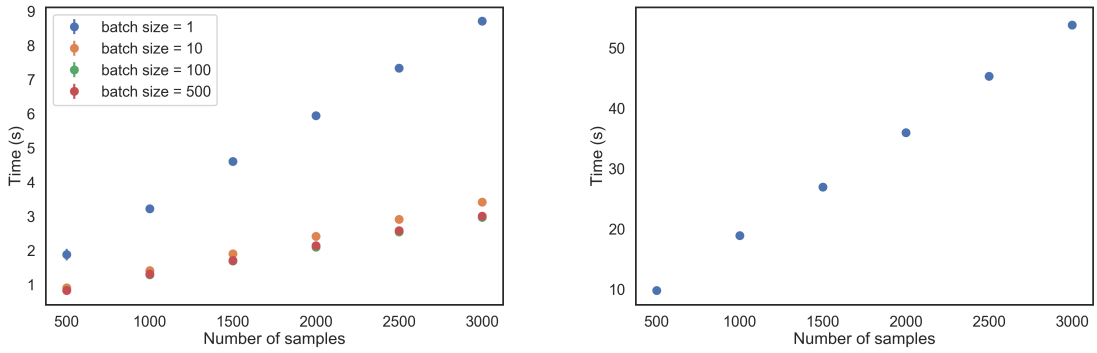
Figure 3.5: Learning curve for the neural networks trained on increasing number of samples from the iMD-VR (orange) and CMD (blue) data sets and tested on a primary abstraction trajectory. The shaded area shows the standard deviations of the MAEs obtained for each fold of the data.

3.2.3 Performance comparison of ACSFs implementations

A comparison of the speed of generating the ACSF representation with two different TensorFlow implementations (described in section 3.1.2) is presented here.

This analysis was performed by generating the ACSF representations of 500, 1000, 1500, 2000, 2500 and 3000 data points from the isopentane + CN data set. The ACSFs generated had 513 features. The wall-clock time taken to calculate the ACSF was measured. Each measurement was repeated 5 times to evaluate the error on the measurement.

As can be seen from Fig. 3.6, using the TensorFlow `map` method appears to give slower results compared to the implementation that uses the `get_next` method from the `tf.data.Iterator` class. This is the case even if the batches are of size 1, which is the case where one would expect the two methods to be equivalent. The reason why generating the ACSFs with a batch size of 1 is faster than generating the representations with the `map` function is due to the fact that the operations in the `map` function are not executed on the GPU as expected, but on the CPU. This is because the `tf.data` API places the whole input pipeline on the CPU. The benefit of using the `map` function should become apparent once it is used with training a model. Normally, while a neural network is training on a particular batch of



(a) Using batches of different size.

(b) Using the TensorFlow in built `map` method.

Figure 3.6: Wall-clock time taken to generate the ACSFs representations for increasing number of samples with two different implementations.

the data, the CPU is preprocessing another part of the data. This reduces the time in which the CPU and the GPU sit idle.

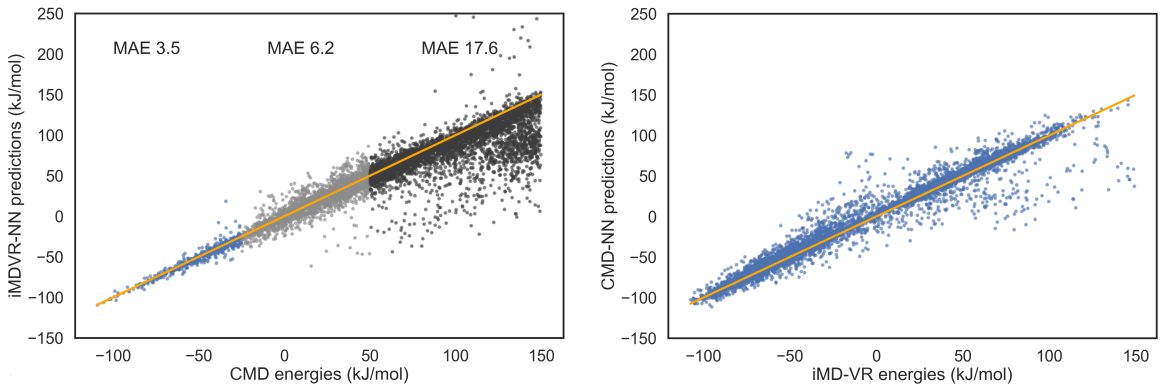
Another observation that can be made is that there is no great improvement when using batches larger than 100. This could be due to the fact that eventually the overhead of copying the data to the GPU becomes the time limiting step.

3.2.4 Training and validating the models

Table 3.1: Mean absolute error (MAE) for the predictions and the cross-predictions of the iMD-VR-NN and the CMD-NN. The values are in $\mathrm{kJ\,mol^{-1}}$ and the error is one standard deviation.

		Predicting on	
		iMD-VR data	**CMD data**
Training on	**iMD-VR-NN**	3.6 ± 5.0	12.4 ± 21.0
	CMD-NN	6.4 ± 11.0	5.1 ± 6.0

For both the iMD-VR and the constrained MD data set, two distinct atomic neural networks were trained on 7621 data points, as this is the maximum number of data points available for the CMD data set. The 7621 configurations from the iMD-VR data set were randomly picked from the 21563 data points. The model trained on iMD-VR data (referred to as iMD-VR-NN) was used to predict the energy of the abstraction trajectory that was left out for testing purposes (as in the previous chapter). Then, it was also used to predict the energies of the 7621 structures from the CMD data set (cross



(a) iMD-VR-NN predicting the CMD data.

(b) CMD-NN predicting the iMD-VR data.

Figure 3.7: Correlation plot for the cross-predictions using the models trained on iMD-VR (3.7a) and CMD (3.7b) data.

prediction). The opposite was done for the model trained on the CMD data (referred to as CMD-NN). The MAE of the predictions and the cross-predictions are shown in Table 3.1.

The results show that the CMD-NN can predict with about the same accuracy the test data from the CMD test trajectory as the data from the iMD-VR data set. This suggests that the configuration space sampled in iMD-VR is covered by CMD training set (the forces applied in VR to the atoms did not distort the molecules outside the configuration spaced sampled by CMD). The iMD-VR-NN does not predict the energies of the CMD data as accurately. In order to understand why, the correlation plots for the iMD-VR-NN and the CMD-NN predictions were analysed (Fig. 3.7a and 3.7b respectively). For the iMD-VR-NN, the correlation plot (Fig. 3.7a) energies were divided in three regions: low energy region, medium energy region and high energy region. The MAE was calculated for each region. This shows that the iMD-VR-NN does better than the CMD-NN at predicting lower energy structures which are not far off the minimum energy path. However, it is evident that the iMD-VR-NN does not predict as accurately the high energy structures. This can be explained by the fact that the iMD-VR sampling gives structures that are closer to the minimum energy path, with very few structures with energy between $100\,\mathrm{kJ\,mol^{-1}}$-$150\,\mathrm{kJ\,mol^{-1}}$. Consequently, the iMD-VR-NN has not learnt this region of the potential as well as the CMD-NN.

3.2.5 Potential energy surface prediction

A relaxed potential energy surface scan of a primary hydrogen abstraction was generated by Dr Lars Bratholm. [3] The scan was performed along the two CH distances shown in Fig. 3.9. The same grid of constraints as for the CMD data set was used. The unconstrained degrees of freedom were optimised at

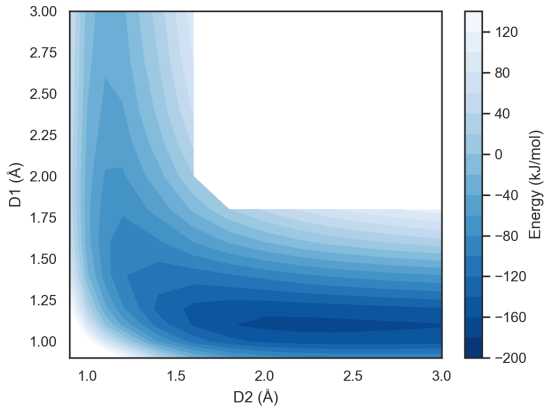


Figure 3.8: Relaxed surface scan of the abstraction of a primary hydrogen by the cyano radical from isopentane. The energies are calculated at CF-uPBE/TZVP level. The degrees of freedom scanned are shown in Fig. 3.9.

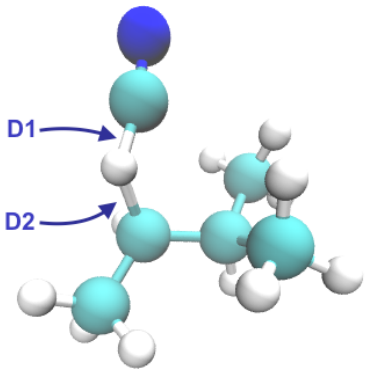
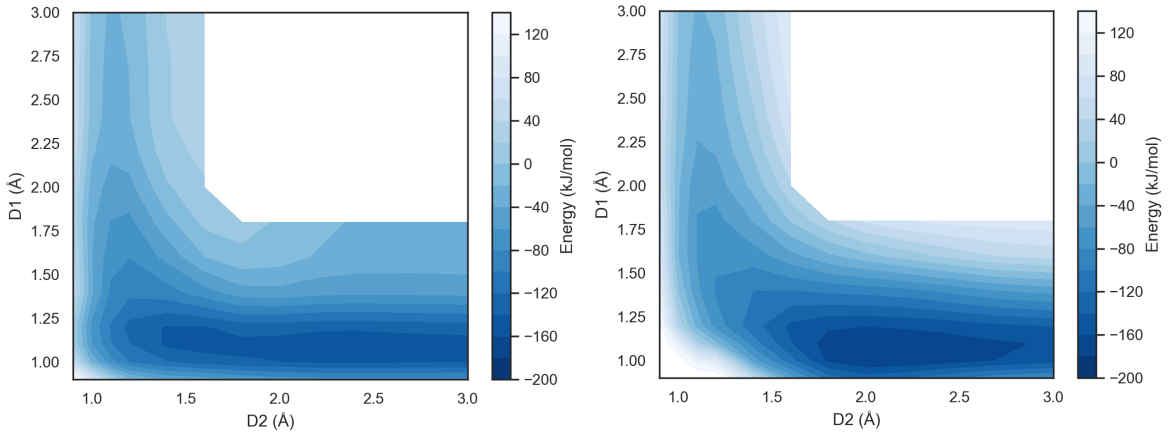


Figure 3.9: Degrees of freedom scanned to generate the surface in Fig. 3.8.

the CF-uPBE0/SVP level of theory and then the energies were re-calculated at the CF-uPBE/TZVP level of theory (as for the iMD-VR and the CMD data set). The energies were then interpolated to give a smooth surface (Fig. 3.8). Only the energies up to 100 kJ mol^{-1} above the reference energy were plotted. Since the geometries have been relaxed, the energies along the minimum energy path are much lower compared to those of the structures in the iMD-VR data set. The iMD-VR-NN and the CMD-NN were used to predict the energies of the structures along the relaxed surface. The resulting surfaces are shown in Fig. 3.10.

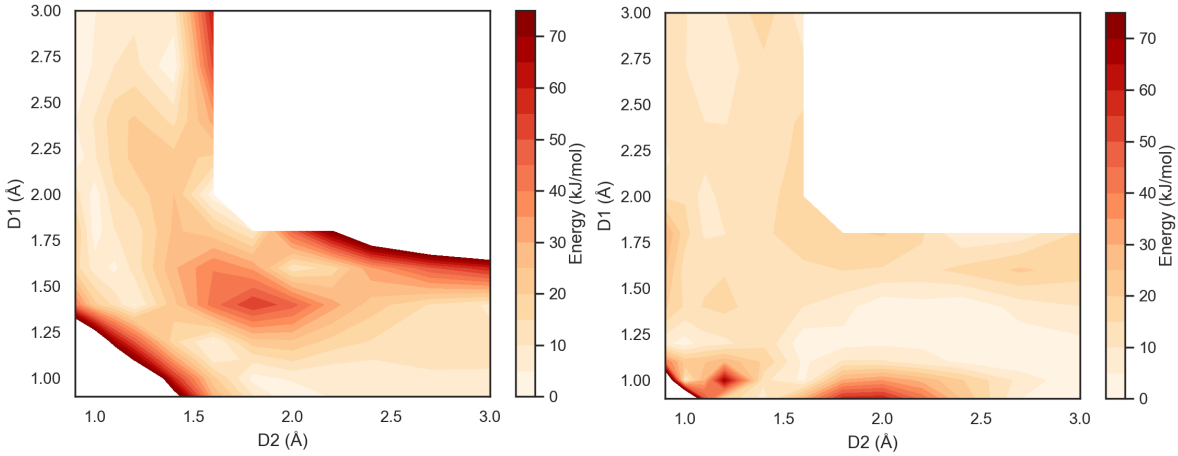
Both iMD-VR-NN and CMD-NN can predict surfaces where the features of the DFT surface are qualitatively reproduced. There appears to be no barrier to the abstraction. [137] The CMD-NN qualitatively reproduces the surface better compared to the iMD-VR-NN. This is due to the fact that the CMD data was generated using the same constraints as for generating the relaxed surface. Consequently, the structures sampled are more similar and therefore easier to predict. The errors in the iMD-VR-NN predictions are more significant in the high energy regions of the potential energy surface (Fig. 3.11a). This is because the iMD-VR samples structures closer to the minimum energy path, as explained in the previous section. Overall, the MAE for the iMD-VR-NN predictions is 66.5 kJ mol^{-1} and for the CMD-NN is 24.1 kJ mol^{-1}. These errors are much larger than those in the previous section for two reasons. Firstly, because the structures to be predicted were chosen with a grid scan of the potential energy surface, which means that they have beens sampled also from regions of the surface that are not present in the training set. Secondly, the structures being predicted have been relaxed, while there were no relaxed structures in the training set.



(a) iMD-VR-NN predictions.

(b) CMD-NN predictions.

Figure 3.10: Predictions for the energies of the relaxed surface scan using the neural networks trained on the iMD-VR and the CMD data sets. The reference surface is shown in Fig. 3.8.



(a) iMD-VR-NN errors.

(b) CMD-NN errors.

Figure 3.11: Difference of the predictions for the relaxed surface (Fig. 3.10a and 3.10b) and the CF-uPBE/TZVP reference energies (Fig. 3.8).

3.3 Conclusion

In this chapter, atomic neural networks were used in combination with the ACSFs to fit the potential energy of isopentane reacting with a cyano radical. Two different ways of generating the data set were compared. The first one was iMD-VR while the second was CMD, both with PM6 as the force engine for the molecular dynamics. Two different atomic NNs were trained on these data sets and their performance was compared.

It was noticed that iMD-VR samples regions closer to the minimum energy path, while with CMD one can obtain structures from regions of high energy. The results showed that the NN trained on the iMD-VR data had larger errors when predicting the data from the CMD data set, as the latter data set contained structures from regions of the potential energy surface that are not sampled by iMD-VR. However, for configurations close to the minimum energy path, both NNs gave errors below $7\,\mathrm{kJ\,mol^{-1}}$ when predicting the energies of the structures of the opposite data set. This is because both data sets have sampled this region effectively. This is an important reminder of the effect of the data set on the performance of a NN for fitting potential energy surfaces.

In conclusion, this chapter showed that using iMD-VR is a suitable method for generating a data set for fitting a potential energy surface of hydrocarbons reacting with cyano radicals. iMD-VR offers the advantage over methods like CMD that enhanced sampling can be done intuitively and no collective variables or degrees of freedom to constrain have to be programmed. This can facilitate and speed up the generation of data.

The PM6 force engine was found to give trajectories that do not show the over stabilisation of the transition state that was previously observed with the DFTB+ force engine. Consequently, PM6 will be used to generate the data set of the larger systems in the following chapter. However, predictions of structures that differ considerably from those along the minimum energy path will result in large uncertainties. A way of detecting this problem could be to use a 'committee of networks'. [138] This involves training multiple models instead of a single one, to then combine the predictions. The most common approach to training a committee is to take a collection of networks with same hyper-parameters, but different random initial weights. They are all trained on the same data and the final prediction is the average of all the predictions. The committee can perform better than the best model used in isolation. [138] In addition, by comparing the variance in performance of the different models, one can get an estimate of the uncertainty for a particular prediction.

Chapter 4

Reaction of cyano radical and squalane

In the previous two chapters, NNs were trained to predict the potential energy surfaces of a cyano radical reacting with small hydrocarbons (methane and isopentane). This chapter builds on what was learnt previously to fit the potential energy surface of a cyano radical reacting with squalane.

Creating a data set for a system as large as squalane can be extremely time consuming. First of all, sampling a large number of geometries is harder than for the smaller systems. If the sampling is based on an enhanced sampling method, evaluating the forces acting on each atom for each time step of the molecular dynamic simulation takes longer due to the much larger number of atoms. Then, refining the energies of each geometry with a more accurate electronic structure calculation takes longer. Consequently, here a different strategy was used compared to the previous two chapters. Instead of training the NN on various conformations of the system of interest (i.e. squalane), the atomic NN was trained on a data set consisting of smaller hydrocarbons (i.e. methane, ethane, isobutane, isopentane and isohexane) reacting with cyano radical. In this way, the transferability of atomic NNs can be investigated. Atomic NNs have been shown in the literature to be transferable when learning the potential energy surface of small organic molecules around geometries corresponding to energy minima. [26] This chapter builds on these findings by studying if atomic NN potentials are also transferable for reactive systems. Transferability for reactive systems would be useful, because it would enable to study reaction mechanisms at a high level of theory, without having to generate high quality data for the large systems, only for the smaller fragments of it. This may enable to study reactive systems that have so far been out of reach due to computational constraints.

4.1 Method

4.1.1 Generating the data sets

The data sets were generated using iMD-VR (with PM6) within the Narupa framework [102] for hydrocarbons of a variety of lengths. One hydrocarbon and the cyano radical were loaded in the Narupa environment and spawned in random non-overlapping positions. The hydrocarbons used were methane, ethane, isobutane, isopentane, 2-isohexane, 3-isohexane and squalane. The iMD-VR simulation was run in the same way as in chapter 3.

The data set generated for methane is different from that in the first chapter, because the force engine used is PM6 instead of DFTB. 81439 configurations were sampled in iMD-VR and were then reduced to 18000. This was done by taking the average of the first 400 frames and the last 400 frames to find the energies of reactants and products. Then, at most 600 frames before and after the mid point between the reactants and products energies were taken. For ethane, the same procedure was followed. In this case, 28969 configurations were obtained with iMD-VR and were then reduced to 7975. For isobutane, 40984 samples were sampled and then pruned to 13113. For isopentane, the trajectories sampled in the previous chapter with iMD-VR were reused. Two different isomers of hexane were sampled: 2-isohexane and 3-isohexane. For 2-isohexane, 36389 configurations were initially sampled and they were then pruned using the same procedure as for methane, but in addition the structures with energy $113\,\mathrm{kJ\,mol^{-1}}$ higher than the energy of the reactants were removed. This left 13084 samples, with a total of 11 abstractions. For 3-isohexane, 45179 samples were obtained and then pruned similarly to 2-isohexane. This left 13 trajectories for a total of 14912 samples. The final molecular system sampled was squalane reacting with the cyano radical. Only one trajectory was generated, as the SCINE implementation used was slow for the generation of forces, which resulted in about 1 time step per second being rendered in iMD-VR. This made it extremely difficult to effectively bias the sampling. It should be noted that since then, the implementation of PM6 in Narupa has been improved and is parallelised. Consequently, it is now easier to sample large systems in Narupa.

In summary, the final data set contained 15 trajectories for methane, 8 trajectories for ethane, 11 trajectories for isobutane (7 primary and 4 tertiary), 19 trajectories for isopentane (11 primary, 3 secondary and 5 tertiary abstractions), 11 trajectories for 2-isohexane (5 primary, 4 secondary and 2 tertiary), 13 trajectories for 3-isohexane (6 primary, 3 secondary and 4 tertiary) and one trajectory for squalane (secondary).

The energies of the structures sampled in iMD-VR were refined with DFT (CF-uPBE/TZVP as in

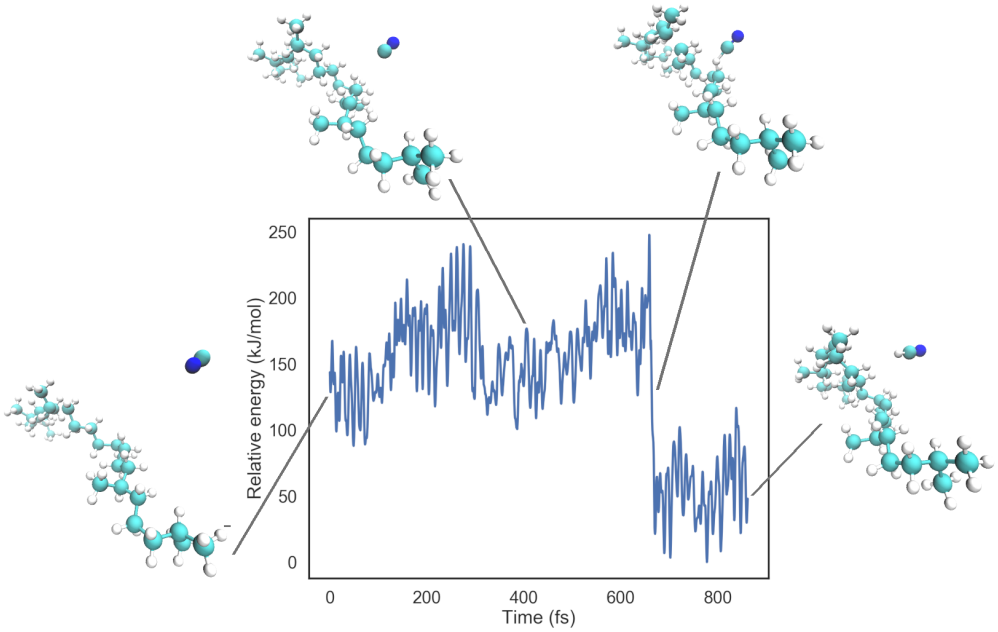


Figure 4.1: Cyano radical abstracting a secondary hydrogen from squalane. It shows how the energy evolves over time. The energies are shown relative to the geometry with lowest energy in the trajectory.

the previous chapter) using MOLPRO [128]. The refined energies of the geometries from the squalane abstraction trajectory are shown in fig. 4.1. However, the energies of the different hydrocarbons have considerably different magnitudes. Consequently, the following procedure was used to shift all the energies so that they were all in the same energy range. The contribution of each atom to the total energy was fitted and subtracted from the total energy. To do this, the Lasso linear model [69] was used:

$$\mathbf{y} = \boldsymbol{\beta}\mathbf{X} + \boldsymbol{\epsilon} \tag{4.1}$$

where $\mathbf{y}$ is the vector of the energies of the different hydrocarbons and cyano radical, $\mathbf{X}$ is a matrix of the features for all the samples (the number of C, H and N atoms in each sample), $\boldsymbol{\epsilon}$ is an intercept term (it has the same value for each sample, i.e. $\epsilon \cdot \hat{\mathbf{i}}$ where $\hat{\mathbf{i}}$ is the unity vector) and $\boldsymbol{\beta}$ is the vector of regression coefficients. The regression coefficient is regularised with L1 regularisation during the optimisation.

To fit the Lasso model, 100 configurations of the reactants of each system were taken. The number 100 was chosen because it seemed enough to represent the average energy of the reactants. The $\mathbf{X}$ matrix has dimensions (N, M), where N is the number of samples and M is the number of element types

present in the systems. In this case, $M = 3$ since there are only H, C and N present in all configurations. This means that a row of $\mathbf{X}$ corresponding to a configuration of $CH_4 + CN$ is $\mathbf{X}_i = [4, 2, 1]$, as there are 4 H atoms, 2 C and 1 N. $\mathbf{y}_i$ is the energy of the $\mathbf{X}_i$ sample minus an arbitrary reference energy (133.1 Ha). The Lasso model is then fit to this data. Once the model is fit, it can predict a shifting factor for hydrocarbon systems with different numbers of atoms. The shifted energies can then be obtained by taking the difference between the shifting factor and the original energy values. The shifting of the energies was done in this way instead of using the atomisation energies so that some of the bonding energy can also be taken into account.

The reason for performing the shifting is as follows. The final layer of each feed forward NN in the atomic NNs (h_1^{l+1}) is a linear combination of the outputs (h_i^l) from the last hidden layer l plus a bias b_i^l:

$$h_1^{l+1} = w_{11}^l h_1^l + w_{12}^l h_2^l + ... + w_{1N}^l h_N^l + b_1^l \tag{4.2}$$

The activation function of the last layer is simply linear, there is no sigmoid or hyperbolic tangent. Thus, h_1^{l+1} corresponds to the 'atomic energy' of the atoms in the system. By shifting the energies, one ensures that the magnitude of h_1^{l+1} is close to zero, and thus the value of b_1^l is not large. In practice, it was found that the NN training is faster if the magnitude of the output to predict is small, hence why the energies of the molecules were shifted. It can be said that the NN trained on the shifted energies learns the 'deviations' from a hypothetical reference structure and energy, rather than learning to predict the absolute energy for a structure.

Six mixed data sets with the same total number of data points (15000) but different ratios of the various species were constructed. The data sets were capped at 15000 data points because larger data sets could not be fit in memory when training the NNs. The composition of the six data sets is shown in table 4.1.

For each mixed data set, at least one full trajectory was left out the training for each different hydrocarbon, so that it could be used for testing.

4.1.2 Software details

The atomic NNs and ACSFs implementations used were the same as in the previous chapter. The current implementation of the atomic NNs in QML expects the input data to have the shape $(1, N_a, M)$, where N_a is the number of atoms in the system and M is the number of features. Therefore, since the NN needs to be able to predict the energy of squalane reacting with the cyano radical, N_a needs to be

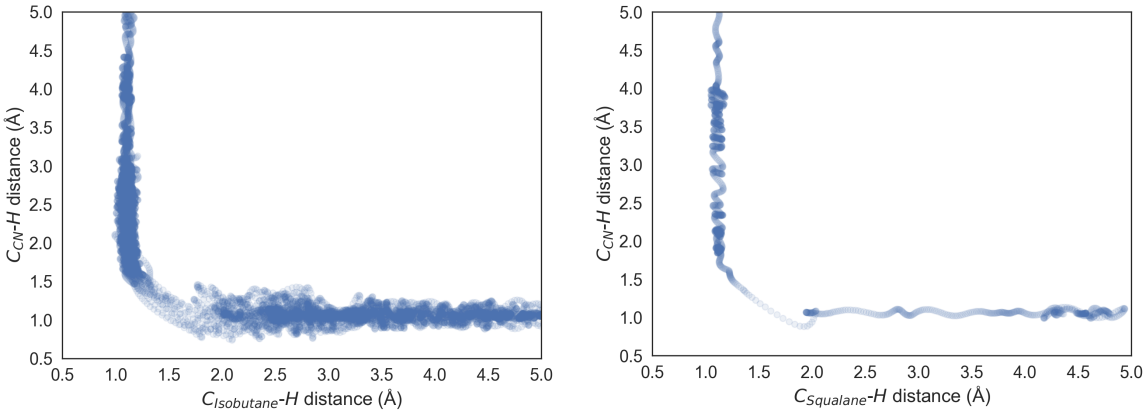
Table 4.1: Composition of the six different training sets used to train the atomic NNs.

Training set	Composition
1	15000 Methane
2	10000 Methane
	5000 Ethane
3	8000 Methane
	4000 Ethane
	3000 Isobutane
4	7500 Methane
	3500 Ethane
	2500 Isobutane
	1500 Isopentane
5	7500 Methane
	7500 Isopentane
6	8000 Methane
	4000 Isopentane
	1500 2-Isohexane
	1500 3-Isohexane

94. This means that the inputs for all smaller molecular systems need to be padded with 0 to reach the correct number of dimensions.

4.1.3 Hyper-parameter optimisation

Both the hyper-parameters of the ACSFs and of the atomic NN were optimised. For the former, these are η, ζ, R_s, θ_s and a cut off radius, [3] while for the latter they are the number of hidden neurons in the first and second hidden layers, batch size, number of iterations, learning rate and L1 and L2 regularisation parameters. They were optimised using a random search. Group 3-fold cross validation [3] was used and the performance of the NN was assessed by looking at the MAE of the predictions. The package used to perform the hyper-parameter optimisation was Osprey. [114] The hyper-parameter optimisation was done using the shifted energies.



(a) Isobutane

(b) Squalane

Figure 4.2: Values of the distances between the cyano carbon and the abstracted hydrogen as a function of the distance between the hydrocarbon's carbon and the abstracted hydrogen. Each data point is plotted with transparency, so that the difference in sampling of various regions can be observed. The plots for all the other hydrocarbons are shown in Appendix E.

4.2 Results and discussion

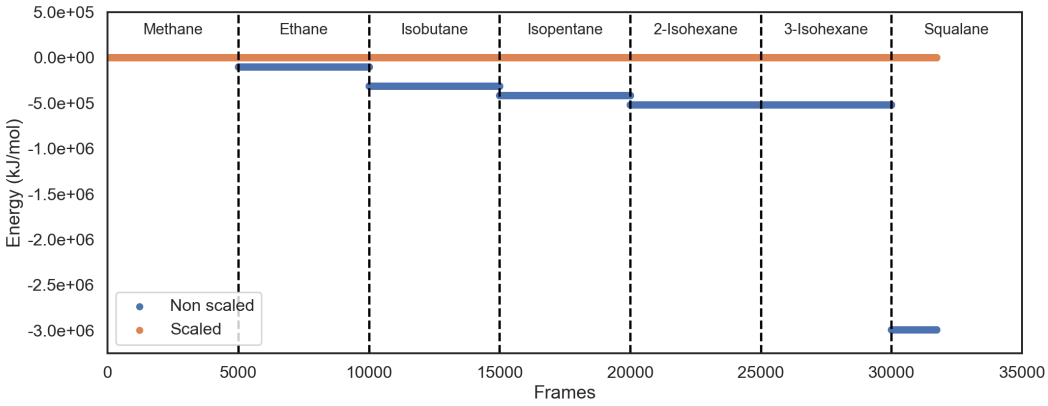
4.2.1 Data sets

Fig. 4.2 shows the values of the $C_{hydrocarbon}$-H and C_{CN}-H distances (for the H being abstracted) during the sampled abstraction trajectories of isobutane and squalane. As can be seen from Fig. 4.2, the structures sampled the most have either a short $C_{hydrocarbon}$-H or a short C_{CN}-H, where the H is the hydrogen being abstracted. These are the structures that are closer to the minimum energy path for abstraction. Transition state structures, where the distance between the hydrocarbon C and the H is larger than 1.2 Å and the cyano C and the H is larger than 1.1 Å, are not sampled as often. This is evident from the lighter colour of the plot, due to the lower density of points in this region. Obtaining additional samples near the transition state configuration would require sampling many more trajectories in iMD-VR. Then, most of the data points near equilibrium would be discarded, while most of the data points near the transition state would be kept. This would give a more homogeneous distribution of samples. However, it would be considerably more time consuming.

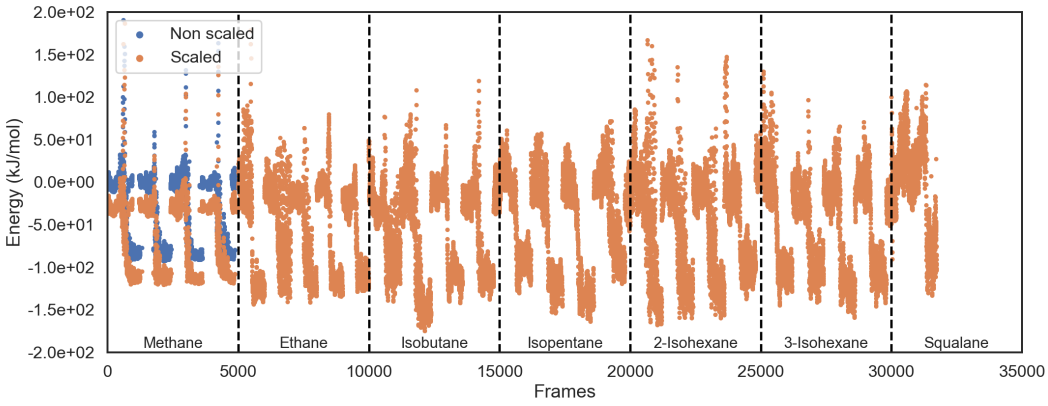
The plots for all the other hydrocarbons are similar to the one for isobutane (Fig. 4.2a), i.e. the structures sampled are in the vicinity of the minimum energy pathway and the transition state is not sampled as frequently. Therefore, they are not shown here, but they can be seen in Appendix E.

After pruning, all the energies were shifted using the Lasso model. The values the energies before

and after shifting are shown in Fig. 4.3. Only 5000 data points per hydrocarbon are shown. The average energy of the methane and cyano radical system was used as the reference energy. The longer the hydrocarbon, the lower the energy of the system. Fig. 4.3a shows a 'zoomed out' view of all the energies. Since the range is of around $3 \times 10^6\,\mathrm{kJ\,mol^{-1}}$, the details of the trajectories cannot be seen. Fig. 4.3b shows a 'zoomed in' view of the trajectories, where the energy range is only around $2 \times 10^2\,\mathrm{kJ\,mol^{-1}}$. This shows how for larger hydrocarbons, the difference in energy between the reactants and the products is larger.



(a) Energies of all the different systems.



(b) Zoom into the range where the scaled energies lay.

Figure 4.3: Values of the energies of all the systems with different sizes of hydrocarbons before (blue) and after (orange) shifting with the Lasso model. Fig. 4.3b is a zoomed in view of Fig. 4.3a in the range of the scaled energies.

4.2.2 Training and validating the models

An atomic NN was trained for each mixed data set with the highest scoring set of hyper-parameters obtained in the hyper-parameter optimisation (as described in section 4.1.3). All the hyper-parameters values are reported in Appendix F.

Table 4.2: Mean absolute error (MAE) for the predictions of the NNs trained on mixed data sets.

Trained on	**Predicting**	**MAE** ($\mathrm{kJ\,mol^{-1}}$)
Training set 1	Methane	0.95 ± 1.18
	Squalane	50.49 ± 51.54
Training set 2	Methane	1.49 ± 1.69
	Ethane	1.71 ± 2.37
	Squalane	640.94 ± 56.17
Training set 3	Methane	0.93 ± 1.12
	Ethane	1.24 ± 1.82
	Isobutane	4.20 ± 6.26
	Squalane	24.87 ± 26.78
Training set 4	Methane	1.35 ± 1.56
	Ethane	1.04 ± 1.50
	Isobutane	3.28 ± 4.60
	Isopentane	3.79 ± 5.32
	Squalane	106.42 ± 13.54
Training set 5	Methane	1.20 ± 1.53
	Isopentane	3.79 ± 6.16
	Squalane	154.02 ± 11.80
Training set 6	Methane	1.79 ± 1.70
	Isopentane	3.14 ± 4.97
	2-Isohexane	5.21 ± 7.40
	3-Isohexane	3.97 ± 5.91
	Squalane	129.90 ± 10.72

The first NN was trained on training set 1, which contained 15000 samples of methane and CN. It was then used to predict the energy of a test trajectory of $CH_4 + CN \longrightarrow CH_3 + HCN$ which had been left out of the training set. It was also tested on a trajectory of CN abstracting a secondary hydrogen from squalane (Fig. 4.1). The MAEs of the predictions are reported in table 4.2 and the predicted energies for the squalane abstraction trajectory are shown in Fig. 4.4. As can be seen from table 4.2, the NN has learnt to predict the energies of methane structures very accurately, as it reaches a MAE of $0.95 \pm 1.18\ \mathrm{kJ\,mol^{-1}}$. However, it does not generalise to squalane. Fig. 4.4 shows that the predicted trajectory does not show a change in energy between the reactants and the products and the changes

in energies due to the stretching motions of the bonds are not matched. The poor performance of this NN in predicting squalane is not surprising, since it could not have learnt about carbon-carbon bonds and secondary hydrogen abstractions.

The model trained on training set 2 (including both methane and ethane) gave a trajectory with a clear difference between the predicted energies of the reactant and the products (Fig. 4.5). However, the changes in energies due to the stretching motions of the bonds are still not captured and there is an offset of about $600\,\mathrm{kJ\,mol^{-1}}$ between the reference energies and the predictions. Potential reasons for this offset are discussed later.

When isobutane was added to the training set (training set 3), the network learnt about tertiary hydrogens. The correlation plot for the energy predictions and the DFT energies started to approach a straight line (green plot in Fig. 4.6). The energies of the products are predicted worse compared to the reactants. This is because in isobutane only primary and tertiary abstractions can be sampled, but the squalane test trajectory is a secondary abstraction.

When the training set contains 7500 methane, 3500 ethane, 2500 isobutane and 1500 isopentane (training set 4), the model can learn about secondary hydrogen abstractions. Now the stretching motions of the bonds are captured better and the energy difference between the products and the reactants is similar to that seen in the reference data. However, there is still an offset between the predictions and the reference values. The energies are under-estimated by about $100\,\mathrm{kJ\,mol^{-1}}$.

The NN trained on the data set with 7500 methanes and 7500 isopentanes (training set 5) gave very similar results to the one with methane, ethane, isobutane and isopentane (training set 4), which suggests that the shorter hydrocarbons still contribute valuable information and reduce the number of isopentane samples required to obtain a good description of squalane (Fig 4.8).

When larger hydrocarbons such as isohexane were added to the training set (training set 6), the performance of the NN did not improve significantly (Fig. 4.9). This suggests that isopentane contains enough information to learn the most important features of the potential energy surface and isohexane does not add much new. However, the offset still did not improve. The reason why there is this offset in the predictions is unclear. Especially because the offset changes value for the data sets with small hydrocarbons and then remains similar for the different data sets that contain isopentane. For the three data sets with isopentane, the NN predicts the system to be more stable than it is. This could be due to the fact that squalane is destabilised by unfavourable backbone conformations that cause steric interactions. This sort of interactions cannot be learnt from short hydrocarbon chains. A way to test this hypothesis would be to perform a hydrogen abstraction on a straight squalane chain and see how the predicitons change.

In order to compare the relative performance of the NNs trained on the six different data sets, the MAEs and the R^2 of the predictions with an offset correction were calculated (table 4.3). This was calculated with the scikit-learn R^2 score function, which can give negative numbers. [139] The correction constant c was calculated by minimising the square difference between the DFT values and the NN predictions minus c with respect to the constant c. Gradient descent was used for the numerical optimisation. This shows the expected results: the data sets with only methane and methane with ethane are the worst. Adding isobutane halves the MAE and adding isopentane halves it again. Isohexane does not have a considerable effect on the MAE.

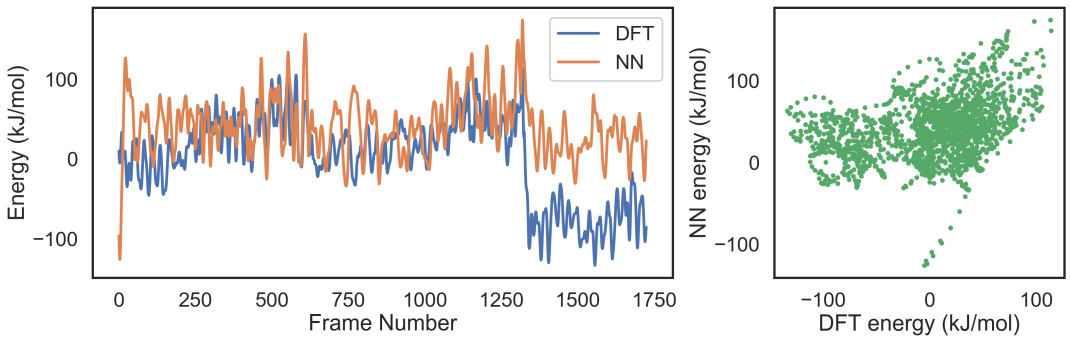


Figure 4.4: Neural network trained on training set 1 predicting the energies of the trajectory of squalane reacting with CN.

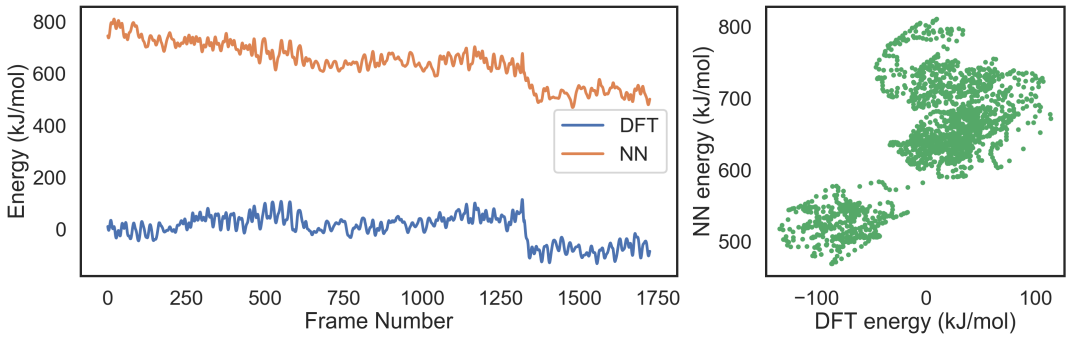


Figure 4.5: Neural network trained on training set 2 predicting the energies of the trajectory of squalane reacting with CN.

The predictions of the NN trained on the data set containing isopentane and isohexane (training set 6) were compared to the PM6 energies for the squalane trajectory (Fig. 4.10). A constant was also removed from the PM6 energies in order to minimise the error from the DFT energies. The PM6 energies have a MAE of 34.8 kJ mol^{-1} compared to the DFT energies, while the NN predictions only have a MAE of 8.2 kJ mol^{-1}. This is also evident from Fig. 4.10, where the NN predictions are closer to the DFT energies compared to the PM6 energies, especially for the products. The energy of the squalane radical and HCN is under-estimated by PM6 by about 50 kJ mol^{-1}.

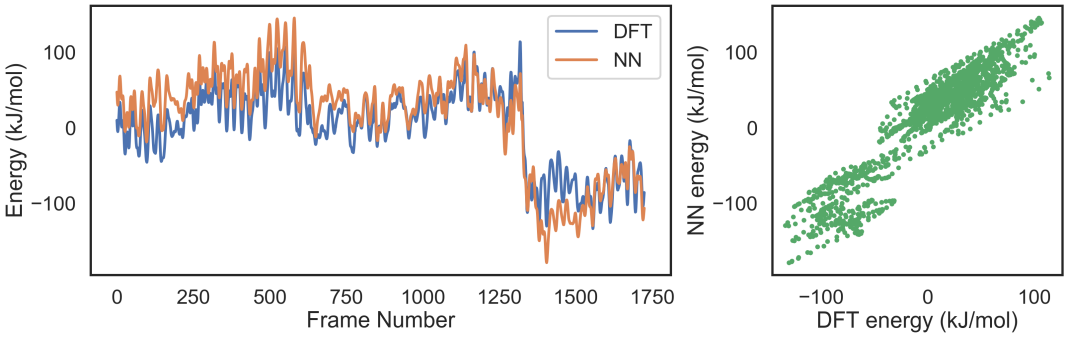


Figure 4.6: Neural network trained on training set 3 predicting the energies of the trajectory of squalane reacting with CN.

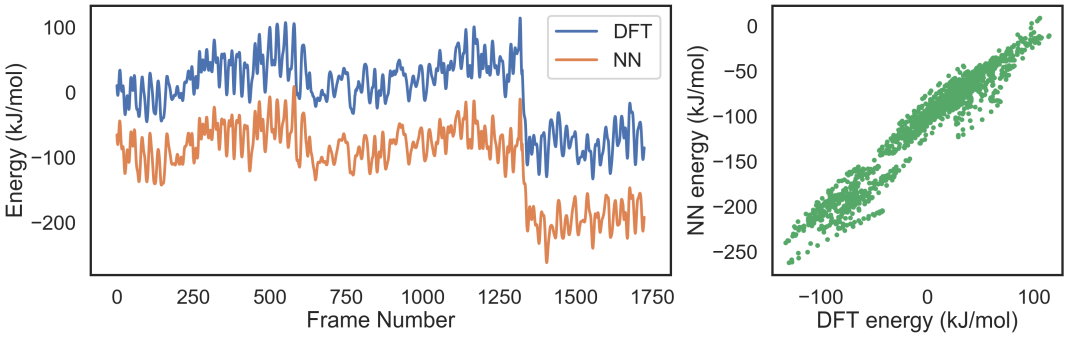


Figure 4.7: Neural network trained on training set 4 predicting the energies of the trajectory of squalane reacting with CN.

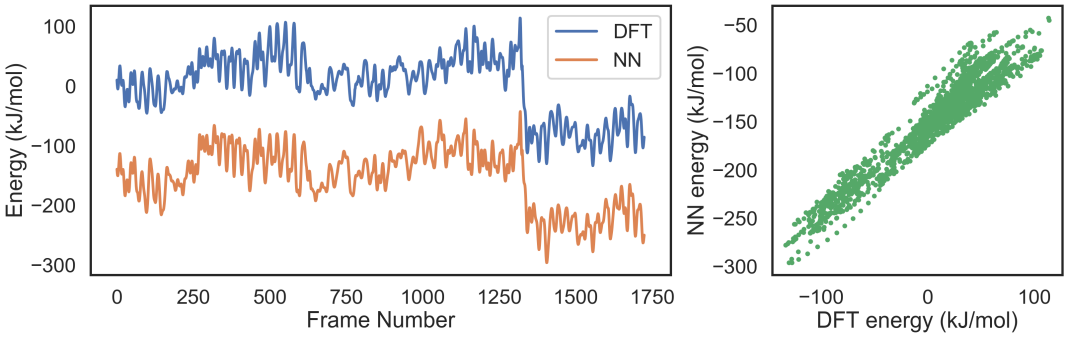


Figure 4.8: Neural network trained on training set 5 predicting the energies of the trajectory of squalane reacting with CN.

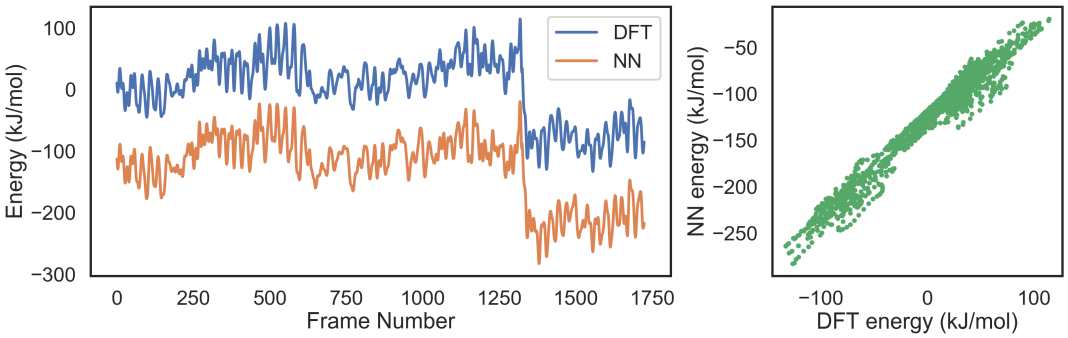


Figure 4.9: Neural network trained on training set 6 predicting the energies of the trajectory of squalane reacting with CN.

Table 4.3: Mean absolute error (MAE) for the corrected squalane predictions of the NNs trained on mixed data sets. The correction is the removal of the energy offset. The scikit-learn R^2 score [139] for the correlation plot is also reported.

Training set	**Corrected MAE** ($\mathrm{kJ\,mol^{-1}}$)	$\mathbf{R^2}$
1	41.52 ± 30.53	0.01
2	43.20 ± 35.91	-0.18
3	20.68 ± 17.01	0.73
4	9.87 ± 9.26	0.93
5	11.54 ± 8.53	0.92
6	8.16 ± 6.9	0.96

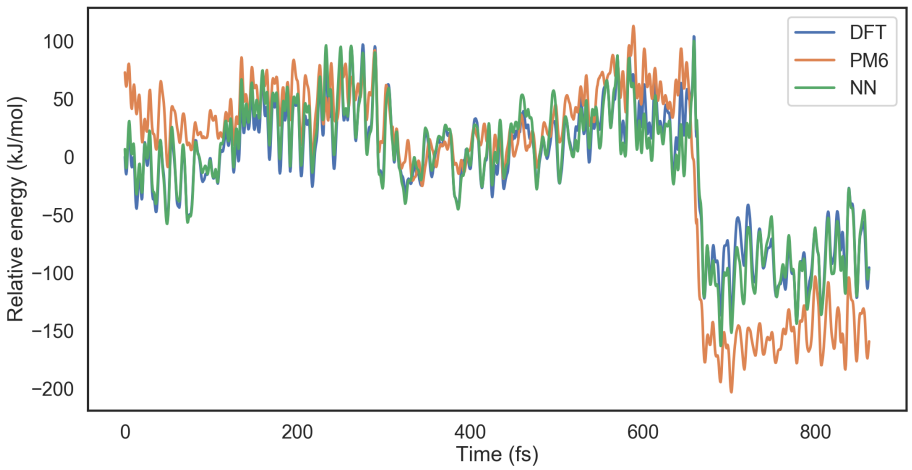


Figure 4.10: Comparison of the NN predictions, the PM6 energies and the DFT energies of the trajectory of squalane reacting with CN. Energy offsets have been removed for both the NN predictions and the PM6 energies, in order to give a fair comparison.

4.2.3 Environment analysis

As was explained in the introduction to ACSFs (section 1.4.2), ACSFs are a 'local representation', which means that each atom in the molecule is represented by one fixed-size vector. Each fixed-size vector represents the environment around a particular atom.

In this section, these fixed-sized vectors that make up ACSFs are analysed. The vectors representing the environments of the atoms in squalane are compared to those representing the environment of atoms in smaller hydrocarbons. This will be done only for the carbon atoms.

To do this, the Manhattan distance between the vectors representing the carbons in squalane and the vectors representing the carbons in smaller hydrocarbons is calculated. For each smaller hydrocarbon, only one abstraction trajectory is used (for isopentane and isohexane, the secondary abstraction trajectories are used).

The Manhattan distance $D_{\text{Manhattan}}$ between two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$D_{\text{Manhattan}}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i| \tag{4.3}$$

The procedure used is explained below for isobutane, but it is the same for all other hydrocarbons. One starts with the first carbon in the first geometry of the squalane with CN abstraction trajectory. The Manhattan distances of the vector representing this first squalane carbon to the vectors representing all the carbons in a isobutane with CN abstraction trajectory are calculated. For example, if a trajectory contained 10 geometries, then 50 distances would be obtained, because there are 5 distances per geometry (as there are 4 carbon atoms in isobutane and one in the cyano radical) and there are 10 geometries. Out of these 50 distances, the *smallest* one is recorded, while the others are discarded. This process is repeated for all the carbons in squalane and CN and for each geometry in the trajectory. Since there are 30 carbons in squalane and one in the cyano radical, at the end there are 31 distance values per geometry in the squalane abstraction trajectory. The distribution of these distances can then be plotted to give an idea of how similar are squalane carbons atoms compared to isobutane carbons. If the distribution shows only peaks at distances close to zero, then the carbons in isobutane are very similar to the carbons in squalane. This procedure was followed for all the small hydrocarbons and the plots of the distance distributions are shown in Fig. 4.11.

For methane, there are no carbons with a Manhattan distance shorter than about 15 to the squalane carbons. With ethane, the situation improves: there are carbons with a distance as small as 6. From isobutane onwards, all hydrocarbons have some carbons with distance from around 2. Isopentane is

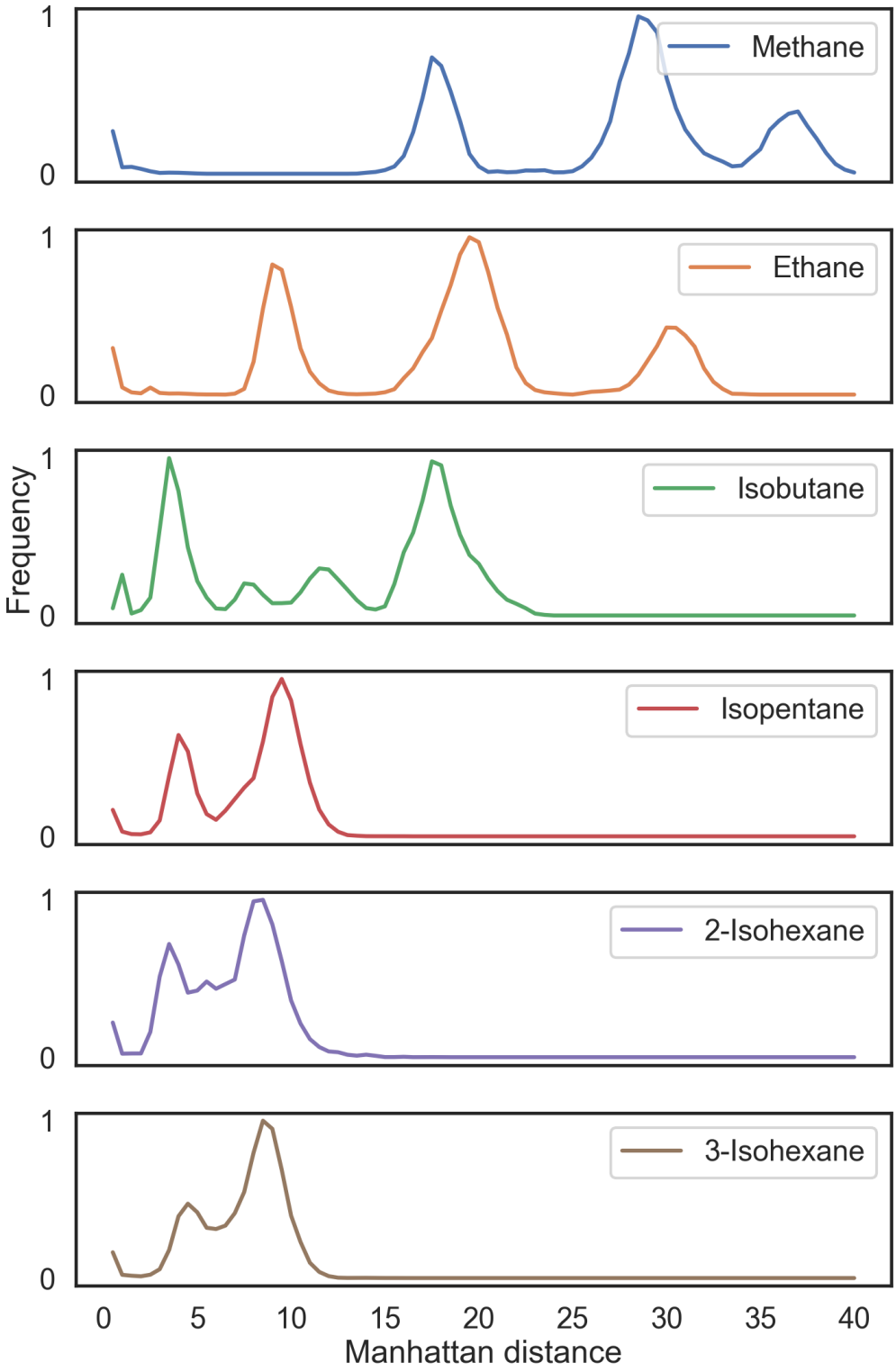


Figure 4.11: Distribution of the shortest Manhattan distances between the ACSF of each carbon in squalane and the carbons in an H-abstraction trajectory from a shorter hydrocarbon.

the first hydrocarbon where there are no carbons with distances larger than about 13.

Fig. 4.11 confirms what was previously stated: isopentane is the shortest hydrocarbon which contains most of the information required to represent squalane. This is because isopentane is the smallest system that contains primary, secondary and tertiary hydrogens. Methane is the most different, as it does not even include carbon-carbon bonds. The carbons in the two isohexanes isomers are only slightly more similar to the squalane carbons compared to those in isopentane.

All the carbons in squalane that have a Manhattan distance between 6-13 to the isopentane carbons were selected, to understand which type of carbon atoms they are. The analysis shows that they are secondary and tertiary carbons (Fig. 4.12). This could be due to the fact that due to the arrangement of the backbone, there are interactions between the atoms that are not captured when using short hydrocarbons. This is in agreement with what was postulated in section 4.2.2, i.e. that there are still effects that are not taken into account, in particular those due to the torsions of the squalane chain.

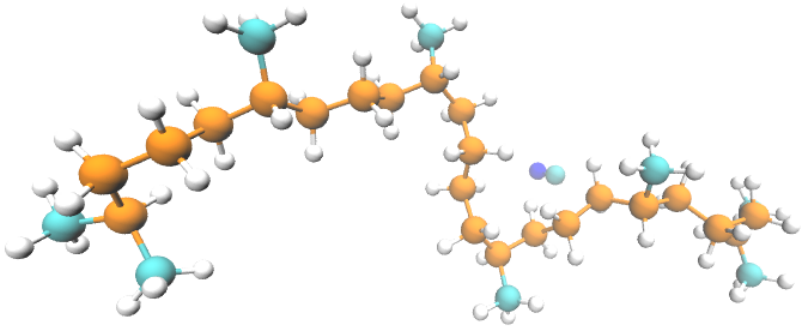


Figure 4.12: Squalane where the carbon atoms that have a Manhattan distance between 6-13 from carbons in an isopentane H-abstraction trajectory have been coloured in orange.

4.2.4 Prediction timings

This chapter has shown how NNs trained on mixed data sets of smaller hydrocarbons can be used to predict the energies of squalane and obtain MAE of the order of $10\,\mathrm{kJ\,mol^{-1}}$ compared to the reference data. In this section, the timings of the NN predictions for 1725 squalane conformations were measured.

All the NNs trained on mixed data set had different hyper-parameters, which affects the speed of prediction. The hyper-parameters that influence these measurements the most are the number of features of the ACSFs and the number of hidden neurons in the hidden layers. Consequently, the

Table 4.4: Average times obtained when evaluating the energy of 1725 CN + squalane configurations using NNs trained on the 6 mixed data sets. The timings are divided in the ACSFs generation and the energy prediction. The total time for each data sample is shown.

Method	**ACSFs time** (ms)	**Energy time** (ms)	**Total time** (ms)
NN 1	7	1	8
NN 2	8	2	10
NN 3	4	1	5
NN 4	4	1	5
NN 5	4	1	5
NN 6	6	1	7

measurements for each NN are shown in table 4.4. When using a NN to predict the energy of a configuration, there are two steps. The first is to evaluate the representation (in this case the ACSF). Once the representation has been generated, it is input into the NN which outputs the total energy. Since the time taken for each of these steps depends on the hyper-parameters, there are two columns in table 4.4 that show the average time taken for each step, and then there is a column that shows the combined time as well. The results show that the average timings are in the order of 0.01 s for each sample.

In order to have a comparison, DFT (CF-uPBE/TZVP) was used to calculate the energies of the 1725 squalane configurations. MOLPRO was used without parallelisation. This took on average in the order of 1000 s per configuration. On the other hand, PM6 took in the order of 0.1 s. The comparison to the NN timings is shown in table 4.5. This is an encouraging results, since the NN appears to be the fastest at predicting the energies, while giving better energy results compared to PM6 (Fig. 4.10).

Table 4.5: Comparison of the average timings for evaluating the energy of a squalane + CN geometry using NNs, CF-uPBE/TZVP (DFT) and PM6.

Method	**Time** (s)
NN	0.01
CF-uPBE/TZVP	1205
PM6	0.1

4.3 Conclusion and further work

In this chapter, the transferability of atomic NNs with ACSF from small hydrocarbons to large hydrocarbons was investigated. Six different data sets containing mixtures of varying length hydrocarbons

were used for training.

In conclusion, it can be said that an atomic NN can learn to predict the energies of a larger system compared to what it was trained on. For this to be possible, the smaller systems in the training set need to capture enough of the molecular interactions that are found in the larger system. For example, to predict the energy of a secondary hydrogen abstraction on squalane molecules, the NN needs to have learnt about C-C bonds and secondary hydrogen abstractions, which are features not present in data sets containing only methane, ethane and isobutane. Isopentane is therefore required to obtain results with a MAE around $12\,\mathrm{kJ\,mol^{-1}}$ (after removing the offset between the predictions and the reference values). This transferability means that computational power can be saved when creating data sets for potential energy surface fitting, because the full system is not necessarily needed.

While the relative NN predictions can reach low MAEs, there is an offset with respect to the reference data. One hypothesis is that the unfavourable back-bone conformation of the squalane chain were not well represented in the training set. Since only one trajectory was used as a test and in this trajectory the squalane back bone did not move considerably, this could cause a constant offset of the predictions. In order to test whether this is the case, an abstraction trajectory where the squalane is in an all-trans configuration could be tested. If there is still an off-set, then another hypothesis will have to be formulated. However, if the offset was found to be caused by a lack of unfavourable bond angle configurations, the next step for this project would be to add a longer hydrocarbon to the training set with twisted back-bones. These trajectories would not necessarily need to include the cyano radical.

In terms of future software development, the memory limit encountered when training on the mixed data set would have to be addressed. It is related to the fact that the current implementation requires padding the representations of the smaller hydrocarbons before inputting them into the NN. This will require changing how the data is input in the atomic NN. This change would enable to train on larger data sets of small molecules.

Another aspect that needs investigating further is how to gain confidence in the prediction for squalane after training. This could be done using a 'committee of networks' [138].

In the longer term, it would be interesting to investigate the quality of the predictions for the forces in addition to the energies. Being able to obtain the forces from the NN is a key requirement for molecular dynamics simulations. This has been successfully done on small systems, such as $\mathrm{H + HBr}$ [5] or organic molecules with up to 8 heavy atoms. [26] Training a NN that can predict forces presents a few challenges and there are various strategies to tackle them. [5, 62] First of all, evaluating the gradients of the NN is computationally expensive and slows down the training procedure considerably.

Another issue is the quantity of data used for training: while each molecule has only one value for the energy, it has $3N$ forces (where N is the number of atoms). This means that efficient ways of transferring and storing data need to be implemented. Other modifications required to train on the forces include adding an extra term to the cost function, comparing the reference forces and the gradient of the network with respect to the Cartesian coordinates. Evaluating the forces would also slow down the timings of predictions.

On the other hand, training on the forces as well as the energies could help obtaining smoother and more accurate potential energy surfaces. [5] Fewer configurations should be needed to learn the shape of the potential, since the gradients give additional information about the shape of the surface around a certain point.

Part II

Recurrent neural networks as molecular generators

Chapter 5

Introduction to machine learning for *de novo* drug design

De novo drug design is the practice of creating drug molecules from scratch and is a key part of the field of cheminformatics. When it was introduced in the 1990s, it mostly involved building molecules in three-dimensions inside a receptor site and scoring the interactions with the receptor. [140] However, these techniques had limited success because they generated molecules that are difficult to synthesise. [141] More recently, combinatorial techniques that attempt to account for synthetic accessibility have been developed, among which are Retrosynthetic Combinatorial Analysis Procedure (RECAP)[142], Breaking of Retrosynthetically Interesting Chemical Substructures (BRICS) [143] and the Design Of Genuine Structures (DOGS). [144]

RECAP is a procedure where molecules that are known to be active on a biological target are collected into a data set and then fragmented based on chemical knowledge. There are rules that govern which type of bonds can be cleaved. When a particular bond is broken, information about the type of bond is stored, so that when molecules are re-assembled only attachment points with the same type as in the original bond are allowed. For example, if a fragment is obtained by breaking an amide bond, that fragment can only be attached with a new amide bond. This yields molecules that are more synthetically accessible. [145] However, there are disadvantages associated with this technique. The building blocks generated from retrosynthetic cleavage do not necessarily have a physical analogue [145] and the fragmentation can break bonds that might be part of a molecular feature that makes the drug active (pharmacophore). [146] Shunichi et al. also reports that often the generated structures are the same as the initial ones. [147] For these reasons, BRICS was developed. This contains a

more elaborate set of rules for obtaining fragments from biologically active compounds. [143] BRICS was shown to generate 10% more fragments from molecules and give more fragments with multiple connection points. [143] Nevertheless, the molecules produced were not always as easy to synthetise as expected. Consequently, this issue was addressed by using virtual reaction schemes for the generation of new molecules. [148] DOGS is an example of such techniques and it mimics a multi-step synthetic pathway. It is based on a library of 83 established organic reactions and a database of about 25000 commercially available and curated building blocks. [144] New molecules are generated by iterative fragment assembly using reactions from the library. The reactions from the library keep being applied to the candidate molecule until the molecular weight exceeds a user defined value or the maximum number of synthetic steps is reached. The candidates generated are scored based on a particular design objective. Usually, this includes similarity to a reference molecule. [144] In this way, DOGS generates molecules that mimic a certain template with regard to size and pharmacophore features. [149] DOGS has been used extensively in the literature. For instance, it has been applied to the design of new highly potent, selective, and patent-free kinase inhibitors. [150] It was also used to generate new starting points for the development of modulators of retinoid X receptors. [151]

The amount of data generated by medicinal chemists has rapidly increased over recent years. There are large databases such as ChEMBL, [152] ZINC [153] and PubChem [154] with millions of compounds. Thanks to the increased computational power available, machine learning methods have revolutionised a variety of fields where large quantities of data are available, such as image recognition, [155] speech recognition [156], language translation [157], etc. Therefore, there has been an effort in the medicinal chemistry community to apply machine learning techniques to improve drug design. [158, 159] A variety of methods have been used, including recurrent neural networks (RNNs), [86, 160, 140] variational auto encoders (VAEs), [161] adversarial auto encoders, [162, 163] generative adversarial networks (GANs) [164] and graph convolutional networks. [165] It is still unclear which method performs best for *de novo* drug design. Up to now, few benchmarking articles have been published. Recently, BenevolentAI introduced a benchmarking suite called GuacaMol. [166] In their paper they assess the ability of a variety of models to reproduce the property distribution of molecules in a data set, as well as their ability to generate novel compounds. Their study shows that different models perform better depending on which task and metric is considered. Recently, Polykovskiy et al. published another benchmark that compares the performance of various generative models. [167] In their publication, the authors mention that while there are standardised benchmarks and data sets for regression and classification tasks in chemistry, there is no such thing for generative models. Possible causes are the lack of general metrics to assess the generated molecules and the fact that different chemical applications have different requirements. They try to tackle this problem by presenting a benchmarking suite with data processing

tools, implementations of the metrics and of the models. [167] In both benchmarks from BenevolentAI and Polykovskiy et al., large data sets were used.

Medicinal chemist have different expectations for generative models depending on the stage of a drug design project. In the early stages, there may be few known compounds, usually very diverse from each other. Here a more exploratory behaviour is needed in order to find other diverse molecules that can be scored through QSAR models or docking. In the later stages, there may be a variety of molecules available and closer analogues to the existing ones are sought after. Generally, the number of molecules in a data set for a medicinal chemistry project is between 10^2 and 10^3, which is much smaller compared to the 10^6 used in the currently available benchmarks. [166, 167] Consequently, the focused generation of new drug molecules is a subject that still requires investigation.

Gupta et al. [86] have applied RNNs with Long Short-Term Memory (LSTM) cells to design drug molecules (these are explained in detail in section 5.1).

In their paper, Gupta et al. [86] perform a two-step training of the RNN. They first train on a large data set, so that the model learns to generate valid molecules. Then, it is fine-tuned to generate molecules similar to a smaller set of target molecules. This process is referred in the literature both as 'fine-tuning' or 'transfer learning'. With this procedure, the authors can generate new peroxisome proliferator-activated receptor gamma inhibitors, trypsin inhibitors and transient receptor potential M8 blockers. Schneider et al. [168] used a similar approach to design molecules with agonistic activity on retinoid X receptors and peroxisome proliferator-activated receptors. They then screened the obtained compounds and synthesised 5 of them. They tested the activity of the synthesised compounds on a variety of receptors and observed that 4 of the compounds revealed nM to low µM activity in cell-based assays. Müller et al. have used RNNs with LSTM to generate *de novo* amino acid sequences. [169]

Olivecrona et al. [140] as well as Popova et al. [160] have combined RNNs with reinforcement learning (introduced in section 5.3) to further improve the learning process. They first trained a RNN on a large data set of molecules, so that the RNN learns to generate valid molecules. They then used reinforcement learning to refine the model to sample molecules with more desirable properties. As a proof of concept, Olivecrona et al. [140] generated molecules that do not contain Sulphur atoms and also molecules that are active towards dopamine type 2 receptors. Popova et al. showed that they can generate molecules with high melting temperatures, molecules with lipophilicity in a particular range and molecules with high predicted activity towards JAK2 proteins. [160]

Many different representations can be used to describe molecules when working with RNNs, but since RNNs have been extensively used as sequence generation models in Natural Language Processing, a

natural choice is to use SMILES strings. [170] SMILES are strings of ASCII characters and they encode the connectivity in the molecules. They have been developed in the late 1980s [171] and they are a language specifically designed for computer use. They are in fact a true language, just with few words and few grammatical rules. [172] This has encouraged the application of natural language processing techniques to chemistry. Very recently, a variant of SMILES more suited to machine learning was introduced: DeepSMILES. [173] DeepSMILES address the problem of unbalanced parentheses and the problem of pairing ring closure symbols. However, since DeepSMILES are not yet widely used by the cheminformatics community, most of the tools available for filtering and analysing molecules can only process the original SMILES. Consequently, the original SMILES will be used in this work.

However, for reinforcement learning, one needs a way of assessing the 'desirability' of a particular molecule that has been generated by the RNN. In this work, the activity (pIC50) of a molecule on a particular target is used. The pIC50 is defined as:

$$\mathrm{pIC50} = -\log_{10} \mathrm{IC50} \tag{5.1}$$

Where the IC50 is the half-maximal inhibitory concentration, i.e. the molar concentration ($\mathrm{mol\,L^{-1}}$) of drug that is required to inhibit a biological process *in vitro* by 50%. [174] The pIC50 cannot be calculated from first principle and has to be determined experimentally. In this work, a feed forward neural network is used to learn and then predict the pIC50 values of generated molecules, based on experimental data.

For this purpose, SMILES are not suitable. This is due on one hand to their variable length size depending on the molecule. Another more important issue is the fact that a SMILES is a sequence where each character has a special function: a 'C' represents a carbon atom, a ')' represents the end of a branch, etc. The model has to learn the different roles of the characters to be able to learn the correlation between the input molecules and the output property. This makes it harder to obtain an accurate model. Consequently, different representations are usually used for this application.

Representations referred to as 'fingerprints' are a common choice. Fingerprints were initially introduced to improve searching for molecular substructures in large chemical databases, but then started being used for similarity searching, clustering, and classification. [175]

Extended-connectivity fingerprints (ECFPs) [175] were introduced to capture molecular features relevant to molecular activity. ECFPs are formed in three steps: [175]

1. Each atom in the molecule is assigned an index. These initial indices are collected into an initial fingerprint.

2. For each atom, an array containing the index of the atom and its immediate neighbours is created. The neighbours are ordered based on their index. Then, a hash function is used to reduce this array to a single index. These new indices replace the old ones in the fingerprint. This step is repeated a pre-defined number of times. After the first iteration, the identifier only contains information about the immediate neighbours, but as the iterations continue it will incorporate information about the environment further away. The number of iterations is specified with the 'radius' parameter, which determines up to which distance the neighbours of a central atom will be considered when creating the identifiers.

3. Once the previous step is finished, any duplicate identifiers are removed.

4. The array is then stored as an array of 1s and 0s or 'on' and 'off' bits, where 'on' bits correspond to substructures that are present in the molecule. This is because one way to interpret the identifiers generated in the previous steps is as indices of a large (2^{32}) array of bits.

There are many different variants of ECFP, Morgan fingerprints (also known as circular fingerprints) being a prominent example. Morgan fingerprints are implemented in the chemistry Python package RDkit, [176] which makes them readily available for use.

5.1 Recurrent Neural Networks (RNNs)

RNNs are an extension of feed forward neural networks to deal with sequential data. The input data to RNNs has an extra dimension compared to feed forward neural networks. The data for the latter has dimensions $(n_{\text{samples}}, n_{\text{features}})$, where n_{samples} is the number of samples and n_{features} is the number of features of the representation used. For RNNs, the input data usually has dimensions $(n_{\text{samples}}, n_{\text{time}}, n_{\text{features}})$, where n_{time} is the number of time steps or the number of sequence elements present.

The first RNNs were sequences of feed forward neural networks (Fig. 5.1). The hidden neurons of these RNNs took as input both the output of neurons from the previous layer and of the neurons in the hidden layer of previous neural network in the time sequence (Fig. 5.1). More explicitly, the output of the hidden neurons $\mathbf{h}_{t=2}$ is calculated by multiplying the vector of weights $\mathbf{w}_{xh}$ with the input layer (x_2 in Fig. 5.1) and summing it to the multiplication of the vector of weights $\mathbf{w}_{hh}$ with the hidden layer $\mathbf{h}_{t=1}$. A bias $\mathbf{b}$ is also added before applying an activation function f. This is expressed explicitly in equation 5.2 and it is shown in the diagram in Fig. 5.1, with the connections shown explicitly for the first neuron of the second unit (shown with dashed lines):

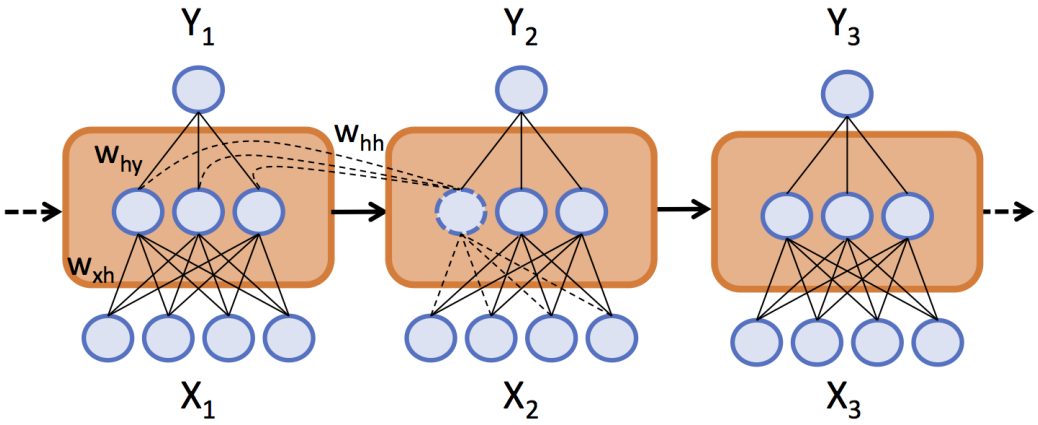


Figure 5.1: Diagram of a simple RNN. Each of the three unit represents one feed forward neural network along the sequence. Each blue circle represents a neuron and each orange box represents a hidden layer. The weights for the connection between the input neurons and the hidden layer are $\mathbf{w}_{xh}$, between the hidden layer and the output layer are $\mathbf{w}_{hy}$ and between the hidden layers are $\mathbf{w}_{hh}$. The connections between the hidden neurons are shown only for the first neuron of the middle unit, to avoid cluttering the diagram.

$$\mathbf{h}_{t=2} = f(\mathbf{w}_{xh} \cdot \mathbf{x}_2 + \mathbf{w}_{hh} \cdot \mathbf{h}_{t=1} + \mathbf{b}) \tag{5.2}$$

It is important to note that the weights are 'shared', meaning that the matrix of weights $\mathbf{w}_{xh}$ that multiply $\mathbf{x}_1$ is the same as the matrix of weights that multiplies $\mathbf{x}_2$ and all successive inputs in the sequence.

Unfortunately, training sequences of feed forward neural networks is difficult, because the of the vanishing and exploding gradients problem. This issue can be understood by analysing the training process. One can view the training of this RNN architecture in the same way as that of a deep multilayer network. The cost function $J(\mathbf{w})$ is the sum of the cost functions $J_t(\mathbf{w})$ at each time step t until the final time step T: [177]

$$J(\mathbf{w}) = \sum_{t=0}^{T} J_t(\mathbf{w}) \tag{5.3}$$

To calculate the gradient of the cost function $J(\mathbf{w})$ with respect to some weight w, one has to take the derivative of equation 5.3.

$$\frac{\partial J(\mathbf{w})}{\partial w} = \sum_{t=0}^{T} \frac{\partial J_t(\mathbf{w})}{\partial w} \tag{5.4}$$

The term $\frac{\partial J_t(\mathbf{w})}{\partial w}$ can be further expanded using the chain rule:

$$\frac{\partial J_t(\mathbf{w})}{\partial w} = \frac{\partial J_t(\mathbf{w})}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial w} \tag{5.5}$$

However, $\mathbf{h}_t$ depends on $\mathbf{h}_{t-1}$, which itself depends on the weight w. In turn $\mathbf{h}_{t-1}$ depends on $\mathbf{h}_{t-2}$, which is also a function of the weight w. Consequently, equation 5.5 can be expanded further using the chain rule:

$$\frac{\partial J_t(\mathbf{w})}{\partial w} = \frac{\partial J_t(\mathbf{w})}{\partial \mathbf{h}_t} \left[\prod_{t \geq i > k} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right] \frac{\partial \mathbf{h}_k}{\partial w} \tag{5.6}$$

The product term can be expressed as: [177]

$$\prod_{t \geq i > k} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{t \geq i > k} diag\left[f'\left(\mathbf{w}_{xh} \cdot \mathbf{x}_i + \mathbf{w}_{hh} \cdot \mathbf{h}_{i-1} + \mathbf{b}\right)\right] \mathbf{w}_{hh} \tag{5.7}$$

where *diag* turns a vector into a diagonal matrix and f' is the derivative of f. The derivative of activation functions such as the sigmoid and hyperbolic functions are always smaller than 1. So, if the magnitude of the weights $\mathbf{w}_{hh}$ is smaller than one, multiplying many gradient terms together will give a value close to zero. If the gradient of the cost function are almost zero, the RNN will train slowly and will struggle to learn long term dependencies. [178] This is the 'vanishing gradient' problem. On the other hand, if the magnitude of $\mathbf{w}_{hh}$ is large, they will overpower the multiplication by the derivative of the activation function. The multiplication of multiple gradient terms that are larger than 1 will grow exponentially. If the gradients are too large, the weights will change significantly from one iteration to the next, making the training noisy and erratic, as well as more likely to diverge. This problem is referred to as the 'exploding gradient' problem. [179, 177]

Long Short-term Memory (LSTM) cells were introduced to tackle this issue. [180, 181] The main idea behind it was to make the term $\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$ have a constant value. Before explaining how this is achieved, the architecture of LSTM cells is explained below.

A cell is a unit that replaces what used to be the hidden layer. The representation of a LSTM cell is shown in Fig. 5.2. They have an additional component, called 'cell state' C_t. The cell state carries information thorughout the whole RNN. It can only be modified with multiplications and additions operations. [182] These operations are a way of selectively adding or removing information from the cell state. Then, instead of having one single hidden layer with one activation function, there are four (blue rectangular boxes in Fig. 5.2). These interact in a special way with each other. They all take as

input a vector x_t and the output of the layers of the previous cell in the sequence h_{t-1}, where t is the current time step. The first three hidden layers are used to calculate a term to modify the cell state. The final hidden layer is combined with the cell state to give the output of the current cell Y_t. This is also passed along to the next cell in the sequence as h_t.

The equation for the different parts of the cells are shown in eq. 5.8-5.13:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.8}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.9}$$

$$d_t = \tanh(W_d \cdot [h_{t-1}, x_t] + b_d) \tag{5.10}$$

$$C_t = f_t * C_{t-1} + i_t * d_t \tag{5.11}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.12}$$

$$h_t = o_t * \tanh(C_t) \tag{5.13}$$

where W_f, W_i, W_d and W_o are the weight matrices and b_f, b_i, b_d and b_o are the biases. $[h_{t-1}, x_t]$ is the concatenation of the output vector from the previous RNN cell and the input vector. x_t is a vector of dimensions equal to the number of unique characters in the data set, f_t, i_t, o_t, d_t and h_t are vectors with dimensions that depend on the architecture. σ is the sigmoid function and tanh is the hyperbolic tangent. The symbol $*$ denotes an element-wise product.

When calculating the gradient of $\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$ from equation 5.7, now there is also a term that depends on the cell state C_t. As is shown in equation 5.11, C_t is a function of f_t, C_{t-1}, i_t and d_t. All of these (except C_{t-1}) are a function of h_{t-1} which is a function of C_{t-1}. Expressing the derivative of C_t with respect to C_{t-1}:

$$\frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial C_t}{\partial f_t}\frac{\partial f_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial i_t}\frac{\partial i_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial d_t}\frac{\partial d_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial C_{t-1}} \tag{5.14}$$

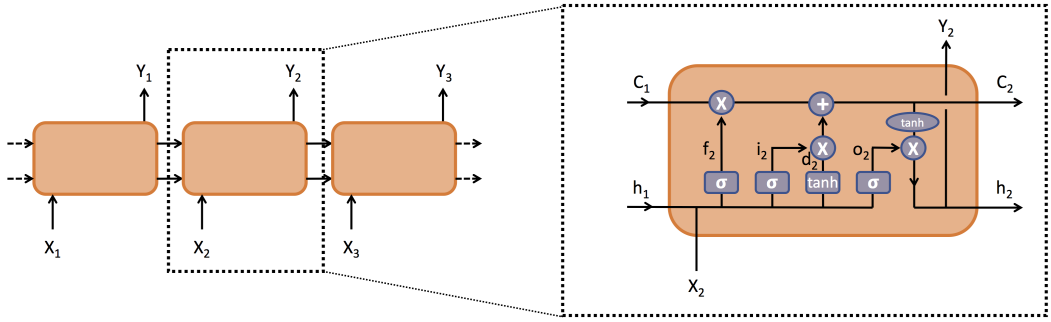


Figure 5.2: Diagram of an LSTM cell. Each blue rectangular unit represents a layer of the RNN with either a sigmoid (σ) or a hyperbolic tangent (tanh) activation function. Each blue round unit represents an element-wise vector operation. [182]

When propagating for n steps in time, one has to multiply the above terms n times. In the limit of large n, this derivative is not guaranteed to converge to zero or to infinity. This helps preventing the vanishing/exploding gradient problem.

Many variations of the cell architectures have been used, [183] but the LSTM is one of the most popular, alongside the Gated Recurrent Unit (GRU). This is a simplified version of the LSTM cell that achieves similar results with less computational effort. [140]

5.2 RNNs learning molecules

As was mentioned earlier, SMILES are the most used representation of molecules when training RNNs. During training, the RNN expects all data points to have the same number of sequence elements, i.e. all SMILES should have the same legnth. A way of achieving this is to padd all SMILES that are shorter than the longest SMILES in the data set. First, a "go" character (`G`) is appended to the beginning of each SMILES string and an "end" character (`E`) is appended at the end. Then, they are padded with `A` characters to make them all the same length. Once the padded SMILES are all the same length, they are then one-hot encoded. [184] This is necessary as the network requires numerical data as an input, rather than characters. One-hot encoding consists in assigning an index to each character used in the SMILES, and then representing each character as a vector of zeroes where only the element corresponding to its index is 1. For example, if one wanted to one-hot encode the SMILES string of formaldehyde (`C=O`), the '`C`' character would be assigned the index 0, '`=`' the index 1 and '`O`' the index 2. Then, the '`C`' would become a vector [1, 0, 0], '`=`' is [0, 1, 0] and '`O`' [0, 0, 1]. So, `C=O` would be [[1, 0, 0], [0, 1, 0], [0, 0, 1]].

The target SMILES during training are the same as the input SMILES, but they are shifted by one to the left. This is because each unit of the RNN predicts the *next* character in the sequence (Fig. 5.3). So, using the padded SMILES of formaldehyde as an example again, the input would be `GC=OEAAAA` one-hot encoded, while the target SMILES would be `C=OEAAAAA`. This is shown in Fig. 5.3. The cost function used to assess the performance at each time step (J_t) during training is usually the categorical cross-entropy: [185]

$$J_t = -\sum_{i}^{N_{\text{classes}}} y_i^{\text{true}} \log(P(y_i^{\text{pred}})) \tag{5.15}$$

where N_{classes} is the number of classes (which here corresponds to the number of unique characters in the SMILES), y_i^{true} is the target value, and y_i^{pred} is the output of a LSTM cell. To obtain a probability distribution for each class, a Softmax function is applied to the LSTM output ($P(y_i^{\text{pred}})$):

$$P(y_i^{\text{pred}}) = \frac{e^{y_i^{\text{pred}}}}{\sum_{j=1}^{N_{\text{classes}}} e^{y_j^{\text{pred}}}} \tag{5.16}$$

In this case, the softmax gives the probability distribution for what the next SMILES character in the sequence should be. The character that is output is then chosen based on this probability distribution. [86]

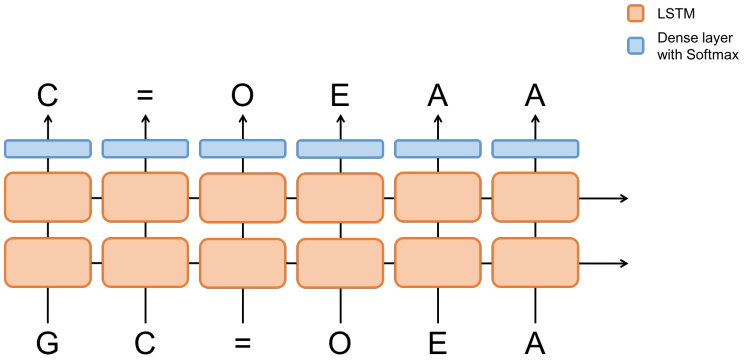


Figure 5.3: Diagram showing the structure of a RNN with 2 layers of LSTM cells and a softmax layer. The input and the output data for the training is shown. For clarity, the SMILES is shown as a string and not as a one-hot encoded vector.

To avoid over-fitting, regularisation can be used. Similarly to the feed forward neural networks used in the first part of this thesis, L1 and L2 regularisations can be used. In addition, a common approach is to use drop-out, which is a technique developed at Google. [186] When training with drop-out, each hidden neuron is randomly omitted with a user defined probability. Usually, at each gradient update

the weights for each hidden neuron are modified to minimise the cost function, but their modification depends also on how all the other hidden neurons are performing. This can lead to co-adaptation, meaning that certain neurons will vary in a way to 'correct' mistakes of other neurons. If each hidden unit cannot rely on other hidden units being present, it needs to perform well in a variety of different contexts. This helps to prevent overfitting. [186]

After the RNN has been trained, it can be used for sampling new SMILES strings. The Softmax can modified by adding a temperature parameter (T, in eq. 5.17).

$$P(y_i) = \frac{e^{y_i/T}}{\sum_{j=1}^{n} e^{y_j/T}} \tag{5.17}$$

The additional temperature parameter T can control whether a more 'explore' or 'exploit' behaviour is used, as increasing T makes the probability of sampling each character more uniform. [86] One can either start by feeding the 'go' character alone, or start from a substring of a SMILES. Once the 'go' character is used as input, the RNN will output a distribution of probabilities for the following character. The character is chosen based on this probability distribution, i.e. it is not always the most likely character that is chosen. The generation ends either when the 'end' character is output or when the maximum length of the string is reached.

5.3 Reinforcement learning

There are many different techniques that come under the umbrella term of 'reinforcement learning'. Here, the focus is on 'policy gradient methods'. In reinforcement learning a policy is a function that takes a state and returns an action. In this case, the RNN is the policy, because it takes in a 'state' (i.e. a fragment of a SMILES string) and returns an 'action' (i.e. the next character to add to the fragment of the SMILES string). The 'action' that the RNN chooses depends on the weights and biases that were optimised during its training on the molecular data sets.

During reinforcement learning, the RNN is used to generate some molecules. The SMILES generated, are scored depending on their desirability. The score is usually called 'reward'. For example, if the goal is to obtain highly polar molecules, then the molecules that are highly polar will receive a high reward, while the others a low one. Then, the weights and the biases of the RNN are modified so that it becomes more likely to sample new SMILES that have higher polarity. [187]

In order to update the parameters of the RNN, two copies of the trained RNN are kept. One is referred to as the 'agent' and the other as the 'prior'. The agent is used to sample a SMILES string.

The probability of the sequence of actions (what sequence of characters is chosen) is the product of the Softmax probability of choosing that character at each time step. For the formaldehyde example `GC=OE`, if we suppose that the Softmax probability of choosing '`C`' after the '`G`' character is 0.8, the probability of choosing '`=`' after '`GC`' is 0.6, for '`O`' after '`GC=`' is 0.65 and for '`E`' after '`GC=O`' is 0.4, then the sequence probability would be $0.8 \times 0.6 \times 0.65 \times 0.4 = 0.1248$.

The agent needs to be modified so that it is more likely to generate sequences that result in high scores. However, it is key that it does not forget the syntax of SMILES strings that it has previously learnt. To attempt solving this problem, an 'augmented log likelihood' $\log P(A)_{\mathbb{U}}$ can be used. [140]

$$\log P(A)_{\mathbb{U}} = \log P(A)_{\text{prior}} + \sigma S(A) \tag{5.18}$$

where $\log P(A)_{\text{prior}}$ is the base e logarithm probability of obtaining the sequence A with the prior, $S(A)$ is the score of the sequence A and σ is a hyper-parameter that weights the importance of the score. The loss function then is:

$$\text{Loss} = \left[\log P(A)_{\text{agent}} - \log P(A)_{\mathbb{U}}\right]^2 \tag{5.19}$$

where $\log P(A)_{\text{agent}}$ is the base e logarithm probability of obtaining the same sequence A with the agent. The loss function is then differentiated with respect to the weights and the biases of the agent RNN, and the parameters are then modified to minimise it.

5.4 Industry collaborations

The work presented in this part of the thesis was done in collaboration with two different companies. Chapter 6 was done while working in collaboration with NovaData Solutions Ltd., a consulting company with expertise in data informatics, computational chemistry and molecular modelling. This chapter focuses on training a LSTM RNN with multiple rounds of fine-tuning to generate Kinase inhibitors. Reinforcement learning is then used to generate molecules with higher activity towards JAK2 proteins. Chapter 7 was done during an internship at GlaxoSmithKline (GSK). In this chapter multiple GRU RNNs are fine-tuned on multiple small data sets with varying properties. The goal was to understand how the data sets affect the performance of RNNs when learning small sets of molecules.

Chapter 6

Generating Kinase inhibitors

6.1 Introduction

In this chapter, RNNs combined with reinforcement learning are applied to generate new Kinase inhibitors. Kinases have received considerable attention in pharmacological research as they play a critical role in cellular signalling. This makes them suitable targets for treating diseases like cancer. [188]

There are a wide variety of Kinases families, among which is the Janus family of Kinases (JAK). This family includes JAK1, JAK2, JAK3 and TYK2, which are inlvolved in mediating protein phosphorylation. This in turn is responsible for controlling cell growth and immunological responses. [189] The JAK2 protein is important for the hematopoietic growth factors, which regulate the differentiation and the proliferation of cells. [190] Mutations in the JAK2 have been linked to development of diseases that eventually progress into myelofibrosis, a type of bone marrow cancer. [190] Due to the large number of patients who remain symptomatic after conventional therapies, the need for novel approaches is high. The success of Kinase inhibitors in the treatment of cancer has led to a great interest from the scientific community. There are currently over 40 Food and Drug Administration (FDA) approved Kinase inhibitors. [188] Among these drugs, many are not specific to one target and have undesirable side effects. Consequently, there is the need to design drugs with cleaner profiles.

This project aims to investigate the ability of RNNs to learn from already existing Kinase inhibitors and generate new ones with improved activity. This work was carried out in collaboration with the company NovaData Solutions ltd., [191] who provided the data sets for learning and analysed the best candidate molecules that the model generated.

6.2 Method

6.2.1 Data sets

The generation of the three data sets used was done by Dr Mike Mazanetz from NovaData Solutions Ltd.

The largest data set contained 597652 molecules from the ChEMBL22 data set. [192] This data set consisted only of drugs tested on various human targets where IC50 values were reported in nM or µM and had an IC50 below 3 µM. Then, the compounds were standardised by removing solvents from the structures, stripping salts by only retaining the largest fragment, clearing any isotopes, aromatising the structures, adding explicit hydrogens and clearing any stereochemistry information.

The second data set contained 29010 Kinase inhibitors that were in ChEMBL database. These were inhibitors to a variety of different Kinases. The smallest data set contained 2091 inhibitors specific to the JAK2 target. For each of the JAK2 inhibitors there was an associated pIC50 value. The molecules in the data sets were represented as SMILES strings.

6.2.2 RNN and reinforcement learning

In order for the RNN to learn the syntax of SMILES strings, it needs to be trained on a large amount of data. To do this, the large ChEMBL data set containing 597652 molecules was used. Of the 597652 samples, 5977 were removed from the training set and used as the validation set.

The RNN used had the same architecture as the one described by Gupta et al. [86]: it was composed of two layers of LSTM cells, each with 256 hidden neurons. The first and second layer were both regularised with drop-out, with drop-out parameters of 0.3 and 0.5 respectively (e.g. a parameter of 0.3 means that in each iteration 30% of the neurons are randomly dropped out). Following the LSTM layer, there was a fully connected layer (a layer where each neuron takes as input the output of *all* the neurons in the previous layer) with a Softmax activation function. The RNN was trained for 22 iterations with a learning rate of 0.001 and the Adam optimiser. [193] The cost function used was the categorical cross-entropy. [185] During training, the temperature parameter in the Softmax was kept to 1 (equation 5.16). This part of the training has the purpose of teaching the RNN the syntax of SMILES for general drug-like molecules. In order for it to learn about Kinase inhibitors, it was fine-tuned twice on smaller data sets containing Kinase inhibitors. Gupta et al. only did one round of fine-tuning, but here it was decided to do two, where the first round was done on a larger set of more

varied Kinase inhibitors and the second one on a more focused set. It was thought that a smoother transition between the ChEMBL and the focused data set may yield a better generative model.

The first round of fine-tuning on the larger set of Kinase inhibitors was done by training for 7 iterations on 27559 Kinase inhibitors and validated on 1451 samples split randomly. After each epoch, 1000 SMILES were generated with a softmax temperature parameter of 0.75 (as this was what was done in the Gupta et al. paper [86]) and the percentage of valid and unique SMILES strings was checked to make sure that the model was not degrading in its ability to generate grammatically valid SMILES strings. The second round of fine-tuning was done in the same way, but using the data set containing 2091 JAK2 inhibitors.

After fine-tuning the model twice, reinforcement learning was used to increase the average pIC50 value of the SMILES generated. Since the pIC50 values cannot easily be calculated from first principles, a neural network model was trained on experimental data to predict pIC50 values of the 2091 JAK2 inhibitors. In order to train this model, the SMILES were transformed to Morgan fingerprints (with a radius of 3) and these were pre-processed by subtracting the mean of each feature. The hyper-parameters of the network were optimised with a random search using three-fold cross validation (training on 1324 samples, testing on 697 samples). The trained model was then used in the reinforcement learning procedure to predict the pIC50 of molecules generated by the RNN.

The procedure used to perform reinforcement learning involves multiple steps. First of all, an 'experience buffer' is populated by generating 60 SMILES and taking the 30 best scoring ones. This experience buffer is a list of tuples containing the SMILES generated by the Agent network at a temperature $T = 0.75$, their sequence log likelihood ($\log P(A)_{\text{agent}}$ in equation 5.19) and the reward that the sequence has received. The reward is calculated by using the feed forward neural network trained on predicting pIC50 values. Once the pIC50 value is predicted, the hyperbolic tangent function is used to get a value between -1 and 1 (eq. 6.1).

$$S(A) = \tanh(\text{pIC50}(A) - 8) \tag{6.1}$$

The value of 8 was used to favour molecules with pIC50 higher than 8. The experience buffer never contained more than 30 samples. Then, the RNN was trained on the SMILES in the experience buffer. During training, the following cost function J was used:

$$J = \left[\log P(A)_{\text{agent}} - \log P(A)_{\mathbb{U}}\right]^2 \tag{6.2}$$

and $\log P(A)_{\mathbb{U}}$ was obtained using $\sigma = 60$, as this was the value used by Olivecrona et al. [140] The minimisation of the cost function was performed using the Adam optimiser, where the gradients were clipped if their magnitude was larger than 3. The learning rate was 0.0001.

After one iteration of training on the SMILES in the buffer, the Agent with modified weights was used to generate 60 more SMILES. The reward for these SMILES was calculated and if it was higher than that of the SMILES already present in the buffer they were replaced. This process was repeated 5 times.

6.2.3 Software details

All the RNNs used were implemented in Python using the Keras API [194] to TensorFlow. [110] Keras was chosen because it enables fast prototyping of different neural network architectures, but also gives access to the TensorFlow computation graph through a 'backend' module. This means that the standard LSTM cells and feed forward neural networks can be easily used, but then any custom modification can easily be made. Here the 'backend' module was used to implement the reinforcement learning part. The architecture of the RNN followed that described by Gupta et al. [86] while the reinforcement learning implementation followed closely that of Olivecrona et al. [140] The difference with the Olivecrona implementations are:

1. The SMILES are one-hot encoded instead of using token indices for the encoding. This means that in the Olivecrona implementations the SMILES are encoded as array of indices rather than array of vectors. The one hot encoding was used because it is present in the Gupta et al. model, which was implemented first.

2. The reinforcement learning scores used here are between -1 and 1, like Olivecrona et al. used in the initial paper. [140] However, when they re-wrote their code after publishing the paper they changed the scores to be between 0 and 1.

3. They check that there are no duplicates in the experience buffer. This is useful but would slow the training down, so was not implemented.

4. Their implementation uses Pytorch instead of TensorFlow. Due to the fact that the RNN was already implemented in TensorFlow, the reinforcement learning was also written in TensorFlow. In addition, Keras does not support PyTorch as the backend.

RDKit [176] was used to assess the validity of SMILES strings and to convert the SMILES to Morgan Fingerprints during the reinforcement learning procedure.

6.3 Results and discussion

6.3.1 Training and fine-tuning

The value of the cost was evaluated on both the training and the validation set as after each training iteration (Fig. 6.1). The cost function smoothly decreases as the number of iterations increases. As can be seen, the cost for the validation set is lower than for the training set. This is counterintuitive, but it is a consequence of a technical aspect of the training procedure as implemented in Keras. During training, the cost function is evaluated for each batch of data points (so that the gradients of the cost function with respect to the weights and biases can be evaluated). At the end of an epoch, the values of the cost function obtained throughout that epoch are averaged. In contrast, the cost function for the validation set is only evaluated at the end of the epoch. Therefore, the model has considerably improved from the first batch of the epoch, which results in a lower validation cost compared to the training cost.

After training on the ChEMBL data set, the RNN was used to generate 1000 SMILES with increasing values of the temperature parameter T in the softmax function (eq. 5.17). Fig. 6.2 shows that as T increases past $T = 0.75$, the percentage of unique SMILES generated remains constant around 95%. On the other hand, the percentage of valid SMILES decreases drastically. This is because as T increases, the probability of sampling each character becomes closer to uniform, and randomly sampling a valid SMILES is unlikely. There is a trade-off between the explore and exploit ability of the model. With a low T, the SMILES sampled will be more similar to those in the training set. With higher T, a wider space is explored but the chances of encountering invalid SMILES also increases. For the rest of this work $T = 0.75$ was used. This is because it gives a high percentage of both unique and valid SMILES.

For the SMILES generated with a temperature of 0.75, 52 molecular properties such as the number of acidic and basic groups, the number of aromatic rings, the number of heavy atoms, the number of Chlorine atoms, the number of double bonds, etc. were calculated for all the SMILES (the full list of calculated properties can be found in Appendix G). A Principal Component Analysis (PCA) [195] was performed to reduce the dimensionality of the data so that the generated molecules and the training set could be compared visually. The properties of the generated molecules were compared to those of the training set by projecting the 52 molecular properties into the space of their first two principal components, as these were found to explain 97% of the variance in the data. The comparison of the properties of the ChEMBL molecules and the generated molecules is shown in Fig. 6.3. The generated molecules lay in the region where the ChEMBL data is most dense. This is expected: the

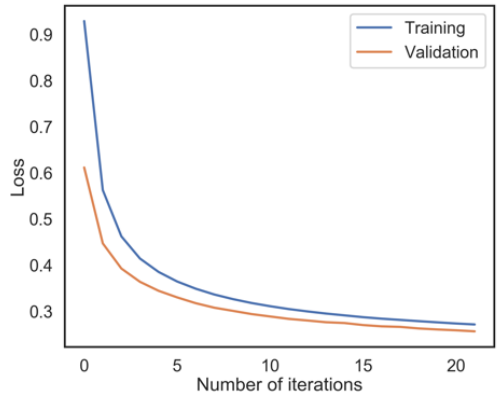


Figure 6.1: Evolution of the loss function during training on the large ChEMBL training set (blue) and on the test set (orange).

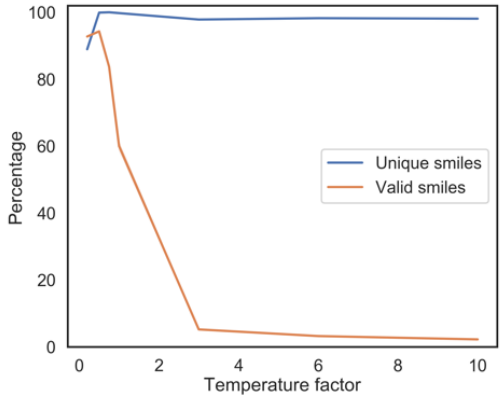


Figure 6.2: Percentage of unique and valid SMILES generated by the RNN after training on the large ChEMBL data set. The model is used to predict 1000 SMILES using different temperature factors in the Softmax function (eq. 5.17).

model learns to generate molecules more similar to molecules that it has seen more often in the training set. However, there are also a number of outliers. These were often found to be molecules with large hydrocarbon chains (Fig. 6.4). The initial data set contains around 4k molecules with carbon chains of at least 8 carbon atoms and around 50k molecules with at least 5. This means that the model has definitely learnt about hydrocarbon chains. During the generation process the next character in the sequence is sampled from a distribution of characters. Consequently, there is a small probability that the model generates molecules with some long carbon chains, longer than those observed in the training set.

After training on the large ChEMBL data set, the RNN was fine-tuned on 29010 Kinase inhibitors for 7 iterations. After each epoch, 1000 SMILES were generated and the percentage of valid and unique SMILES strings was evaluated to make sure that the RNN was not forgetting the SMILES syntax that it had learnt on the large ChEMBL data set. The results in Fig. 6.5 show that the number of unique SMILES generated remains close to 100%, but the number of valid SMILES is lower than after the training on the large ChEMBL data set. This could be due to the fact that the SMILES in the Kinase inhibitors data set are quite different from those in the ChEMBL data set and the weights in the RNN are being modified too drastically. This causes the RNN to 'forget' some of the rules that it had learnt from the large ChEMBL data set and hence produce fewer valid SMILES. The too rapid change of the weights may be improved in the future by reducing the learning rate or clipping the gradients. The model after 6 iterations was chosen as the model to use for further fine-tuning on the JAK2 inhibitors. This is because the number of valid SMILES has started increasing again, but there is no improvement

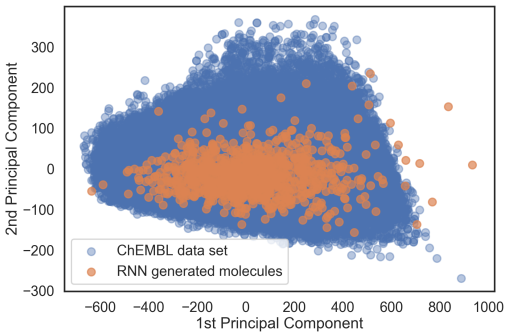


Figure 6.3: Comparison of the properties of the molecules in the large ChEMBL data set (blue) and the properties of the molecules generated by the RNN after training (orange). The properties are projected onto the first two principal components of the training data.

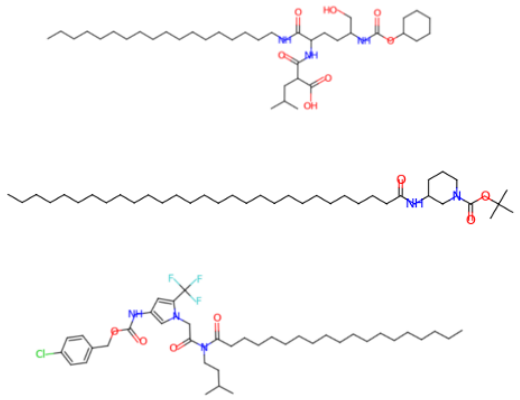


Figure 6.4: Examples of RNN generated molecules that do not overlap with the ChEMBL data set space.

after 7 iterations. After each epoch in the second round of fine-tuning 1000 SMILES were generated to check the evolution of the percentage of valid and unique SMILES (Fig. 6.6). The percentage of valid SMILES is worse than after the first round of fine-tuning. It does, however, increase with the number of iterations, but it does not reach 50%. Since the model after 7 iterations had the highest number of valid SMILES it was used for reinforcement learning.

Both the model after the first and second round of fine-tuning were used to generate 1000 SMILES. The same 52 molecular properties mentioned earlier were calculated for both the generated molecules and the molecules in the JAK2 data set. PCA was performed again on the JAK2 data and the properties of the generated molecules were projected on the new basis. The comparison of the properties in the space of the two principal components is shown in Fig. 6.7 for the molecules generated before fine-tuning, after 1 round of fine-tuning and after 2-rounds of fine-tuning. In this basis, it looks that even before fine-tuning the overlap between the generated molecules and the JAK2 data set is quite high, although there is a large number of outliers (Pannel A of Fig. 6.7). Pannel B in Fig. 6.7 shows that after fine-tuning on the large Kinase data set, the distribution of generated molecules overlaps only in part with the JAK2 molecules and mostly causes a reduction in the number of outliers. After fine-tuning on the JAK2 the overlap increases, but the number of outliers increases again. Even though the RNN generates fewer valid SMILES after the second round of fine-tuning, the molecules that are generated tend to have more similar properties to those in the JAK2 set.

Then, the molecules that corresponds to the outliers in pannel C of Fig. 6.7 were analysed and are

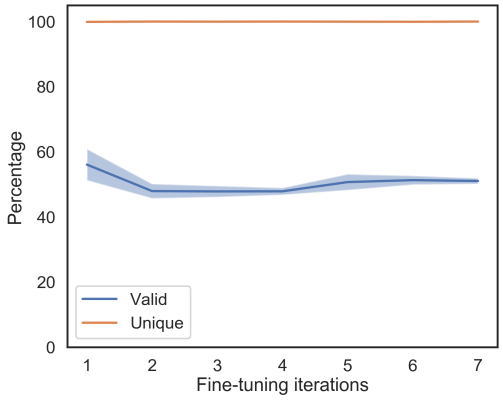


Figure 6.5: First round of fine-tuning on the data set containing 29010 Kinase inhibitors. The evolution of the percentage of valid and unique SMILES is checked on 1000 predictions after each iteration of fine-tuning.

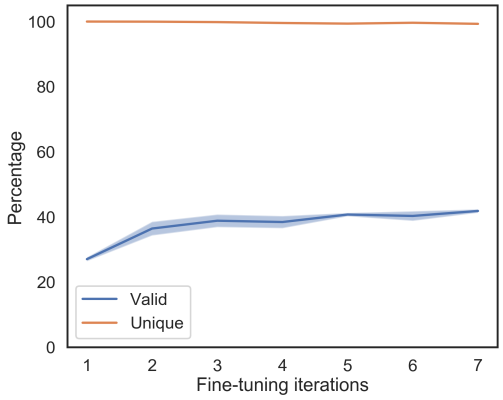


Figure 6.6: Second round of fine-tuning on the data set containing 2091 JAK2 inhibitors. The evolution of the percentage of valid and unique SMILES is checked on 1000 predictions after each iteration of fine-tuning.

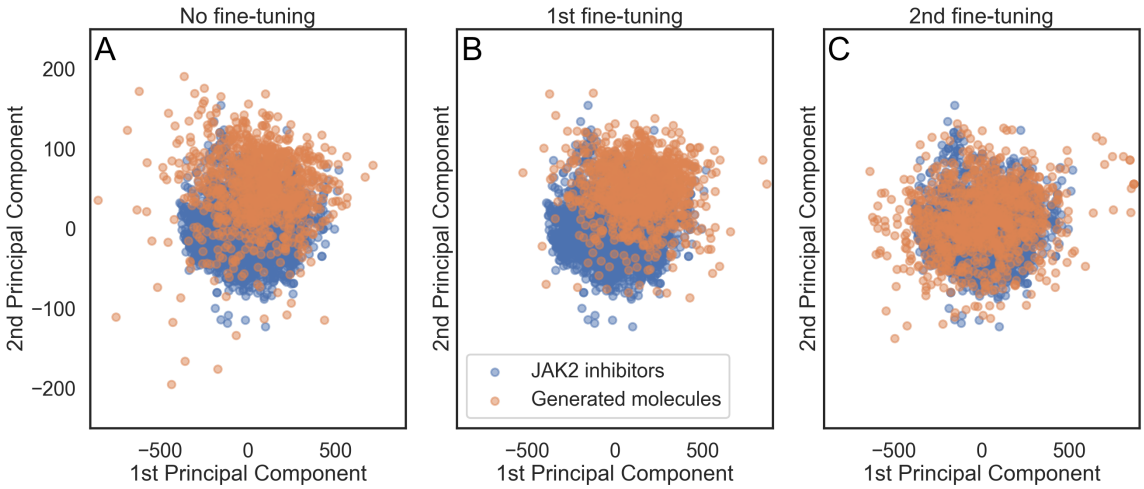


Figure 6.7: Comparison of the properties of the molecules in the JAK2 data set (blue) and those of the molecules generated after training on the large ChEMBL data set (A), the first (B) and second (C) round of fine-tuning (orange).

shown in Fig. 6.8. As can be seen from the figure, the outliers with large negative values of the 1st principal component tend to have large molecular weights, while the outliers with large positive values of the 1st principal component tend to have low molecular weights. Upon further analysis of the molecular weight (Fig. 6.9), one can see that the distribution of molecules generated by the RNN matches quite well that of the JAK2 fine-tuning set, but it is wider. Its tails reach further, so there are molecules that are generated with higher and lower molecular weight compared to what is found

in the fine-tuning set.

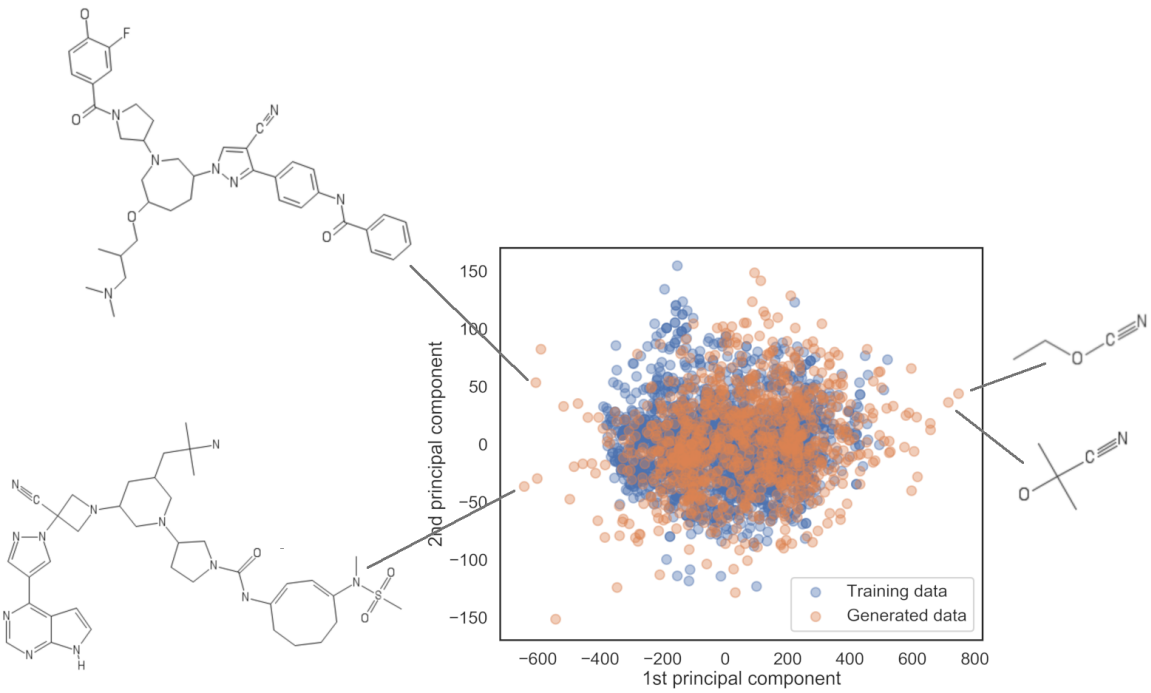


Figure 6.8: Analysis of the outliers generated by the RNN after the second round of fine-tuning.

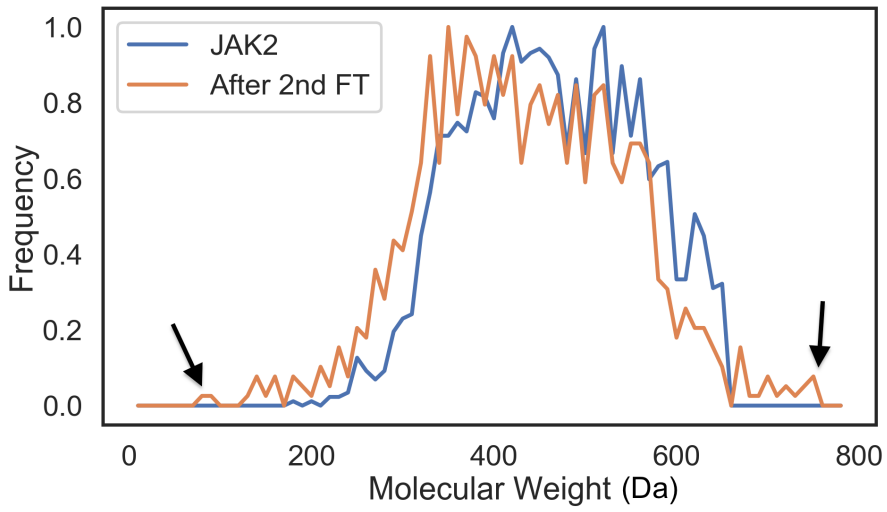


Figure 6.9: Analysis of the molecular weight of molecules generated by the RNN after the second round of fine-tuning. These are compared with the molecular weight of the molecules in the JAK2 fine-tuning set. The arrows point out that there are molecules genrated by the RNN that are heavier and lighter compared to those found in the JAK2 set.

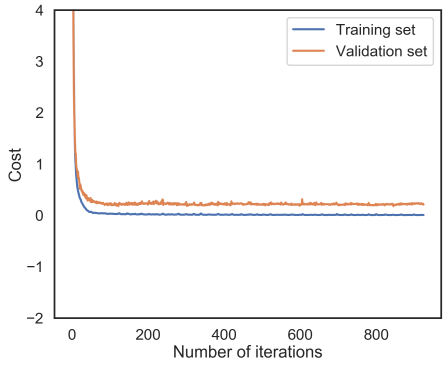


Figure 6.10: Value of the cost function during the training of the model learning pIC50 values for JAK2 inhibitors. The cost function was evaluated on the training and validation set.

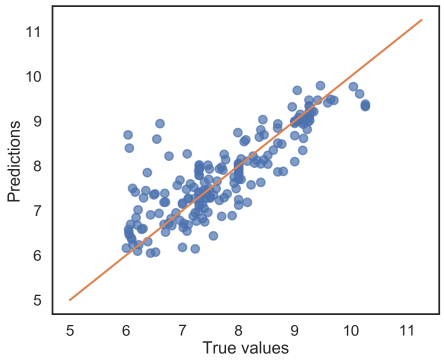


Figure 6.11: Correlation plot for prediction of the pIC50 values of the test set, containing 209 JAK2 molecules. The R^2 score [139] was 0.68.

6.3.2 Reinforcement learning

The next step was to use reinforcement learning to try and shift the distribution of generated molecules to a region with higher pIC50 values. A feed forward neural network was trained to predict the pIC50 values of the JAK2 inhibitors. The best model obtained during hyper-parameter optimisation had a cross-validation MAE of 0.41 M. The best hyper-parameters were: 2 hidden layers with 393 and 21 hidden neurons respectively, L1 regularisation parameter of 4.7×10^{-8}, L2 regularisation parameter of 1.7×10^{-5}, a learning rate of 3.9×10^{-4}, a batch size of 19 and 925 iterations. Then, the model was trained again on 1788 samples and tested on 209 to see how the performance increases with increasing data set size. The MAE decreased to 0.38 M. The final model was trained on 1986 data points and tested on 105, to try and maximise the amount of data available. The evolution of the training and test loss as well as the correlation plot for the predictions are shown in Fig. 6.10 and Fig. 6.11 respectively. The validation error does not reach values as low as the training loss and the correlation plot shows that there are a few outliers with error larger than 2 M. However, it is unknown what accuracy is necessary for reinforcement learning, so a MAE around 0.4 M seemed like an acceptable starting point.

Then, the RNN was modified using the reinforcement learning procedure described in section 6.2.2. Fig. 6.12 shows a comparison of the distributions of pIC50 values for the molecules generated after the two rounds of fine-tuning and after reinforcement learning. It is evident that the reinforcement learning encourages the model to generate molecules with higher predicted pIC50 values. However, the reinforcement learning only has the objective of maximising the score based on the predicted pIC50, which means that the molecules could be modified in unexpected ways to maximise this score if no other constraints are enforced. This could result in more molecules that are unsuitable for medicinal

chemistry purposes.

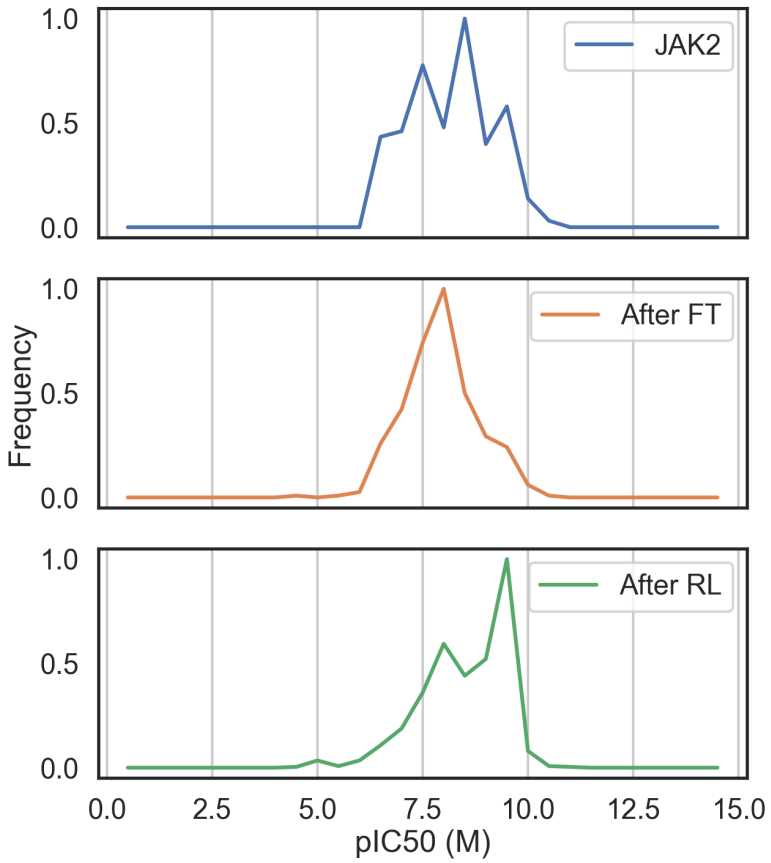


Figure 6.12: Comparison of the predicted pIC50 values for the molecules generated by the model after fine-tuning (FT) and after the reinforcement learning (RL) with the experimental pIC50 values of the molecules in the JAK2 data set.

6.4 Conclusions and further work

In this chapter, the ability of RNNs to generate new JAK2 inhibitors was investigated. This was done by first training on a large ChEMBL data set and then performing two rounds of fine-tuning on more focused data sets. Afterwards, the RNN was modified further using reinforcement learning to increase the number of generated molecules with higher pIC50 on JAK2.

The results of fine-tuning showed that the RNN can learn the distribution of smaller data sets, but there is a drastic decrease of the number of valid SMILES generated. The reason for this is not entirely clear. It could be due to the fact that the SMILES in the smaller data sets are quite different from

those in the ChEMBL data set, and the network is forgetting about all the variety of molecules that it had learnt. In the process, it also forgets some of the rules required for valid SMILES. This aspect requires further investigation. In the next chapter, the fine-tuning process will be investigated in more details. Multiple data sets containing molecules with a varying degree of similarity to each other and to the initial training set will be used. This will enable to understand how the distribution of RNN-generated molecules changes depending on the fine-tuning data set. In addition, a more systematic study of the number of data points required to perform fine-tuning will be carried out.

The reinforcement learning procedure showed that the RNN can be modified to produce more high-scoring molecules. However, further analysis is required to understand whether the molecules generated change in the expected way or they 'hack' the scoring process and obtain high rewards without developing the features required to be potential drug candidates.

Chapter 7

Fine-tuning recurrent neural networks

In the previous chapter, the ability of RNNs to generate JAK2 inhibitors was investigated. The RNN was first trained on a large data set of SMILES strings and then fine-tuned on a small data set of molecules with a desired set of properties. It was observed that after fine-tuning, the ability of the RNN to generate valid SMILES had significantly deteriorated: after being trained on the large data set, the RNN generated valid SMILES with 95% probability, while after fine-tuning on the small data set only about 50% of the generated SMILES were syntactically valid. So far, RNNs fine-tuning benchmarks have been mostly focusing on large data sets. [166, 167] Understanding how RNNs can be fine-tuned on small data sets is of key importance for medicinal chemistry, where often only small data sets of molecules (10^2 to 10^3 samples) with desired properties are available. This will be investigated in this chapter by fine-tuning on several data sets of different sizes and with different degrees of molecular diversity.

The work in this chapter was carried out in collaboration with GlaxoSmithKline (GSK). GSK gathered data sets and cleaned the data, while I focused on training the RNNs and analysing the results.

7.1 Method

7.1.1 Data sets

All of the data sets used in this chapter were pruned by Dr Peter Pogány from GSK.

The data set for the initial RNN training was a subset of ChEMBL23 [152], obtained by removing any compounds with bad valence or with poorly defined bonds, with any isotope labelling and any element other than N, O, C, S, F, Cl, Br and I.

When pruning medicinal chemistry data sets, it is common to apply the Lipinski filter. [196] This means that any compound with more than 5 hydrogen bond donors, 10 hydrogen bond acceptors, a molecular mass of more than 500 Da and an octanol-water partition coefficient ($\log P$) larger than 5 would be removed. However, some medicinal data sets do contain molecules with molecular weight higher than 500 Da. To make sure the RNN learns about larger molecules too, here only molecules with molecular weight higher than 650 Da were removed and the other components of the Lipinski filter were not applied.

The molecules were standardized using ChemAxon JChem toolkit Standardiser [197] and InChIs [198] were calculated to make sure there were no duplicate structures. This left with 1363545 compounds.

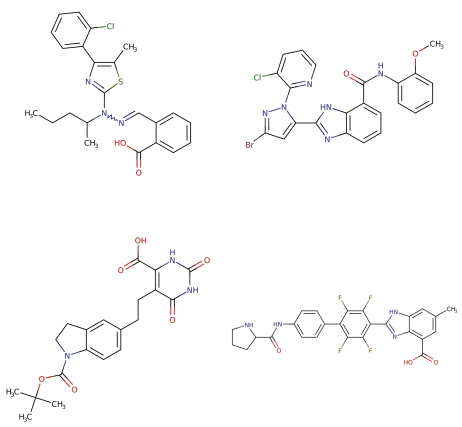
Then, the REOS filter was applied. The REOS (Rapid Elimination Of Swill) filter flags any molecule with ligands that have been recognised in the literature to have non-drug-like functionalities. [199] This reduced the number of structures to 1204109.

The fine-tuning data sets included a GSK data set (MMP12 [200]), 6 ChEMBL data sets (DHODH, METAP2, PLD1, SLC9A1, SLC22A12, P2X7) and 8 patent data sets (US-20090018134-A1, US-20090286778-A1, US-20100016279-A1, US-20120157425-A1, WO-2010079443-A1, WO-2011075515-A1, WO-2012053186-A1, WO-2012067965-A1). The patent data sets were chosen by first downloading a list of available bioassays with human data from SureChEMBL. [201] Then, this list was submitted to the GoStar database [202] to download only compounds with associated pIC50. The structures were deduplicated and only those patent data sets with at least 800 compounds were kept. A small description of all the data sets is reported below, and a few structures from each data set are shown in Fig. 7.1-7.3 (the image files shown in these figures were obtained from Dr Peter Pogány).

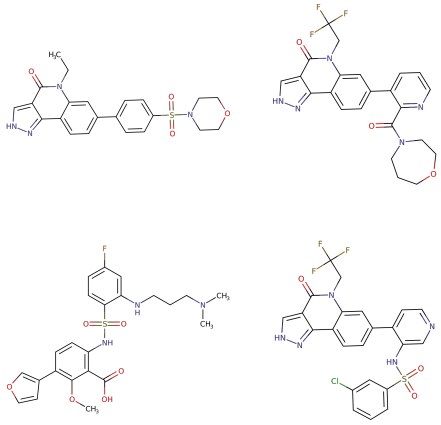
1. DHODH: contains inhibitors to the Dihydroorotate Dehydrogenase, an enzyme required to produce DNA and RNA. [203]

2. METAP2: contains inhibitors of the Methionine Aminopeptidase 2, a protein that removes the N-terminal methionine from nascent proteins. Initially they were developed as anti-cancer agents, but since they induced considerable weight-loss they were also investigated for treating obesity. [204]

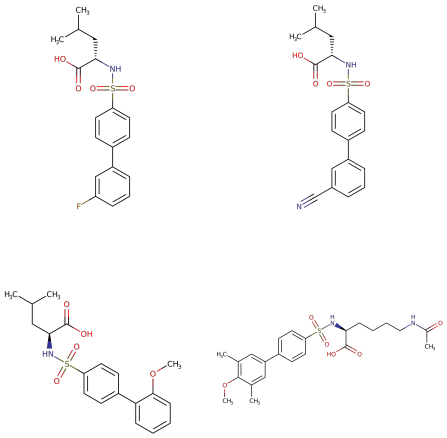
3. PLD1: contains inhibitors of Phospholipase D1, a protein involved in numerous cellular pathways,



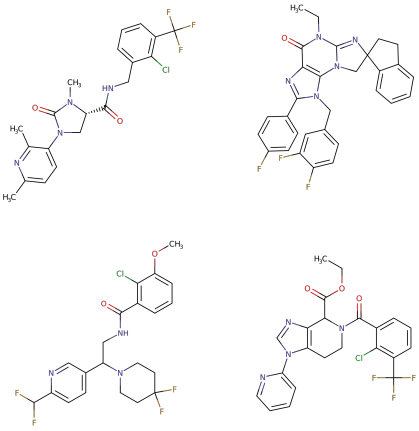
(a) DHODH



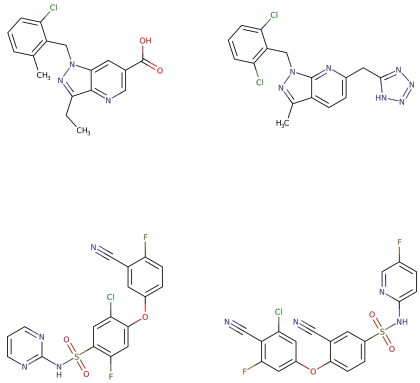
(b) METAP2



(c) MMP12

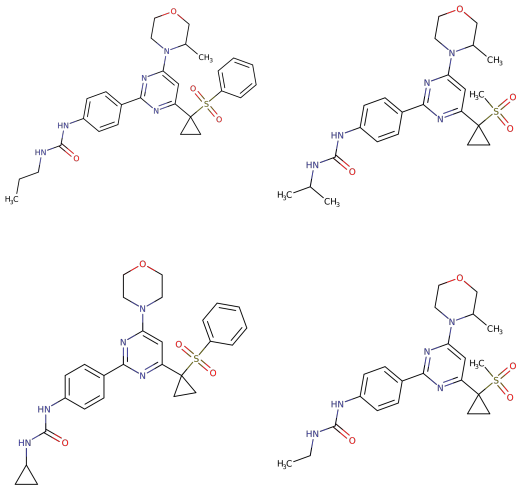


(d) P2X7

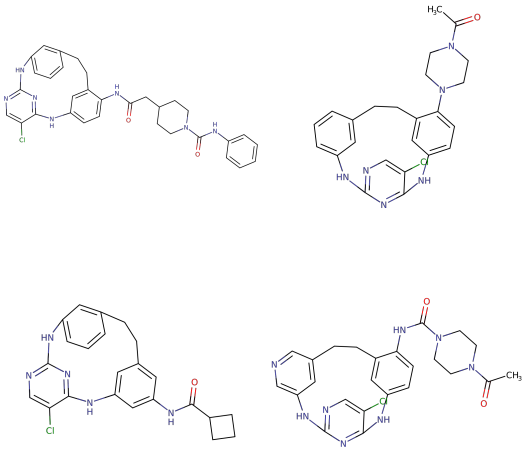


(e) SLC22A12

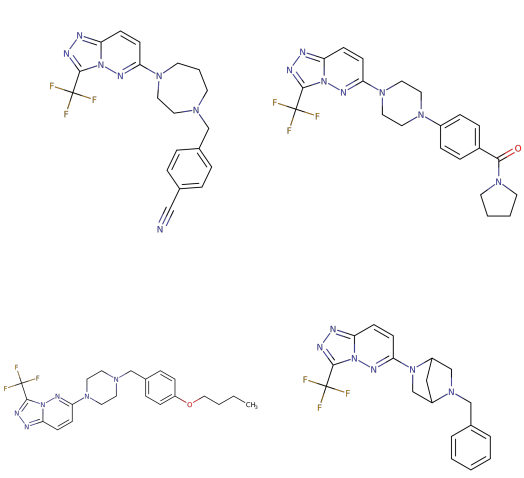
Figure 7.1: Representative structures for the full ChEMBL and GSK medicinal chemistry data sets used for transfer learning.



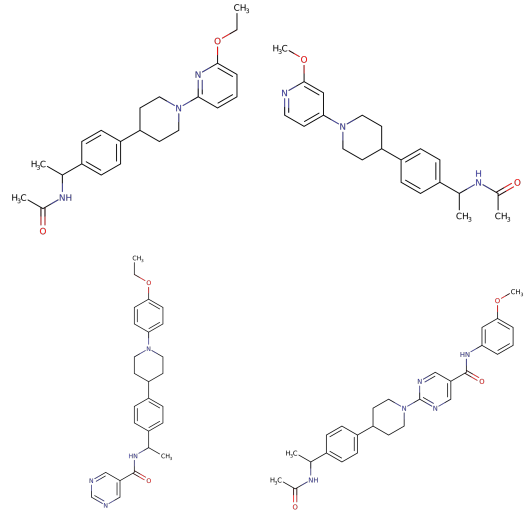
(a) US-20090018134-A1



(b) US-20090286778-A1

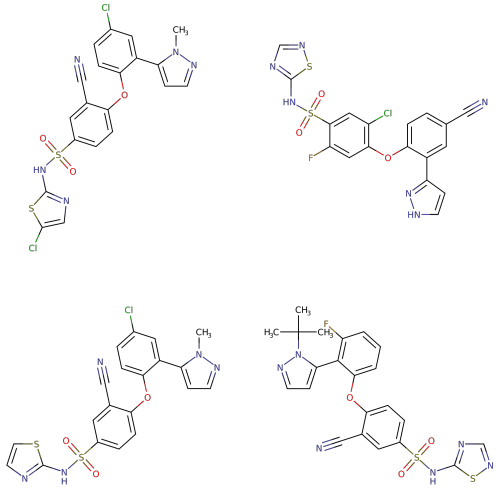


(c) US-20100016279-A1

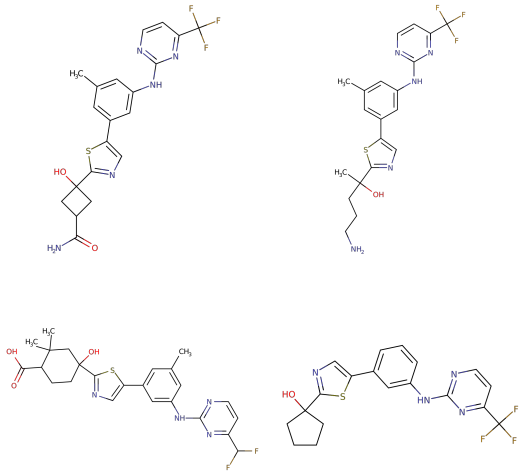


(d) US-20120157425-A1

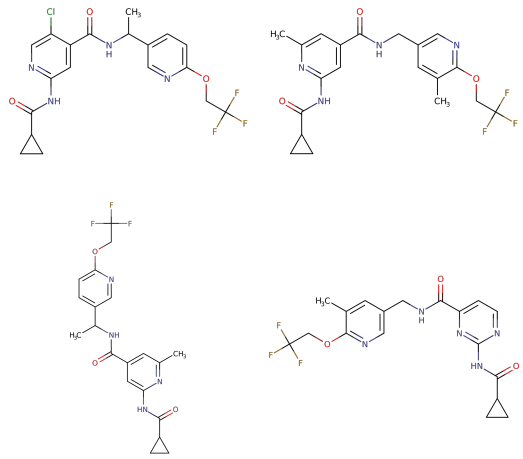
Figure 7.2: Representative structures for the medicinal chemistry data sets coming from the patents with identifier starting with 'US' used for transfer learning.



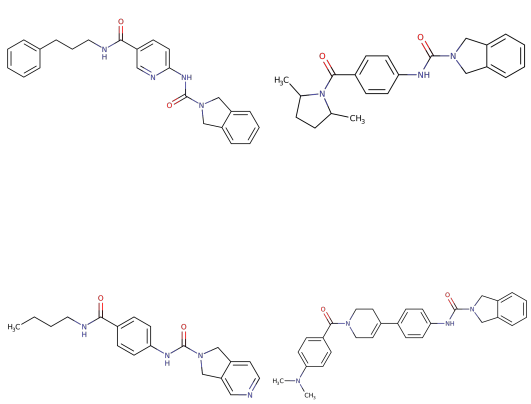
(a) WO-2010079443-A1



(b) WO-2011075515-A1



(c) WO-2012053186-A1



(d) WO-2012067965-A1

Figure 7.3: Representative structures from the patents with identifier starting with 'WO' medicinal chemistry data sets used for transfer learning.

including signal transduction, membrane trafficking, and the regulation of mitosis. [205] It may play a role in neurodegenerative diseases. [206]

4. SLC9A1: contains inhibitors to the sodium/hydrogen exchanger 1, which is a membrane protein involved in the regulation of cell pH and volume. They have been investigated as potential cancer drugs. [207]

5. SLC22A12: contains drugs affecting the solute carrier family 22, member 12. This is a protein that regulates the levels of urate in the blood and is primarily found in the kidneys. [208]

6. P2X7: contains inhibitors to the P2X purinoceptor 7, which is a receptor for adenosine triphosphate (ATP). It acts as a gate of ion channel and is responsible for the formation of membrane pores permeable to large molecules. [209]

7. MMP12: contains molecules affecting the matrix metalloproteinase-12, which is a protein involved in the breakdown of the extracellular matrix. [210]

8. US-20090018134-A1: This data set contains compounds containing at least two heterocycles. In each heterocycle, there is at least one nitrogen or oxygen atom as the hetero-atom. These compounds target proliferating diseases such as cancer. [211]

9. US-20090286778-A1: This data set contains kinase inhibitors targeting inflammatory and autoimmune disorders, as well as cancer. It contains compounds where six membered rings are joined together to form a large macrocycle. [212]

10. US-20100016279-A1: These compounds have been developed for the prevention or treatment of androgen-receptor associated conditions. The data set contains heterocyclic compounds with at least one system of at least two heterocycles fused together (sharing a bond) or both sharing a bond with a common ring only containing carbon atoms. [213]

11. US-20120157425-A1: This data set contains heterocyclic compounds with hydrogenated pyridine rings. These are not fused with other rings and do not have double bonds between ring members or between ring members and non-ring members ligand atoms. These compounds target the inhibition of acetyl-CoA carboxylase enzyme, which is involved in the metabolism of lipids and therefore is related to diseases including diabetes, dyslipidemia, hypertension, and cardiovascular diseases. [214]

12. WO-2010079443-A1: This data set contains sulfonamide derivatives that are used for the treatment of pain. These are heterocyclic compounds containing at least two hetero rings, where at least one ring has nitrogen and sulfur atoms. [215]

13. WO-2011075515-A1: The molecules covered by this patent include pyrimidine amines which are potent inhibitors of spleen tyrosine kinase. These are useful in the treatment and prevention of diseases such as asthma and rheumatoid arthritis. There are many spiro compounds, which are compounds where 2 or more rings are linked together by one common atom. It also contains compounds with many rings joined together, i.e. where rings share a bond, as well as heterocycles. [216]

14. WO-2012053186-A1: This data set contains heterocyclic compounds with two or more rings where the only hetero atom is a nitrogen. These compounds are involved in the treatment or prevention of disorders and diseases in which voltage gated sodium channels are involved. [217]

15. WO-2012067965-A1: This data set also contains heterocyclic compounds with nitrogen as the hetero atom. These molecules act on the NAMPT enzyme, which has the role of phosphorylating molecules in many physiologically essential processes. [218]

The patent data sets contain many macrocycles and systems of fused rings. These were used because some of the molecular generators used at GSK in the past had struggled in learning compounds containing these features (unpublished results). Consequently, they were used here to see if the RNN can learn these compounds.

For the MMP12 and the ChEMBL data sets (except PLD1 and SLC9A1), reduced versions were created. The reduced versions of the data sets were obtained using the Bemis-Murko framework [219] implemented in GSK in-house software. [220] In this framework, side chains and hetero-atoms are removed from the molecules in the data base, so that one is just left with their 'scaffold'. Then, a variety of methods can be used to measure the similarity between the scaffolds and cluster them. Here the similarity metric used was the Tanimoto similarity. The Tanimoto similarity is one of the most common similarity metrics for comparing molecular fingerprints. It is a value between 0 and 1, where 1 corresponds to identical molecules. If two molecules have N_a and N_b bits set in their fingerprints, where N_c is the number of common bits, then the Tanimoto similarity coefficient is: [221]

$$C_{\text{Tanimoto}} = \frac{N_c}{N_a + N_b - N_c} \tag{7.1}$$

Here, ECFPs were used to represent the molecules. The ECFPs had 1024 bits and a radius of 2 (referred to as ECFP_4, where '4' is the effective diameter of the largest feature). [175] The clustering algorithm used was the sphere exclusion clustering. [222] The clustering works by first calculating the Tanimoto similarity between the molecules in the data set. Then, the list of molecules is sorted by placing first the molecules that have more 'similar molecules' (i.e. a Tanimoto similarity higher than a

cut-off, here 0.8). Then, the clustering algorithm begins by considering the first molecule in the list a centroid for the first cluster. All molecules with a Tanimoto similarity above the cut-off are considered part of this cluster. All molecules that are part of the cluster are flagged and can no longer be picked by the algorithm to become centroids. After the first cluster has been populated, the next molecule in the list that is not already part of the cluster is picked to be the next centroid. The process is then repeated. By the end, the molecules that have not been assigned to a cluster become 'singletons'. Here the largest 2-4 clusters were kept, depending on the size of the initial data set.

Table 7.1: Characteristics of the data sets used for fine-tuning the RNNs. The empty cells represent missing data.

data set	**Size**	**Number of clusters**	**Self-similarity**	**Similarity to ChEMBL**
MMP12 full	2500	86	0.581	0.780
DHODH full	505	90	0.394	0.821
METAP2 full	516	68	0.357	0.781
PLD1 full	126	22	0.500	-
SLC9A1 full	251	38	0.349	-
SLC22A12 full	340	45	0.373	0.892
P2X7 full	2480	236	0.416	0.739
MMP12 reduced	890	3	0.613	0.815
DHODH reduced	256	4	0.428	0.775
METAP2 reduced	281	2	0.647	0.785
SLC22A12 reduced	247	4	0.543	0.931
P2X7 reduced	814	3	0.424	0.753
US-20100016279-A1	817	17	0.695	0.791
WO-2012053186-A1	1281	9	0.685	0.722
US-20090286778-A1	887	21	0.666	0.706
US-20120157425-A1	1025	24	0.561	0.741
WO-2012067965-A1	1417	28	0.557	0.729
US-20090018134-A1	905	19	0.849	0.945
WO-2011075515-A1	1391	23	0.755	0.851
WO-2010079443-A1	1029	21	0.632	0.726

Table 7.1 shows the total size of all the data sets and the total number of clusters in them. It also shows the 'median self-similarity' of each data set. This value is calculated by creating an $N \times N$ matrix M (where N is the number of compounds in a data set), where each element m_{ij} of the matrix is the Tanimoto similarity (on $ECFP_4$) between compound i and j. Then, the median of this matrix is taken. A low value indicates high diversity of the molecules in the data set, whereas higher values indicate low diversity (the molecules are more similar to each other). The number of clusters and the

median self-similarity of each data set is visualised in Fig. 7.4 to 7.6. For each data set, the clusters present are shown as circles of different size, where the radius of the circle represents the size of the cluster and the colour represents the median self-similarity. Data sets that are coloured in red are more varied than data sets coloured in blue. Table 7.1 also shows a value of the similarity to the initial large ChEMBL data set. This was evaluated by calculating the similarity of each compound in a data set and all the compounds in the large ChEMBL data set. The smallest value was retained and this process was repeated for each molecule in the small data set. At the end, the median of these similarity values was taken.

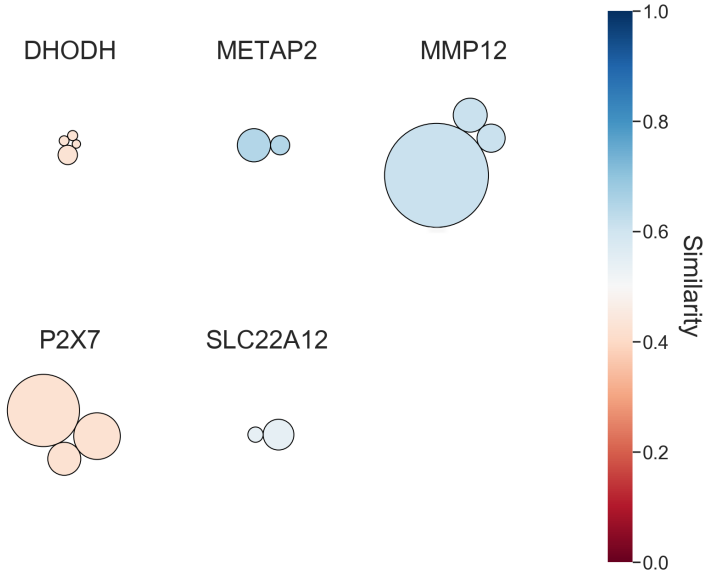


Figure 7.4: Visualisation of the size of the clusters present in the reduced ChEMBL data sets. The radius of each circle corresponds to the size of the cluster, while the colour shows the self-similarity of that data set.

7.1.2 Training and fine-tuning RNNs

Here the software used for training the RNN was that developed by Olivecrona et al. [140] Their Python software uses Gated Recurrent Units (GRU, Fig. 7.7) in each layer instead of LSTMs. As discussed in the introduction, these follow the same principles as the LSTM cells, but are more computationally efficient.

The model was trained for 5 epochs on the ChEMBL23 data set with 1204109 compounds with a batch size of 32, using the Adam optimiser with the default parameters. The initial learning rate was 0.001

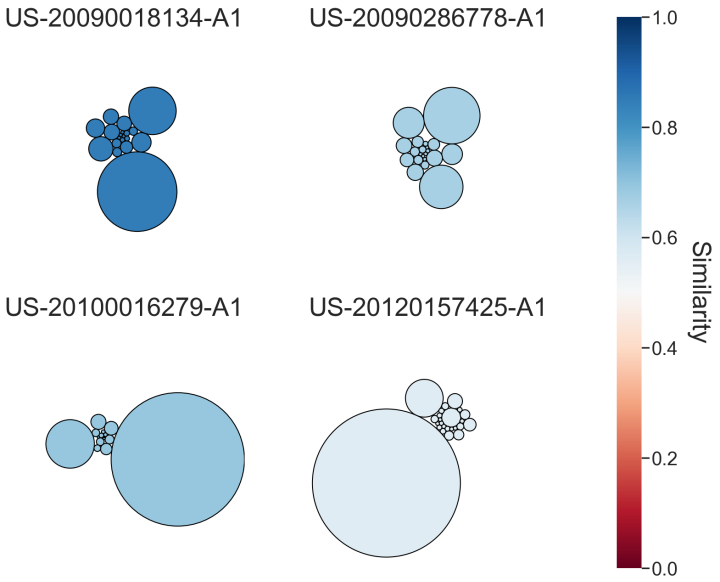


Figure 7.5: Visualisation of the size of the clusters present in the US patents data sets. Each cluster contains similar drug molecules. The radius of each circle corresponds to the size of the cluster, while the colour shows the self-similarity of that data set.

and there was a 0.03 learning rate decay every 500 steps, i.e. the rate decreased by 3% every 500 steps. A batch size of 32 was chosen (instead of 128 like in the Olivecrona publication [140]) in order to be able to study the fine-tuning on the small data sets. The cost function used during training was the categorical cross-entropy (equation 5.15).

After training the RNN on the ChEMBL23 data set, it was fine-tuned on the small data sets. In order to understand what is the smallest data set that can be used for fine-tuning, subsets of increasing size were created for each small data set. The data points in the subsets were sampled randomly. The smallest number of samples used was 32, then 64 and then increasing by 64 up to when all of the data set is used or up to 512. After 512, the subsets increase by 128 until 1417 is reached. Each training experiment was run 5 times using a different random selection of samples, in order to gain an understanding of the variance in the results.

The RNN was trained for 5, 10, 15 and 20 epochs on each subset for each of the small data sets. After fine-tuning, 4000 SMILES were generated by the RNN. Five different metrics were used to assess the results:

1. Validity: the percentage of valid SMILES among the 4000 sampled. This was assessed using the

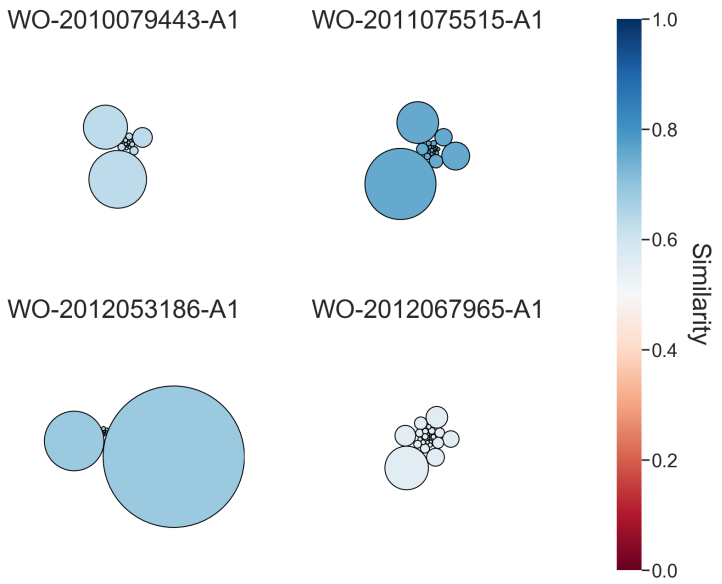


Figure 7.6: Visualisation of the size of the clusters present in the WO patents data sets. Each cluster contains similar drug molecules. The radius of each circle corresponds to the size of the cluster, while the colour shows the self-similarity of that data set.

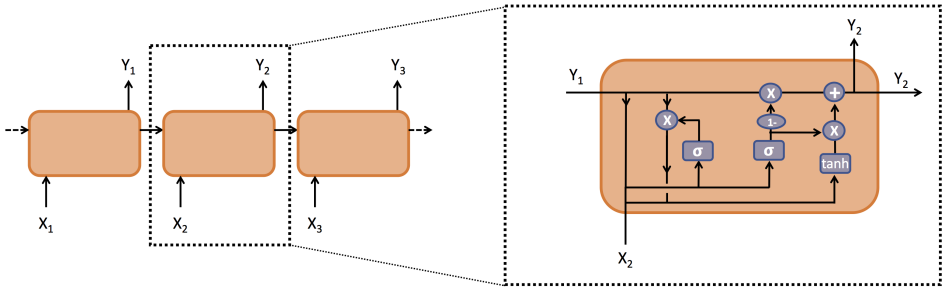


Figure 7.7: Diagram of a GRU cell. Each blue rectangular unit represents a layer of the RNN with either a sigmoid (σ) or a hyperbolic tangent (tanh) activation function. Each blue round unit represents an element-wise vector operation. [182]

RDKit Python package, [176] with the `rdkit.Chem` module.

2. Uniqueness: the percentage of unique SMILES among the 4000 sampled.

3. Novelty: the percentage of SMILES that were not present in the fine-tuning (sub)set on which the RNN was trained.

4. Frechet ChemNet Score [223]: this is a measure of the similarity between the distribution of

generated data and the distribution of the molecules in the training set. It uses the activations of the penultimate layer of the ChemNet model as the representation for the molecules. [224] ChemNet is a deep convolutional neural network for chemical property prediction (properties such as toxicity, activity on a particular target or the solvation free energy of molecules [224]) and takes as input 2D molecular diagrams in the form of an 80x80 image. The image has 4 colour channels where each atom and bond pixel is assigned a 'colour' based on its properties, such as atomic number, partial charge, valence, hybridization, etc. [224] Luckily, the ChemNet implementation automatically converts SMILES to the image representations, so SMILES could be used as an input. From the images, the convolutional part of ChemNet learns a representation for the molecule and then uses it to predict its properties. Taking the penultimate layer of a trained ChemNet corresponds to taking the representation that it has learnt internally. Here, it was used to compare how similar the molecules are to each other.

To do this, the mean m and the covariance matrix $\mathbf{C}$ of each feature (each element of the representation vector) were calculated. Then, the Frechet Distance was obtained as follows:

$$\text{Frechet Distance} = ||\mathbf{m}_{\text{ref}} - \mathbf{m}_{\text{gen}}||_2^2 + \text{Tr}\left(\mathbf{C}_{\text{ref}} + \mathbf{C}_{\text{gen}} - 2(\mathbf{C}_{\text{ref}}\mathbf{C}_{\text{gen}})^{1/2}\right) \tag{7.2}$$

where $\mathbf{m}_{\text{ref}}$ and $\mathbf{m}_{\text{gen}}$ are the vectors of the mean values for each feature of the reference data set and the generated molecules respectively, $\mathbf{C}_{\text{ref}}$ and $\mathbf{C}_{\text{gen}}$ are the covariance matrices for the reference data set and the generated molecules respectively, $||...||_2$ is the euclidean norm of a vector and Tr is the trace of a matrix. From the Frechet ChemNet distance, the Frechet ChemNet Score was calculated in the same way as in the GuacaMol paper: [166]

$$\text{Frechet Score} = e^{-0.2 \cdot \text{Frechet Distance}} \tag{7.3}$$

5. Kullback-Leibler (KL) score [225, 166]: this is a measure of the divergence between the distributions of the physicochemical properties of the generated molecules and the training set. The properties used were: BertzCT (an index of molecular complexity) [226], $\log P$, the molecular weight, the topological polar surface area, the number of H-bonds acceptors, the number of H-bonds donors, the number of rotable bonds, the number of aliphatic rings and the number of aromatic rings. KL divergence has its roots in information theory, where one of the primary goals is to quantify how much information is in data. [227] The KL divergence gives a measure of how much information is lost when approximating a distribution $p(x)$ with another distribution

$q(x)$:

$$D_{\mathrm{KL}}(p, q) = \sum_{i=1}^{N} p(x_i) \cdot \log \frac{p(x_i)}{q(x_i)} \tag{7.4}$$

The larger the KL divergence D_{KL}, the more information is lost. In this case, the larger D_{KL}, the larger the difference between the reference and the generated distributions of molecules. The KL score is calculated from the KL divergence as in the GuacaMol paper: [166]

$$\text{KL score} = e^{-D_{\mathrm{KL}}} \tag{7.5}$$

7.2 Results and discussion

7.2.1 Training the RNN

Training the RNN on the large ChEMBL data set took about 1.5 h and it generated 90.6% valid SMILES. Then, fine-tuning was performed on subsets of increasing size of the full MMP12, full DHODH, full METAP2, full PLD1, full SLC9A1, full SLC22A12, full P2X7, reduced METAP2, reduced MMP12, reduced DHODH, reduced SLC22A12, reduced P2X7, US-20100016279-A1, WO-2012053186-A1, US-20090286778-A1, US-20120157425-A1, WO-2012067965-A1, US-20090018134-A1, WO-2011075515-A1 and WO-2010079443-A1 data sets. Each training was repeated five times. In total, this took about 33 h.

After generating molecules with each trained model, the five metrics described in the Method section (section 7.1.2) were evaluated. The results are presented below.

Validity

For most RNNs, the percentage of valid SMILES generated after fine-tuning on 32 samples for 5 epochs was above 78%. The only exception was the RNN fine-tuned on the US-20090286778 data set, where the percentage of generted valid SMILES was close to 50%.

The reason behind this is likely to be that US-20090286778 is the data set that is the least similar to the ChEMBL training set (see Table 7.1). In US-20090286778, all compounds contain a non-peptidic macrocycle. However, the ChEMBL data set contains only 160 macrocycles (8 membered rings or larger) and almost all of them are cyclic peptides. Consequently, the RNN undergoing fine-tuning on US-20090286778 has to learn considerably more chemistry than the RNNs fine-tuned on the other data

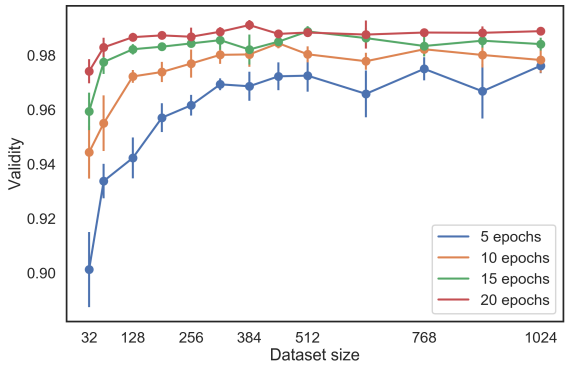


Figure 7.8: The validity of the generated SMILES after fine-tuning on US-20090018134-A1.

sets.

The second lowest performing RNN is the one trained on WO-2011075515. After 5 epochs on 32 data points it generates fewer than 80% valid SMILES. This data set has a much higher similarity to the initial ChEMBL data set (0.851) compared to US-20090286778 (0.706). However, this data set contains a large number of structures with quaternary carbon atoms, spiro and bridged compounds. The initial ChEMBL data set contains around 17000 spiro compounds and 20000 bridged compounds (out of around 1.2 million compounds). Due to the small percentage of these compounds in the initial ChEMBL data set, the RNN cannot learn these compounds well. Therefore, the effect is similar to the RNN trained on US-20090286778, where during fine-tuning the RNN has to learn considerably different chemistry, so the initial validity is low.

In general, the percentage of generated valid SMILES is greater than 85% for all RNNs trained for at least 10 epochs on data sets with more than 192 data points.

Uniqueness and Novelty

The uniqueness and novelty metric show different trends compared to the validity metric, as they do not increase with data set size and number of epochs. To summarise, the following general trends were observed:

- The uniqueness of the generated SMILES first decreases and then increases with data set size and decreases with the number of epochs (Fig. 7.9).
- The novelty of the generated SMILES decreases with both data set size and the number of epochs (Fig. 7.10).

Fig. 7.9 and 7.10 show that as the data set size increases past 256 data points, the number of novel SMILES decreases and the number of unique SMILES increases. This is because with larger data sets the RNN learns to generate more molecules similar to the fine-tuning set without repeating the same few SMILES many times.

On the other hand, the trend of the uniqueness for data set sizes below 256 is not as intuitive. It could be explained as follows. With low data set size and many epochs, the model generates only a few unique SMILES. It is possible that the RNN takes the most common characters and makes sequences out of them, as this is a good way of minimising the cost function when not much data is available. The initial decrease in Fig. 7.9 can be due to the fact that the model was previously trained on the large ChEMBL data set and was generating almost only unique SMILES. As the RNN is trained on the small data sets, the weights are being modified towards generating 'the most likely sequences' which result in low values of the cost function. If few epochs are performed, the weights are not modified as much, so the number of unique SMILES decreases more slowly. As the data set size increases the number of unique SMILES generated increases, because the model starts to learn the distribution of the data set rather than generating only the most likely sequences.

The uniqueness is therefore related to both the number of gradient updates that are performed on the parameters (weights and biases inside the RNN) and the number of samples available. The uniqueness of the molecules generated when training on a small data set goes to zero as the number of gradient updates increases, but it increases as the data set size increases. Consequently, the combination of these two factors result in the trend shown in Fig. 7.9, where there is a minimum for small data set sizes with a certain number of gradient updates.

With low data set sizes the novelty of the generated SMILES is high, because few of the molecules generated are identical to those in the training set. As the size of the data set increases, more molecules already in the training set start being generated, hence the decrease in the novelty metrics (Fig. 7.10). This is in agreement with the idea that as the data set size increases, the RNN generates a distribution increasingly similar to the training set. To confirm this, the Frechet ChemNet score has to be analysed.

Frechet ChemNet Score

The Frechet ChemNet score is shown as a function of data set size in Fig. 7.11 and 7.12 for data sets WO-2012067965-A1 and WO-2010079443-A1 respectively. This shows the trend that is seen for all data sets: the Frechet ChemNet score increases with data set size until it reaches a plateau. Higher values of the Frechet ChemNet score mean that the distributions of the data set and the generated

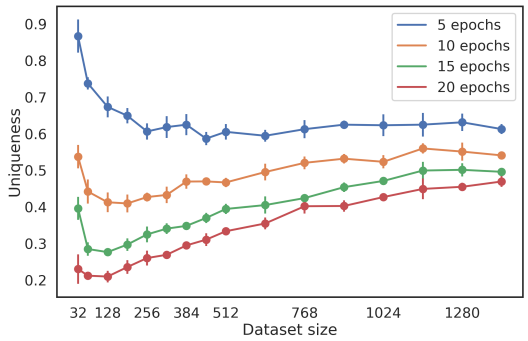


Figure 7.9: Evolution of the fraction of unique SMILES generated with the WO-2012067965-A1 data set as the fine-tuning set.

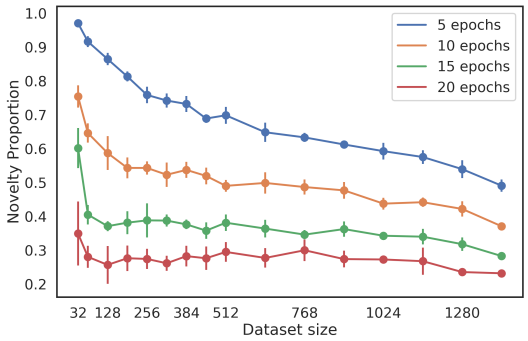


Figure 7.10: Evolution of the fraction of novel SMILES generated with the WO-2012067965-A1 data set as the fine-tuning set.

molecules are more similar. The original authors of the Frechet ChemNet distance [223] considered that a Frechet ChemNet distance of 1.62 (corresponding to a Frechet ChemNet score of 0.723) was a good level of similarity between two molecular sets.

Usually, fine-tuning for more epochs also increases the similarity of the distributions, but only up to a certain point. For most data sets, doing more than 15 epochs is unnecessary because the Frechet ChemNet score does not improve considerably. For some data sets, there is no considerable difference even between 10 and 15 epochs (for example WO-2012067965-A1, as can be seen in Fig. 7.11).

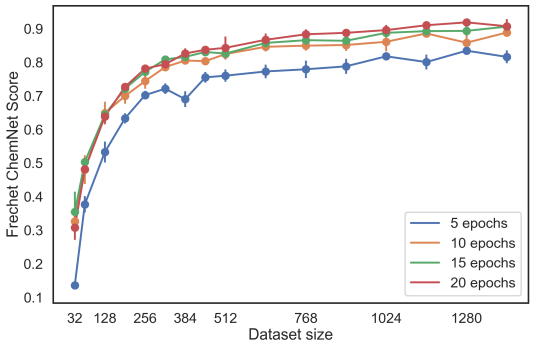


Figure 7.11: Evolution of the Frechet ChemNet Score between the WO-2012067965-A1 data set and the molecules generated by the RNN trained on it for different numbers of epochs.

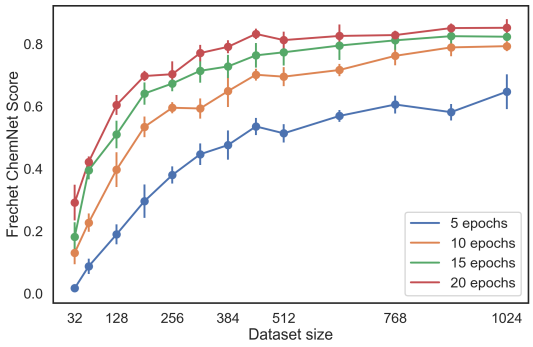


Figure 7.12: Evolution of the Frechet ChemNet Score between the WO-2010079443-A1 data set and the molecules generated by the RNN trained on it for different numbers of epochs.

Another general observation that can be made is that for most data set, around 300 samples and at least 10 epochs are needed to reach a Frechet ChemNet score of 0.8. There are a few exceptions:

1. WO-2010079443-A1: with 320 samples and 10 epochs, the Frechet ChemNet score is 0.71 (Fig. 7.12).

2. Full DHODH: with 320 samples and 10 epochs, the Frechet ChemNet score is 0.77.

3. Full P2X7: with 320 samples and 10 epochs, the Frechet ChemNet score is 0.71.

Understanding these exceptions may give a better understanding of what makes a data set easier or harder to learn with a RNN. It is interesting to notice that these exceptions are not the same as for the validity, where the RNN was struggling most with the US-20090286778 and WO-2011075515 data sets. Here the RNN generates high percentages of valid SMILES, but the distribution of the generated molecules is not as similar to the fine-tuning set.

One factor that makes comparisons difficult is that the size and number of the clusters in the data sets vary considerably, as well as the similarity of the molecules in each cluster. One would expect that a data set with many clusters and a low similarity would make it harder to reach a high Frechet ChemNet score. However, WO-2010079443-A1 has fewer clusters and is less diverse than WO-2012067965-A1, but yet it takes longer to reach a Frechet ChemNet score of 0.8 (Fig. 7.11 and 7.12). This seems to suggest that the molecular diversity and the number of clusters are not the best predictor for the Frechet ChemNet score.

To get a better understanding of this, the Frechet ChemNet score was evaluated after fine-tuning on 256 samples for 10 epochs. It was then plotted as a function of the self-similarity of each data set (Fig. 7.13). From this data, there does not appear to be a strong correlation between the Tanimoto self-similarity and the Frechet ChemNet score. The R^2 is 0.15 and the coefficient is 0.16. Consequently, there must be other factors at play.

It was thought that maybe the complexity of the SMILES strings themselves could affect the Frechet ChemNet score. In this case, SMILES strings that are more 'complex' are considered to be those that are longer, the ones that contain a high number of unique characters and the ones with more branches and rings.

Consequently, for each data set the average/standard deviation of the SMILES length (i.e. how many characters there are in each SMILES), the number of unique characters and of the number of characters that need a 'pair' were calculated. The characters that need a pair are brackets (both curly and square brackets) and numbers that are used to represent rings. The Frechet ChemNet score (after 10 epochs on 256 data points) was plotted as a function of each of these properties to observe visually whether there is a correlation between them. A linear regression model was fitted to evaluate more quantitatively the correlation between the Frechet ChemNet score and each of these variables (Fig. 7.14).

As can be seen from Fig. 7.14, none of the properties appeared to be correlated to the Frechet ChemNet score. The average value of the SMILES length does not change considerably between the different

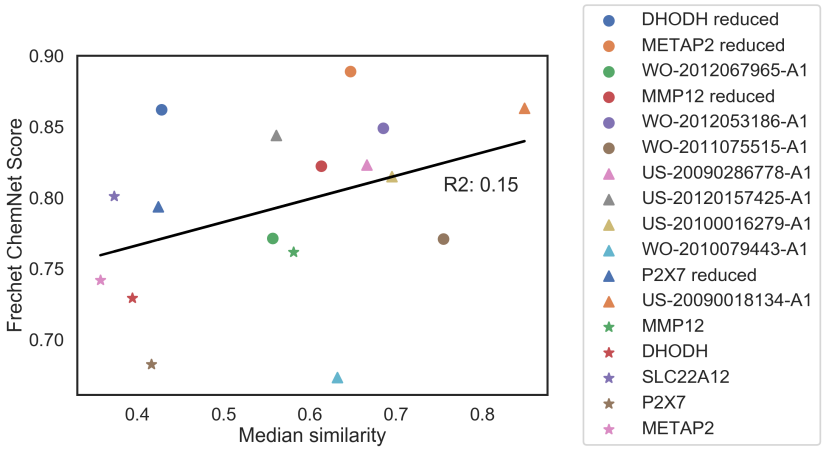
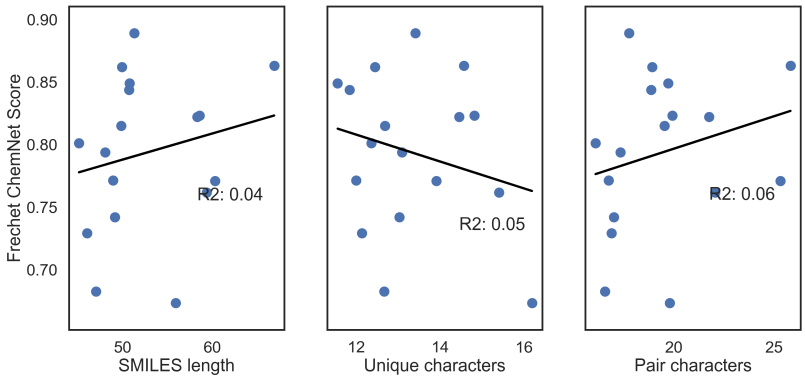


Figure 7.13: Frechet ChemNet score after 10 epochs on 256 data points as a function of the median Tanimoto similarity of the $ECFP_4$ fingerprints. The linear fit shows that there is no correlation between the two.

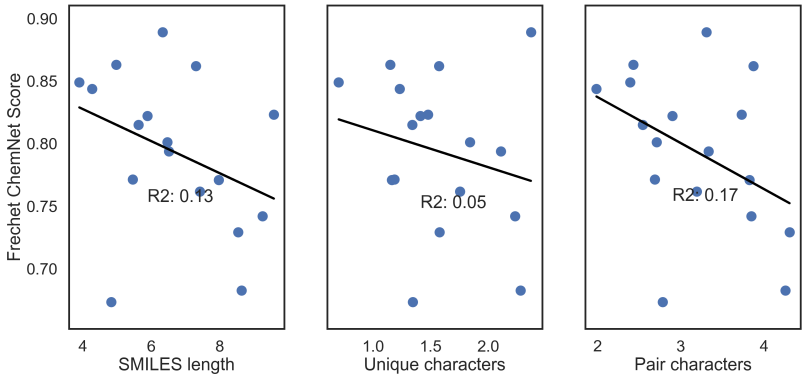
data sets and data sets with a mean SMILES length around 50 can have a Frechet ChemNet score below 0.7 and above 0.85. Similar trends are seen for the other properties. The largest R^2 observed is only 0.17. This can be rationalised by realising that an increased complexity of the SMILES should mostly affect the validity of the generated SMILES rather than the Frechet ChemNet score. Since the trends observed for the validity and the Frechet ChemNet score are different, the complexity should not be a good predictor of the Frechet ChemNet score.

Another factor that was investigated was a correlation between the similarity of the molecules in the initial large ChEMBL data set and the fine-tuning sets. However, this was again found not to be a good predictor of the Frechet ChemNet score.

In conclusion, from this analysis it remains unclear what affects the Frechet ChemNet score and why certain data sets reach higher values quicker than others. An additional investigation that could help to understand what is happening would be to calculate a large number of physicochemical properties for the fine-tuning data sets and the generated molecules. Then, a Principal Component Analysis (PCA) could be carried out and the results could be projected into the space of the largest 2 principal components (similarly to chapter 6). This would enable to visualise the similarity between the distributions in this space, and compare with the Frechet ChemNet scores.



(a) Average of the properties



(b) Standard deviation of the properties

Figure 7.14: Frechet ChemNet score after 10 epochs on 256 data points as a function of different properties for all the data sets. The properties are the SMILES length, of the number of unique characters and of the number of characters that need a 'pair'.

KL divergence score

The KL divergence score follows very simial trends to the Frechet ChemNet scores, but the differences between the data sets are not as large (Fig. 7.15 and 7.16).

Consequently, the KL score is not so informative about the overall performance of the RNN, but it confirms that the generated SMILES have similar physicochemical properties compared to the molecules in the fine-tuning data sets.

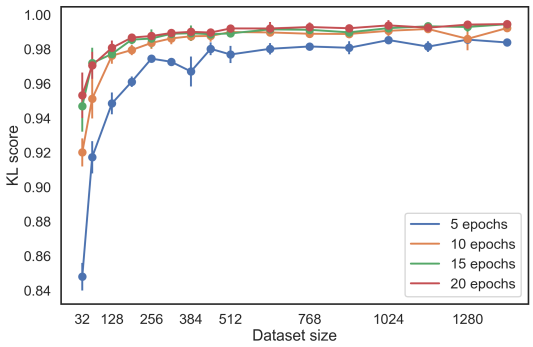


Figure 7.15: Evolution of the KL divergence Score between the WO-2012067965-A1 data set and the molecules generated by the RNN trained on it for different numbers of epochs.

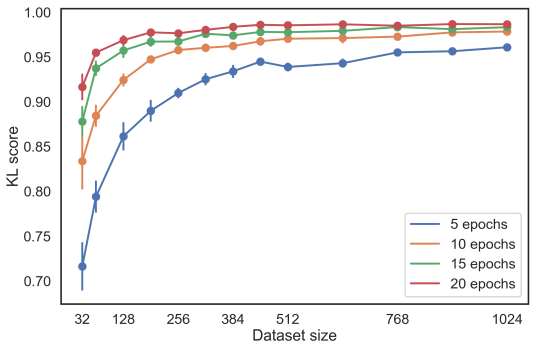


Figure 7.16: Evolution of the KL divergence Score between the WO-2010079443-A1 data set and the molecules generated by the RNN trained on it for different numbers of epochs.

7.2.2 General observation on fine-tuning

From the metrics analysed in the previous sections, it is clear that the number of fine-tuning epochs that had to be performed in order to reach high value of the Frechet score varied considerably depending on the data set. In general terms, the following fine-tuning categories were encountered:

1. **Dataset size below 200:**

   In this case, fine-tuning is difficult and the results depend considerably on the data set. It requires careful filtering of the molecules, so that the diversity is as low as possible. A large number of fine-tuning epochs are needed to learn the distribution of the data set. However, with this data set size increasing the number of epochs causes the novelty and uniqueness to decrease, which means that a high number of SMILES have to be generated in order to obtain a reasonable number of viable drug candidates. To get an idea of how many SMILES to generate in this case, one could first generate 4000 SMILES and check the percentage of unique SMILES and the percentage of novel SMILES among the unique ones. Then, depending on how many new drug candidates one wants (N_{dc}), the number of molecules to generate (N_{g}) would be obtained as:

   $$N_{\mathrm{g}} = \frac{N_{\mathrm{dc}}}{N_{\mathrm{nu}}} \tag{7.6}$$

   where N_{nu} is the fraction of novel and unique SMILES.

2. **Dataset size between 200-400:**

   This case also necessitates a high number of epochs. However, with this data set size using a larger number of epochs does not have as drastic an effect on the percentage of unique and novel

SMILES generated. This means that fewer SMILES can be generated. This number can still be estimated in the same way as for the previous case.

3. **Dataset size above 400:**
   Here between few epochs can be sufficient to reach a Frechet ChemNet score above 0.7. Since the number of epochs needed is lower, the percentage of SMILES that need to be generated is lower, as the percentage of novel and unique SMILES will be much higher than in the previous cases.

7.3 Conclusions

This chapter investigated in more details the fine-tuning of RNNs on small data sets. A great variety of data sets were used, including GSK in house data, ChEMBL data sets and data sets from patents. Five different metrics were used to assess the performance of the RNNs: the percentage of unique, valid and novel molecules, the Frechet ChemNet score and the Kullback-Leibler (KL) score.

These metrics were found to evolve differently with respect to the data set size and the number of fine-tuning epochs performed. For example, the novelty decreases with data set size and number of epochs, the uniqueness decreases with the number of epochs, but shows first a decrease with data set size and then an increase. The Frechet ChemNet score increases with data set size and number of epochs until it reaches a plateau.

Consequently, the number of fine-tuning epochs has to be chosen carefully depending on the data set size. Depending on the number of fine-tuning epochs chosen and the number of data points available, the number of molecules that need to be generated varies. This is because the percentage of unique and novel SMILES produced is dependent on the number of gradient updates performed during training.

It has to be noted that there are a multitude of other hyper-parameters that have not been investigated in this study. For example, the effect of the learning rate, regularisation parameters or the batch size. The effect of all these hyper-parameters would be useful to investigate in the future.

In addition, it would be interesting to investigate further what causes the Frechet ChemNet score to increase at different rates for different data sets. As said in the previous section, something that could be done would be to calculate a large number of physicochemical properties for the fine-tuning data sets and the generated molecules and then perform PCA. Then, the results could be projected into the space of the largest 2 principal components and plotted (similarly to chapter 6). This would enable to

visualise the similarity between the distributions in this space, and see if it sheds light on the trends of the Frechet ChemNet scores.

General conclusions

In this thesis, two types of neural networks were used to tackle two different chemical problems. In the first part, it was investigated how atomic neural networks and Atom Centred Symmetry Functions (ACSFs) can be used to fit the reactive potential energy surface of large organic systems. In-house software was developed and applied to fit the potential energy surface of squalane reacting with a cyano radical. In the second part, recurrent neural networks were used as molecular generators for medicinal chemistry projects. The training consisted of a two-parts procedure, which involved a fine-tuning step (also known as transfer learning). From both these projects, a few conclusions can be drawn.

First of all, as it is well documented in the machine learning literature [14, 77], it was observed that the performance of machine learning algorithms strongly depends on the quality of the data set. The construction of data sets is far from trivial and requires careful thought.

In the case of fitting potential energy surfaces, multiple choices had to be made when generating the data set. One question to address was how to efficiently sample the relevant regions of configuration space. Here, an innovative approach that exploits human intuition to accelerate sampling was used: real-time ab initio interactive molecular dynamics in virtual reality (iMD-VR). This method allows users to quickly sample geometries close to the minimum energy reaction pathway. [3] This sampling technique was compared with a more traditional one: constrained molecular dynamics (CMD). iMD-VR and CMD yielded different distributions of samples and, unsurprisingly, each neural network was found to perform better in regions of configuration space where the density of data points was higher.

The pruning of data set also influences the performance of neural networks. For example, in chapter 7, in order to bias the neural network to generate molecules pertinent to specific medicinal chemistry projects, the initial data sets were reduced by applying multiple chemical filters (e.g. molecular weight filters, PAINS filters, etc.). When pruning the raw data, there can be a multitude of steps that are required to obtain the final data sets, and keeping track of them can be difficult. This could be

facilitated by the introduction of standardised software solutions to keep track of large data sets and their changes over time. This would help improving the reproducibility of the work in this field.

Secondly, another hurdle when training neural networks is hyper-parameter optimisation. This is not only crucial to the performance of neural networks, but it is also extremely computationally expensive. Neural networks have to be trained multiple times with different combinations of hyper-parameters, and the number of possible combinations increases exponentially with the number of variable hyper-parameters. Consequently, since a brute-force approach is usually computationally unfeasible, the optimisation requires careful thought. In this thesis, all hyper-parameters were optimised using an open source Python package called Osprey. [114] This program offers many useful features, among which the use of SQL data bases to store the results, the compatibility with Scikit-learn [113] interfaces and the fact that it can train multiple neural networks in parallel. However, further development of hyper-parameter optimisation software will significantly facilitate this critical step in the training pipeline.

Despite the above mentioned hurdles, it is evident that machine learning algorithms can be used to tackle chemical problems of practical interest. Here it was showed how the potential energy surface of large reactive systems can be accurately and efficiently calculated using neural networks. Furthermore, general guidelines were given on how to best train recurrent neural networks to generate molecules for medicinal chemistry projects. In the near future, these techniques will become part of the standard computational chemist tool-box. However, until then, the software solutions will need to mature, as currently considerable domain expertise is required to use them. While it is encouraging that the number of open source implementations is increasing, they are not always reliable. Research in this field will benefit considerably from a stronger collaboration between software engineers and research scientists.

Contributions

The contributions described in this thesis have led to the following publications. Some of these have not yet been submitted at the time of writing.

- S. Amabilino, L. A. Bratholm, S. J. Bennie, A. C. Vaucher, M. Reiher, D. R. Glowacki, 'Training neural nets to learn reactive potential energy surfaces using interactive quantum chemistry in virtual reality', J. Phys. Chem. A 2019, 123, 20, 4486-4499.

  Summary of contribution:
  This publication describes the work presented in chapter 3, where two data sets of isopentane reacting with CN were generated. The first one was generated by SA using interactive molecular dynamics in virtual reality (iMD-VR), while LAB generated the second one using constrained molecular dynamics (CMD). SJB and M. B. O'Connor collaborated with ACV and MR to implement PM6 in iMD-VR. After generating the data sets, LAB used a Gaussian process to optimise the hyper-parameters and SA trained two neural networks on the CMD and iMD-VR data sets. The NNs implementation was developed by SA with help from LAB. The NNs performances were then compared and were found to be of comparable quality, with the NN trained on the CMD data performing better on structures of high energy and the NN trained on iMD-VR data performing better on the structures close to the minimum energy path. This contribution showed that iMD-VR enables to sample meaningful structures along the minimum energy path of a reaction. SA wrote the first draft of the paper, with subsequent help from LAB, M. B. O'Connor and DRG.

- S. Amabilino, L. A. Bratholm, S. J. Bennie, M. B. O'Connor, D. R. Glowacki, 'Training atomic neural networks using fragment-based data generated in virtual reality', *in preparation.*

  Summary of contribution:
  This manuscript describes the work presented in chapter 4. Six data sets containing different hydrocarbons of increasing size (from methane to hexane) reacting with CN were generated using iMD-VR. The implementation of iMD-VR was the same as the one in the previous publication. SA optimised the hyper-parameters and trained six NNs on each of the six data sets. The implementation of the NNs was the same as the one in the previous publication. The NNs obtained were then used to predict the energy of squalane reacting with CN. The NNs obtained were found to be transferable, as long as they were trained on data sets containing at least some isopentane. This confirmed that as long as the key features of a system are well represented in a data set, having the full system in the data set is not necessary. SA wrote the first draft of the paper, with subsequent help from DRG.

- S. Amabilino, P. Pogány, S. D. Pickett, D. Green, 'Guidelines for RNN based transfer learning molecular generation of focussed libraries', *in preparation.*

  Summary of contribution:
  This manuscript describes the work presented in chapter 7. RNNs were trained using a fine-tuning procedure on a large variety of data sets. The data sets were collected and pruned by

PP. The software used was adapted by SA from the work of Olivecrona et al. [140] SA and PP analysed the results together while at GSK, but PP performed further analysis using GSK software after SA finished her internship. SA generated most of the scripts used to analyse and visualise the results. The results showed that when few data points are available, like at the beginning of a drug design project, it is difficult to train RNNs to generate new molecules. SA and PP worked together on the draft, with subsequent help from SDP and DG.

In addition, I also briefly worked on boxed molecular dynamics [10], as the initial plan was to used it for enhanced sampling of the CN + hydrocardon reactions. However, iMD-VR was used instead, because its implementation became available before that of boxed molecular dynamics. Nevertheless, the work done on boxed molecular dynamics was later included in a publication:

- R. J. Shannon, S. Amabilino, M. B. O'Connor, D. V. Shalashilin, D. R. Glowacki, 'Adaptively accelerating reactive molecular dynamics using boxed molecular dynamics in energy space', J. Chem. Theory Comput. 2018, 14, 9, 4541-4552.

  Summary of contribution:
  This publication describes the use of boxed molecular dynamics in energy space to accelerate rare event sampling without specifying a particular reaction coordinate. Both RJS and SA worked on an implementation of boxed molecular dynamics in energy space in NVT (canonical ensemble), with help from MBO. RJS performed the accelerated sampling of chemical reactions and found that the discovery of reactions was several orders of magnitude faster compared to unbiased molecular dynamics, but the ratios of products formed were similar. RJS wrote the first draft of the paper, with subsequent help from DRG.

Appendix A

Example of ACSFs for a toy system

The toy system of choice to explain how the ACSFs are constructed is NH_2Cl. The following coordinates are used:

	x	y	z
N	0	0	0
Cl	1	0	0
$H1$	0	1	0
$H2$	0	0	1

The angles are unrealistic, but they make the example clearer. The first step is to construct the two-body terms. For clarity, only two values for the R_s parameter are used: $R_s = [0, 1]$, and one value for η ($\eta = 1$) is used. The dimension of the tensor containing the two-body term is $[N_{\text{samples}}, N_{\text{atoms}}, N_{\text{elements}} \times N_{R_s}]$, where N_{samples} is the number of configurations, N_{atoms} is the number of atoms, N_{elements} is the number of elements and N_{R_s} is the number of R_s values. Since in this systems there are 3 element types and 2 values of R_s, the representation tensor will have dimensions $[1, 4, 6]$. The cut-off length here is just considered to be long enough that the f_c terms are equal to 1.

For the N atom, the representation is constructed as follows. One starts by calculating all the terms in the sum of equation 1.17 and those corresponding to the same atom type are summed together. Since there are no terms where the neighbouring atom is a nitrogen, one term remains zero.

$$\begin{aligned} G^2_N(R_s = 0) &= [e^{-\eta(R_{NCl})^2},\ 0,\ e^{-\eta(R_{NH1})^2} + e^{-\eta(R_{NH2})^2}] \\ &= [0.37, 0.0, 0.74] \end{aligned} \tag{A.1}$$

Where R_{NCl}, R_{NH1} and R_{NH2} are the distance from the central N atom to the Cl, to the first and second H respectively. The same thing is done but with a different R_s value:

$$\begin{aligned} G^2_N(R_s = 1) &= [e^{-\eta(R_{NCl}-1)^2},\ 0,\ e^{-\eta(R_{NH1}-1)^2} + e^{-\eta(R_{NH2}-1)^2}] \\ &= [1.0, 0.0, 2.0] \end{aligned} \tag{A.2}$$

The two vectors obtained are then concatenated together, so that the full two-body term for the N atom is:

$$G_N^2 = [0.37, 0, 0.74, 1.0, 0, 2.0] \tag{A.3}$$

For the first of the two H atoms, the two-body term is given by the concatenation of the terms $G_{H1}^2(R_s = 0)$ and $G_{H1}^2(R_s = 1)$:

$$\begin{aligned} G_{H1}^2(R_s = 0) &= [e^{-\eta(R_{H1Cl})^2},\ e^{-\eta(R_{H1N})^2},\ e^{-\eta(R_{H1H2})^2}] \\ &= [0.14, 0.37, 0.14] \end{aligned} \tag{A.4}$$

$$\begin{aligned} G_{H1}^2(R_s = 1) &= [e^{-\eta(R_{H1Cl}-1)^2},\ e^{-\eta(R_{H1N}-1)^2}, e^{-\eta(R_{H1H2}-1)^2}] \\ &= [0.84, 1.0, 0.84] \end{aligned} \tag{A.5}$$

The two-body term is obtained in this way for all the remaining atoms. Then, the three-body terms have to be evaluated. The parameters η and ζ only take one value each, but the parameters R_s and θ_s take multiple values. In this example, $\eta = 1$, $\zeta = 1$, $R_s = [0, 1]$, $\theta_s = [\pi, \pi/2]$. The number of possible 2 neighbours in this system is 4: HH, NH, ClH, NCl. The dimensions of the three-body term tensor is $[N_{samples}, N_{atoms}, N_{element\ pairs} \times N_{R_s} \times N_{\theta_s}]$. So, in this case it is $[1, 4, 16]$. The procedure to calculate them is the same as for the two-body terms. One starts by evaluating the terms in the sum of equation 1.19 and summing together those that correspond to the same pair of neighbours. For example, for the N atom:

$$\begin{aligned} &G_N^3(R_s = 0, \theta_s = \pi) = \\ &\quad [(1 + \cos(\theta_{NH1H2} - \pi))e^{-\left(\frac{R_{NH1}+R_{NH2}}{2}\right)^2}, \quad 0, \\ &\quad (1 + \cos(\theta_{NClH1} - \pi))e^{-\left(\frac{R_{NCl}+R_{NH1}}{2}\right)^2} + (1 + \cos(\theta_{NClH2} - \pi))e^{-\left(\frac{R_{NCl}+R_{NH2}}{2}\right)^2}, \quad 0] \end{aligned} \tag{A.6}$$

This process is repeated for all possible combinations of R_s and θ_s values. Then, all of the vectors are concatenated together to give the three-body term for the N atom.

Once the two- and three-body terms have been calculated for all atoms, they are concatenated together, so that the final ACSF descriptor has shape:

$$[N_{samples}, N_{atoms}, N_{elements} \times N_{R_s} + N_{element\ pairs} \times N_{R_s} \times N_{\theta_s}]$$

Appendix B

Note on additivity schemes

There have been efforts in the past to reduce thermochemical properties to a sum of atomic, bonds and group properties. These techniques are called 'additivity schemes' [228] and are useful because the calculations can replace more time consuming experiments otherwise required to obtain certain molecular properties. They also take into account non-bonded interactions, because they also affect the molecular properties. These can be trickier to codify, because they involve different parts of molecules being close in space to each other and this is not always evident from bonding information alone. For example, predicting the properties of ring structures requires unique strain corrections which cannot always be derived from the groups themselves. [228] For systems like squalane, these additivity schemes can produce reasonable results. By looking at Fig. B.1, it can be seen that there are 8 C-(C)(H)3 groups, 16 C-(C)2(H)2 groups and 6 C-(C)3(H) groups. In addition there are 10 gauche interactions that cannot be avoided. This results in an enthalpy of formation at 298 K of:

$$\begin{aligned}\Delta H &= 8 \times (-41.8\,\mathrm{kJ\,mol^{-1}}) + 16 \times (-20.9\,\mathrm{kJ\,mol^{-1}}) + 6 \times (-10.0\,\mathrm{kJ\,mol^{-1}}) + 10 \times (3.3\,\mathrm{kJ\,mol^{-1}}) \\ &= -695.8\,\mathrm{kJ\,mol^{-1}}\end{aligned} \tag{B.1}$$

While the experimental enthalpy of formation for squalane in the gas phase at 298 K is $\Delta H_{exp}(gas) = -861.97\,\mathrm{kJ\,mol^{-1}}$. [229] The discrepancy between these two values shows that there are clearly additional interactions that are not accounted for. Since atomic neural networks learn to predict the energy of each atom, based on what atoms are spatially close to each other, they should learn to include automatically the interactions that are difficult to predict using additivity schemes. However, the interactions considered when using representations like the ACSFs only include the atoms that are within a certain cut-off from each atom. There have been attempts to learn from fragments of a larger system with atomic neural networks. Gastegger et al. [230] used the systematic molecular fragmentation (SMF) approach to compare the performance of an atomic neural network to predict the energies of linear all-trans alkane chains of varying length. SMF works by generating overlapping fragments of a system and calculating the properties of the fragments. Then, the properties are summed together and the contribution from the overlapping regions are subtracted. [231] The authors reported that the networks were able to utilise the information in the molecular fragments more efficiently compared to SMF.

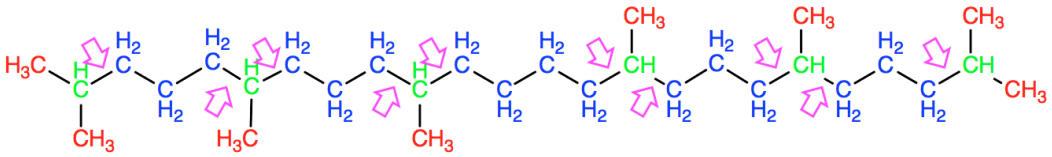


Figure B.1: Diagram showing all the different Benson groups that can be identified in squalane: C-(C)(H)3, C-(C)2(H)2 and C-(C)3(H). The pink arrows show all of the gauche interactions.

Appendix C

Hyper-parameters of chapter 2

For the chapter on fitting the potential energy surface of methane (chapter 2), the hyper-parameters used were:

- For the feed forward neural network with the Coulomb matrix: number of iterations: 500, L1 regularisation parameter: 3.81×10^{-7}, L2 regularisation parameter: 5.46×10^{-5}, learning rate: 8×10^{-4}, number of neurons in the first hidden layer: 32, number of neurons in the second hidden layer: 298, batch size: 15.
- For the feed forward neural network with SLATM: number of iterations: 2401, L1 regularisation parameter: 3.73×10^{-6}, L2 regularisation parameter: 2.13×10^{-7}, learning rate: 4×10^{-4}, number of neurons in the first hidden layer: 483, number of neurons in the second hidden layer: 70, batch size: 15.
- For the atomic neural network with atomic SLATM: number of iterations: 2493, L1 regularisation parameter: 3.13×10^{-5}, L2 regularisation parameter: 8.56×10^{-7}, learning rate: 3.26×10^{-3}, number of neurons in the first hidden layer: 91, number of neurons in the second hidden layer: 36, batch size: 99.
- For the atomic neural network with the unoptimised ACSF: number of iterations: 349, L1 regularisation parameter: 9.56×10^{-7}, L2 regularisation parameter: 5.56×10^{-7}, learning rate: 7.74×10^{-5}, number of neurons in the first hidden layer: 70, number of neurons in the second hidden layer: 104, batch size: 124.
- For the atomic neural network with the optimised ACSF: number of iterations: 869, L1 regularisation parameter: 1.79×10^{-7}, L2 regularisation parameter: 2.11×10^{-5}, learning rate: 1.65×10^{-3}, number of neurons in the first hidden layer: 320, number of neurons in the second hidden layer: 49, batch size: 67.

Appendix D

Reduction of the hyper-parameter number in ACSFs

The hyper-parameters of the ACSFs were reparametrised following the procedure designed by Dr Lars Bratholm and described in a recent publication. [3] This was done to reduce the the number of correlated hyper-parameters, in order to speed up hyper-parameter optimisation. The original formulation of the ACSFs from Smith et al. [26] for the 2-body (G_i^2) and 3-body (G_i^3) terms was described in eq. 1.17 and 1.19 respectively, but they are shown below for convenience:

$$G_i^2 = \sum_{\substack{j \neq i \\ Z_j = Z}} e^{-\eta(R_{ij} - R_s)^2} f_c(R_{ij}) \tag{D.1}$$

$$G_i^3 = 2^{1-\zeta} \sum_{\substack{j \neq i, j \neq k \\ Z_j = Z_1 \\ Z_k = Z_2}} (1 + \cos(\theta_{ijk} - \theta_s))^\zeta e^{-\eta\left(\frac{R_{ij} + R_{ik}}{2} - R_s\right)^2} f_c(R_{ij}) f_c(R_{ik}) \tag{D.2}$$

In the following explanation, radial and angular basis functions refer to $f(R)$ and $g(\theta)$ of equations D.3 and D.4 respectively.

$$f(R) = e^{-\eta(R - R_s)^2} \tag{D.3}$$

$$g(\theta) = 2^{1-\zeta}(1 + \cos(\theta - \theta_s))^\zeta \tag{D.4}$$

Usually, a grid of values of R_s and θ_s are used to create the ACSFs. Here, N_r and N_a are used to refer to the number of R_s and θ_s values used in the grid. The values of R_s range from $r_{\min}$ to the cut-off radius R_c and the values of θ_s range from 0 to π.

In order to reduce the number of hyper-parameters, the number of radial and angular basis function is kept the same (i.e. $N_r = N_a = N_{\text{basis}}$). Choosing a good value for η and ζ depends on the number of basis functions (N_{basis}), as this affects how wide the gaussian functions need to be to overlap enough with each other. Consequently, η and ζ are re-written as a function of N_{basis} and a new precision parameter τ. τ is defined such that the value where two neighbouring radial basis functions intersect is $1/\tau$ and the value where two neighbouring angular basis functions intersect is $2/\tau$. For N_{basis} radial

basis functions, with R_s range from r_{min} to R_c (included), the distance d between the centres of any two neighbouring basis function is:

$$d = \frac{R_c - r_{\text{min}}}{N_{\text{basis}} - 1} \tag{D.5}$$

From this definition, neighbouring basis functions will intersect at a distance of $R_s + d/2$. This means that η can be expressed as a function of N_{basis}, R_c, r_{min} and τ by solving the equation:

$$f\left(R_s + \frac{d}{2}\right) = \frac{1}{\tau} \tag{D.6}$$

which results in:

$$\eta = \frac{4\log(\tau)(N_{\text{basis}} - 1)^2}{(R_c - r_{\text{min}})^2} \tag{D.7}$$

The value of ζ can be derived in a similar way. For N_{basis} basis functions, with θ_s in the range 0 to π, the distance d between the centres of the basis functions is:

$$d = \frac{\pi}{N_{\text{basis}} - 1} \tag{D.8}$$

It follows that a given basis function will intersect with a neighbouring one at a distance of $\theta_s + d/2$ from its centre. This means that θ can be expressed as a function of N_{basis} and τ by solving the equation:

$$g\left(\theta_s + \frac{d}{2}\right) = \frac{2}{\tau} \tag{D.9}$$

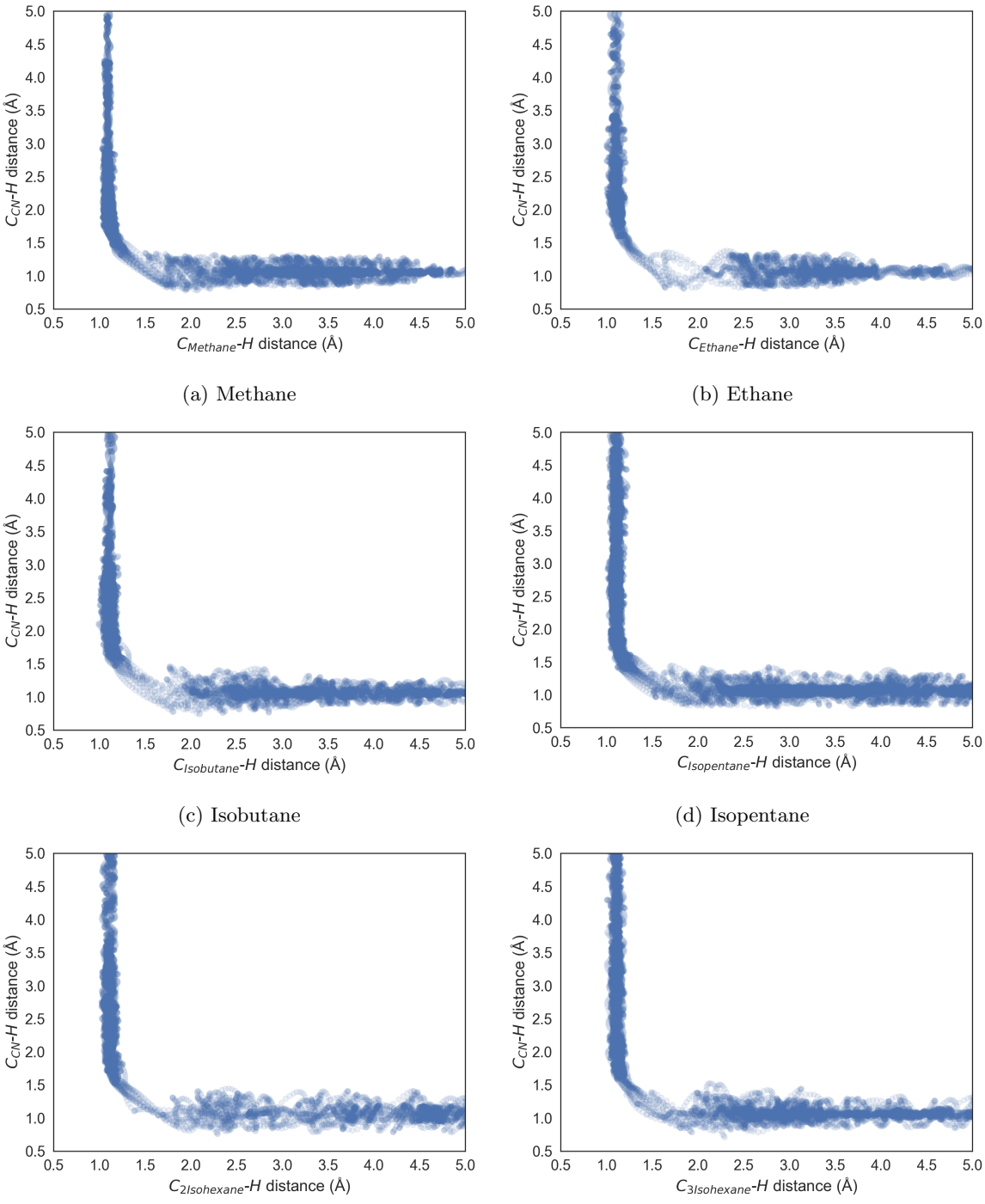
Resulting in:

$$\zeta = -\frac{\log(\tau)}{2\log(\cos(\frac{\pi}{4N_{\text{basis}}-4}))} \tag{D.10}$$

Appendix E

Visualisation of data sets for different hydrocarbons

Fig. E.1 shows the values of the $C_{hydrocarbon}$-H and C_{CN}-H distances (for the H being abstracted) during the sampled abstraction trajectories of methane, ethane, isobutane, isopentane, 2-isohexane, 3-Isohexane and squalane.



(a) Methane

(b) Ethane

(c) Isobutane

(d) Isopentane

(e) 2-Isohexane

(f) 3-Isohexane

Figure E.1: Values of the distances between the cyano carbon and the abstracted hydrogen as a function of the distance between the hydrocarbons carbon and the abstracted hydrogen. Each data point is plotted with transparency, so that the difference in sampling of various regions can be observed.

Appendix F

Hyper-parameters of chapter 4

The hyper-parameters for the NNs trained on each mixed data set are shown in table F.1. The hyper-parameter shown in the table are the keywords used in the QML program. Their meaning is:

`iterations`	Number of training epochs
`l1_reg`	L1 regularisation parameter
`l2_reg`	L2 regularisation parameter
`learning_rate`	Learning rate in the Adam optimisation
`hidden_layer_sizes`	Number of neurons in each hidden layer
`batch_size`	Size of the mini-batches used in the optimisation
`n_basis`	number of values to use in R_s and θ_s in the ACSFs
`r_min`	First value of R_s
`r_cut`	Cut-off radius in the ACSFs
`tau`	Parameter used to calculate η and ζ in the ACSFs

The hyper-parameter `tau` (τ) is used to calculate η and ζ as follows:

$$\eta = 4\log(\tau) \times \left(\frac{n_{\text{basis}} - 1}{r_{\text{cut}} - r_{\text{min}}}\right)^2 \tag{F.1}$$

$$\zeta = -\frac{\log(\tau)}{(2\log(\cos(\pi/(4n_{\text{basis}} - 4))))} \tag{F.2}$$

Table F.1: Hyper-parameters used in chapter 4 when training the NNs on the different mixed data sets.

Parameter	**Training set 1**	**Training set 2**	**Training set 3**	**Training set 4**	**Training set 5**	**Training set 6**
`iterations`	639	1338	965	1181	1424	900
`l1_reg`	1.5e-4	6.4e-7	1.0e-6	2.5e-4	1.5e-6	1.9e-4
`l2_reg`	3.5e-7	2.2e-5	4.2e-7	4.1e-5	8.7e-5	2.2e-8
`learning_rate`	2.3e-3	1.2e-3	4.3e-4	7.1e-4	7.0e-4	1.5e-3
`hidden_layer_sizes`	(272,179)	(393,154)	(280,326)	(94,174)	(235,144)	(62,142)
`batch_size`	23	24	22	26	43	23
`n_basis`	15	19	13	13	12	16
`r_min`	0.8	0.8	0.8	0.8	0.8	0.8
`r_cut`	4.3	3.4	3.6	3.4	3.7	3.1
`tau`	1.7	1.4	1.7	1.9	2.2	1.8

Appendix G

Molecular properties for PCA

In chapter 6 section 6.2.2, 52 molecular 2D descriptors were calculated using the Molecular Operating Environment (MOE) software. The properties that were calculated were:

number of H-bond acceptor atoms
number of acidic atoms
number of aromatic atoms
number of basic atoms
number of atoms
number of H-bond donor atoms
number of H-bond donor + acceptor atoms
number of heavy atoms
number of hydrophobic atoms
number of boron atoms
number of bromine atoms
number of carbon atoms
number of chlorine atoms
number of fluorine atoms
number of hydrogen atoms
number of iodine atoms
number of nitrogen atoms
number of oxygen atoms
number of phosphorus atoms
number of sulfur atoms
number of rotatable single bonds
fraction of rotatable single bonds
number of aromatic bonds
number of bonds
number of double bonds
number of heavy-heavy bonds
number of rotatable bonds
fraction of rotatable bonds
number of single bonds
number of triple bonds
number of chiral centers
number of unconstrained chiral centers
sum of formal charges
octanol/water distribution coefficient (pH=7)
octanol/water partition coefficient
log solubility in water
molar refractivity
molecular flexibility
lipinski acceptor count
lipinski donor count
lipinski druglike test
lipinski violation count
reactivity
number of rings
topological polar surface area (A**2)
van der waals surface area (A**2)
van der waals volume (A**3)
vdw acceptor surface area (A**2)
vdw donor surface area (A**2)
vdw hydrophobe surface area (A**2)
molecular weight (CRC)

Appendix H

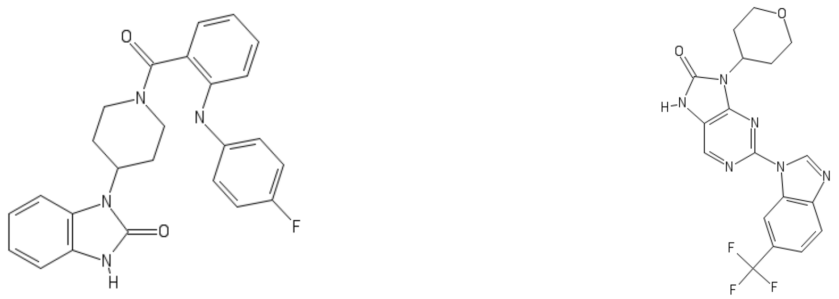
Testing experimentally the RNN predictions

NovaData Solutions attepted to evaluate experimentally the properties of the best candidates generated by the RNN. The work described in this section was carried out by Dr Michael Mazanetz.

First of all, 864 valid SMILES that had been generated by the RNN after reinforcement learning were taken and their Tanimoto similarity [221] to compounds already available in Enamine, [232] Asinex, [233] Ambinter, [**?**] Maybridge, [234] Princeton, [235] Otava, [236] Chembridge, [237] ChemDiv [238] was checked.

The Tanimoto similarity was calculated based on the MACCS fingerprints [239] and only compounds with a similarity higher than 0.6 were kept. The highest similarity observed was 0.85. It has to be noted that this is not a very high degree of similarity. For example, in Fig. H.1 is shown an example of a molecule from the Enamine database and one generated by the RNN, with a Tanimoto similarity of 0.79.

Then, the feed forward neural network was used to predict the pIC50 values of the compounds. The top 56 compounds with the highest values were kept and induced fit docking was used to check which molecules bind best to the 4P7E protein [240] (Fig. H.2). Induced fit docking is different from standard virtual docking, because it takes into account structural changes in the receptor rather than considering it a rigid receptor. [241] The best 30 compounds were purchased and an Z'-LYTE biochemical assay



(a) Molecule from Enamine database

(b) Molecule generated by the RNN

Figure H.1: Example of a molecule from the Enamine database and one generated by the RNN, with a Tanimoto similarity of 0.79.

[242] was ran.

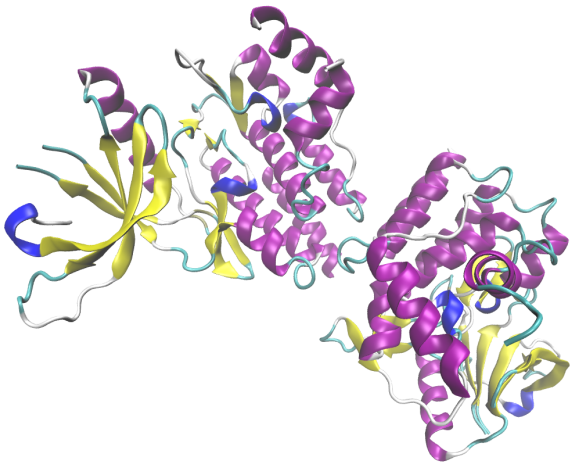


Figure H.2: Structure of the 4P7E protein

This type of assays works by having a protein substrate labelled with two fluorophores, one at each end of the substrate. The two fluorophores make a fluorescence resonance energy transfer (FRET) pair. There is a first reaction where the Kinase phosphorylates the substrate. Then, there is a second reaction with a protease that cleaves all non-phosphorylated substrates. Cleaved substrates are no longer fluorescent. If a drug molecule is added and it inhibits the Kinase, the substrates will not be phosphorylated and the fluorescence will be lost. [243] The results showed that none of the compounds tested had more than 40% inhibition of the Kinase at 100 µM. This result is not conclusive though as to whether the molecules generated by the RNN are good inhibitors or not. In order for this test to have been conclusive, closer analogues should have been synthesised. However, this is an expensive and long process, so better analysis of the molecules output by the RNN should be done before undertaking such endeavour.

Bibliography

[1] D. G. Truhlar, R. Steckler, and M. S. Gordon, "Potential energy surfaces for polyatomic reaction dynamics," *Chemical Reviews*, vol. 87, no. 1, pp. 217–236, 1987.

[2] J. Behler, "Neural network potential-energy surfaces for atomistic simulations," in *Chemical Modelling: Applications and Theory Volume 7*, vol. 7, pp. 1–41, The Royal Society of Chemistry, 2010.

[3] S. Amabilino, L. A. Bratholm, S. J. Bennie, A. C. Vaucher, M. Reiher, and D. R. Glowacki, "Training neural nets to learn reactive potential energy surfaces using interactive quantum chemistry in virtual reality," *The Journal of Physical Chemistry A*, vol. 123, no. 20, pp. 4486–4499, 2019.

[4] R. J. Allen, D. Frenkel, and P. R. ten Wolde, "Simulating rare events in equilibrium or nonequilibrium stochastic systems," *The Journal of Chemical Physics*, vol. 124, no. 2, p. 024102, 2006.

[5] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnum, and R. Komanduri, "Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks," *The Journal of Chemical Physics*, vol. 130, no. 13, p. 134101, 2009.

[6] L. Raff, R. Komanduri, M. Hagan, and S. Bukkapatnam, *Neural networks in chemical reaction dynamics*. OUP USA, 2012.

[7] E. Gurdia, R. Rey, and J. Padr, "Potential of mean force by constrained molecular dynamics: A sodium chloride ion-pair in water," *Chemical Physics*, vol. 155, no. 2, pp. 187 – 195, 1991.

[8] Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics method for protein folding," *Chemical Physics Letters*, vol. 314, no. 1, pp. 141 – 151, 1999.

[9] A. Barducci, M. Bonomi, and M. Parrinello, "Metadynamics," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1, no. 5, pp. 826–843, 2011.

[10] D. R. Glowacki, E. Paci, and D. V. Shalashilin, "Boxed molecular dynamics: A simple and general technique for accelerating rare event kinetics and mapping free energy in large molecular systems," *The Journal of Physical Chemistry B*, vol. 113, no. 52, pp. 16603–16611, 2009. PMID: 19961166.

[11] R. C. Bernardi, M. C. Melo, and K. Schulten, "Enhanced sampling techniques in molecular dynamics simulations of biological systems," *Biochimica et Biophysica Acta (BBA) - General Subjects*, vol. 1850, no. 5, pp. 872 – 877, 2015. Recent developments of molecular dynamics.

[12] C. Abrams and G. Bussi, "Enhanced sampling in molecular dynamics using metadynamics, replica-exchange, and temperature-acceleration," *Entropy*, vol. 16, no. 1, pp. 163–199, 2014.

[13] K. Yao, J. E. Herr, D. Toth, R. Mckintyre, and J. Parkhill, "The tensormol-0.1 model chemistry: a neural network augmented with long-range physics," *Chem. Sci.*, vol. 9, pp. 2261–2269, 2018.

[14] J. Behler, "Constructing high-dimensional neural network potentials: A tutorial review," *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1032–1050, 2015.

[15] M. A. Collins, "Molecular potential-energy surfaces for chemical reaction dynamics," *Theoretical Chemistry Accounts*, vol. 108, pp. 313–324, Dec 2002.

[16] J. E. Stone, J. Gullingsrud, and K. Schulten, "A system for interactive molecular dynamics simulation," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 191–194, ACM, 2001.

[17] M. O'Connor, H. M. Deeks, E. Dawn, O. Metatla, A. Roudaut, M. Sutton, L. M. Thomas, B. R. Glowacki, R. Sage, P. Tew, M. Wonnacott, P. Bates, A. J. Mulholland, and D. R. Glowacki, "Sampling molecular conformations and dynamics in a multiuser virtual reality framework," *Science Advances*, vol. 4, no. 6, 2018.

[18] C. W. Bauschlicher, S. R. Langhoff, and P. R. Taylor, "Accurate quantum chemical calculations," *Advances in Chemical Physics*, vol. 77, pp. 103–161, 1990.

[19] Z. Gershgorn and I. Shavitt, "An application of perturbation theory ideas in configuration interaction calculations," *International Journal of Quantum Chemistry*, vol. 2, no. 6, pp. 751–759, 1968.

[20] T. Helgaker, P. Jorgensen, and J. Olsen, *Molecular electronic-structure theory*. John Wiley & Sons, 2014.

[21] I. Shavitt and R. J. Bartlett, *Many-body methods in chemistry and physics: MBPT and coupled-cluster theory*. Cambridge university press, 2009.

[22] C. Møller and M. S. Plesset, "Note on an approximation treatment for many-electron systems," *Phys. Rev.*, vol. 46, pp. 618–622, Oct 1934.

[23] "Introduction to computational quantumchemistry: Theory." `http://rsc.anu.edu.au/~agilbert/gilbertspace/uploads/Chem3108.pdf`. Accessed: 2020-01-20.

[24] D. R. Glowacki, A. J. Orr-Ewing, and J. N. Harvey, "Product energy deposition of cn + alkane h abstraction reactions in gas and solution phases," *The Journal of Chemical Physics*, vol. 134, no. 21, p. 214508, 2011.

[25] H. M. Le and L. M. Raff, "Molecular dynamics investigation of the bimolecular reaction beh + h2 to beh2 + h on an ab initio potential-energy surface obtained using neural network methods with both potential and gradient accuracy determination," *The Journal of Physical Chemistry A*, vol. 114, no. 1, pp. 45–53, 2010. PMID: 19852450.

[26] J. S. Smith, O. Isayev, and A. E. Roitberg, "Ani-1: an extensible neural network potential with dft accuracy at force field computational cost," *Chem. Sci.*, vol. 8, pp. 3192–3203, 2017.

[27] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, "Assessment and validation of machine learning methods for predicting molecular atomization energies," *Journal of Chemical Theory and Computation*, vol. 9, no. 8, pp. 3404–3419, 2013. PMID: 26584096.

[28] S. Mohr, M. Eixarch, M. Amsler, M. J. Mantsinen, and L. Genovese, "Linear scaling dft calculations for large tungsten systems using an optimized local basis," *Nuclear Materials and Energy*, vol. 15, pp. 64 – 70, 2018.

[29] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Phys. Rev.*, vol. 136, pp. B864–B871, Nov 1964.

[30] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Phys. Rev.*, vol. 140, pp. A1133–A1138, Nov 1965.

[31] J. C. Slater, "A simplification of the hartree-fock method," *Phys. Rev.*, vol. 81, pp. 385–390, Feb 1951.

[32] A. D. Becke, "Density functional calculations of molecular bond energies," *The Journal of Chemical Physics*, vol. 84, no. 8, pp. 4524–4529, 1986.

[33] A. D. Becke, "Densityfunctional thermochemistry. iii. the role of exact exchange," *The Journal of Chemical Physics*, vol. 98, no. 7, pp. 5648–5652, 1993.

[34] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch, "Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields," *The Journal of Physical Chemistry*, vol. 98, no. 45, pp. 11623–11627, 1994.

[35] E. R. Davidson and D. Feller, "Basis set selection for molecular calculations," *Chemical Reviews*, vol. 86, no. 4, pp. 681–696, 1986.

[36] W. J. Hehre, R. F. Stewart, and J. A. Pople, "Selfconsistent molecularorbital methods. i. use of gaussian expansions of slatertype atomic orbitals," *The Journal of Chemical Physics*, vol. 51, no. 6, pp. 2657–2664, 1969.

[37] J. Foresman and E. Frish, "Exploring chemistry," *Gaussian Inc., Pittsburg, USA*, 1996.

[38] A. J. Cohen, P. Mori-Sánchez, and W. Yang, "Insights into current limitations of density functional theory," *Science*, vol. 321, no. 5890, pp. 792–794, 2008.

[39] A. Sisto, D. R. Glowacki, and T. J. Martinez, "Ab initio nonadiabatic dynamics of multichromophore complexes: A scalable graphical-processing-unit-accelerated exciton framework," *Accounts of Chemical Research*, vol. 47, no. 9, pp. 2857–2866, 2014. PMID: 25186064.

[40] G. Seifert and J.-O. Joswig, "Density-functional tight binding - an approximate density-functional theory method," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 2, no. 3, pp. 456–465, 2012.

[41] S. Grimme, C. Bannwarth, and P. Shushkov, "A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions of large molecular systems parametrized for all spd-block elements (z = 1-86)," *Journal of Chemical Theory and Computation*, vol. 13, no. 5, pp. 1989–2009, 2017. PMID: 28418654.

[42] W. Thiel, "Semiempirical quantum-chemical methods," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 2, pp. 145–157, 2014.

[43] C. Bannwarth, S. Ehlert, and S. Grimme, "Gfn2-xtban accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions," *Journal of Chemical Theory and Computation*, vol. 15, no. 3, pp. 1652–1671, 2019. PMID: 30741547.

[44] H. Jiri, R. Jan, and H. Pavel, "On the performance of the semiempirical quantum mechanical pm6 and pm7 methods for noncovalent interactions," *Chemical Physics Letters*, vol. 568-569, pp. 161 – 166, 2013.

[45] A. F. Oliveira, G. Seifert, T. Heine, and H. A. Duarte, "Density-functional based tight-binding: an approximate dft method," *Journal of the Brazilian Chemical Society*, vol. 20, no. 7, pp. 1193–1205, 2009.

[46] M. Elstner and G. Seifert, "Density functional tight binding," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2011, p. 20120483, 2014.

[47] "Interpolation vs. fitting." `http://www.math.stonybrook.edu/~scott/Book331/Interpolation_vs_Fitting.html`. Accessed: 2019-11-02.

[48] B. J. Braams and J. M. Bowman, "Permutationally invariant potential energy surfaces in high dimensionality," *International Reviews in Physical Chemistry*, vol. 28, no. 4, pp. 577–606, 2009.

[49] S. McKinley and M. Levine, "Cubic spline interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[50] F. V. Prudente and J. S. Neto, "The fitting of potential energy surfaces using neural networks. application to the study of the photodissociation processes," *Chemical Physics Letters*, vol. 287, no. 5, pp. 585 – 589, 1998.

[51] N. Sathyamurthy, G. E. Kellerhals, and L. M. Raff, "Quantum mechanical scattering studies using 2d cubic spline interpolation of a potentialenergy surface," *The Journal of Chemical Physics*, vol. 64, no. 5, pp. 2259–2261, 1976.

[52] M. A. Collins, "Molecular potential-energy surfaces for chemical reaction dynamics," *Theoretical Chemistry Accounts*, vol. 108, no. 6, pp. 313–324, 2002.

[53] J. Espinosa-Garcia, M. Monge-Palacios, and J. C. Corchado, "Constructing potential energy surfaces for polyatomic systems: Recent progress and new problems," *Advances in Physical Chemistry*, vol. 2012, 2012.

[54] G. G. Maisuradze and D. L. Thompson, "Interpolating moving least-squares methods for fitting potential energy surfaces: Illustrative approaches and applications," *The Journal of Physical Chemistry A*, vol. 107, no. 37, pp. 7118–7124, 2003.

[55] Y. Guo, A. Kawano, D. L. Thompson, A. F. Wagner, and M. Minkoff, "Interpolating moving least-squares methods for fitting potential energy surfaces: Applications to classical dynamics calculations," *The Journal of Chemical Physics*, vol. 121, no. 11, pp. 5091–5097, 2004.

[56] Y. Guo, L. B. Harding, A. F. Wagner, M. Minkoff, and D. L. Thompson, "Interpolating moving least-squares methods for fitting potential energy surfaces: An application to the h2cn unimolecular reaction," *The Journal of Chemical Physics*, vol. 126, no. 10, p. 104105, 2007.

[57] A. Warshel and R. M. Weiss, "An empirical valence bond approach for comparing reactions in solutions and in enzymes," *Journal of the American Chemical Society*, vol. 102, no. 20, pp. 6218–6226, 1980.

[58] D. R. Glowacki, A. J. Orr-Ewing, and J. N. Harvey, "Non-equilibrium reaction and relaxation dynamics in a strongly interacting explicit solvent: F + cd3cn treated with a parallel multi-state evb model," *The Journal of Chemical Physics*, vol. 143, no. 4, p. 044120, 2015.

[59] R. A. Rose, S. J. Greaves, F. Abou-Chahine, D. R. Glowacki, T. A. A. Oliver, M. N. R. Ashfold, I. P. Clark, G. M. Greetham, M. Towrie, and A. J. Orr-Ewing, "Reaction dynamics of cn radicals with tetrahydrofuran in liquid solutions," *Phys. Chem. Chem. Phys.*, vol. 14, pp. 10424–10437, 2012.

[60] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.

[61] S. Si, C.-J. Hsieh, and I. Dhillon, "Computationally efficient nyström approximation using fast transforms," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2655–2663, PMLR, 20–22 Jun 2016.

[62] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, "Machine learning of accurate energy-conserving molecular force fields," *Science Advances*, vol. 3, no. 5, 2017.

[63] P. O. Dral, A. Owens, S. N. Yurchenko, and W. Thiel, "Structure-based sampling and self-correcting machine learning for accurate calculations of potential energy surfaces and vibrational levels," *The Journal of Chemical Physics*, vol. 146, no. 24, p. 244108, 2017.

[64] S. Jothilakshmi and V. Gudivada, "Chapter 10 - large scale data enabled evolution of spoken language research and applications," in *Cognitive Computing: Theory and Applications* (V. N. Gudivada, V. V. Raghavan, V. Govindaraju, and C. Rao, eds.), vol. 35 of *Handbook of Statistics*, pp. 301 – 340, Elsevier, 2016.

[65] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[66] B. L. Kalman and S. C. Kwasny, "Why tanh: choosing a sigmoidal function," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 4, pp. 578–581 vol.4, June 1992.

[67] S. Lawrence, C. L. Giles, and A. C. Tsoi, "Lessons in neural network training: Overfitting may be harder than expected," in *AAAI/IAAI*, pp. 540–545, Citeseer, 1997.

[68] E. Phaisangittisagul, "An analysis of the regularization between l2 and dropout in single hidden layer neural network," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 174–179, Jan 2016.

[69] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[70] R. HECHT-NIELSEN, "Iii.3 - theory of the backpropagation neural network**based on nonindent by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593611, june 1989. 1989 ieee.," in *Neural Networks for Perception* (H. Wechsler, ed.), pp. 65 – 93, Academic Press, 1992.

[71] A. V. Ooyen and B. Nienhuis, "Improving the convergence of the back-propagation algorithm," *Neural Networks*, vol. 5, no. 3, pp. 465 – 471, 1992.

[72] "Backpropagation algorithm." `http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`. Accessed: 2019-11-05.

[73] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[74] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," *The Journal of Chemical Physics*, vol. 145, no. 17, p. 170901, 2016.

[75] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Phys. Rev. Lett.*, vol. 98, p. 146401, Apr 2007.

[76] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," *Phys. Rev. B*, vol. 87, p. 184115, May 2013.

[77] J. Behler, "Neural network potential-energy surfaces for atomistic simulations," in *Chemical Modelling: Applications and Theory Volume 7*, vol. 7, pp. 1–41, The Royal Society of Chemistry, 2010.

[78] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, "Neural network models of potential energy surfaces," *The Journal of Chemical Physics*, vol. 103, no. 10, pp. 4129–4137, 1995.

[79] K. T. No, B. H. Chang, S. Y. Kim, M. S. Jhon, and H. A. Scheraga, "Description of the potential energy surface of the water dimer with an artificial neural network," *Chemical Physics Letters*, vol. 271, no. 1, pp. 152 – 156, 1997.

[80] R. Withnall, B. Z. Chowdhry, S. Bell, and T. J. Dines, "Computational chemistry using modern electronic structure methods," *Journal of Chemical Education*, vol. 84, no. 8, p. 1364, 2007.

[81] R. Ramakrishnan and O. A. von Lilienfeld, *Machine Learning, Quantum Chemistry, and Chemical Space*, pp. 225–256. John Wiley and Sons, Inc., 2017.

[82] B. Huang and O. A. von Lilienfeld, "The "dna" of chemistry: Scalable quantum machine learning with" amons"," *arXiv preprint arXiv:1707.04146*, 2017.

[83] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," *Physical review letters*, vol. 108, no. 5, p. 058301, 2012.

[84] J. Sun, *Learning over Molecules: Representations and Kernels*. PhD thesis, Harvard College, 2014.

[85] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 134, no. 7, p. 074106, 2011.

[86] A. Gupta, A. T. Müller, B. J. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider, "Generative recurrent networks for de novo drug design," *Molecular informatics*, vol. 37, no. 1-2, p. 1700111, 2018.

[87] M. H. Segler and M. P. Waller, "Neural-symbolic machine learning for retrosynthesis and reaction prediction," *Chemistry–A European Journal*, vol. 23, no. 25, pp. 5966–5971, 2017.

[88] N. Artrith and J. Behler, "High-dimensional neural network potentials for metal surfaces: A prototype study for copper," *Phys. Rev. B*, vol. 85, p. 045439, Jan 2012.

[89] J. Behler, "Runner–a neural network code for high-dimensional potential-energy surfaces," *Lehrstuhl für Theoretische Chemie, Ruhr-Universität Bochum*, 2018.

[90] "List of torchani github releases." `https://github.com/aiqm/torchani/releases`. Accessed: 2019-11-4.

[91] "Github repository of tensormol." `https://github.com/jparkhill/TensorMol`. Accessed: 2019-11-4.

[92] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "Schnet - a deep learning architecture for molecules and materials," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241722, 2018.

[93] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[94] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature communications*, vol. 8, p. 13890, 2017.

[95] "Schnetpack." `https://github.com/atomistic-machine-learning/schnetpack`. Accessed: 2019-11-06.

[96] S. Lorenz, M. Scheffler, and A. Gross, "Descriptions of surface chemical reactions using a neural network representation of the potential-energy surface," *Phys. Rev. B*, vol. 73, p. 115431, Mar 2006.

[97] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnum, and R. Komanduri, "Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks," *The Journal of Chemical Physics*, vol. 130, no. 13, p. 134101, 2009.

[98] S. J. Greaves, R. A. Rose, T. A. Oliver, D. R. Glowacki, M. N. Ashfold, J. N. Harvey, I. P. Clark, G. M. Greetham, A. W. Parker, M. Towrie, *et al.*, "Vibrationally quantum-state–specific reaction dynamics of h atom abstraction by cn radical in solution," *Science*, vol. 331, no. 6023, pp. 1423–1426, 2011.

[99] P. Davidovits, C. E. Kolb, L. R. Williams, J. T. Jayne, and D. R. Worsnop, "Mass accommodation and chemical reactions at gasliquid interfaces," *Chemical Reviews*, vol. 106, no. 4, pp. 1323–1354, 2006. PMID: 16608183.

[100] I. R. Sims, J.-L. Queffelec, D. Travers, B. R. Rowe, L. B. Herbert, J. Karthäuser, and I. W. Smith, "Rate constants for the reactions of cn with hydrocarbons at low and ultra-low temperatures," *Chemical Physics Letters*, vol. 211, no. 4, pp. 461 – 468, 1993.

[101] S. B. Morales, C. J. Bennett, S. D. L. Picard, A. Canosa, I. R. Sims, B. J. Sun, P. H. Chen, A. H. H. Chang, V. V. Kislov, A. M. Mebel, X. Gu, F. Zhang, P. Maksyutenko, and R. I. Kaiser, "A crossed molecular beam, low-temperature kinetics, and theoretical investigation of the reaction of the cyano radical (cn) with 1,3-butadiene (c4h6). a route to complex nitrogen-bearing molecules in low-temperature extraterrestrial environments," *The Astrophysical Journal*, vol. 742, p. 26, nov 2011.

[102] M. OConnor, S. J. Bennie, H. M. Deeks, A. Jamieson-Binnie, A. J. Jones, R. J. Shannon, R. Walters, T. J. Mitchell, A. J. Mulholland, and D. R. Glowacki, "Interactive molecular dynamics in virtual reality from quantum chemistry to drug binding: An open-source multi-person framework," *The Journal of Chemical Physics*, 2019.

[103] "Narupa." `www.gitlab.com/intangiblerealities`.

[104] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," *The Journal of Chemical Physics*, vol. 76, no. 1, pp. 637–649, 1982.

[105] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," *The Journal of Chemical Physics*, vol. 81, no. 8, pp. 3684–3690, 1984.

[106] B. Aradi, B. Hourahine, and T. Frauenheim, "Dftb+, a sparse matrix-based implementation of the dftb method," *The Journal of Physical Chemistry A*, vol. 111, no. 26, pp. 5678–5684, 2007. PMID: 17567110.

[107] J. P. Perdew, K. Burke, and M. Ernzerhof, "Generalized gradient approximation made simple," *Phys. Rev. Lett.*, vol. 77, pp. 3865–3868, Oct 1996.

[108] D. E. Woon and T. H. Dunning, "Gaussian basis sets for use in correlated molecular calculations. iii. the atoms aluminum through argon," *The Journal of Chemical Physics*, vol. 98, no. 2, pp. 1358–1371, 1993.

[109] T. B. Adler, G. Knizia, and H.-J. Werner, "A simple and efficient ccsd(t)-f12 approximation," *The Journal of Chemical Physics*, vol. 127, no. 22, p. 221106, 2007.

[110] 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. Software available from tensorflow.org.

[111] A. Khorshidi and A. A. Peterson, "Amp: A modular approach to machine learning in atomistic simulations," *Computer Physics Communications*, vol. 207, pp. 310 – 324, 2016.

[112] A. Christensen, L. Bratholm, F. Faber, B. Huang, A. Tkatchenko, K. Mller, and O. von Lilienfeld, "Qml: A python toolkit for quantum machine learning," 2017. Software available from https://github.com/qmlcode/qml.

[113] 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, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[114] R. T. McGibbon, C. X. Hernndez, M. P. Harrigan, S. Kearnes, M. M. Sultan, S. Jastrzebski, B. E. Husic, and V. S. Pande, "Osprey: Hyperparameter optimization for machine learning," Sept. 2016.

[115] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[116] M. Anthony and S. B. Holden, "Cross-validation for binary classification by real-valued functions: theoretical analysis," in *COLT*, pp. 218–229, Citeseer, 1998.

[117] G. E. Bullock and R. Cooper, "Reactions of cyanogen radicals with hydrocarbons," *Trans. Faraday Soc.*, vol. 67, pp. 3258–3264, 1971.

[118] C. Anastasi and D. U. Hancock, "Reaction of cn radicals with ch 4 and o 2," *Journal of the Chemical Society, Faraday Transactions 2: Molecular and Chemical Physics*, vol. 84, no. 1, pp. 9–15, 1988.

[119] C. Goy, D. Shaw, and H. Pritchard, "The reactions of cn radicals in the gas phase," *The Journal of Physical Chemistry*, vol. 69, no. 5, pp. 1504–1507, 1965.

[120] J. Chen, X. Xu, X. Xu, and D. H. Zhang, "Communication: An accurate global potential energy surface for the oh + co -¿ h + co2 reaction using neural networks," *The Journal of Chemical Physics*, vol. 138, no. 22, p. 221104, 2013.

[121] J. Li, B. Jiang, and H. Guo, "Permutation invariant polynomial neural network approach to fitting potential energy surfaces. ii. four-atom systems," *The Journal of Chemical Physics*, vol. 139, no. 20, p. 204103, 2013.

[122] "Abstraction of hydrogen from isopentane by cyano radical." `https://vimeo.com/310557619`. Accessed: 2020-01-20.

[123] J. J. P. Stewart, "Optimization of parameters for semiempirical methods v: Modification of nddo approximations and application to 70 elements," *Journal of Molecular Modeling*, vol. 13, pp. 1173–1213, Dec 2007.

[124] M. P. Haag, A. C. Vaucher, M. Bosson, S. Redon, and M. Reiher, "Interactive chemical reactivity exploration," *ChemPhysChem*, vol. 15, no. 15, pp. 3301–3319, 2014.

[125] A. C. Vaucher and M. Reiher, "Minimum energy paths and transition states by curve optimization," *Journal of chemical theory and computation*, vol. 14, no. 6, pp. 3091–3099, 2018.

[126] J. Hutter, M. Iannuzzi, F. Schiffmann, and J. VandeVondele, "cp2k: atomistic simulations of condensed matter systems," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, no. 1, pp. 15–25, 2014.

[127] G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of Chemical Physics*, vol. 126, no. 1, p. 014101, 2007.

[128] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, and M. Schütz, "Molpro: a general-purpose quantum chemistry program package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 2, no. 2, pp. 242–253, 2012.

[129] R. Polly, H.-J. W. *, F. R. Manby, and P. J. Knowles, "Fast hartre-fock theory using local density fitting approximations," *Molecular Physics*, vol. 102, no. 21-22, pp. 2311–2321, 2004.

[130] F. Weigend, "Accurate coulomb-fitting basis sets for h to rn," *Phys. Chem. Chem. Phys.*, vol. 8, pp. 1057–1065, 2006.

[131] C. Adamo and V. Barone, "Toward reliable density functional methods without adjustable parameters: The pbe0 model," *The Journal of Chemical Physics*, vol. 110, no. 13, pp. 6158–6170, 1999.

[132] W. P. Hess, J. Durant Jr, and F. P. Tully, "Kinetic study of the reactions of cyanogen radical with ethane and propane," *The Journal of Physical Chemistry*, vol. 93, no. 17, pp. 6402–6407, 1989.

[133] C. E. Rasmussen and C. Williams, "Gaussian processes for machine learning cambridge," *MA: MIT Press [Google Scholar]*, 2006.

[134] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[135] S. ichi Amari, "A universal theorem on learning curves," *Neural Networks*, vol. 6, no. 2, pp. 161 – 166, 1993.

[136] M. Malshe, L. M. Raff, M. G. Rockley, M. Hagan, P. M. Agrawal, and R. Komanduri, "Theoretical investigation of the dissociation dynamics of vibrationally excited vinyl bromide on an ab initio potential-energy surface obtained using modified novelty sampling and feedforward neural networks. ii. numerical application of the method," *The Journal of Chemical Physics*, vol. 127, no. 13, p. 134105, 2007.

[137] S. J. Greaves, R. A. Rose, T. A. A. Oliver, D. R. Glowacki, M. N. R. Ashfold, J. N. Harvey, I. P. Clark, G. M. Greetham, A. W. Parker, M. Towrie, and A. J. Orr-Ewing, "Vibrationally quantum-state–specific reaction dynamics of h atom abstraction by cn radical in solution," *Science*, vol. 331, no. 6023, pp. 1423–1426, 2011.

[138] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[139] "Scikit-learn." `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html`.

[140] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of Cheminformatics*, vol. 9, p. 48, Sep 2017.

[141] J. Bajorath and J. Bajorath, *Chemoinformatics and computational chemical biology*. Springer, 2011.

[142] X. Q. Lewell, D. B. Judd, S. P. Watson, and M. M. Hann, "Recap retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry," *Journal of chemical information and computer sciences*, vol. 38, no. 3, pp. 511–522, 1998.

[143] J. Degen, C. Wegscheid-Gerlach, A. Zaliani, and M. Rarey, "On the art of compiling and using 'drug-like' chemical fragment spaces," *ChemMedChem*, vol. 3, no. 10, pp. 1503–1507, 2008.

[144] M. Hartenfeller, H. Zettl, M. Walter, M. Rupp, F. Reisen, E. Proschak, S. Weggen, H. Stark, and G. Schneider, "Dogs: reaction-driven de novo design of bioactive compounds," *PLoS computational biology*, vol. 8, no. 2, p. e1002380, 2012.

[145] M. Hartenfeller and G. Schneider, *De Novo Drug Design*, pp. 299–323. Totowa, NJ: Humana Press, 2011.

[146] K. Heikamp, F. Zuccotto, M. Kiczun, P. Ray, and I. H. Gilbert, "Exhaustive sampling of the fragment space associated to a molecule leading to the generation of conserved fragments," *Chemical Biology & Drug Design*, vol. 91, no. 3, pp. 655–667, 2018.

[147] S. Takeda, H. Kaneko, and K. Funatsu, "Chemical-space-based de novo design method to generate drug-like molecules," *Journal of Chemical Information and Modeling*, vol. 56, no. 10, pp. 1885–1893, 2016. PMID: 27632418.

[148] L. Hoffer, C. Muller, P. Roche, and X. Morelli, "Chemistry-driven hit-to-lead optimization guided by structure-based approaches," *Molecular Informatics*, vol. 37, no. 9-10, p. 1800059, 2018.

[149] T. Rodrigues, D. Reker, M. Welin, M. Caldera, C. Brunner, G. Gabernet, P. Schneider, B. Walse, and G. Schneider, "De novo fragment design for drug discovery and chemical biology," *Angewandte Chemie International Edition*, vol. 54, no. 50, pp. 15079–15083, 2015.

[150] T. Rodrigues and G. Schneider, "Flashback forward: reaction-driven de novo design of bioactive compounds," *Synlett*, vol. 25, no. 02, pp. 170–178, 2014.

[151] D. Merk, F. Grisoni, L. Friedrich, E. Gelzinyte, and G. Schneider, "Scaffold hopping from synthetic rxr modulators by virtual screening and de novo design," *Med. Chem. Commun.*, vol. 9, pp. 1289–1292, 2018.

[152] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, *et al.*, "The chembl database in 2017," *Nucleic acids research*, vol. 45, no. D1, pp. D945–D954, 2016.

[153] T. Sterling and J. J. Irwin, "Zinc 15 - ligand discovery for everyone," *Journal of Chemical Information and Modeling*, vol. 55, no. 11, pp. 2324–2337, 2015. PMID: 26479676.

[154] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant, "PubChem Substance and Compound databases," *Nucleic Acids Research*, vol. 44, pp. D1202–D1213, 09 2015.

[155] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[156] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, and T. Sainath, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.

[157] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks," October 2013.

[158] S. Ekins, A. C. Puhl, K. M. Zorn, T. R. Lane, D. P. Russo, J. J. Klein, A. J. Hickey, and A. M. Clark, "Exploiting machine learning for end-to-end drug discovery and development," *Nature materials*, vol. 18, no. 5, p. 435, 2019.

[159] D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, "Deep learning for molecular designa review of the state of the art," *Mol. Syst. Des. Eng.*, vol. 4, pp. 828–849, 2019.

[160] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," *Science advances*, vol. 4, no. 7, p. eaap7885, 2018.

[161] T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath, and H. Chen, "Application of generative autoencoder in de novo molecular design," *Molecular informatics*, vol. 37, no. 1-2, p. 1700123, 2018.

[162] E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper, and A. Zhavoronkov, "Adversarial threshold neural computer for molecular de novo design," *Molecular pharmaceutics*, vol. 15, no. 10, pp. 4386–4397, 2018.

[163] D. Polykovskiy, A. Zhebrak, D. Vetrov, Y. Ivanenkov, V. Aladinskiy, P. Mamoshina, M. Bozdaganyan, A. Aliper, A. Zhavoronkov, and A. Kadurin, "Entangled conditional adversarial autoencoder for de novo drug discovery," *Molecular pharmaceutics*, vol. 15, no. 10, pp. 4398–4405, 2018.

[164] E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik, and A. Zhavoronkov, "Reinforced adversarial neural computer for de novo molecular design," *Journal of chemical information and modeling*, vol. 58, no. 6, pp. 1194–1204, 2018.

[165] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2224–2232, Curran Associates, Inc., 2015.

[166] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "Guacamol: Benchmarking models for de novo molecular design," *Journal of chemical information and modeling*, 2019.

[167] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, *et al.*, "Molecular sets (moses): A benchmarking platform for molecular generation models," *arXiv preprint arXiv:1811.12823*, 2018.

[168] D. Merk, L. Friedrich, F. Grisoni, and G. Schneider, "De novo design of bioactive small molecules by artificial intelligence," *Molecular Informatics*, vol. 37, no. 1-2, p. 1700153, 2018.

[169] A. T. Mller, J. A. Hiss, and G. Schneider, "Recurrent neural network model for constructive peptide design," *Journal of Chemical Information and Modeling*, vol. 58, no. 2, pp. 472–479, 2018. PMID: 29355319.

[170] M. Skalic, J. Jimnez, D. Sabbadin, and G. De Fabritiis, "Shape-based generative modeling for de novo drug design," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1205–1214, 2019. PMID: 30762364.

[171] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[172] "Smiles - a simplified chemical language." `https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html`. Accessed: 2019-06-13.

[173] A. Dalke, "Deepsmiles: An adaptation of smiles for use in," 2018.

[174] C. Selvaraj, S. K. Tripathi, K. K. Reddy, and S. K. Singh, "Tool development for prediction of pic 50 values from the ic 50 values-a pic 50 value calculator.," *Current Trends in Biotechnology & Pharmacy*, vol. 5, no. 2, 2011.

[175] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[176] G. Landrum, "Rdkit: open-source cheminformatics software," 2016.

[177] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR, abs/1211.5063*, vol. 2, 2012.

[178] "Why lstms stop your gradients from vanishing: A view from the backwards pass." `https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html`. Accessed: 2019-11-18.

[179] Y. Bengio, P. Simard, P. Frasconi, *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[180] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[181] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[182] "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed: 2019-06-12.

[183] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[184] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 1019–1027, Curran Associates, Inc., 2016.

[185] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison.," in *Interspeech*, vol. 13, pp. 1756–1760, 2013.

[186] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[187] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[188] P. Wu, T. E. Nielsen, and M. H. Clausen, "Fda-approved small-molecule kinase inhibitors," *Trends in Pharmacological Sciences*, vol. 36, no. 7, pp. 422 – 439, 2015.

[189] H. Patterson, R. Nibbs, I. McInnes, and S. Siebert, "Protein kinase inhibitors in the treatment of inflammatory and autoimmune diseases," *Clinical & Experimental Immunology*, vol. 176, no. 1, pp. 1–10, 2014.

[190] M. B. Sonbol, B. Firwana, A. Zarzour, M. Morad, V. Rana, and R. V. Tiu, "Comprehensive review of jak inhibitors in myeloproliferative neoplasms," *Therapeutic Advances in Hematology*, vol. 4, no. 1, pp. 15–35, 2013. PMID: 23610611.

[191] "Novadata solutions ltd.." `https://novadatasolutions.co.uk`. Accessed: 2019-07-02.

[192] A. P. Bento, A. Gaulton, A. Hersey, L. J. Bellis, J. Chambers, M. Davies, F. A. Krger, Y. Light, L. Mak, S. McGlinchey, M. Nowotka, G. Papadatos, R. Santos, and J. P. Overington, "The ChEMBL bioactivity database: an update," *Nucleic Acids Research*, vol. 42, pp. D1083–D1090, 11 2013.

[193] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[194] F. Chollet *et al.*, "Keras: The python deep learning library," *Astrophysics Source Code Library*, 2018.

[195] I. Jolliffe, *Principal Component Analysis*, pp. 1094–1096. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[196] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings," *Advanced Drug Delivery Reviews*, vol. 23, no. 1, pp. 3 – 25, 1997. In Vitro Models for Selection of Development Candidates.

[197] "Chemaxon stardizer, jchem 17.11.0, chemaxon," 2017.

[198] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, "Inchi, the iupac international chemical identifier," *Journal of Cheminformatics*, vol. 7, p. 23, May 2015.

[199] W. Walters and M. A. Murcko, "Prediction of drug-likeness," *Advanced Drug Delivery Reviews*, vol. 54, no. 3, pp. 255 – 271, 2002. Computational Methods for the Prediction of ADME and Toxicity.

[200] S. D. Pickett, D. V. S. Green, D. L. Hunt, D. A. Pardoe, and I. Hughes, "Automated lead optimization of mmp-12 inhibitors using a genetic algorithm," *ACS Medicinal Chemistry Letters*, vol. 2, no. 1, pp. 28–33, 2011. PMID: 24900251.

[201] G. Papadatos, M. Davies, N. Dedman, J. Chambers, A. Gaulton, J. Siddle, R. Koks, S. A. Irvine, J. Pettersson, N. Goncharoff, A. Hersey, and J. P. Overington, "SureChEMBL: a large-scale, chemically annotated patent document database," *Nucleic Acids Research*, vol. 44, pp. D1220–D1228, 11 2015.

[202] "Global online structure activity relationship (gostar)database." `https://gostardb.com/gostar/`, 2019.

[203] "Dihydroorotate dehydrogenase." `https://www.tocris.com/pharmacology/dihyroorotate-dehydrogenase`. Accessed: 2019-08-15.

[204] A. A. Joharapurkar, N. A. Dhanesha, and M. R. Jain, "Inhibition of the methionine aminopeptidase 2 enzyme for the treatment of obesity," *Diabetes, metabolic syndrome and obesity: targets and therapy*, vol. 7, p. 73, 2014.

[205] "Uniprotkb q13393 (pld1 human)." `https://www.uniprot.org/uniprot/Q13393`. Accessed: 2019-08-15.

[206] E. Bae, H. Lee, Y. Jang, S. Michael, E. Masliah, D. Min, and S.-J. Lee, "Phospholipase d1 regulates autophagic flux and clearance of α-synuclein aggregates," *Cell death and differentiation*, vol. 21, no. 7, p. 1132, 2014.

[207] S. K. Parks, Y. Cormerais, J. Durivault, and J. Pouyssegur, "Genetic disruption of the phi-regulating proteins na+/h+ exchanger 1 (slc9a1) and carbonic anhydrase 9 severely reduces growth of colon cancer cells," *Oncotarget*, vol. 8, no. 6, p. 10225, 2017.

[208] "Slc22a12." `https://www.proteinatlas.org/ENSG00000197891-SLC22A12/tissue`. Accessed: 2019-08-15.

[209] "Uniprotkb - q9z1m0 (p2rx7 mouse)." `https://www.uniprot.org/uniprot/Q9Z1M0`. Accessed: 2019-08-15.

[210] "Mmp12 matrix metallopeptidase 12 homo sapiens (human)." `https://www.ncbi.nlm.nih.gov/gene/4321`. Accessed: 2019-08-15.

[211] K. G. Pike and J. J. Morris, "Compounds-945," 2009. US Patent App. 12/170,128.

[212] A. P. Combs, R. B. Sparks, E. W. Yue, H. Feng, M. J. Bower, and W. Zhu, "Macrocyclic compounds and their use as kinase inhibitors," 2014. US Patent 8,871,753.

[213] R. H. Bradbury, A. A. Rabow, and N. J. Hales, "Bicyclic derivatives for use in the treatment of androgen receptor associated conditions-155," 2011. US Patent 8,003,649.

[214] G. J. Roth, M. Fleck, T. Lehmann-Lintz, H. Neubauer, and B. Nosse, "Compounds, pharmaceutical compositions and uses thereof," 2015. US Patent 9,006,450.

[215] S. Beaudoin, M. C. Laufersweiler, C. J. Markworth, B. E. Marron, D. S. Millan, D. J. Rawson, S. M. Reister, K. Sasaki, R. I. Storer, P. A. Stupple, N. A. Swain, C. W. West, and S. Zhou, "Sulfonamides derivatives," 2010.

[216] K. K. Childers, A. M. Haidle, M. R. Machacek, and A. B. Northrup, "Aminopyrimidines as syk inhibitors," 2014. US Patent 8,759,366.

[217] T. Yamagishi, K. Kawamura, Y. Arano, and M. Morita, "Arylamide derivatives as ttx-s blockers," 2016. US Patent 9,302,991.

[218] M. L. Curtin, B. K., S. Howard, R. H. , R. F. Clark, K. R., W. Omar, J. S. , C. Michaelides, Michael and, T. Anil, V. , H. Mack, T. M., H. Ramzi, S. , and M. A. Pliushchev, "Nampt and rock inhibitors," 2011.

[219] G. W. Bemis and M. A. Murcko, "The properties of known drugs. 1. molecular frameworks," *Journal of Medicinal Chemistry*, vol. 39, no. 15, pp. 2887–2893, 1996. PMID: 8709122.

[220] "Bemis-murcko clustering." `https://docs.chemaxon.com/display/docs/Bemis-Murcko+clustering`. Accessed: 2019-11-28.

[221] P. Willett, "Similarity-based virtual screening using 2d fingerprints," *Drug Discovery Today*, vol. 11, no. 23, pp. 1046 – 1053, 2006.

[222] D. Butina, "Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets," *Journal of Chemical Information and Computer Sciences*, vol. 39, no. 4, pp. 747–750, 1999.

[223] K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer, "Fréchet chemnet distance: a metric for generative models for molecules in drug discovery," *Journal of chemical information and modeling*, vol. 58, no. 9, pp. 1736–1741, 2018.

[224] G. B. Goh, C. Siegel, A. Vishnu, and N. Hodas, "Using rule-based labels for weak supervised learning: a chemnet for transferable chemical property prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 302–310, ACM, 2018.

[225] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[226] S. H. Bertz, "The first general index of molecular complexity," *Journal of the American Chemical Society*, vol. 103, no. 12, pp. 3599–3601, 1981.

[227] "Kullback-leibler divergence explained." `https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained`. Accessed: 2019-07-11.

[228] N. Cohen, "Revised group additivity values for enthalpies of formation (at 298 k) of carbon-hydrogen and carbon-hydrogen-oxygen compounds," *Journal of Physical and Chemical Reference Data*, vol. 25, no. 6, pp. 1411–1481, 1996.

[229] A. XU-WU and Y. RI-HENG, "Determination of heats of combustion and enthalpies of formation of ρ-chlorobenzoic acid, squalane and perchloro-m-dicyanobenzene by means of rotating-bomb combustion calorimetry [j]," *Acta Chimica Sinica*, vol. 8, 1982.

[230] M. Gastegger, C. Kauffmann, J. Behler, and P. Marquetand, "Comparing the accuracy of high-dimensional neural network potentials and the systematic molecular fragmentation method: A benchmark study for all-trans alkanes," *The Journal of chemical physics*, vol. 144, no. 19, p. 194110, 2016.

[231] M. A. Collins, M. W. Cvitkovic, and R. P. Bettens, "The combined fragmentation and systematic molecular fragmentation methods," *Accounts of chemical research*, vol. 47, no. 9, pp. 2776–2785, 2014.

[232] "Enamine ltd.." `https://enamine.net`. Accessed: 2019-07-03.

[233] "Asinex." `http://www.asinex.com`. Accessed: 2019-24-09.

[234] "Ambinter." `https://www.maybridge.com`. Accessed: 2019-24-09.

[235] "Princeton." `https://princetonchemical.com/`. Accessed: 2019-24-09.

[236] "Otava chemicals." `https://www.otavachemicals.com`. Accessed: 2019-24-09.

[237] "Chembridge." `https://www.chembridge.com`. Accessed: 2019-24-09.

[238] "Chemdiv." `http://www.chemdiv.com`. Accessed: 2019-24-09.

[239] A. Cereto-Massagu, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallv, and G. Pujadas, "Molecular fingerprint similarity search in virtual screening," *Methods*, vol. 71, pp. 58 – 63, 2015. Virtual Screening.

[240] C. J. Menet, S. R. Fletcher, G. Van Lommen, R. Geney, J. Blanc, K. Smits, N. Jouannigot, P. Deprez, E. M. van der Aar, P. Clement-Lacroix, L. Lepescheux, R. Galien, B. Vayssiere, L. Nelles, T. Christophe, R. Brys, M. Uhring, F. Ciesielski, and L. Van Rompaey, "Triazolopyridines as selective jak1 inhibitors: From hit identification to glpg0634," *Journal of Medicinal Chemistry*, vol. 57, no. 22, pp. 9323–9342, 2014. PMID: 25369270.

[241] W. Sherman, T. Day, M. P. Jacobson, R. A. Friesner, and R. Farid, "Novel procedure for modeling ligand/receptor induced fit effects," *Journal of Medicinal Chemistry*, vol. 49, no. 2, pp. 534–553, 2006. PMID: 16420040.

[242] F. Rininsland, W. Xia, S. Wittenburg, X. Shi, C. Stankewicz, K. Achyuthan, D. McBranch, and D. Whitten, "Metal ion-mediated polymer superquenching for highly sensitive detection of kinase and phosphatase activities," *Proceedings of the National Academy of Sciences*, vol. 101, no. 43, pp. 15295–15300, 2004.

[243] "Z'-lyte kinase assay kits." `https://www.thermofisher.com/uk/en/home/industrial/pharma-biopharma/drug-discovery-development/target-and-lead-identification-and-validation/kinasebiology/kinase-activity-assays/z-lyte.html`. Accessed: 2019-07-03.