Materials Design with Machine Learning

by

Ian Benlolo

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for
Master of Science in Physics

Department of Physics
Faculty of Science
University of Ottawa

Examining Committee

The following served on the Examining Committee for this thesis.

Internal Member: Adina Luican-Mayer
Professor, Department of Physics
University of Ottawa

Internal Member: Stefanie Czischek
Assistant Professor, Department of Physics
University of Ottawa

Supervisor(s): Isaac Tamblyn
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

In the quest to advance materials design, this thesis integrates Machine Learning (ML) techniques with Density Functional Theory (DFT) data. A novel representation called splashdown is formulated to capture long-range interactions, an aspect often neglected by material representations. A project known as ORGANIZER leads to the creation of a pivotal database, culminating in the discovery of a new organic solid-state lasing molecule that doubled the state-of-the-art emission gain cross-section. Concurrently, a monte-carlo based optimizer, aMC, is tested, demonstrating superior performance to gradient-based methods without the need for expensive gradient computation. Enhanced Graph Neural Networks (GNN)s predict High Entropy Alloy (HEA) catalysts for oxygen reduction reaction, halving necessary DFT computations and unveiling a new HEA catalyst with a 0.27V overpotential. The splashdown representation compares to state-of-the-art ones like MBTR and SOAP in predicting long-range interactions. Collectively, these efforts highlight the transformative potential of ML and some adjacent fields in materials science.

Acknowledgements

I would like to express my deepest appreciation to all those who have provided the possibility to complete this thesis. I would like to start by extending my sincere gratitude to my supervisor, Isaac. His enduring support, guidance, and insightful critiques have played a crucial role in my journey. His kind nature coupled with brutal honesty pushed me to elevate my work and broaden my perspective. The opportunity to work in this exhilarating field under his supervision has truly been a privilege.

I am deeply indebted to Prof. Stephen Whitelam, who led the aMC project. His teachings about efficient and neat work, coupled with the invaluable lesson of consistently revisiting the fundamental question - "What am I trying to show?" - have profoundly shaped my approach to scientific inquiry.

I am grateful for my colleagues at the CLEAN lab, who were always ready for constructive and enriching discussions. Their insights and inputs often gave my work fresh angles and new directions. Their presence significantly enhanced the vibrancy of the academic environment.

A big thanks to the National Research Council of Canada, the University of Ottawa, NSERC, and the Vector Institute for funding my studies and compute resources.

And last but certainly not least, my heartfelt thanks go out to my family, friends and loved ones. Their unflinching support and occasional distractions in the form of laughter and camaraderie helped keep my spirits high during the intensive periods of my studies. The role they played in this academic endeavour is beyond measure and I am forever thankful.

The journey of completing this thesis has been an enlightening one, thanks to all of you. This accomplishment would not have been possible without your unwavering faith and constant encouragement.

Table of Contents

List of Tables

List of Figures

Abbreviations

ACSF Atom-Centred Symmetry Functions 68, 69

BO Born-Oppenheimer 5, 6

CGCNN Crystal Graph Convolutional Neural Networks 41

CM Coulomb Matrix 67, 70–72, 91, 122

CME Coulomb Matrix Eigenvalue 70, 71, 92

CNN Convolutional neural networks 30, 32, 37

DFT Density Functional Theory iv, vi, 5–7, 10, 36, 87, 89, 94

GBT Gradient Boosted Trees 42, 43

GCN Graph Convolutional Network 39–41

GD Gradient Descent 43–45, 54

GGA Generalized-Gradient Approximations 9, 10

GNN Graph Neural Networks iv, 32, 33, 37, 39, 40

HEA High Entropy Alloy iv, 2, 3, 12, 17–19, 87, 94

HF Hartree-Fock 6, 9

HK Hohenberg-Kohn 6, 7

HPLC High Performance Liquid Chromatography 63, 64

IID Independent and Identically Distributed 48

KRR Kernel Ridge Regression 42, 91

LDA Local Density Approximation 9

LSDA Local Spin Density Approximation 10

MBTR Many-Body Tensor Representation 65, 70–72, 90–92, 122

ML Machine Learning iv, 50, 65, 66, 88, 94

MLIP Machine-Learned Interaction Potentials 73, 74

MPNN Message Passing Neural Network 37, 39

MS Mass Spectrometry 63, 64

NN Neural Network x, 27, 52, 53, 82

OER Oxygen Evolution Reaction 18, 20

ORR Oxygen Reduction Reaction 3, 12, 18–20

OSLD Organic Solid-State Lasing Devices 56, 57, 60, 62, 63

PBE Perdew-Burke-Ernzerhof 10, 36

RNN Recurrent Neural Network 30, 37, 84, 86

RSS Residual Sum of Squares 24, 25

SGD Stochastic Gradient Descent 45–47

SOAP Smooth Overlap of Atomic Positions 65, 68, 92

VASP Vienna *Ab-Initio* Simulation Package 10, 11

XC Exchange-Correlation 8–10

Chapter 1

Introduction

Two overarching themes in science have nearly always been **learning** and **modelling**. Christopher Bishop notes their difference in the book "Pattern Recognition and Machine Learning" [19] with the two classic examples; Kepler and Newton. After extensive amounts of observations, Kepler was able to carefully fit his empirical observations and come up with the laws of planetary motion. Newton's three laws of planetary motion are a classic example of learning - or pattern recognition. Newton, however, laid out the foundations of classical mechanics and brought immense advancements in the understanding of the classical world - modelling. Kepler's equations, introduced around 100 years before Newton's, can also be derived from his fundamental laws; hence the scientists' inclination towards first principles. The opposite statement can be made when extending from two-body to three or more body systems where we run into chaotic behaviours and the first-principles viewpoint fails us.

Kepler and Newton's examples demonstrate that there are two main approaches to modelling: learning-based and first principles-based. Learning-based methods offer practical solutions for handling vast amounts of data, but they have the downside of producing limited principles, making them less universal and transferable. In contrast, first principles-based modelling provides a deeper understanding and greater universality, but it can become expensive, or even infeasible to apply directly to practical problems when situations become complicated.

These methods are in no way mutually exclusive. We can refer to Galileo who allegedly dropped objects of different mass from the leaning tower of Pisa. He leveraged both methods to transform the world of physics. Using empirical data - the times of the fall - he constructed a model of motion: a learning-based method akin to Kepler's. He then extended this model to conceptualize the laws of falling bodies, a principle that serves

as one of the fundamental premises in Newton's law of motion: a first principles-based approach. This intertwining of learning and modelling has remained a cornerstone of scientific exploration. Galileo's approach illustrates how empirical data and first principles can be combined to create models that are both practical and theoretically profound.

The modelling approach is appealing, but it faces practical challenges when dealing with complex situations. This is partly due to the inability to effectively represent complex models and functions, particularly in molecular dynamics studies where accurately representing interatomic potential energy has proven challenging. As a result, expensive quantum mechanical methods are often needed for high accuracy. However, in recent years, machine learning, especially deep learning, has emerged as a promising solution to overcome these limitations. This presents a unique opportunity to reassess the theoretical foundations of various scientific fields that involve modelling and learning, either to develop new methodologies or to critically evaluate existing methodologies and solve previously intractable problems.

1.0.1 The Energy Crisis

This brings us to a seemingly intractable challenge: solving the global energy crisis. Based on current trends, greenhouse gases are predicted to cause catastrophic environmental changes if global mean temperatures are not limited to an increase of 1.5°C by the year 2030. This limit requires the transformation of almost all energy systems; from the way we power our economies to the way we feed our growing populations [22, 202]. Despite a significant decrease in greenhouse gasses (1.1% annual growth from 2010 to 2019 compared to 2.6% from 2000 to 2009[1]), the dependence on fossil fuels remains unsustainable. It is a cause of great environmental concern. To avoid catastrophes, the shift towards renewable sources of energy must accelerate globally. Novel and more efficient energy storage mechanisms and materials are necessary to do so.

Then, a novel type of material shockingly appeared which may be the key to solving this problem. In 2004, two principal investigators shook the whole of the materials science community with their papers on materials composed of seemingly random combinations of elements: the Yeh et al. group studying CuCoNiCrAlxFe [245] and the Cantor et al. group studying FeCrMnNiCo [32]. This in turn sparked nearly 20 years of research efforts into what is known as HEAs.

Material alloying has, throughout human history, been an essential tool in everyday life. Many of the techniques employed in the past followed certain conventions, which in general

[1] Emissions Gap Report 2022, UNEP

started with a principal component and added small amounts of various other components to enhance materials as desired. A large difference with HEAs is the number of different components and their concentrations. These tend to be composed of more than 4 different elements at around equal concentrations. The reason for this will be discussed at more length in chapter Section 2.3.

These materials immense potential in various applications due to their unique properties, such as high strength, ductility, and resistance to wear and corrosion [154]. These properties are attributed to their complex, multi-component nature, which leads to a high configurational entropy and promotes the formation of simple, solid-solution structures as opposed to fragile intermetallics [246,254]. In the context of energy storage, HEAs have recently attracted attention as potential catalysts for the oxygen reduction reaction Oxygen Reduction Reaction (ORR), a key process in fuel cells and metal-air batteries [243].

The ORR is a fundamental electrochemical reaction that involves the reduction of oxygen to water or hydroxide ions, releasing a large amount of energy in the process. However, the ORR is kinetically hindered, requiring a catalyst to enhance its efficiency. Traditionally, precious metals such as platinum have been used as catalysts for these reactions, but their scarcity and high cost limit their widespread use [223] and justify research in cheaper and better alternatives.

While a breakthrough for solving quantum mechanics problems has not yet been achieved, we have successfully applied deep learning techniques to molecular modelling at the nanoscale, leading to small advancements in materials design for clean energy storage. This thesis discusses some work done in this search for new electrocatalysts, with High Entropy Alloys as the candidate surfaces. A large focus is on the machine learning methods employed. In an adjacent field, machine learning optimizers will be explored as they have a large hand in training these aforementioned deep learning networks. Lastly, another problem will be tackled; the storage of these vast amounts of molecule data efficiently. Thus, the goal of this dissertation is to offer a clear demonstration of these technologies and present some important applications made possible by these tools.

In the following chapters, we first introduce the background and methods in molecular modelling. We will then briefly start with linear models and the least squares method in Section 2.5.1, to set a basis for explaining machine learning techniques in Section 2.5.3. Then, we will discover some representation learning techniques which can be a very powerful tool in modelling physical systems. This will lead us to learn about graph neural networks (Section 2.5.4). To tie all these together, in Section 2.5.6 we discover the backbone of all these methods; optimizers. Lastly, we learn about databases and their role in science and molecule discovery with self-driving labs and why there is a need for **molardb**.

Chapter 2

Methods

2.1 Density Functional Theory

The electronic structure of solids is a very broad term, yet it can be simply defined as the wavefunctions and energies associated with electrons. This definition, however, masks the incredibly complex nature of interactions between electrons and nuclei that govern the wavefunctions in the solid. There are on the order of 10^{23} individual particles that interact with each other to form a solid, and all these interactions contribute to the electronic structure. However difficult, it is of great importance to our daily lives to understand the electronic structure of materials.

The electronic structure governs most properties of materials. Whether it is a conductor of electricity, an insulator, or somewhere in between (semiconductor), its colour, hardness, thermal conductivity, optical properties, and magnetic properties to name a few, are all determined by how the electrons and nuclei interact [147].

Any way we see it, to understand the physical properties of materials we must first understand the quantum methodologies that govern them. To do so, it is almost satisfactory to start with the many-electron Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},\mathbf{R},t) = \hat{H}\Psi(\mathbf{r},\mathbf{R},t). \tag{2.1}$$

The Hamiltonian, $\hat{H}$, is defined as

$$\begin{aligned}\hat{H} = &-\frac{\hbar^2}{2m_e}\sum_i \nabla_i^2 - \frac{\hbar^2}{2}\sum_j \frac{\nabla_j^2}{m_j} \\ &- \frac{e^2}{4\pi\epsilon_0}\sum_{i,j}\frac{Z_j}{|\mathbf{r}_i - \mathbf{R}_j|} \\ &+ \frac{e^2}{8\pi\epsilon_0}\sum_{i\neq i'}\frac{1}{|\mathbf{r}_i - \mathbf{r}_{i'}|} + \frac{e^2}{8\pi\epsilon_0}\sum_{j\neq j'}\frac{Z_j Z_{j'}}{|\mathbf{R}_j - \mathbf{R}_{j'}|}\end{aligned} \tag{2.2}$$

where $\mathbf{r} = (r_1, \ldots, r_N)$ are the coordinates of the electrons and $R = (R_1, \ldots, R_M)$ the nucleus in a d-dimensional Euclidean space. m_j denotes the mass of the nucleus, m_e the mass of an electron and Z_j the nuclear charge. i's denote electron indices and position vectors and j's denote nuclei and the indices span to N_e and N_i respectively. ∇ denotes the gradient operator, $\hbar$ the reduced Planck constant and ϵ_0 the permittivity of free space, also known as the electric constant. The wavefunction, $\Psi(\mathbf{r}, \mathbf{R}, t)$, is normalized to 1. We use the atomic unit here, and the spin indices are omitted for simplicity. Equation (2.2) consists of three lines. The first represents the kinetic energy of the electrons and nuclei, the second the Coulomb interaction between electrons and nuclei, and finally the Coulomb interactions among all electrons and all nuclei.

DFT is a widely used quantum theory widely employed in condensed matter physics, material science, computational physics, and quantum chemistry to forecast mechanical and electronic properties of materials [171, 186, 229]. We use DFT to construct the Hamiltonian of a material at equilibrium. In the following, we describe the basic theories underlying DFT.

2.1.1 The Born-Oppenheimer and Hartree-Fock approximations

In Equation (2.2), the only term that can be considered small is $1/M_j$. Ions vibrate around their equilibrium positions, while electrons move around their corresponding ions and since ions are much heavier than electrons (more than 1000 times), they tend to move much slower than electrons [28]. This implies that the electrons can be considered as particles that follow the nuclear motion adiabatically, meaning that they are "dragged" along with the nuclei without requiring a finite relaxation time.

This leads to the Born-Oppenheimer (BO) ansatz, which assumes that ion motion is much slower than an electron in motion so that ions can be regarded as fixed in space (in

other words setting M_j to ∞) when solving electron motion. This way, the electrons can be treated as moving in an external potential generated by the "frozen" ions. As a result, the system's ground state wave function Ψ can be factorized into an electronic part and a nuclear part, allowing the two components to be treated separately. More concisely,

$$\Psi(\mathbf{r}, \mathbf{R}) = \Psi_e(\mathbf{r}, \mathbf{R})\Psi_c(\mathbf{R}), \tag{2.3}$$

where the subscripts e and c denote electronic and nuclear respectively. Ψ_e depends on the electronic spatial coordinates and the nuclear positions, and Ψ_c depends only on the nuclear positions. This gives rise to a new Hamiltonian:

$$H_{BO} = -\frac{\hbar}{2m_e}\sum_i \nabla_i^2 + \frac{e^2}{8\pi\epsilon_0}\sum_{i\neq i'}\frac{1}{|\mathbf{r}_i - \mathbf{r}_{i'}|} - \frac{e^2}{4\pi\epsilon_0}\sum_{i,j}\frac{Z_j}{|\mathbf{r}_i - \mathbf{R}_j|} \tag{2.4}$$

Although this method simplifies the problem, describing the motion of numerous electrons in an equilibrium potential field is still a complex many-body problem. As a result, further simplification is needed to solve the many-body Schrödinger equation for systems containing many electrons.

As the BO method remains intractable, we introduce another common approximation, the Hartree-Fock (HF) method [94]. Here, we simplify the interactions between electrons even more. This method assumes that each electron moves in the average field created by all the other electrons, rather than considering the interactions between individual pairs of electrons explicitly. This reduces the problem to a single-electron one, making it easier to solve.

We keep this explanation short as much more can be found in [147]. We will, however, note that the Hartree-Fock method has a major limitation in that it only considers the average effect of electron-electron repulsion and neglects instantaneous repulsion. We therefore now introduce more recent methods introduced in 1964 by Pierre Hohenberg and Walter Kohn.

2.1.2 Hohenberg-Kohn Theorem

The Hohenberg-Kohn (HK) theorem [102] introduced in the 1960s greatly reduced the complexity of the electronic Schrödinger and provides the foundation for DFT. The HK theorem is surprisingly simple. It reduces the fully interacting many-electron problem

further to determine the ground state single-particle density

$$\rho(\mathbf{r}) = N \sum_{i=2}^{N} \int d\mathbf{r}_i \Psi^*(\mathbf{r}_i)\Psi(\mathbf{r}_i) \tag{2.5}$$

where N is the number of electrons and Ψ is the electronic wavefunction. The HK theorem states that there exists a one-to-one correspondence between the external potential and the ground-state electronic density, meaning that a given density uniquely determines the potential and vice versa. Formally,

$$E = E[\rho(\mathbf{r})] \tag{2.6}$$

where $\rho(\mathbf{r})$ is the electron density. Hohenberg and Kohn's theorem also states that all properties of a system, including excited state properties, can be expressed as exact functionals[1] of the ground state electronic density. This means that once the ground state electronic density is known, all other properties of the system can be determined exactly. Therefore we obtain all properties of a system by minimizing a unique, universal energy functional $E[\rho(\mathbf{r})]$ which is the focus of DFT. Proofs of these are quite simple and can be seen in the original paper [102] or in Chapter 6 of [147]. The total energy function generally takes the form

$$E[\rho(\mathbf{r})] = T[\rho(\mathbf{r})] + \frac{1}{2}\int d\mathbf{r}d\mathbf{r'}\frac{\rho(\mathbf{r})\rho(\mathbf{r'})}{|\mathbf{r}-\mathbf{r'}|} + \int d\mathbf{r}\rho(\mathbf{r})V_{\text{ext}}(\mathbf{r}) + E_{XC}[\rho(\mathbf{r})]. \tag{2.7}$$

The first term is the kinetic energy of the system, the second term is the Hartree energy otherwise known as the Coulomb interaction energy. The third term is the electric energy caused by the electron-nucleus interaction and the final term represents the exchange (E) and correlation (C) energy. This accounts for all unknown physics. Exchange energy refers to the energy required to exchange the position of two identical particles, such as electrons, due to the Pauli exclusion principle[2]. Correlation energy refers to the energy associated with the interaction between electrons that cannot be accounted for by the classical Coulombic interactions.

We now have a function to minimize Equation (2.7), however there remain two un-

[1] A functional is simply a real-valued function of a function.

[2] Simply put, the Pauli Exclusion Principle states that no two identical fermions (particles with half-integer spin) can occupy the same quantum state simultaneously within a quantum system - in other words, have the same four quantum numbers. This leads to a specific symmetry property of the total wave function of the system: it must be antisymmetric with respect to the exchange of any pair of electrons. Meaning if you swap two electrons, the wave function changes sign.

knowns; $T[\rho(\mathbf{r})]$ and $E_{XC}[\rho(\mathbf{r})]$. This brings us to further approximations.

2.1.3 Kohn-Sham Equation

Kohn and Sham proposed a system to replace the complicated interacting many-body system with one that can be solved more easily to subsequently minimize Equation (2.7) [125]. The auxiliary system they proposed is composed of **non-interacting** electrons but possesses the same electron density as the original system. In other words, the Kohn-Sham (KS) equations replace the kinetic energy of interacting particles in a system with the kinetic energy of non-interacting particles that share the same density. Any discrepancies between these kinetic energies are then factored into the exchange-correlation energy term, enabling the ground state density of an interacting system to be expressed as a sum of contributions from N independent orbitals. The density is formally described as

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} \varphi_i^*(\mathbf{r})\varphi_i(\mathbf{r}), \tag{2.8}$$

and the full interacting many-body problem for the ground-state energy function is

$$E_{KS} = T_S[\rho(\mathbf{r})] + \int d\mathbf{r} V_{\text{ext}}\rho(\mathbf{r}) + E_{\text{Hartree}} + E_{II} + E_{XC}(\rho(\mathbf{r})). \tag{2.9}$$

Similarly to Equation (2.7), V_{ext} is the external potential due to the nuclei and any other external fields and E_{II} is the interaction between the nuclei. Comparing the Hohenberg-Kohn (Equation (2.7)) and Kohn-Sham (Equation (2.9)) equations, E_{XC} can be written as

$$E_{XC} = T[\rho] + E_{\text{int}}[\rho] - T_s[\rho] - E_{\text{hartree}}[\rho]. \tag{2.10}$$

Here square brackets denote functionals. This shows that E_{XC}, the Exchange-Correlation (XC) energy, is just the difference between the kinetic and the internal interaction energies of the true interacting many-body system from those of the fictitious independent-particle system with electron–electron interactions replaced by the Hartree energy. With a proper $E_{XC}[\rho]$ describing the true exchange and correlation energies, the KS method provides a feasible way of approximating ground-state properties of the many-body electron system.

2.1.4 Exchange-Correlation Potential

Another way of describing the XC energy is by describing it as a hole. This "hole" denotes the decrease in the probability of finding another electron in the region around each electron. This is due to the Pauli exclusion principle and the correlation caused by repulsive Coulomb interaction [147].

The KS equation is simpler and more rigorous than the HF approximation. The HF approximation replaces the ground state of a many-electron problem with an auxiliary independent-particle problem. The KS equation has led to significant approximations that are the basis of most calculations aiming to make predictions based on first-principles calculations. The equation can only be practically soluble in computational science by obtaining an accurate and easy-to-express XC term, which is presumably the key to accurately solving quantum many-body problems. Great effort has been made to explore the representations of XC energies. The Local Density Approximation (LDA) [35,177,216] and Generalized-Gradient Approximations (GGA) [138,174,175] are the most notably and remarkably accurate ones among a variety of different approximation methods that have been proposed.

In 1965, Kohn and Sham proposed LDA in which they made the approximation that solids can be treated like a homogenous electron gas. In this limit, the effects of exchange and correlation are known to be local in character, which is the basis for LDA. The XC energy is then an integral over all space with the XC density at each point assumed to be the same as in a homogenous electron gas with the same density. The energy is first split into the exchange and correlation

$$E_{XC}^{LDA}[\rho(\mathbf{r})] = \int d\mathbf{r}\rho(\mathbf{r})\epsilon_{XC}^{LDA}[\rho(\mathbf{r})], \tag{2.11}$$

where $\epsilon_{XC}[\rho(\mathbf{r})]$ is generally split to $\epsilon_{XC}^{LDA}[\rho(\mathbf{r})] = \epsilon_{X}^{LDA}[\rho(\mathbf{r})] + \epsilon_{C}^{LDA}[\rho(\mathbf{r})]$.

The LDA approach is most suitable for systems with relatively smooth densities. An extension to LDA is the spin-polarized L(S)DA which includes the spin polarization $E_{XC}^{LSDA}[n^{\uparrow}, n^{\downarrow}]$. The LDA tends to underperform in strongly correlated systems and those with highly varying densities.

Due to this big limitation, as electron densities are typically not uniform, the GGA methods were introduced. Initially mentioned in the original paper by Kohn and Sham as "gradient expansion approximation" and later developed by Herman et al. [98], these methods introduce the gradient of the density $\nabla\rho(\mathbf{r})$ on top of the notion of local density. The first step that can be taken beyond the local approximation is including a functional of

the magnitude of the gradient of the density $|\rho^\sigma|$. This was first mentioned in the original paper by Kohn and Sham. This however does not lead to significant improvement over Local Spin Density Approximation (LSDA)- it often leads to worse results as the gradients in real materials are large and the expansion breaks down [147].

The most common GGA functionals are the Perdew-Burke-Ernzerhof (PBE) [174] and Perdew-Wang (PW91) [176]. GGAs can be thought of as a Taylor expansion with LSDA as a starting point. We define the functional as a generalized form of Equation (2.11):

$$E_{XC}^{GGA}[\rho(\mathbf{r})] = \int_\Omega d\mathbf{r}\rho(\mathbf{r})\epsilon_x^{\text{hom}}(\rho)F_{XC}(\rho, |\nabla\rho|, \dots) + \epsilon_{\text{c}}^{\text{hom}}(\rho)F_{\text{c}}(\rho, |\nabla\rho|, \dots) \tag{2.12}$$

where "hom" is short for homogeneous, referring to the homogeneous electron gas. $\epsilon_{\text{x/c}}$ is the exchange/correlation energy energy per particle and F is a dimensionless function. In the PBE functional, only the first derivative is computed.

2.1.5 DFT in practice

With the ?? equation and when the XC functional is determined the next step is to deal with the external potential that the system is subjected to. We use pseudopotentials to circumvent the computational complexity that arise from modelling the interactions between core electrons and the atomic nucleus. These pseudopotentials effectively replace the original coulomb potential term in the Schrodinger equation with a modified effective potential. Various methods have been introduced to deal with this, such as empirical [44], norm-conserving [91] and ultrasoft pseudopotentials [230] among others, the last one being the most common form. The external potential is important because it sets the stage for the entire electronic structure problem. It defines the basic environment in which electrons are placed, and around which they move. Once you know the external potential, along with the XC functional, you can solve the Kohn-Sham equations to find the electron density and, subsequently, various ground-state properties of the system.

DFT calculations are supported by various packages, namely the Vienna Vienna *Ab-Initio* Simulation Package (VASP) [126–129], ABINIT [82], Quantum ESPRESSO [74], WIEN2K [20] etc. Currently, VASP is the most commonly used package in materials science due to its efficiency, its vast computational methods and its large number of implemented pseudopotentials. The self-consistent method starts with an initial guess for the electron density. The effective potential that corresponds to this density is then calculated and substituted back into the KS equation to solve for the wavefunction. The wavefunction is then used to calculate a new electron density, which is compared with the initial guess.

If the difference between the initial guess and newly computed electron density meets a convergence criterion, VASP exits the calculation and outputs the results. Otherwise, the iterative procedure is continued with the last computed electron density instead of the initial guess. This iterative method is depicted in Figure 2.1.

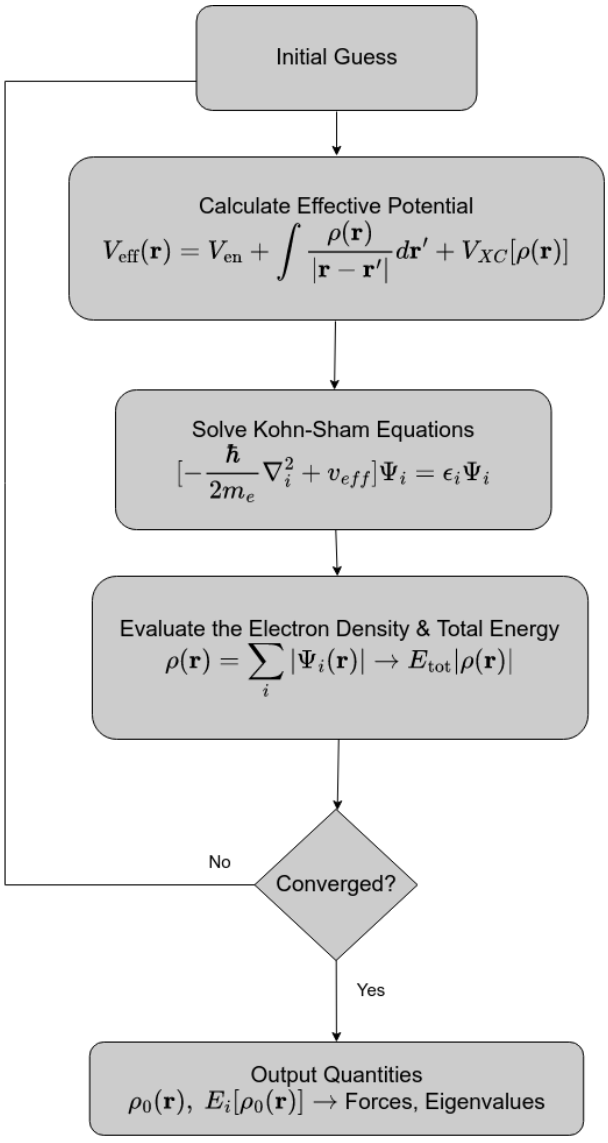


Figure 2.1: Flowchart of the self-consistent, iterative process of VASP for solving for the ground state of the Kohn-Sham equations. Flowchart adapted from [147].

2.2 Alloys, Crystals

Crystalline solids may be described as ordered repetitions of atoms or groups of atoms in three dimensions, with an infinite assortment of atoms (which we call periodic). Because of that intractable amount of atoms, reliable ab-initio DFT computations of these systems are unassailable. However, crystals are structures with translational periodicity governed by geometry [199] of a repeating motif, which can be leveraged to greatly simplify calculations. In the following, we describe the structural properties of alloys and crystals, the different types of lattices and how and why they are used. We then introduce HEAs, a new class of alloys gaining interest in materials science for their fascinating properties. Lastly, we introduce the ORR and summarize research on HEAs for ORR.

2.2.1 Lattice Structure

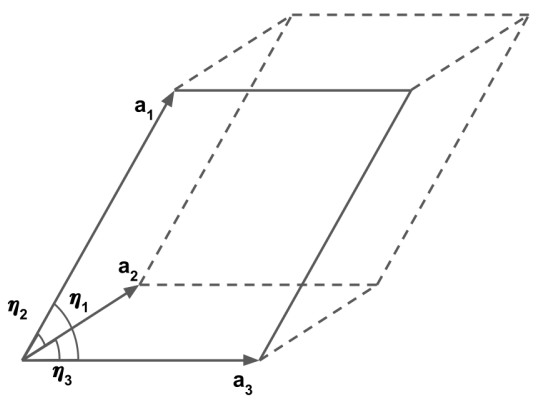
Lattice nodes are what we call the points on the three-dimensional lattice formed by the translation vectors T_n, where n is some integer $1, \dots d$.

$$T_n = T(n_1, n_2, \dots) = n_1\vec{a_1} + n_2\vec{a_2} + n_3\vec{a_3} + \dots \tag{2.13}$$

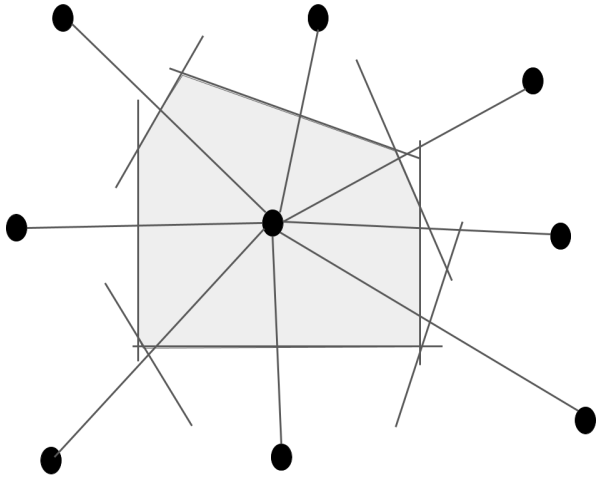
defines the set of all translations in space where the vectors $\vec{a_i}$ define the basis of n-space. An ideal crystal has an identical repeating unit which is related to this global symmetry operation. For simplicity and utility, we mainly worry about three-dimensional space. a_i in Equation (2.13) are called the *primitive translation vectors*. We usually use these to define crystal axes, however, other axes may be used when they have a simple relation to the symmetry of the structure.

Translations in one dimension can be expressed as a multiple of the periodicity length a; $T(n) = na$ where n can be any integer. The primitive cell, which is any cell of length a, can be chosen freely, but the most symmetric cell is the one that is symmetric about the origin $(-a/2, a/2)$ (in two dimensions). This particular choice ensures that each cell centred on a lattice point n encompasses all the points that are closer to that lattice point than to any other point.

This is a classic example of constructing the *Wigner–Seitz cell*, which is defined as the volume that encloses all points in space that are closer to this particular lattice point than to any other. The Wigner-Seitz cell is (in real space) the most symmetric and compact cell that can be constructed around a lattice point, and it is often used to describe the properties of crystals [105, 147]. This can be visualized in Figure 2.2(b). The Wigner-Seitz cell can fill all space just as the unit cell can.



(a) Unit cell with lattice constants (a_1, a_2, a_3) and (η_1, η_2, η_3).



(b) The Wigner-Seitz primitive cell.

Figure 2.2: Primitive cells. (a) the primitive cell of a space lattice in three dimensions, (b) the Wigner-Seitz cell construction, which is done by (1) drawing lines to connect lattice points to all nearby lattice points; (2) at the midpoint and normal to these lines, drawing new lines or planes. The smallest volume enclosed by these lines is the Wigner-Seitz primitive cell.

A common choice for a two-dimensional lattice is the parallelogram constructed from the two primitive translation vectors. The final parallelepiped defined by the three basis vectors is called the unit cell. The lattice constants are the three-unit lengths a_1, a_2, a_3 and three angles η_1, η_2 and η_3 between these vectors, as shown in Figure 2.2(a). Several unit cells can be defined, wherein the smallest one is called the *primitive cell*. The crystal structure is described by repeating the unit cell by an infinite set of vectors. To express an atom's coordinates, we must use fractional coordinates of the unit cell. So a point in the center of the cell, regardless of the cell dimensions and shape, is at $\tau = (0.5, 0.5, 0.5)$, and a point at the origin is at $(0, 0, 0)$.

In reciprocal space, the lattice is defined by three vectors a_1', a_2' and a_3' which are derived using the equations

$$a_1' = 2\pi \frac{a_2 \times a_3}{V}, \; a_2' = 2\pi \frac{a_3 \times a_1}{V}, \; a_3' = 2\pi \frac{a_1 \times a_2}{V}, \tag{2.14}$$

where $V = a_1 \cdot (a_2 \times a_3)$, the scalar triple product. This is, geometrically, the volume of the parallelepiped in Figure 2.2(a). The equivalent of a unit cell in reciprocal space is called the *first Brillouin Zone*.

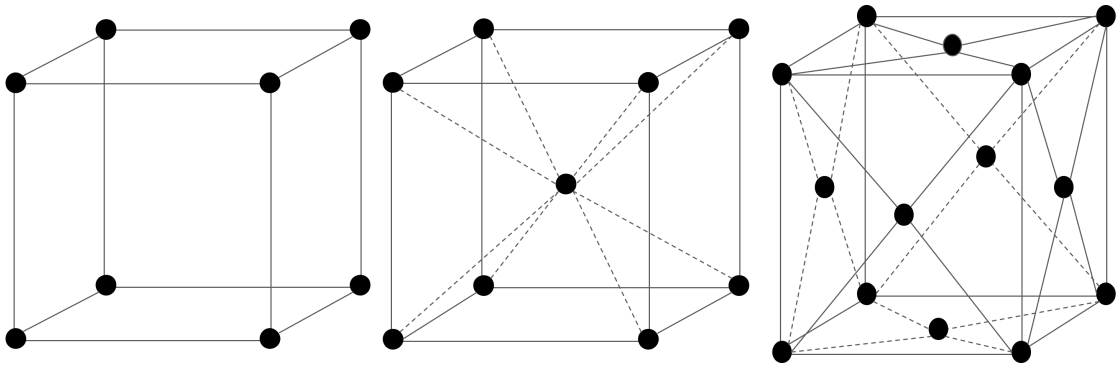
2.2.2 Types of Lattices

Crystal lattices can be mapped to themselves by not only a translation operation but by symmetry operations as well. A typical example is a rotation about an axis passing through a lattice point. In doing so, we form what is called a *point group*, since this operation forms a group of lattices that leave one point fixed.

In three dimensions, there are seven basic types of symmetries that are observed in a crystal lattice. These systems are defined based on the lengths and angles between the edges of the unit cell that define the crystal lattice and amount to 14 lattice types. The set of translations, which generates the entire periodic crystal by repeating the basis (the positions and types of atoms in the primitive cell), is a lattice of points in space called the *Bravais lattices*. Each lattice type also has a number of associated lattice structures. These are listed here:

1. Cubic system: characterized by three axes of equal length at right angles to each other. The cubic system contains 3 lattices which can be seen in Figure 2.3.

2. Tetragonal system: characterized by three axes, two of which are equal in length and



(a) sc lattice. (b) bcc lattice. (c) fcc lattice.

Figure 2.3: The cubic space group. (a) shows the simple cubic (sc) lattice, (b) shows the body-centred cubic (bcc) lattice and (c) shows the face-centred cubic (fcc) lattice.

perpendicular to each other, while the third axis is longer or shorter and perpendicular to the other two. The tetragonal system contains 2 lattices.

3. Orthorhombic system: characterized by three axes of different lengths, all perpendicular to each other. The orthorhombic system contains 4 lattices.

4. Monoclinic system: characterized by three axes of different lengths, two of which intersect at an oblique angle, while the third is perpendicular to the plane of the other two. The monoclinic system contains 2 lattices.

5. Triclinic system: characterized by three axes of different lengths, all intersecting at oblique angles. The triclinic system contains 1 lattice and is the general lattice.

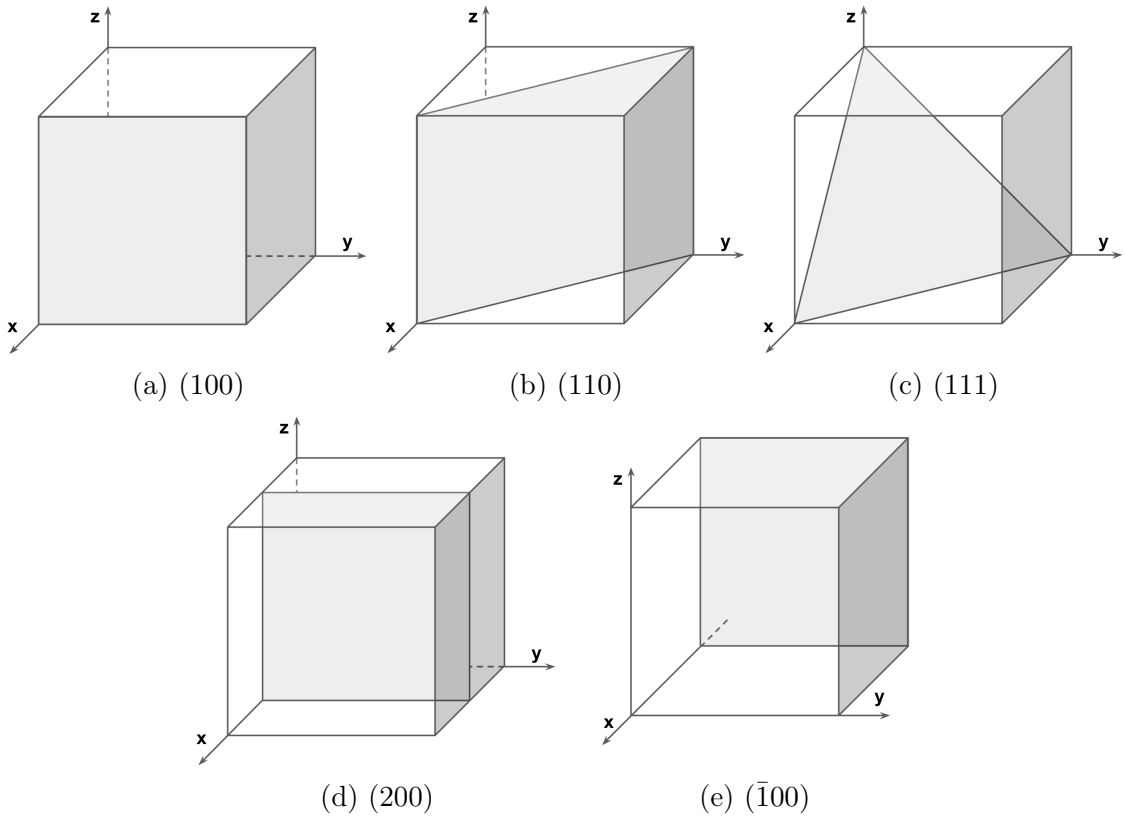
6. Hexagonal system: characterized by four axes, three of which are in a common plane and intersect at 120 degrees, while the fourth axis is perpendicular to this plane. The hexagonal system contains 1 lattice.

7. Trigonal (or rhombohedral) system: characterized by three axes of equal length, all intersecting at angles that are not 90 degrees. The trigonal system contains 1 lattice.

In addition to the seven lattice systems and 14 Bravais lattices, there amount to 230 crystallographic space groups in three dimensions. These are the group formed by the translation and point operations (rotations, reflections, inversions about a point). Having this classification system is advantageous in numerous ways. For example, these can be used for predicting certain physical or chemical properties as certain space groups exhibit

similar properties. One notable example is the family of perovskite materials, which are orthorhombic structures that tend to have a similar cubic phase at high temperatures [86]. Moreover, this classification system may be used for organizing and searching crystals in large databases.

2.2.3 Index System for Crystal Planes



(a) (100) (b) (110) (c) (111)

(d) (200) (e) ($\bar{1}$00)

Figure 2.4: Indices of important planes in the cubic crystal unit cell. The planes (100), (200) and ($\bar{1}$00) in (a), (d), (e) respectively are parallel.

Crystalline solids exhibit anisotropy, which means that they have different physical properties when measured in different directions. To simplify the identification of specific planes where certain physical properties can be observed, crystal faces are described using *Miller indices*. These indices are three whole numbers that describe the orientation perpendicular to a plane. We typically use the symbols h, k, and l to represent these indices. In

classical crystallography, it is logical to restrict our focus to planes where the indices have no common factors since the orientation of the crystal face is the only relevant information and not its distance from an arbitrary origin [147].

To study Miller indices, we first start with *Weiss parameters*, originally introduced in the early 19^{th} century [237]. These are a set of three numbers used to describe the relationship between crystal faces or planes and the crystallographic axes. The sequence of numbers always follows the pattern of $a : b : c$, where a, b, and c represent the orientation of the planes with respect to the a-axis, b-axis, and c-axis, respectively. When a unit face or plane intersects all three crystallographic axes at ratios that correspond to their axial ratios, it has Weiss parameters of $(1 : 1 : 1)$. When a plane is parallel to one or two crystallographic axes, the Weiss parameter is infinity (∞) since the plane never intersects the axis.

The Miller indices and Weiss parameters are related in a reciprocal manner. The Miller indices of a face or set of planes can be obtained by taking the reciprocal of its Weiss parameters. To do this, the Weiss parameters are inverted and then multiplied by the lowest common denominator. We are then left with three integers denoted (hkl), which when enclosed as such are also called the index of the plane. This reciprocal relationship between the Miller indices and Weiss parameters means that large Weiss parameters correspond to small Miller indices. For planes that are parallel to a crystallographic axis, the Miller index is zero because when a large Weiss parameter is inverted it tends to zero as $1/\infty \to 0$.

For the plane whose intersects are $4, 1, 2$, the reciprocals are $\frac{1}{4}, \frac{1}{1}, \frac{1}{2}$ and the smallest three integers with the same ratio are (142). Some common miller indices for the cubic system can be seen in Figure 2.4. The miller indices denoted (hkl) often denote a single plane or a set of parallel planes. A plane that intercepts an axis at a negative value is denoted with a minus symbol on top of the index like in Figure 2.4(e).

The understanding of lattice structures forms the basis for exploring the formation and behaviour of alloys. Different types of lattice structures, such as cubic, hexagonal, and tetragonal lattices, play a crucial role in accommodating and arranging multiple elements in alloys. The lattice parameters, which describe the lengths and angles between lattice vectors, influence interatomic distances, lattice symmetry, and the formation of solid solutions. By modifying the arrangement of atoms within crystallographic planes, alloying introduces lattice strain, which affects material properties such as mechanical strength, thermal stability, and electrical conductivity. This interplay between lattice structures and alloys enables the design and engineering of materials with tailored properties to meet specific industrial requirements, spanning aerospace, automotive, and electronics applications. We now introduce HEAs, a special type of alloy that has shown amazing uses as of late.

2.3 High Entropy Alloys

There are slightly varying definitions of HEAs. Some regard them from a composition point of view and some from an entropy point of view. Regardless, the main idea behind HEAs is the creation of a single-phase solution alloy rather than one with intermetallic or intermediate phases [32, 154, 245]. Here we will define them as alloys having four or more elements with a configurational entropy of more than 1.5R (1.5 times the gas constant). Yeh et al. (Ref. [245]) also added the requirement of each compound having a concentration between 5 to 35%. It is thought that these alloys introduce a new realm of possibilities for alloying, with the benefit of overcoming enthalpies of compound formation. Though not always, [5, 120], intermetallics generally want to be avoided as they are potentially harmful and complex [245]. Single-phase HEAs however have shown amazing properties, like their mechanical properties which have been cited as remarkably better than their elemental components [153, 154, 211]. More relatable, they have also shown promise in Oxygen Evolution Reaction (OER) [183, 256], CO_2 reduction reactions [172], ORR [14] and so on.

Considering an alloy created by mixing two elements X and Y, it can either form a solid solution or one of more intermetallic phases. This depends on the relative free energies calculated as follows:

$$X + Y = XY_{\text{solution}} : \Delta G_{\text{mix}} = \Delta H_{\text{mix}} - T\Delta S_{\text{mix}} \tag{2.15}$$

$$X + Y = XY_{\text{intermetallic}} : \Delta G_{\text{f}} = \Delta H_{\text{f}} - T\Delta S_{\text{f}}. \tag{2.16}$$

Here, ΔG_{mix}, ΔH_{mix} and $T\Delta S_{\text{mix}}$ are the Gibbs free energy, the enthalpy and the entropy of mixing respectively; ΔG_{f}, ΔH_{f} and $T\Delta S_{\text{f}}$ are the values for formation of an intermetallic compound with XY stoichiometry.

The phases present in an alloy at thermodynamic equilibrium depend on whether the Gibbs free energy of mixing (Equation (2.15)) is more or less negative than the energies of formation (Equation (2.16)) of all the possible intermetallic compounds comprised of X and Y (*i.e.* X_iY_j for $i, j = 1, 2, 3, \dots$). Different cases can occur here where, for example, the mixture may not form a single solid solution, but instead separate into two distinct solid solutions with different compositions, crystal structures, and/or lattice parameters. These issues become even more pronounced when more complex alloys are mixed with more elements - the number of possible phases that can co-exist increases dramatically according to the Gibbs phase rule[3].

[3]Gibbs phase rule states that $p + n = c + 1$ gives the relation between the number of phases p,

The primary reason for the stability of the solid solution phase in high entropy alloys is their high configurational entropy, particularly at high temperatures [247]. Initially, studies on high entropy alloys assumed that these behave like an ideal solid solution, where mixing enthalpies and non-configurational entropies of mixing were believed to be insignificant. In such cases, the alteration in Gibbs free energy due to mixing is primarily due to the configurational entropy, as

$$\Delta S_{\text{mix}} = -R \sum_{i=1}^{n} c_i \ln(c_i) \tag{2.17}$$

Where c_i is the mole fraction of the ith component and R is the ideal gas constant [222]. When a solution satisfies $T\Delta S_{\text{mix}} >> \Delta H_{\text{mix}}$, then with Eq. Equation (2.17), Eq. Equation (2.15) becomes

$$\Delta G_{\text{mix}} = -T\Delta S_{\text{mix}} = RT \sum_{i=1}^{n} c_i \ln(c_i) \tag{2.18}$$

which is the change in Gibbs free energy due to mixing for a non-equiatomic solution. In the case of an equiatomic solution, however, Eq. Equation (2.18) simplifies to

$$\Delta G_{\text{mix}} = RT \ln(n) \tag{2.19}$$

where n is the number of elements in the system. Based on Eq. Equation (2.19), two alloys in a system would result in an entropy of $\ln(2)R = 0.69R$ and five would be $1.61R$. The general consensus based on Ref. [114] is that "high entropy" is at four or more elements, which is rounded to $\Delta S_{\text{conf}} \geq 1.5R$. This high configurational entropy threshold is used as a rule of thumb for classifying HEAs.

Additionally, Yeh et. al [245] stated that it is generally sufficient to have $5 - 35\%$ concentrations with a minimum of five constituent elements in order to stabilize a single-phase solid solution [114]. This is the case at high temperatures only, at room temperature the formation of an ideal solid is unfeasible [97], with the presence of intermetallics and precipitates.

One growing area of research for HEAs is catalysis. HEA surfaces have highly disordered configurations and high chemical complexity. This suggests the potential for additional diversity and density of adsorption sites. There has been research into these for the OER [249], ORR [14], CO_2 and CO reduction reaction (CO_2/CORR) [158]. In the following section, we introduce ORR, its importance in the battle against climate change, and the role HEAs can play in this.

the components c in an alloy under equilibrium conditions at constant pressure, and n the number of thermodynamic degrees of freedom in the system.

2.4 Oxygen Reduction Reaction

ORR is an important process in the human body - it governs how our bodies store chemical energy. Beyond this, it is involved in low-temperature fuel cells, metal-air cells, oxygen sensors, and the preparation of hydrogen peroxide in the field of energy. It is such a common mechanism that it is the one which governs metal corrosion. ORR is noted to be the limiting factor in the chemical reactions that govern hydrogen fuel cells [51, 215] for example. More generally, in electrocatalysis its high overpotential [4] and complex kinetic mechanism which limits new developments for fuel cells [215] and the lack of sufficiently good ORR catalysts limits the overall efficiency of these devices [49, 68, 69]. This is where extensive research has been put for the past few decades. We now introduce the mechanism of ORR.

For the electrolysis of water, two electrodes are placed in a container of water and a potential is placed across them [187]. With a high enough potential, water oxidation and oxygen evolution occur at the anode and the evolution of hydrogen with the reduction of protons occurs on the cathode. The overpotential for the splitting of water reportedly ranges from 0.55 to 0.77V at $10\text{mA}/\text{cm}^2$ while the overpotential for H_2 evolution may be an order of magnitude smaller [251]. Because of its four-step process with additional activation energies and intermediates, OER tends to have lower efficiency than the evolution of hydrogen [9]. Therefore the slow reaction kinetics of OER at the anode in a water-splitting device severely constrains the efficiency of the hydrogen evolution reaction at the cathode and overall water splitting [233] and the discovery of materials with lower overpotentials at both the anode and cathode have grown in interest.

In water electrolysis, water is split into hydrogen and oxygen gasses like so:

$$2\,H_2O\,(l) \longrightarrow 2\,H_2\,(g) + O_2(g). \tag{2.20}$$

A simple schematic of this process can be seen in Figure 2.5. At standard conditions, this reaction (Equation (2.20)) requires a Gibbs free energy of 237.2 kJ/mol, or 4.92 eV [252] and under acidic conditions it can be split into two half-reactions:

hydrogen evolution which is the reduction of protons at the cathode;

$$4\,H^+ + 4\,e^- \longrightarrow 2\,H_2\,(g), \tag{2.21}$$

[4] The overpotential of an electrochemical reaction is the additional cost necessary to drive a reaction towards favourable products at a certain rate. It is usually the difference between the increased potential and the thermodynamically required potential [8].

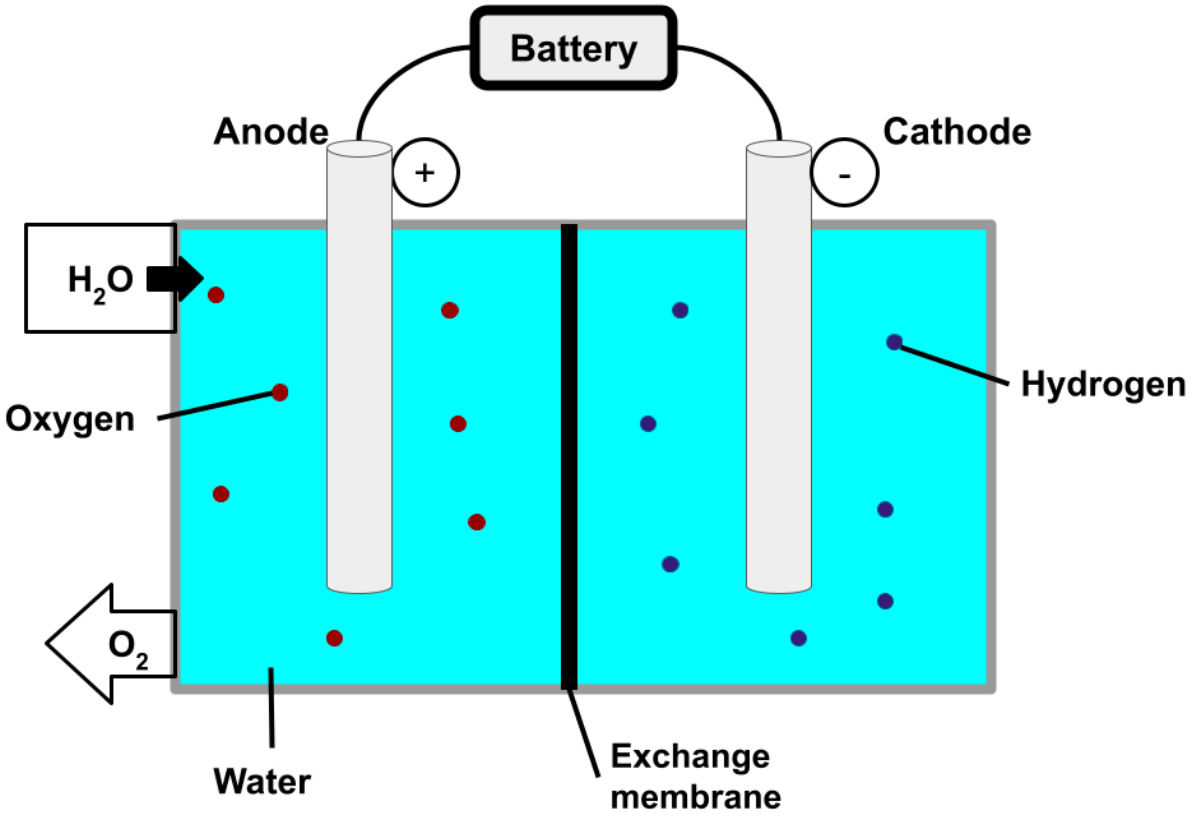


Figure 2.5: A simple schematic of a water electrolyzer. At the anode (+) side are the oxygen molecules in red and at the cathode (−) are the hydrogen atoms in blue. An exchange membrane that allows hydrogen ions to pass through. At the cathode we have the HER denoted in Equation (2.21) and at the anode occurs the OER denoted in Equation (2.22). We note water being added to the system and oxygen exiting.

and oxygen evolution which is the oxidation of water:

$$2\,H_2O\,(l) \longrightarrow O_2\,(g) + 4\,H^+ + 4\,e^-. \tag{2.22}$$

It has long been an open question why some materials perform better (have a lower overpotential) than others. Some metals may have lower overpotential for electrolysis due to their high reactivity and exchange current density. These imply a high reactivity of the

electrode, which results in a lower overpotential.

2.5 Statistical Learning

Statistical learning is a discipline that is leveraged in most fields of research. This holds true for astrophysicists studying cosmic radiation [71], to economists studying the commonalities between high school teachers and sumo wrestlers in [140]. It is, of course, also used by condensed matter physicists studying materials design.

We can define statistical learning in various ways. In the words of [109], *statistical learning refers to a vast set of tools for understanding data.* The authors then go on to state that *these tools can be classified as supervised or unsupervised.* The term supervised comes from the data used in learning being labelled, and unsupervised refers to unlabeled data.

This thesis focuses on supervised learning, where for each observation of the *input variables* $x_i,\ i = 1, \ldots, n$ there is an associated *output variable* y_i. The output variables can vary in nature, ranging from quantitative measurements (ie continuous variables) to qualitative (where they can be split into *classes*). Throughout this work, *input variables* will be called *predictors* or *features* and the *output variables* will be denoted *response*, *dependent variable*. We wish to then fit a model that relates the response to the predictors, with the goal of accurately predicting the response in future observations. This is done through a process called training. We assume there is some relationship between predictors x_i and responses Y, and wish to approximate a function f such that

$$Y = f(X) + \epsilon, \tag{2.23}$$

where ϵ is a random error term which is independent of X and has a mean of zero. In the linear form, for example, we hypothesize that equation (2.23) is of the form

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{n} X_j \hat{\beta}_j \tag{2.24}$$

where $X = (X_1, X_2, \ldots, X_p)$ is our vector of inputs, $\hat{\beta}_0$ is the intercept, otherwise known as the *bias*. A variable with a "hat" (eg. $\hat{Y}$) denotes a predictor of the variable (Y in this case).

Linear regression, decision trees, support vector machines and principal component analysis are only some of the extremely powerful methods that fall into the set comprising statistical learning. Although some may group these methods in the realm of ml, we separate them despite the fact that regression, for example, is a staple for any machine learning framework (see Section 2.5.2).

2.5.1 Fitting a model

In order to fit a model, we must know what we are fitting. Given a situation, we will first define the problem at hand, and then we will decide what we are trying to solve and how we will model this problem. For example, say we are trying to model the linear expansion coefficients of metals by observing how these materials expand and contract with changes in temperature. It can be shown that supposing they do not go through a phase change, the change in these objects' lengths is measured by

$$\Delta L = \alpha L_i \Delta T, \tag{2.25}$$

where L_i is the length of the material before heat is added and α is the linear expansion coefficient. In this case, a scientist will control the change in heat added to the object and measure the change in length. They will then try to fit the function

$$\hat{L}(T_i) = \hat{\beta}_1 \Delta T_i + \hat{\beta}_0. \tag{2.26}$$

In order to model Equation (2.26), a very common method is the *least squares* method;

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \tag{2.27}$$

where RSS stands for Residual Sum of Squares (RSS) and is, as the name suggests, the sum of all squared residuals. Here N is the total number of data points, and y_i is the true value.

Equation (2.27) can be rewritten as

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \tag{2.28}$$

which we can differentiate with respect to β in order to minimize, yielding the normal equations

$$\mathbf{X}^T (\mathbf{y} - \mathbf{X}\beta) = 0. \tag{2.29}$$

The unique solution is (assuming $\mathbf{X}^T\mathbf{X}$ is nonsingular) is

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.30}$$

In Figure 2.6, we show an example of linear regression.

In this case, we used a model with a known analytic solution. If we were to not use this

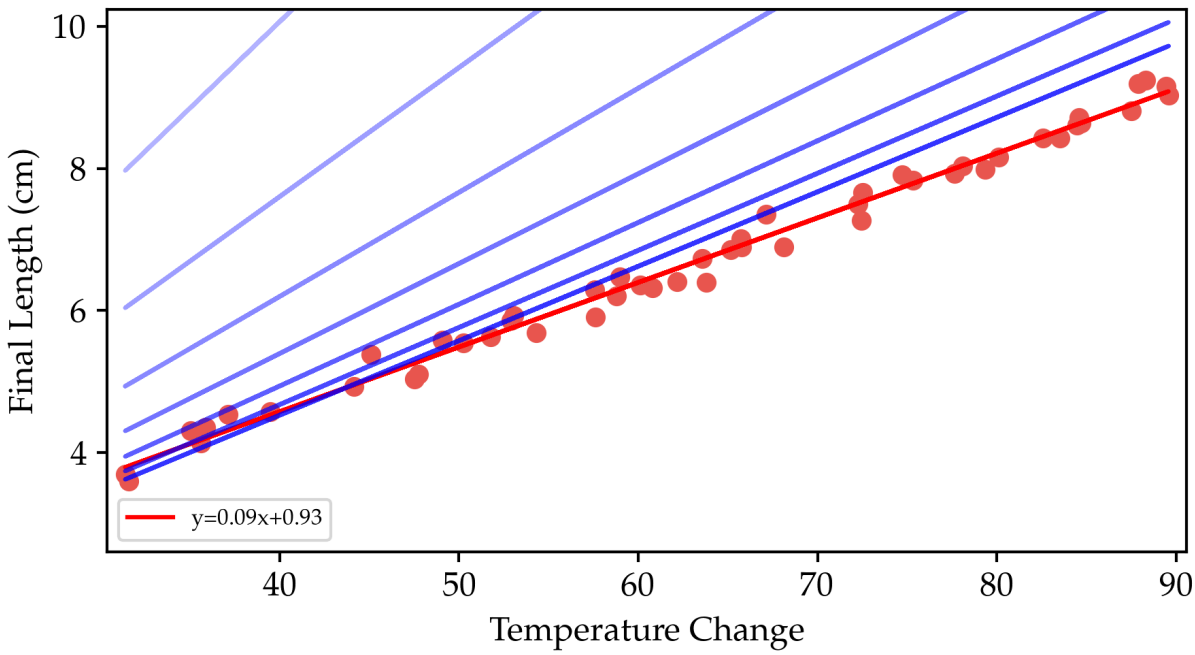


Figure 2.6: Linear regression example. The line of best fit (red) was calculated using the RSS method. Blue is an example of learning with gradient descent, where we start with a random prediction (the lightest shade) and get better and better after each update (darker colours). The final curve attained after training for only 10 epochs with a learning rate of $5e-5$ is $y = 0.09x + 0.99$. This is not quite as good a fit (defining goodness as the RSS) but improves as training progresses. Better results are expected with fine-tuning of the initial state and a longer training time. The data used can be found in Table A.1.

solution, we would initiate a process called *training*. Starting with an educated guess for the form of the model, we initialize its parameters randomly. In the case of our regression problem, these would be $\hat{\beta}_0$ and $\hat{\beta}_1$. We would then use a *loss function* (or just *loss*) to compute a distance metric from our goal by evaluating a performance measure. Usually, this measure is specific to the task being carried out. In a classification task, a common measure is *accuracy* which is the proportion of examples for which the model produces the correct output. In regression, however, some common measures are the mean squared error Equation (2.31a) and the mean absolute error Equation (2.31b). In short, the former reduces large outliers while the latter reduces overall error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.31a}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{2.31b}$$

Using our data, our model and our loss function, we are ready to train. Our goal is to find new values of $\hat{\beta}_0$ and $\hat{\beta}_1$ such that the loss is minimized. We will therefore update the weights using the loss function, specifically by subtracting the gradients;

$$\hat{\beta}_i = \hat{\beta}_1 - \epsilon\frac{\partial L}{\partial \hat{\beta}_i}. \tag{2.32}$$

Here, ϵ is called the *learning rate* which defines the step size by which we update our predictors. Equation (2.32) is called gradient descent, where we "descend" the gradient to lower loss values. Figure 2.6 shows an example of gradient descent, where we slowly get closer and closer to the analytical solution at each iteration. This method can be seen in blue in Figure 2.6. Much more can be said about gradient descent, but we will discover *optimizers* more in Section 2.5.6.

This brings us to neural networks, a class of tools in statistical learning. These can broadly and informally be characterized as tools where data and algorithms are used to imitate human intelligence [109].

2.5.2 Neural Networks

The term *neural networks* has evolved over many years. We will, however, learn of its building blocks and of its many uses, for example in self-driving cars [36, 117, 200], image recognition [67, 112] and large language models helping people complete their everyday tasks [25, 41, 227].

Neural networks were first developed as models for the human brain. They are regression or classification models, typically represented by diagrams as in Figure 2.7. Each unit (circle) represents a *neuron* and the connections (the links) represent synapses. For regression, the output layer typically contains $K = 1$ neurons and for classification, K may be the number of classes. These networks are easily extended to multiple quantitative responses (by increasing K values).

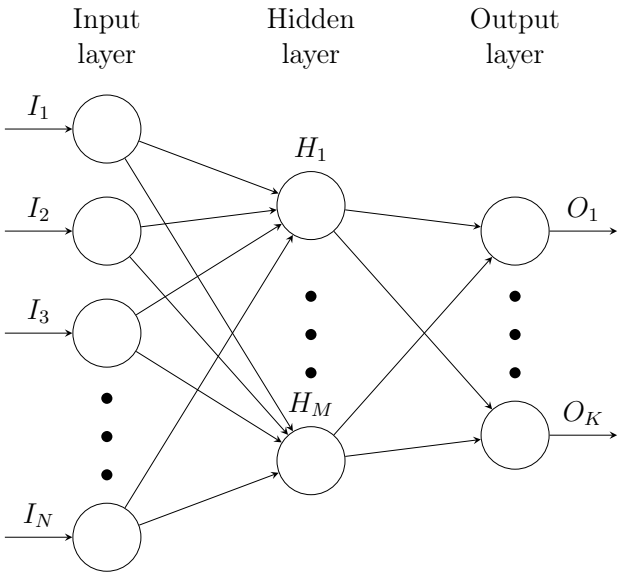


Figure 2.7: A single hidden layer, feed-forward neural network. For simplicity, the biases have been omitted.

A single (hidden) layer, feed-forward NN is also known as a single-layer fully connected network. This is simply a function $f(x, w)$ used to approximate the mapping $\mathbf{f} : \mathbb{R}^{N_{in}} \mapsto \mathbb{R}^{N_{out}}$ where x denotes the input and R the parameters. Derived features, H_m are linear combinations of the inputs, and the outputs O_k are linear combinations of the H_m. Mathematically, Figure 2.7 can be described as

$$f(w, x) = \mathcal{L}^{\text{out}} \circ \mathcal{L}^{H} \circ I, \tag{2.33}$$

where $\circ$ denotes composition (ie $\mathcal{A} \circ \mathcal{B} \equiv \mathcal{A}(\mathcal{B})$). This can naturally be extended to any number of layers. $\mathcal{L}$ denote the operations mapping layers. $\mathcal{L}_i$, the operation mapping the neurons n_{i-1} to n_i is denoted as

$$n_i = \mathcal{L}_i(n_{i-1}) = \varphi(\mathbf{w}_i n_{i-1} + b_i). \tag{2.34}$$

Here, φ is some nonlinear transformation of its inputs called an *activation function*, $\mathbf{w}$ is the weights and b is the bias. Some activation functions are more common than others, some due to special properties of their gradients and others for no apparent reason at all. These are necessary since without them, we would have a linear model and we would be adding no complexity whatsoever to our model since if $f^1(x) = W^T x$ and $f^2(h) = x^T w$

then $f^2(f^1(x)) = w^T W^T(x)$.

Figure 2.8 shows some common activation functions with different scaling factors which can, in fact, also be learned[5].

Specific activation functions are often used to restrict the range of outputs of the networks' layers. A few a listed below. Tanh and Sigmoid are known as squashing functions, which are common activation functions used to limit outputs to a certain range.

Sigmoid Squashing function otherwise known as the logistic function, depicted in Figure 2.8(a). It is a differentiable, monotonically increasing function which asymptotes at 0 as it approaches $-\infty$, and 1 as it approaches $+\infty$. It is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.35}$$

This is commonly used to model probabilities (so commonly used at the output layer in logistic regression) but suffers from a vanishing gradient at increasing and decreasing values of x.

Hyperbolic Tangent (tanh) Another common squashing function depicted in Figure 2.8(b). It is also differentiable and monotonically increasing, but asymptotes at -1 and 1 as it approaches $-\infty$ and $+\infty$. It is defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.36}$$

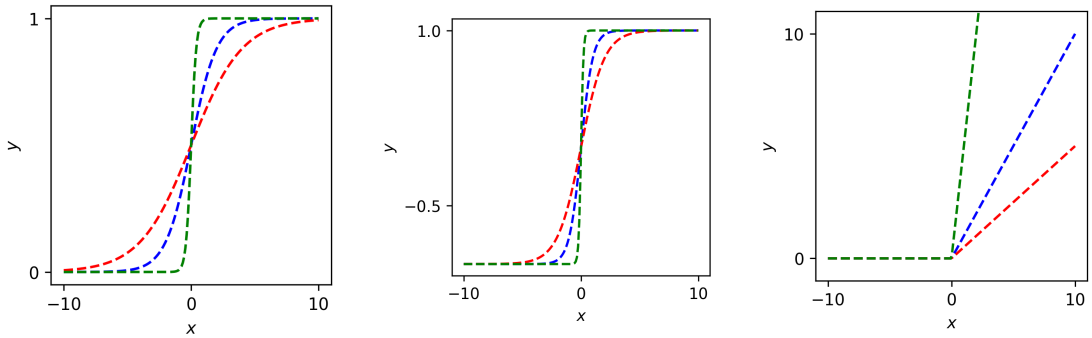
where e is the base of the natural logarithm.

Rectified Linear Unit (ReLU) This is the default activation function recommended for use with most feedforward neural networks. It is plotted in Figure 2.8(c). ReLU is nearly a linear function as it contains two piecewise linear functions, and is therefore similar to linear models in that they are easy to optimize with gradient-based methods. It also preserves many of the properties that make linear models generalize well [83]. It is defined as

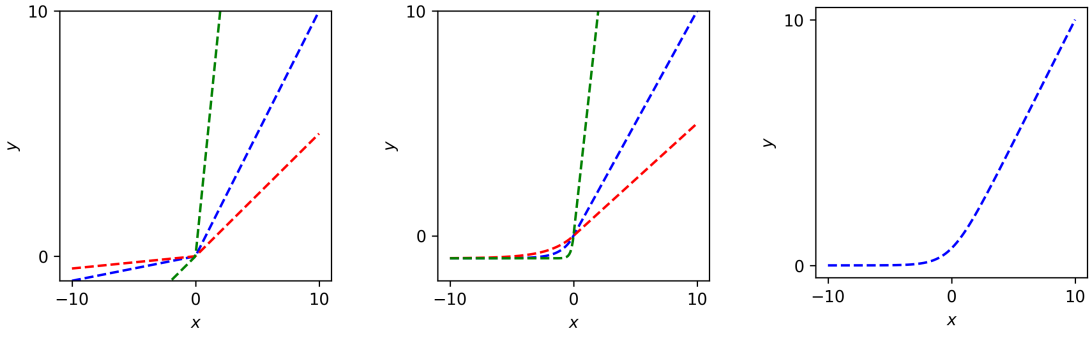
$$\text{ReLU}(x) = \max(0, x), \tag{2.37}$$

and thus ranges from $[0, +\infty]$. Some downsides are that it is not differentiable at zero and has a zero gradient in the negative part. Some "soft" versions have been introduced to overcome these downsides.

[5]Though a recent study shows that learning the scaling parameter can be equivalent to adding layers with fixed activation functions and some weight constraints [4], therefore it may not be necessary.



(a) Sigmoid (b) Tanh (c) ReLU



(d) Leaky ReLU (e) SELU (f) Softplus

Figure 2.8: Common activation functions. a) The sigmoid b) Tanh, c) ReLU d) Leaky ReLU e) SELU and f) Softplus activation functions. Included are $\varphi(sx)$, different scaling factors in blue ($s = 1$), green ($s = 5$) and red ($s = 0.5$) where appropriate. This parameter controls the rate of activation, with larger values causing more activation (a steeper curve).

SoftPlus [54] is known as a soft version or RELU and can be shown in Figure 2.8(f). It is defined as

$$\text{softplus}(x) = \log(1 + e^x). \tag{2.38}$$

The idea behind Softplus is to approximate ReLU, however without the non-differentiable part at the origin. One interesting aspect of the softplus function is that its derivative is exactly the sigmoid function (2.35).

Leaky ReLU (lReLU) [145] is another "soft" version of the ReLU which is defined as

$$\text{lReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases} \tag{2.39}$$

which shows negligible improvements over ReLU.

Scaled Exponential Linear Units (SELU) was introduced in [123] and is shown to induce self-normalizing properties - it favours lower absolute values of neurons. It is shown in Figure 2.8(e) and is defined as

$$\text{SELU}(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^x - 1) & \text{otherwise} \end{cases} \tag{2.40}$$

where α and λ are some pre-defined constants. The main advantage noted of SELU is that it prevents gradient issues during training like exploding or vanishing gradients due to its self-normalization properties.

In general, the design of a network - the number of layers, number of units and connections among them, defines its *architecture*.

Convolutional neural networks (CNN) [135, 136] are a type of feedforward neural network that is commonly used for image recognition and computer vision tasks. CNNs use a technique called convolutions to extract features from input images, which are then used to classify the image. These are typically stacked with *pooling* layers which consist of a subsampling operation that aggregates statistics of local features [83]. Because CNNs can automatically learn the features that are most relevant for a given task, they have become a popular choice in many computer vision applications. Their spatial invariance which allows for shifts in the inputs, their weight sharing scheme which allows powerful models to be trained with less memory and their robustness to noise are only a few of the benefits that CNNs offer. See [226] for a thorough overview of computer vision, which heavily utilizes the power of CNNs.

Recurrent Neural Network (RNN)s are another type of neural network that process sequential data such as speech or text. RNNs employ feedback loops to pass information from one time step to the next, allowing them to maintain a memory of previous inputs. RNNs are commonly used in natural language processing tasks, such as language translation and sentiment analysis, though recent advances in these fields seem to favour the transformer model [231] instead. Goodfellow et al. [83] (Chapter 10) give a good introduction to sequential modelling with recurrent (and recursive) NNs.

Autoencoders are a type of neural network that is used for unsupervised learning, meaning they are not given any labelled data to learn from. They are trained to copy their inputs to their outputs. They do so by having two parts: an encoder and a decoder which respectively encode and decode the inputs. Because it is forced to prioritize certain aspects of its inputs, it can be useful in learning the properties of datasets. These are typically

used for tasks such as image compression (dimensionality reduction) or anomaly detection (feature learning), where the network is trained to reconstruct input data with as little error as possible.

Though many other neural network classes exist, and with as much importance in many fields, we keep this section with some brief mentions and move on to other methods of statistical learning relevant to this dissertation.

2.5.3 Representation Learning

Representation learning is a large field of study in machine learning which focuses on learning features (or representations) for data, without manually engineering them. It is the main topic of discussion in many conferences, one even being dedicated to it[6]. Traditionally, features would often be handcrafted by domain experts, which can be time-consuming, error-prone and limiting to the scope of analysis by introducing biases and assumptions about the data. Representation learning allows us to discover features automatically from the raw data. The idea behind representation learning is learning a low-dimensional feature space where each feature represents a different aspect of the data. These learned representations can then be used for a wide range of tasks, including classification, clustering, and visualization.

We can consider feedforward networks trained by supervised learning as a kind of representation learning. It is the last layer of the network which is typically a linear classifier such as a softmax regressor, the rest of the network learns to provide a good representation of this classifier. In Figure 2.7 for example, we could extract the hidden layer's outputs H_i and use that as extracted features. Classes that were previously not linearly separable in the input features may now be so.

Often an issue that arises is not being able to utilize large amounts of unlabelled data. Since this does not fall into the realm of supervised learning, and training on a small amount of labelled data may lead to severe overfitting, we can look towards representation learning to perform semi-supervised learning. We can use the unlabelled data to learn some good representations, which can aid in solving the supervised task [83]. Otherwise, we can be left with a lot of unused and therefore wasted data and an underperforming model.

In a large dataset, it can be hard for a data scientist to extract useful features without specific domain knowledge. In a simple dataset like MNIST [137], which is a large database

[6]International Conference on Learning Representations (ICLR).

of hand-drawn digits, a data scientist may wonder how to manually extract useful features. Using a model to automatically generate features can lead to amazing ones, which can oftentimes additionally be visualized. An appropriate model, in this case, would be the CNN, as introduced above. This would learn hierarchical representations of the image features. Each successive convolution learns more abstract and separated (spatially) features; earlier convolutions may learn lower-level features like edges and lines, and deeper layers may capture full objects and shapes to then combine these to capture a full image [130].

Representation learning, a concept central to machine learning, is the process of converting raw data into a format that is more manageable and useful for predictive tasks [18]. In physics and material science, this concept has great potential, especially when dealing with the complexities of atomic and molecular systems. Efficiently representing atomic systems and capturing physical and chemical properties, makes these well-suited for predictive tasks in material design and discovery. Graph-based representations have gained much attention due to their ability to encode the structures of molecules and materials, viewing them as graphs with atoms as graph nodes and bonds as edges connecting them [77, 242].

This leads us to the next chapter, focusing on GNNs. These are innovative in the field of representation learning, enabling robust modelling of graph-based representations of physical systems, and capturing the complex, non-local interactions among their components. By mapping the problem of material property prediction to a graph problem, GNNs offer a new pathway towards efficient, scalable, and accurate predictions in material science [240, 253].

2.5.4 Graph Neural Networks

2.5.4.1 Principles of Graphs

Any collection of objects (nodes) with some interactions (edges) between pairs of these objects can be described as a *graph*. These are ubiquitous data structures capable of describing complex systems. In social network research, it is possible to represent individuals as nodes and friendships with an edge. Or in the biological domain, it is possible to represent proteins with nodes representing proteins and edges with various interactions.

The power of graphs stems from their focus on relationships between points, as well as their generality [93]. These benefits, of course, extend to its applications in materials science. Machine learning models on graphs are a fast-growing discipline with many great new advancements in recent years. For example, its uses range from the biomedical field where models created with the capability to generate proteins with certain structures [236]

have been created, to tech companies like Uber and Pinterest using graph learning for (respectively) food recommendations for millions of customers with thousands of options [108] and image recommendation systems on millions of users with billions of image options.

GNNs are a class of algorithms which operate on graphs. We formally define graphs as a tuple $G = (V, E)$ of the set of vertices $v \in V$ and edges $e_{v,w} = (v, w) \in E$ comprising the graph. We will typically be interested in simple graphs, with no loops (a node with an edge to itself), and at most one edge between two nodes (however, these may be weighted).

Various ways of representing graphs exist, but a convenient one is using the *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$. This is done by (arbitrarily) ordering the nodes in the graph such that each column and row of $\mathbf{A}$ indexes to a particular node. The information encoded in the adjacency matrix is the presence of an edge: $\mathbf{A}[i, j] = 1$ if $(i, j) \in E$ and $\mathbf{A}[i, j] = 0$ otherwise. There are often attributes associated with the elements of a graph. For example, in a graph representing atom-atom interactions, we may be interested in different interaction energies associated with these atoms or even the distance between these atoms. We can therefore extend the edge notation to include an edge or relation type τ, e.g. $(i, \tau, j) \in E$ and we can define an adjacency matrix $\mathbf{A}_\tau$ per edge type. These *multi-relational* graphs can be summarized by an adjacency tensor $A \in \mathbb{R}^{|V|\times|R|\times|V|}$ where R is the set of edge-level features. We can naturally also include node-level features with a $X_N \in \mathbb{R}^{|V|\times M}$ matrix or graph-level features with a $X_G \in \mathbb{R}^K$ where M and K are the node and graph features respectively.

2.5.4.2 Graph Analysis

As we know, graphs are a powerful data structure and prior to the advent of modern deep-learning approaches on graphs, many methods were used for machine learning on graphical data. These methods can be used for all types of learning tasks including classification and regression.

Traditional approaches followed the basic trends in machine learning prior to the advent of deep learning. This would entail using domain knowledge and heuristics to extract graph statistics or features. These would then be used as input to standard machine learning classifiers like the logistic regressor.

To do so, one naturally begins with questions about the graph(s) in question. What properties or statistics can be used to distinguish graphs from one another? What about distinguishing nodes from one another? To do so we now cover some more basic graph definitions following [93].

Node degree This simply counts the number of edges incident to a node,

$$d_u = \sum_{v \in V} A[u, v]. \tag{2.41}$$

Node centrality The node degree may not necessarily measure the *importance* of a node in a graph, therefore the node centrality. Different measures of importance can be used, one of which is the *eigenvector centrality*. This encompasses a node's degree and how important a node's neighbours are. The node centrality, e_u, is defined as

$$e_u = \frac{1}{\lambda} \sum_{v \in V} A[u, v] e_v \ \forall u \in V \tag{2.42}$$

where λ is a constant. This equation is a recurrence relation and shows a node's centrality is proportional to the average centrality of its neighbours. One can show the vector of centrality values $\mathbf{e}$ is given by the largest eigenvalue of $\mathbf{A}$. It can also be shown that this metric ranks the likelihood of visiting a node on a random walk of infinite length on the graph [159].

Clustering coefficient This metric measures the proportion of closed triangles in a node's local neighbourhood. It measures how tightly clustered a node's neighbourhood is, so a value of 1 means all of node v's neighbours are also neighbours of each other. The local variant of this coefficient c_u is formally defined as

$$c_u = \frac{|(v_1, v_2) \in E : v_1, v_2 \in N_u|}{\binom{d_u}{2}}, \tag{2.43}$$

where N_u is the neighbourhood of u.

So far we have seen only a small sample of the node-level statistics possible for node classification, now we will a simple metric which can be used for graph-level classification and regression.

Bag of nodes One straightforward way of defining a feature for an entire graph is to collect statistics from individual nodes and combine them. A couple of examples of these statistics include degrees, centralities, and clustering coefficients which can be used to generate histograms or other summary measures. This approach provides an overall representation of the graph that relies only on local node-level details. However, it may not capture some essential global properties of the graph, which is a large limitation of this

approach.

2.5.4.3 Graph Learning Tasks

In contrast to classic machine learning methods, here we model components of graphs or entire graphs in graph space, as opposed to data points in Euclidean space. There exist four main tasks for learning on data structured like graphs.

Node classification. Here, the goal is to predict labels, which could be a type, category or attribute, associated with all nodes in a graph. One may assume this involves learning from a small set of labelled data, however, there may be instances of node classification involving generalization across disconnected graphs, for example.

In a typical machine learning task, it is generally assumed that our data are *independent and identically distributed* (i.i.d.). That is, when we build our models, we assume our data points are all statistically independent of all other data points. We would also have no way of generalizing to other models if our data weren't identically distributed and be required to model these dependencies if they exist. In node classification, this i.i.d. assumption is broken; an edge is what clearly states a relation to other nodes, which introduces much complexity to this task.

Relation Prediction. Here, instead of being interested in predicting nodes, we are interested in predicting relationships between nodes. This task is known by different names, such as link prediction, graph completion, or relational inference, depending on the application domain. Relation prediction is one of the most popular machine learning tasks with graph data, alongside node classification, and has a myriad of applications in the real world. Some examples of its use include recommending content on social media platforms [248] and predicting drug side effects [257] among many others.

Graph classification and regression. A classic example of graph classification is predicting a molecule's solubility or toxicity given its structure [77]. In graph classification or regression, we seek to model graphs but instead of predicting individual components of graphs, we want to make independent predictions for each graph. In this case, we do not break the i.i.d. assumption, and we want to learn a mapping from graphs to labels. One example of graph classification is in population-based disease prediction [165].

Clustering and community detection. Clustering is the graph analogue of unsupervised clustering. To illustrate this, we use the example of a social network, where users interact with each other on a messaging platform. In this context, we ask the question: does the network exhibit a community structure, where users tend to interact more frequently

with others in the same community rather than with those outside their community? The task of community detection involves identifying groups, which is challenging as it requires inferring hidden community structures from input graphs. We find applications of this in locating crime rings in social networks [201] to the study of how diseases spread between communities [197].

2.5.4.4 Graph Neural Networks in Materials Science

We have seen some interesting methods of graph neural networks have been applied in biology, epidemiology and social networks, the question that remains is how have these been applied in the materials science field.

One example is the use of machine learning models to predict the formation energies of crystals [10, 116, 161, 167, 208]. This is a challenging task because it requires predicting the energy required to form a crystal from its constituent atoms, which can involve complex interactions between many atoms. However, machine learning models have been shown to be highly accurate at predicting formation energies, even for materials that have not been synthesized yet. They have also been useful in screening large databases of crystals quickly and efficiently, which is often not feasible using conventional simulation techniques.

Disordered materials are those that lack a well-defined long-range order in their atomic or molecular structure. Examples of this include gasses, gels and polymers. The disorder can influence and sometimes even dominate material properties [33]. Graph neural networks can be extended to model various forms of disordered systems and predict their local and global properties. However, there is a lack of large datasets of disordered systems, particularly labelled datasets with consistent measured or simulated properties [189].

Doping is a process used to introduce impurities into a material, typically a semiconductor, in order to alter its electrical properties. This process is often used to create materials with specific electrical properties that are required for electronic devices. In some cases, doping can disrupt the regular arrangement which can introduce disorder. The introduced impurities may not occupy a single specific site in the crystal lattice but instead have a probability of occupying multiple sites simultaneously. This is called substitutional disorder which introduces problems in the modelling, simulation and even synthesis and characterization of materials. Models have been introduced to address these issues; [39] introduce MEGNet with a learnable embedding for atoms trained with doped crystal sites. They showed how to perform multi-fidelity training by encoding low-fidelity (PBE) DFT data in the global state of the MEGNet to predict band gaps of different levels of DFT (ie more or less accurate). Wand et al. [234] also use a model trained on undoped crystals

to predict certain properties of doped crystal structures. In [64], the authors use a graph model to predict the properties of 2D materials with small defects.

Molecular dynamics (MD) simulations are very expensive, and some recent research tries to speed these up with GNNs. For example, [166] use GNNs to predict forces for MD using link prediction representing the force between atoms with good accuracy. In [188] the authors use graph networks to predict partial charges in metal-organic frameworks for simulations of gas adsorption.

A large area of study in materials science is the synthesis of new materials, which refers to the process of creating materials with specific properties by combining different elements or compounds in a controlled way. This is the central idea of this thesis. In [110], the authors employ a positive-unlabelled[7] machine learning framework and implement GNNs as a classifier in which the model outputs crystal-likeness scores.

2.5.4.5 Neural Networks on Graph

Understanding the various learning tasks available with graph data, and getting a brief overview of classical methods, we now review some relevant deep learning methods on graphs. The central idea is generating representations of nodes or graphs that vary depending on the underlying graph structure and associated features.

Designing advanced encoders for graph-structured data presents a major challenge because our conventional deep-learning methods are not directly applicable. For instance, CNNs are specifically designed to operate on grid-structured inputs, such as images, while RNN are designed to work with sequential data, such as text. To create a deep neural network that can process general graphs, we must use a novel type of deep learning architecture called the GNN [26, 50, 80, 85, 206, 220].

2.5.4.6 Message-Passing Neural Networks

Typically, GNNs operate as *Message Passing Neural Network (MPNN)s*, the central idea being the sharing of information between local nodes. During each message-passing iteration, a node's hidden embedding h_i^k is aggregated from i's neighbourhood N_i. This update

[7] Positive-unlabelled learning otherwise known as PU learning is a type of machine learning technique used when we have a dataset that has only two classes, but the negative class is not well-defined or difficult to obtain. In PU learning, we have a set of labelled examples that belong to the positive class and a set of unlabeled examples that may belong to either the positive or negative class. The PU learning algorithm typically adjusts the class priors to account for the fact that the unlabeled examples may contain both positive and negative examples.

can be expressed as [93]

$$h_i^{k+1} = \text{UPDATE}(h_u^k, F_{\text{agg}}(h_j^k \ \forall j \in N_i)) \tag{2.44}$$

where UPDATE and F_{agg} are differentiable functions. At each iteration k, the aggregation function f_{agg} takes as input the set of embeddings of the nodes in the node i's neighbourhood and produces some message. This process can be visualized in Figure 2.9.

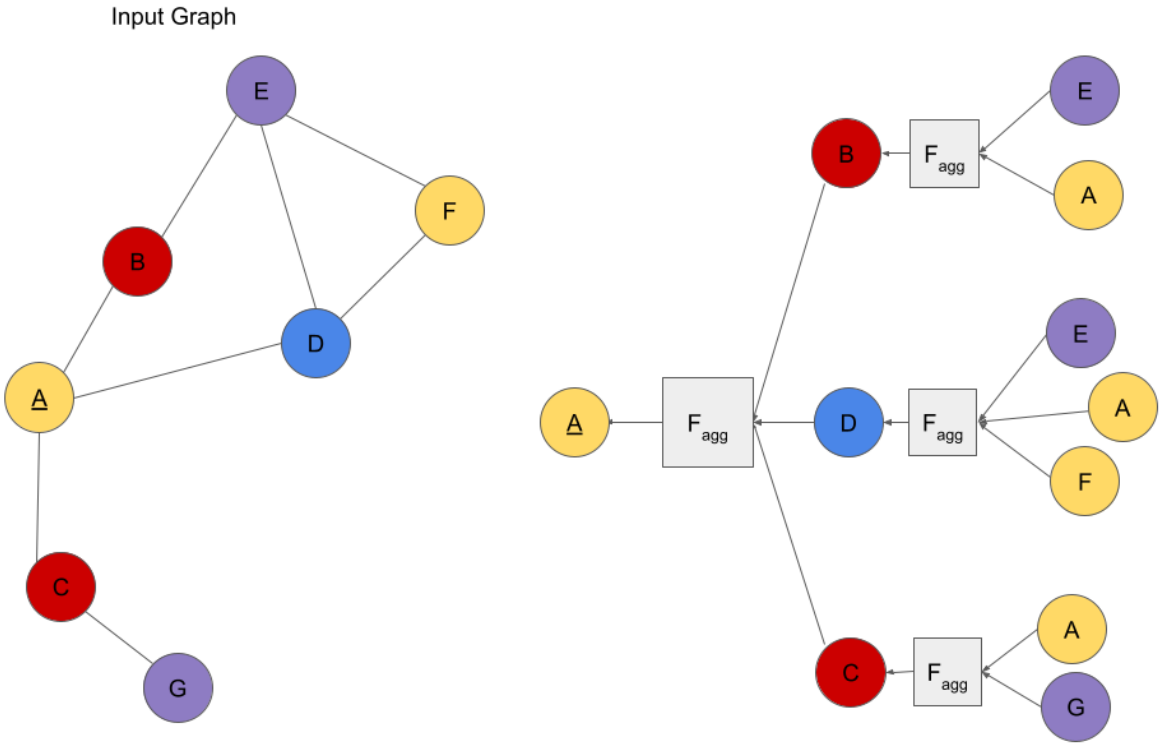


Figure 2.9: Visualization of the message-passing mechanism. The target node is underlined (node "A") which uses the function f_{agg} to aggregate its neighbourhood's information (*i.e.* nodes B, C, D). These then repeat this process.

The intuition is that with each iteration, we propagate more information about the surroundings to all nodes' representations. At iteration k, a node learns information from its k-hop neighbourhood. So after 1 iteration, a node learns information from its direct neighbourhood, then in the second it gets from its second neighbours (2-hop neighbourhood),

etc.

One might now wonder what information these embeddings actually encode. There are two forms to this; the first is structural information about the graph and the second regards features of nodes. Structural information may include, for example, information about the degrees of related nodes. The second form, *features* could entail an encoding of all features in the nodes k-hop neighbourhood.

The basic GNN message passing is defined as follows:

$$h_u^{k+1} = \sigma\left(W_{\text{self}}^k h_u^k + W_{\text{neigh}}^k \sum_{n \in N_u} h_n^k\right), \tag{2.45}$$

where the aggregate function F_{agg} is a simple sum over all neighbours $\sum_{N_u} h_n$ [93]. The bias term is omitted for simplicity (and without loss of generality) and the UPDATE function is a linear projection of the current representation with the aggregated message. A non-linearity is then applied element-wise as follows

$$UPDATE(h_u^k, m_{n(U)}) = \sigma(W_{\text{self}}^k h_u^k + W_{\text{neigh}}^k m_{N(u)}) \tag{2.46}$$

Although not widely appreciated, the standard GNN can also be written in terms of standard graph notation as follows:

$$H^t = \sigma\left(AH^{k-1}W_{\text{neigh}}^k + H^{k-1}W_{\text{self}}^k\right). \tag{2.47}$$

Here, H^k is the matrix of node representations at layer t, and A is the adjacency matrix. This highlights the sparsity in implementing a GNN using matrix operations.

Despite their popularity, basic message-passing schemes have notable shortcomings. Firstly, their expressive power is bound by the first-order Weisfeiler-Lehman test (1-WL) [244]. It has also been shown that many combinatorial problems cannot be solved by MPNNs [203]. Furthermore, the paradigm of MPNNs has been shown to contain other fundamental flaws such as over-smoothing, where the repeated aggregation of local information make representations indistinguishable for nearby nodes [29, 162, 163] and over-squashing where the aggregated information from farther nodes get compressed due to the exponential increase in information at every node [2].

One notable example of an MPNN is the Graph Convolutional Network (GCN) [50, 122]. We now introduce this powerful model, but first explain the difference between spectral and spatial GNNs [37, 240]. Spectral GNNs rely on spectral graph theory [43] where signals

on graphs are filtered using the eigendecomposition of the graph Laplacian (defined as $L = D - A$, the difference between the degree matrix D and then adjacency matrix A). Spatial GNNs however consider local neighbourhoods of nodes and perform operations on them. We highlight these two classes because different graph models generally fall into one of those classes. The GCN falls into the spatial class, but can also be considered (implicitly) a spectral low-pass filter on the eigenvalues.

2.5.4.7 Graph Convolutional Networks

Convolutional neural networks are extremely efficient architectures that exploit the local translational invariance of signal classes. Their success was, in 2013, introduced to a new class of data - graphs [26]. In 2016, Kipf and Welling [122] introduce simplifications to these spectral graph convolutions which allow for faster training times and higher predictive accuracy in many cases.

The GCN layers can be explained similarly to Equation (2.44), where we apply some non-linear function to the input features and some form of the node embedding (h_j, which is usually the adjacency matrix). The authors start with a simple form of this equation. Taking the $N \times D$ feature matrix X and some form of the adjacency matrix A, a simple layer-wise propagation rule (the $UPDATE$ function) can be [122]

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}), \tag{2.48}$$

where $H^{(l)}$ is the output to layer l, $W^{(l)}$ is a learnable weight matrix and σ is some nonlinear function. Two limitations arise from this simple yet powerful model. First, when we multiply a node's feature vector by A, it sums up the feature vectors of all the neighbouring nodes but doesn't include the node itself (unless there are self-loops in the graph). To overcome this limitation, we can "fix" it by adding self-loops to the graph. This means that we include the node itself by adding the identity matrix to A ($\hat{A} = A + I$).

The second limitation is related to the scale of the feature vectors. Normally, matrix A is not normalized, which means that when we multiply it with the feature vectors, it can significantly change their scale. We can understand this by looking at the eigenvalues of A. To address this issue, we can normalize A by ensuring that all its rows sum to one. This is achieved by dividing each row of A by the corresponding node degree, *i.e.* $D^{-1}A$ which now corresponds to taking the average of neighbouring node features.

In practice, using a symmetric normalization further enhances the dynamics of the model by multiplying the feature vectors with $D^{-1/2}AD^{-1/2}$ [122]. With this symmetric

normalization, the model goes beyond simply averaging neighbouring node features and by combining these two techniques, we essentially arrive at the propagation rule introduced in the research paper by Kipf et al. [122]

$$f(H^{(l)}, A) = \sigma(\hat{D}^{\frac{1}{2}}\hat{A}\hat{D}^{\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.49}$$

where $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

In 2018, Xie et al. introduced the Crystal Graph Convolutional Neural Networks (CGCNN) [242], a variant of GCN that is specifically designed for predicting the properties of inorganic crystalline materials. The CGCNN model focuses on both the global and the local structure of a crystal by encoding the crystal graphs into a unified representation. This consists of two kinds of graphs: an atom-graph and a bond-graph. The atom-graph is represented by a matrix $F^{(0)}$ whose rows are the feature vectors of atoms in the crystal. The bond-graph, on the other hand, captures the local environments of atoms and is represented by a three-dimensional tensor S.

The atom feature vectors are updated with the function

$$f(F^{(l)}, S) = \sigma(SF^{(l)}W^{(l)}), \tag{2.50}$$

where S represents the bond-graph, $F^{(l)}$ is the atom feature matrix at layer l, and $W^{(l)}$ is a learnable weight matrix.

The CGCNN model incorporates spatial and chemical information in an inorganic crystal structure and learns a mapping from this graph representation to a target property of the crystal. The atom features are summed to produce the crystal graph feature, which is used to predict the properties of the crystal. The CGCNN model significantly outperforms conventional machine learning models for predicting properties of inorganic crystalline materials and has shown that GCNs can be effectively used for a wide range of applications.

2.5.5 More Notable Methods

In this section, we introduce two notable methods of statistical learning: ridge regression (with the kernel trick), and a class of *tree* methods known as gradient-boosted trees.

Ridge regression, as the name suggests, is a type of regression with an added regularization term known as L2-regularization [101]. Otherwise known as ridge regularization, this technique adds the squared norm of the models' coefficients to the loss function. This

penalizes large coefficients as to prevent any over-dependence on any one parameter and prevent over-fitting[8]. The loss function is given by

$$L(\beta) = ||Y - X\beta|| + \lambda||\beta||_2^2, \tag{2.51}$$

where β are the coefficients, Y is the response variable, and λ is the regularization parameter denoting how much regularization to add. $||\cdot||_2$ denotes the 2-norm, otherwise known as the Euclidean distance. We see that setting $\lambda = 0$ results in ordinary least squares as we saw in Section 2.5.1.

Kernel Ridge Regression (KRR) is a technique that adds on top of ridge regression a "kernel trick" which allows the model to learn more complex, nonlinear relationships [204]. It is a mapping to a higher-dimensional feature space that allows a linear algorithm to compute nonlinear features without explicitly computing a transformation. A common kernel function is the Radial Basis Function (RBF) kernel

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{2.52}$$

where x and x' are datapoints and σ is a parameter. To apply this kernel we use

$$L(\beta) = ||Y - K\alpha|| + \lambda||\beta||_2^2, \tag{2.53}$$

where α is a vector of dual coefficients and $K_{ij} = K(x_i, x_j)$. The KRR has seen great success in many fields like bioinformatics and finance [214].

Another class of models we now introduce is the Gradient Boosted Trees (GBT). This is an ensemble learning[9] technique which builds tree models concurrently, with each successive tree correcting errors of the previous one [66]. Gradient boosting looks like

$$F(x) = \sum_{k=1}^{K} \beta_k h(x; \theta_k), \tag{2.54}$$

where $F(x)$ is the prediction, β_k are the weights assigned to each tree, $h(x; \theta_k)$ are the decision trees, and K is the number of trees. At each iteration, a tree is fitted to the negative gradient of the loss with respect to the current predictions. An optimized and scalable way of running GBT is using XGBoost [40]. It implements regularization and parallelization

[8] When a model "memorizes" the input data, including any noise, and therefore generalizes poorly [109].

[9] Where we use multiple weak (smaller, less accurate models) and combine their predictions to create a large, more powerful model.

and is one of the most widely used GBT software packages alongside LightGBM [118] and CatBoost [180].

2.5.6 Optimizers

Often taken for granted in machine learning is the recording of gradients and backpropagation through loss landscapes. This is, in the popular frameworks like Pytorch [170] and Tensorflow [1], usually done under the hood with simple function calls. Gradient Descent (GD) and its stochastic and adaptive variants are the generic approaches to minimizing the error terms like in Equation (2.31). GD is thought to first have been introduced in 1847 by Claude Lemaréchal [139].

More recently, a class of algorithms called the Metropolis Monte Carlo (MC) was introduced to simulate molecular systems [89, 150, 192]. Equilibriating a molecular system is similar to training neural networks in that both involve optimizing parameters interacting in a nonlinear fashion with many degrees of freedom. However, not many algorithms involving these schemes are used in training neural networks by minimizing a loss function. Some notable exceptions exist [190, 212, 228].

In the following, we explain how classic optimizers like GD work, with some mentions of notable variants, we then learn the Metropolis MC algorithm and how this can be used in training neural networks. We then also introduce an adaptive variant following [238].

2.5.6.1 Gradient Descent

Most deep learning algorithms involve some sort of optimization, referring to optimization as the minimization or maximization of some function $f(x)$. Maximization can be seen as a minimization algorithm of $-f(x)$ therefore we usually only discuss minimization algorithms.

We start with the function $f(x)$ and some objective function[10] we wish to minimize. Given that the derivative $f'(x)$ is the slope at a point x, and tells us how a small change in x with affect $f(x)$:

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x). \tag{2.55}$$

We use this trick from calculus to minimize functions - we know how to change x in order to minimize our cost function $f(x)$. This is done by moving x in small steps with the

[10] See Section 2.5.1 for what an objective function is.

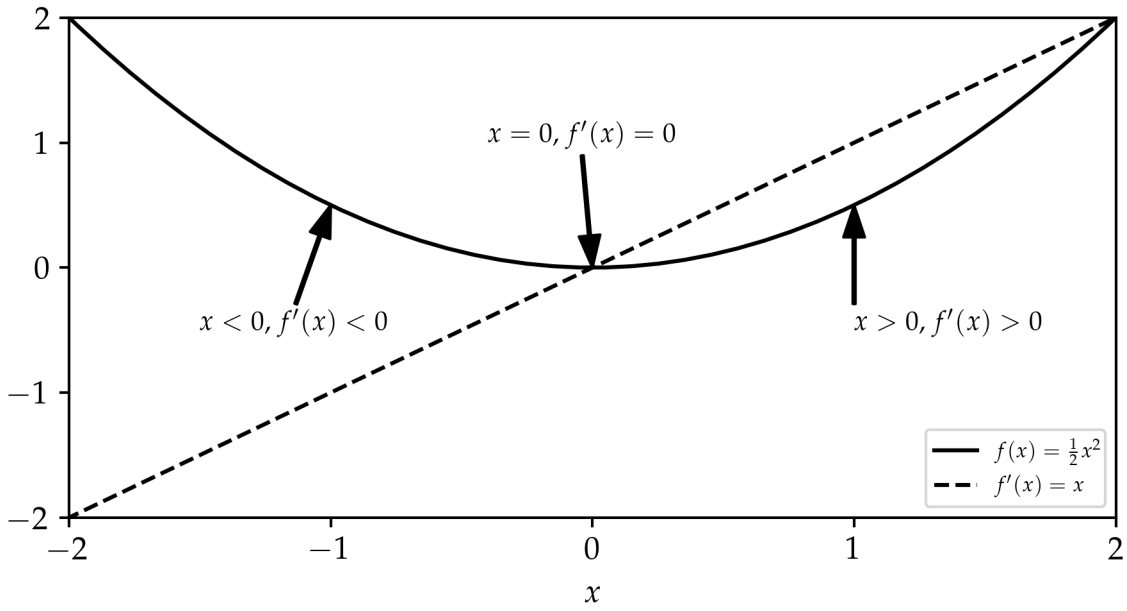


Figure 2.10: Gradient descent. A function $f(x) = \frac{1}{2}x^2$ and its derivative $f'(x) = x$ are depicted. At $x < 0$ we see $f'(x) < 0$ so we can decrease f by moving to the right. At $x = 0$, $f'(x) = 0$ so we have reached the minimum. The GD algorithm would end here and at $x > 0$ we have $f'(x) > 0$ therefore we would take steps to the left. This illustrates for gradient descent would take steps "downhill" - against the gradient - to reach a minimum. This figure was adapted from [83].

opposite sign of the derivative. This technique is what is known as the **gradient descent** and was first introduced by the French mathematician Augustin-Louis Cauchy in 1847 [34]. This is depicted in Figure 2.10.

An important aspect of GD-based algorithms is the learning rate - the rate at which we take steps against the gradient. To minimize our cost function, we would propose a new point at each iteration like so:

$$x' = x - \epsilon \nabla_x f(x) \tag{2.56}$$

where ϵ is the **learning rate**. This is the step size we take in the direction of the steepest descent. It can be chosen in various ways, a common approach being to set it to some small constant arround 1e−3. Another approach known as line search is to evaluate $f(x - \epsilon \nabla_x f(x))$ for different ϵ values and choose the one that results in the smallest resulting cost function [83].

Most machine learning algorithms are based on the GD algorithm and its many variants. The basic GD algorithm we have seen thus far has a few issues. Since we update our parameters in a single batch, we may run into memory problems. This is also slow to compute and is prone to get stuck in a local minimum of the loss function. To overcome these, **mini-batch GD** was introduced where we learn on smaller batches of the data. Then, a stochastic variant was introduced where we learn on randomly sampled smaller batches. This is known as Stochastic Gradient Descent (SGD). This is usually much faster, performs more frequent updates, is more robust to noisy and non-stationary data, and can escape from local minima. However, it may require more iterations to converge than GD and the learning rate needs to be carefully tuned to ensure convergence.

An important extension to SGD is the momentum [179]. As we use mini-batches, there is more variation in the loss of each batch. Momentum is an optimization technique that introduces inertia or a "memory" effect to the parameter updates. By considering the accumulated gradient history in addition to the current gradient, momentum helps the optimization algorithm navigate through flat regions and shallow local minima (of the loss landscape) more efficiently. It smooths out the parameter update trajectory and increases the persistence of updates, resulting in faster convergence. The momentum term is typically between 0 and 1, higher values providing a greater influence on accumulated gradients, reducing oscillations and enhancing the exploration of the parameter space.

Momentum in SGD has demonstrated improved convergence speed, robustness against local minima, and better handling of ill-conditioned problems [46,83]. Our gradient update (Equation (2.55)) now becomes a two-step process according to

$$u = \gamma u - \epsilon \nabla_\theta f(x;\theta) \tag{2.57a}$$

$$\theta = \theta + u. \tag{2.57b}$$

where θ are the parameters of the model. Momentum can be thought of as in the mechanics perspective. The learning algorithm is like a ball moving downhill. A small hill will not stop the ball because of its momentum [83]. A problem that can arise from this, however, is the ball overshooting the minimum. To solve this, the Nesterov momentum [225] was created which adds an additional corrective factor. This uses the derivative of the projected position in the calculation of the new position for the variable. Equation (2.57) becomes

$$v = \alpha v - \epsilon \nabla_\theta \left[\frac{1}{m} \sum_i L(f(x;\theta + \alpha v), y) \right] \tag{2.58a}$$

$$\theta = \theta + v \tag{2.58b}$$

The difference between this and standard momentum is where the gradient gets evaluated - in Nesterov, we evaluate the gradient after the velocity is applied.

Another common technique useful with the SGD algorithm is adapting the learning rate as training proceeds. An early approach to this was the delta-bar-delta algorithm [107]. This uses a simple heuristic: if the partial derivative of the loss with respect to a model parameter remains the same sign then the learning rate increase and if it changes sign then it decreases. However this applies to full-batch optimization, with some newer models being optimized to mini-batches.

The **AdaGrad** algorithm [53] adapts the learning rate of all model parameters by scaling them inversely proportional to the square root of the sum of all historical squared values of the gradient. Consequently, parameters with the largest partial derivative of the loss have a rapid decrease in their learning rate and those with small partial derivatives have a small decrease in their learning rates. This has been shown to result in more gently sloped directions of parameter space [83]. This can be described formally as

$$v = v + (\nabla_\theta f(x;\theta))^2, \tag{2.59a}$$

$$\theta = \theta - \frac{\eta}{\sqrt{v+\epsilon}} \nabla_\theta f(x;\theta). \tag{2.59b}$$

Here, ϵ is a small factor to prevent division by 0. A benefit of Adagrad and other adaptive learning rate methods is that they are much less sensitive to the learning rate. One problem with this algorithm is it accumulates the squared gradients from the beginning of training. It may result in a premature and excessive decrease in the learning rate.

To avoid this problem the root-mean-squared propagation (RMSProp) [100] algorithm was introduced. The accumulating gradient is changed to an exponential moving average, as the name suggests. This discards further history so that the algorithm can converge rapidly after finding a convex part of the loss landscape. This algorithm introduces another parameter ρ which is the decay rate. It is as follows:

$$v_t = \rho v_{t-1} + (1-\rho)(\nabla_\theta f(x;\theta))^2 \tag{2.60a}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t+\epsilon}} \nabla_\theta f(x;\theta_{t-1}). \tag{2.60b}$$

In 2015, the Adaptive Momentum Estimation (Adam) [121] optimizer was introduced which builds on AdaGrad and RMSProp in that it includes both learning rate scaling and momentum. This optimizer has the benefits of including invariant parameter update magnitudes with gradient rescaling, bounded step sizes controlled by a hyperparameter,

compatibility with non-stationary objectives, handling of sparse gradients, and natural step size annealing [121]. The moment vectors (m_t and v_t) are typically initialized to 0 and their update rule is

$$g_t = \nabla_\theta f(x; \theta_{t-1}) \tag{2.61a}$$

$$m_t = \frac{1}{1-\beta_1^t}\left(\beta_1 m_{t-1} + (1-\beta_1) g_t\right) \tag{2.61b}$$

$$v_t = \frac{1}{1-\beta_2^t}\left(\beta_2 v_{t-1} + (1-\beta_2) g_t^2\right) \tag{2.61c}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}. \tag{2.61d}$$

α here is the step size, and ϵ is again a small hyperparameter to prevent division by zero. β_1 and β_2 are the decay rates for the exponential moving averages. The authors define the fraction is Equation (2.61d) as the signal-to-noise ratio as it divides the signal (the gradient) by the "noise" (the second order gradient). They also divide by the square root of the gradient, inspired by RMSProp (Equation (2.60)).

Currently, there is no consensus on which algorithm to choose. Some research shows that is it hard to beat SGD significantly, however, hyperparameter tuning is much less important when using adaptive variants [207]. Currently, the most popular algorithm is Adam. These, however, bring us to a new class of algorithms: Monte-Carlo and evolutionary-based algorithms.

2.5.6.2 Metropolis Monte Carlo

The MC algorithm (otherwise known as Metropolis-Hastings or just Metropolis) was first introduced in the 1950s by a team of physicists trying to simulate interacting particles and investigating properties of molecular systems [150]. The MC algorithm has been used to train neural networks [190, 212, 228] but it is not widely used for these purposes. Here, we introduce the MC algorithm and show how it can be used to train modern neural networks.

MC simulation is a class of numerical methods that use randomness in the sampling - a type of monte carlo technique. Monte carlo (different from MC, which here denotes Metropolis Monte Carlo) algorithms use randomness to approximate statistical values with a measurable error. One well-known application is computing the integral (or area) of a function. Different numerical methods can be used like the Newton-Cotes formulas which work by estimating the integrand at equally spaced points [27]. A simple Monte-Carlo algrithm to do this would be to generate random points on the domain of the function and

then determining whether each point falls within the integration domain by comparing it to the function being integrated. Points that meet the criteria are accepted, and those that do not are rejected. By calculating the ratio of accepted points to the total generated points and scaling it by the volume or area of the bounding region, an estimate of the integral can be obtained.

The Metropolis algorithm works by sampling from a probability distribution and is widely used today in fields ranging from industrial engineering [58, 134, 194] to economics and finance [24, 65, 75, 78] because it is conceptually easy to understand, efficient and theoretically justified [131]. It is a type of *evolutionary algorithm*. That is, a class of black-box optimization algorithms [103, 119] that consist of heuristic-based search procedures inspired by nature (referring to "evolution") [198]. At every iteration (a "generation" of a species), a population of parameters (the "genotypes" comprising that species) is perturbed (random "mutations" of the genes) and some objective function value (the "evolutionary fitness" of the species) is evaluated. The central goal being "survival of the fittest".

Suppose we wish to compute an expectation

$$\mu = \mathbb{E}[g(X)] \tag{2.62}$$

but are unable to compute it exactly so we use the MC method. If we are able to simulate X_i Independent and Identically Distributed (IID) random variables with the same distribution as X, we can define

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i) \tag{2.63}$$

and $Y_i = g(X_i)$. We know Y_i are IID with mean μ and variance

$$\sigma^2 = \text{var}\{g(X)\} \tag{2.64}$$

and by the central limit theorem, we know that

$$\hat{\mu} \approx N(\mu, \frac{\sigma^2}{n}) \tag{2.65}$$

where $\hat{\mu}$ is the sample mean of our predictors Y_i. The variance is estimated by

$$\hat{\sigma}^2 \approx \frac{1}{n}\sum_{i}(Y_i - \hat{\mu})^2, \tag{2.66}$$

with $\hat{\mu}$ being the sample mean. We notice the accuracy is inversely proportional to the

square root of the sample size, however, it is limited since for each additional significant figure, a 10× increase in accuracy requires a 100× increase in the sample size [72].

For the Metropolis-Hastings algorithm, we add the notion of Markov chains to our MC simulation algorithm. These are a study of stochastic processes where the state space is finite or countable. It is generally used when it is hard to sample from our probability distribution. We do this because a large number of parameters and complex probabilities become intractable. In simple terms, a Markov process is one where the conditional distribution of a state space X_{n+1} given X_n does not depend on n [72]. *i.e.*, we do not care about the history, but only the state at step n. A common algorithm to do this is the Metropolis MC [96, 150].

Given a transition matrix T, a Markov chain is time-homogeneous if

$$P(X_{i+1} = a | X_i = b) = T_{ba} \ \forall i, \ \forall a, b \in X \tag{2.67}$$

where X is the set of all possible values for our *stochastic*[11] variables.

The Metropolis algorithm is a method to generate a Markov chain with a desired distribution π. It ensures that we yield a Markov chain that is ergodic[12] and stationary[13] with respect to π. That is, if our state space $X_{(t)} \approx \pi$ then $X_{(t+1)} \approx \pi$ - meaning it will converge to π. We start by selecting a candidate for the next "move", from a probability matrix Q, and accept or reject depending on if the posterior probability of the jump is higher than our initial probability.

1. Initialize $X_1 = x_1$.
2. For $i = 1 \dots n$
   (a) Sample y from $Q(y|x_i)$.
   (b) Compute the acceptance probability

$$A = \min\left(1, \frac{\pi(y)Q(x_i|y)}{\pi(x_i)Q(y|x_i)}\right). \tag{2.68}$$

   (c) Accept y with probability A and set $x_{i+1} = y$, otherwise set $x_{t+1} = x_t$.

[11] Stochastic is a fancy word for *random*.

[12] A Markov chain is ergodic when you can get to any state from any state. See [55, 56] for more details on the ergodicity of stochastic processes.

[13] A stationary distribution is one that remains unchanged as time goes on.

For proof of why this works we refer the reader to the original paper by Metropolis et al [150]. This algorithm has been touted as one of the most influential algorithms of all time and has found widespread applications across various scientific disciplines. This algorithm serves as a powerful tool for generating random samples from complex probability distributions that may be challenging to sample directly. One prominent application lies within statistical physics, where it was originally devised to simulate the behavior of atomic systems at equilibrium. It has also been instrumental in studying the Ising model for ferromagnetism and other lattice systems. Beyond physics, the Metropolis algorithm has been widely adopted in fields such as computational chemistry, where it aids in the prediction of molecular conformations and the exploration of potential energy surfaces [63]. In Bayesian statistics and ML, the algorithm plays a pivotal role in posterior distribution estimation, parameter inference, and modeling [132]. It especially has uses for sampling from high-dimensional functions. Furthermore, it has proven valuable in financial modeling for option pricing and risk assessment [78]. It is a versatile and indispensable tool for solving complex probabilistic problems in diverse domains.

One application of the Metropolis algorithm is estimating complicated probability distributions. For example, the probability distrbution function of

$$P(x) = \frac{\exp(-x^2)(2 + \sin(5x) + \sin(2x))}{\int_{-\inf}^{\inf} \exp(-x^2)(2 + \sin(5x) + \sin(2x))} \tag{2.69}$$

is too complicated to find analytically as the integral is impossible to solve. The Metropolis algorithm allows us to get $P(x)$ by sampling from the unnormalized function $f(x) = \exp(-x^2)(2 + \sin(5x) + \sin(2x))$ following the algorithm above. Using a symmetric proposal distribution, the metropolis acceptance criterion (Equation (2.68)) becomes $A = \min\left(1, \frac{\pi(y)}{\pi(x_i)}\right)$ as $Q(x_i|y) = Q(y|x_i)$ [14]. Figure 2.11 shows the results of running Metropolis to approximate $P(x)$.

Population-based evolutionary algorithms have been used to train networks with many parameters by minimizing a loss function [61, 62, 155] as is generally done with gradient descent. In Witelam et al., [238], we show the ability of a zero-temperature[15] Metropolis

[14] In plain words, the probability of sampling x given y is the same as the probability of sampling y given x.

[15] Zero temperature implies that any moves that result in an increase in loss are disregarded. This decision is based on the proven effectiveness of gradient-descent methods in machine learning and the intuition drawn from Gaussian random surfaces, which suggests that loss surfaces exhibit more downward directions when the loss values are high. By adopting a zero temperature approach, the optimization process prioritizes directions that lead to a decrease in loss, benefiting from the reliability of gradient

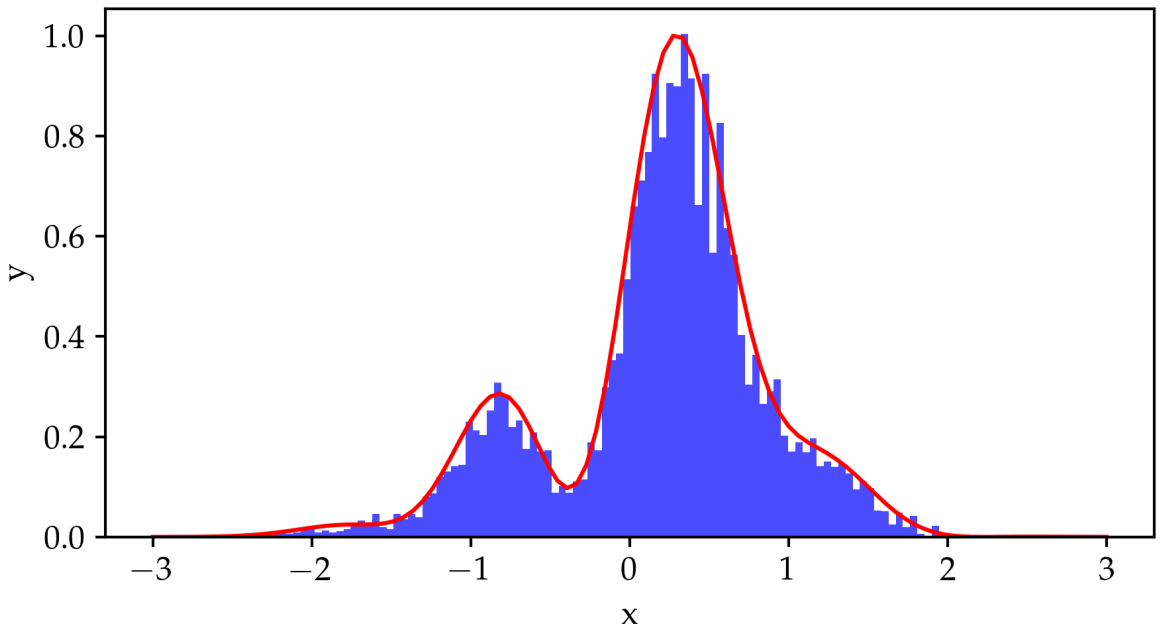


Figure 2.11: Metropolis algorithm example. We find the probability distribution of Equation (2.69) using Metropolis sampling. The red curve shows the normalized curve, and the blue histogram shows the results of running Metropolis for $n = 10$k steps with a normal distribution $\mathcal{N}(x_i, 1)$ for step proposals and starting at point -1.

MC algorithm and an adaptive variant for training neural networks. In the following we follow some of the methods from [238]. We show that the ability of Metropolis MC to train a fully-connected neural network is similar for networks with hundreds or even millions of parameters: there is not necessarily a sharp decline of acceptance rate with increasing network size as is usually claimed. This is done by taking a neural network of fixed structure, proposing a perturbation to all weights and biases, and accepting the change if the loss does not increase.

The proposed mutation to a models parameters is

$$\theta_i \to \theta_i + \epsilon_i \tag{2.70}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ conditional on the loss not increasing. σ needs to be chosen with

descent and capitalizing on the characteristics of loss surfaces at higher loss values [7, 48].

care in simple MC optimization in order to ensure our optimization algorithm reaches a minimum by generating diverse sets of trajectories. See Appendix A.3 for a brief study on diversity of MC trajectories. When the scale of each move σ is small, the value of each parameter θ_i evolves according to the Langevin equation [239]

$$\frac{d\theta_i}{dn} = -\frac{\sigma}{\sqrt{2\pi}} \frac{1}{|\nabla U(\boldsymbol{\theta})|} \frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} + \eta_i(n), \tag{2.71}$$

where n is the training time in epochs, η is a Gaussian white noise with zero mean and variance $<\eta_i(n)\eta_j(n')> = (\sigma^2/2)\delta_{ij}\delta(n-n')$. This means that small stochastic perturbations of the weights and biases (with the zero-temperature condition) are equivalent to noisy clipped or normalized gradient descent on the loss (*i.e.* Equation (2.32)) [238].

2.5.6.3 Adaptive Monte Carlo

As with the adaptive learning rate methods introduced in Section 2.5.6.1, in [238] we introduced an adaptive Monte Carlo (aMC) method. To this end, we introduce the idea of changing the size of each move from Equation (2.70) to adapt as training progresses. We propose the following change

$$\epsilon_i \sim \mathcal{N}(\mu_i, \sigma^2). \tag{2.72}$$

The parameter μ_i is initially set to zero and updated according to

$$\mu_i \to \mu_i + \epsilon(\epsilon_i - \mu_i), \tag{2.73}$$

where ϵ here is a hyperparameter of the model. This is similar to the momentum in the gradient-based schemes introduced in Section 2.5.6.1. The idea is to shift the center of each move μ_i towards the last accepted move, ϵ_i, with the goal of increasing the probability that a future move will be accepted. To remove the need for a large search space, a simple learning-rate scheduling $\sigma \to 0.95\sigma$ (and $\mu_i = 0$) was introduced after n_s (another hyperparameter) consecutive rejected moves. The aMC algorithm, therefore, follows the following four steps, as introduced in [238]:

1. *Current State.* Record the current state of the neural network θ. Sample the data (the batch), and compute the loss $U(\theta)$. If signal norm is on, calculate the values λ_i from Equation (2.74).

2. *Move Proposal.* Propose a move using Equation (2.70) for each NN parameter with $\sigma_i = \lambda_i\sigma$. Evaluate the new loss $U(\theta')$ from the proposal and accept if $U(\theta') \leq U(\theta)$ and go to step 3. otherwise we reject the move and go to step 4.

3. *Accept move.* Make the proposed move $\theta \to \theta'$ and set $n_{cr} = 0$. Update each step size using Equation (2.73). Return to step 1.

4. *Reject move.* Set $n_{cr} \to n_{cr} + 1$. If $n_{cr} = n_s$ then set $n_{cr} = 0$, $\sigma \to 0.95\sigma$ and set $\mu_i = 0, \forall i$. Return to step 1.

This algorithm requires only the computation of the loss and not the gradients at any point. The memory cost of aMC is the cost to calculate the loss, to store two versions of the model, and (if $\epsilon \neq 0$ and signal norm is on) to store the values μ_i and λ_i for each parameter of the NN. The computational cost of a move is drawing N Gaussian random numbers and calculating the loss function. This requires fewer computations than gradient-based methods as there is no gradient computation.

2.5.6.4 Signal Norm

For select deep networks, it may be useful to choose λ_i specifically to keep the scale of updates for each neuron equal. This follows ideas applied to gradient-based methods (adaptive GD as seen in Section 2.5.6.1) [45]. This concept is introduced as **signal-norm**. When it is turned off, we set all $\lambda_i = 1$ from step 2, however, when it is on, we normalize inputsc to a neuron by setting

$$\lambda_i = \left(N_{\text{data}}^{-1} \sum_{\alpha=1}^{N_{\text{data}}} \sum_{i' \to j}^{N_j} (S_{i'}^{\alpha})^2 \right)^{-1/2}, \tag{2.74}$$

where N_j is the "fan-in" of neuron j, S_i^{α} is the output of neuron i given one particular evaluation α of the network. For the full derivation of why this works see [238]. Under Equation (2.74), weights on connections that feed into a neuron receiving **many** other connections will experience a smaller move scale than weights on connections that feed into a neuron receiving few connections. Similarly, weights on connections fed by more active neurons will experience a smaller basic move scale than weights on connections fed by relatively inactive neurons. The idea is to shift move proposals towards the last accepted move thereby increasing the chances of generating moves that will be accepted. This equation does not apply to neurons that are a bias (we set $\lambda_i = 1$).

2.5.6.5 Tests

In [238], we tested whether MC and aMC can train shallow and deep neural networks as well as GD. To that end, we ran a series of tests comparing MC, aMC, and GD. First, we show that for certain problems, the acceptance rate of aMC is much higher than the Metropolis algorithm at lower loss values, and is more insensitive to network width, depth, and size. Then, we show that the momentum-like term in aMC speeds up learning and can learn high-frequency features much like adaptive methods do when compared to vanilla GD. Furthermore, we show that the MC method is able to train a simple recurrent neural network, specifically where gradient methods fail. Lastly, we show that aMC can train deep networks in which gradient methods fail due to small gradients. See Section 3.2 for some of these results.

2.6 ORGANIZER: Molecule Database

The following chapter is adapted from Theophile Gaudin's PhD thesis at the University of Toronto [70]. It will be submitted as an article coauthored by me in the near future.

In order to meet the increasing energy demands of society, there is a pressing need for the development of new materials. Self-driving laboratories have emerged as a promising avenue to drive advancements in material discovery. Early self-driving laboratories [60, 84, 146], equipped with single robotic platforms, have made significant progress in this field. However, they still face limitations in their ability to discover materials at the required pace, primarily due to the vastness of the chemical space and limited instrument throughput. To overcome these challenges, the incorporation of additional laboratories and instruments into the self-driving laboratory process holds immense potential. By leveraging the combined resources and expertise of multiple self-driving labs, a more comprehensive exploration of the chemical space can be achieved, greatly increasing the chances of material discovery.

Among the many materials discovery and acceleration platforms [42, 87, 106], there is a clear lack of data management and sharing. Some lightweight approaches exist and are often employed such as sharing data in memory, which is very susceptible to loss of data. Other approaches include sharing files in different formats, namely parquet, CSV, Matlab files etc. The preferred approach is definitely backing up the data to some server where they are safeguarded. With this method, however, classifying and tracking data becomes a big issue. It is essential to establish good practices so this can be done, effectively accelerating this important field of research.

Databases are a common way to store and share data. Examples of databases in the field of chemical engineering are ChemOS [191] and Materials Experiment and Analysis Database [217] which respectively use SQLite and MySQL and although these projects encapsulate the benefits of implementing relational database management systems, they are either not accessible enough or generic and modular enough to be applied to arbitrary material discovery campaigns. Here lies the need for ORGANIZER, with its modularity.

To facilitate the realization of a network of multiple self-driving laboratories across physical locations, a cloud-based software framework known as ORGANIZER has been proposed. ORGANIZER serves as a distributed system, where instruments function as asynchronous workers that can seamlessly join or leave the system as needed, for instance, during maintenance. The framework encompasses two essential components: a database that meticulously tracks experiment information and results, and a work dispatcher that efficiently schedules experiments. In this work, the software framework behind ORGANIZER is thoroughly described, showcasing its capabilities through an experimental campaign fo-

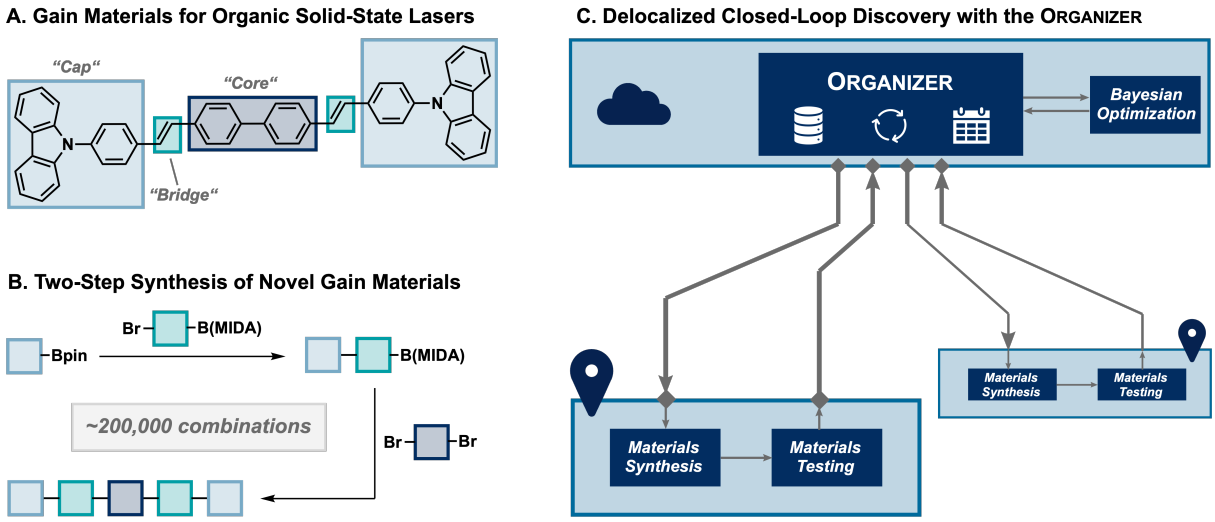


Figure 2.12: **A.** Gain materials for Organic Solid-State Lasing Devices (OSLD)s. Exemplified with 4,4-Bis[(N-carbazole)styryl]- biphenyl (BSBCz). **B.** Two-step synthesis of a space of approx. $200,000$ novel gain materials for OSLDs. **C.** A delocalized self-driving laboratory using `ORGANIZER` as a central platform for experiment orchestration and data storage. Image taken from [70].

cused on organic lasers. This campaign serves as a demonstration of `ORGANIZER`'s potential to provide a flexible and scalable platform for the discovery of novel materials.

By enabling effective collaboration among self-driving labs and offering a delocalized approach to chemistry experimentation, `ORGANIZER` stands as a pioneering effort in this field. The framework presents a promising solution to overcome the limitations faced by early self-driving laboratories and offers a pathway to accelerate the discovery of materials essential for addressing societal energy needs.

To showcase `ORGANIZER`, we worked on an application of it as the central experiment planning hub in a large-scale, delocalized, asynchronous optimization campaign targeting materials for OSLDs. These materials hold promise for the next generation of laser devices, owing to their accessibility, facile tunability and flexible device processing compared to established inorganic crystal lasers [113]. At the same time, the structure-property relationships for molecular lasing have remained largely unknown. For these reasons, we set out a large-scale discovery campaign for novel OSLD gain materials, inspired by the 4,4-bis[(*N*-carbazole)styryl]biphenyl (BSBCz) [217] scaffold. Specifically, we identified the molecular pentamers of the general "cap-bridge-core-bridge-cap" structure as a promising

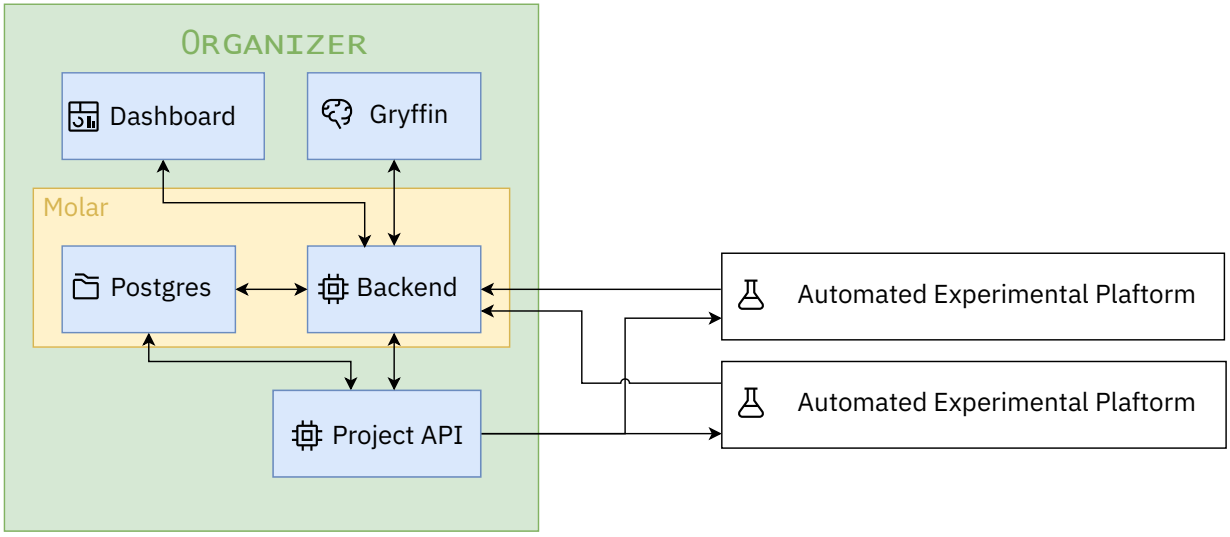


Figure 2.13: ORGANIZER and its components. Figure taken from [70].

search space for OSLD gain materials. These molecules can be prepared in an automated fashion following a novel two-step, iterative SuzukiMiyaura coupling scheme as shown in Figure 2.12 [21, 76].

To optimize experimental throughput and accommodate the availability of different reactants in our laboratories, we adopted a parallelized approach by distributing synthesis efforts across five robotic platforms located at four different sites. Coordinated and streamlined execution of these experiments was made possible through the implementation of ORGANIZER, in a cloud-based platform serving as the central hub for data storage and experiment planning. Within the closed-loop workflow, experiment recommendations were generated directly in the ORGANIZER database and subsequently downloaded and executed locally on the corresponding robotic platforms. Continuous progress updates and obtained experimental data were seamlessly uploaded to ORGANIZER, facilitating asynchronous orchestration of these experiments. Through the implementation of this distributed campaign, we successfully achieved rapid and parallel execution of numerous experiments, ultimately leading to the discovery of novel and high-performing gain materials for OSLDs.

2.6.1 Design

ORGANIZER has been designed to offer adaptability to diverse material discovery campaigns, although the need for custom code will arise when catering to specific project requirements.

The framework follows a modular approach, comprising four key modules: a centralized database, a backend system for efficient data management and authentication, a recommendation system leveraging machine learning techniques, and communication drivers tailored to facilitate seamless interaction between self-driving labs and instruments. Notably, Modules 1 and 2 which encompass the centralized database and backend system have already been made available on GitHub under the name Molar[16]. The recommendation system, Module 3, make use of Gryffin [95] to provide valuable insights and prioritize experiments based on prior results. Module 4, the communication drivers, vary in their implementation depending on the application or hardware employed. The interactions between these components are illustrated in Figure 2.13, providing a visual representation of the orchestration within `ORGANIZER`. Furthermore, to cater to the specific needs of this material discovery campaign, two additional modules have been developed: a visualization dashboard enabling experts to monitor progress and manage lab inventory, and a project-specific API ensuring efficient experiment dispatching while preventing duplication. The comprehensive design and development of `ORGANIZER`, alongside its additional modules, demonstrate its potential as a versatile and powerful tool in advancing material discovery.

2.6.1.1 MolarDB

In the realm of high-throughput experiments and material discovery campaigns, the focus is often directed toward the ultimate outcome while inadvertently neglecting the role of data. The fast-paced nature of some projects, coupled with the urgency to achieve results, contributes to the lack of attention given to data. However, within our specific project, we recognize that this approach falls short on two fundamental fronts. Firstly, the data collected during these campaigns possess an intrinsic value that extends beyond the immediate context, warranting its preservation and usability for future endeavours. Secondly, as our project aims to foster collaboration among researchers and laboratories, it becomes imperative to establish a centralized repository for data to avoid duplicating efforts and facilitate the sharing of valuable insights.

To address these needs and enhance data management and accessibility, we have developed a Python package called **Molar**. Molar serves as a comprehensive solution by streamlining the access and management of databases through the utilization of industry-standard tools, notably PostgreSQL [88] and Docker [149]. By leveraging these well-established technologies, Molar provides a user-friendly interface that simplifies the installation and

[16]This can be found under the `aspuru-guzik-group/molar` repository. See https://github.com/aspuru-guzik-group/molar.

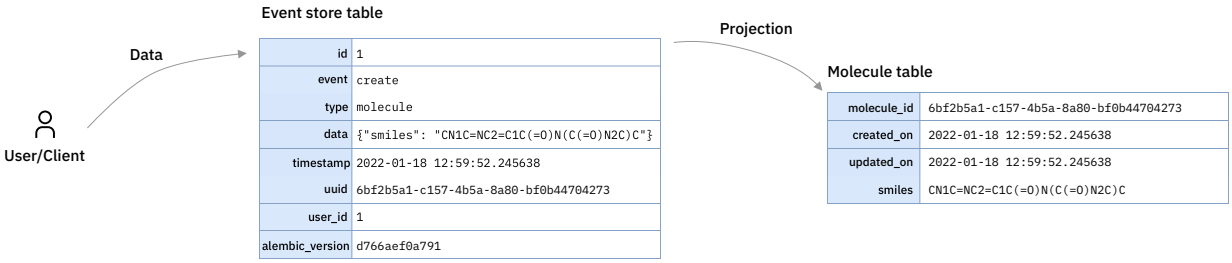


Figure 2.14: Molar event sourcing. Data from the user are sent to the event-store table before being propagated throughout the database. Keeping records of the event store allow for "rollbacks" of the database if data are found to be incorrect or fraudulent.

deployment processes on any Docker-enabled machine with these two straightforward Bash commands:

```
pip install molar[backend]
molarcli install local
```

This enables researchers and scientists to quickly integrate Molar into their existing workflow without the burden of complex setup procedures.

To accommodate the dynamic nature of data structures generally encountered in high-throughput campaigns, Molar incorporates a pivotal feature: the addition of an extra json field to its database tables. This innovative approach allows for the capture and representation of complex relationships between both known and novel types of data, ensuring the flexibility required for efficient data management and analysis. By embracing this adaptive design, Molar allows users to seamlessly incorporate new data types as they emerge, eliminating the need for rigid pre-defined structures that might hinder the exploration of unanticipated data sizes.

Recognizing the importance of data security and access control, Molar incorporates robust mechanisms to safeguard sensitive information and provide secure access to authorized users. The package integrates built-in user and authentication management, employing a JWT-based authentication [115] approach to authenticate and authorize users effectively. This comprehensive authentication framework ensures that only allowed individuals can access the data, mitigating security vulnerabilities that may arise from unauthorized access attempts.

Moreover, to safeguard against unintended errors and data corruption, Molar implements event-sourcing within its architecture. This approach guarantees the integrity of

data by recording all data modifications within a designated table prior to their propagation. The event-sourcing mechanism can be seen in Figure 2.14. The event-sourcing mechanism provides an added layer of protection and facilitates data recovery by allowing database "rollbacks" to any desired point in time.

Molar's inherent modularity and extensibility stem from its foundation as a Docker container-based solution and its structure versioning using Alembic [221]. This tool helps track inventory in different laboratories which allows the proper dispatching of jobs when materials allow it. It also helps in tracking currently running jobs and finally viewing overall results from anywhere. The custom structure built for the OSLD discovery can be seen in Figure 2.15.

The project's API has been specifically designed to handle the storage of large files resulting from high-performance liquid chromatography characterization and other calculations. To ensure efficient and secure storage, we have seamlessly integrated a REST API that establishes a connection with a minio storage server [152]. `ORGANIZER` further enhances usability by providing a convenient authentication feature, allowing users to authenticate with a single POST call, streamlining the authentication process. The following code snippets demonstrate how user credentials can be easily verified from any submodule within the system.

```
import requests

def verify_user(token: str) -> bool:
    response = requests.post(
        f"{MOLAR_BACKEND_URL}/login/
␣␣␣␣␣␣␣␣test-token?database_name={DATABASE_NAME}",
        headers={"Authorization": f"Bearer␣{token}"},
    )
    if response.status_code != 200:
        raise Exception(response.json()["detail"])
    user = response.json()
    if not user["is_active"]:
        raise Exception("Your␣account␣is␣not␣activated!")
    return user
```

where the *token* variable corresponds to the JWT access token provided by the backend to the user. Since REST APIs may not be the most user-friendly, we have also implemented a Python client and web frontend. The Python client offers a way to interact with the `ORGANIZER` backend. Results are returned in a Pandas [164] dataframe which should be

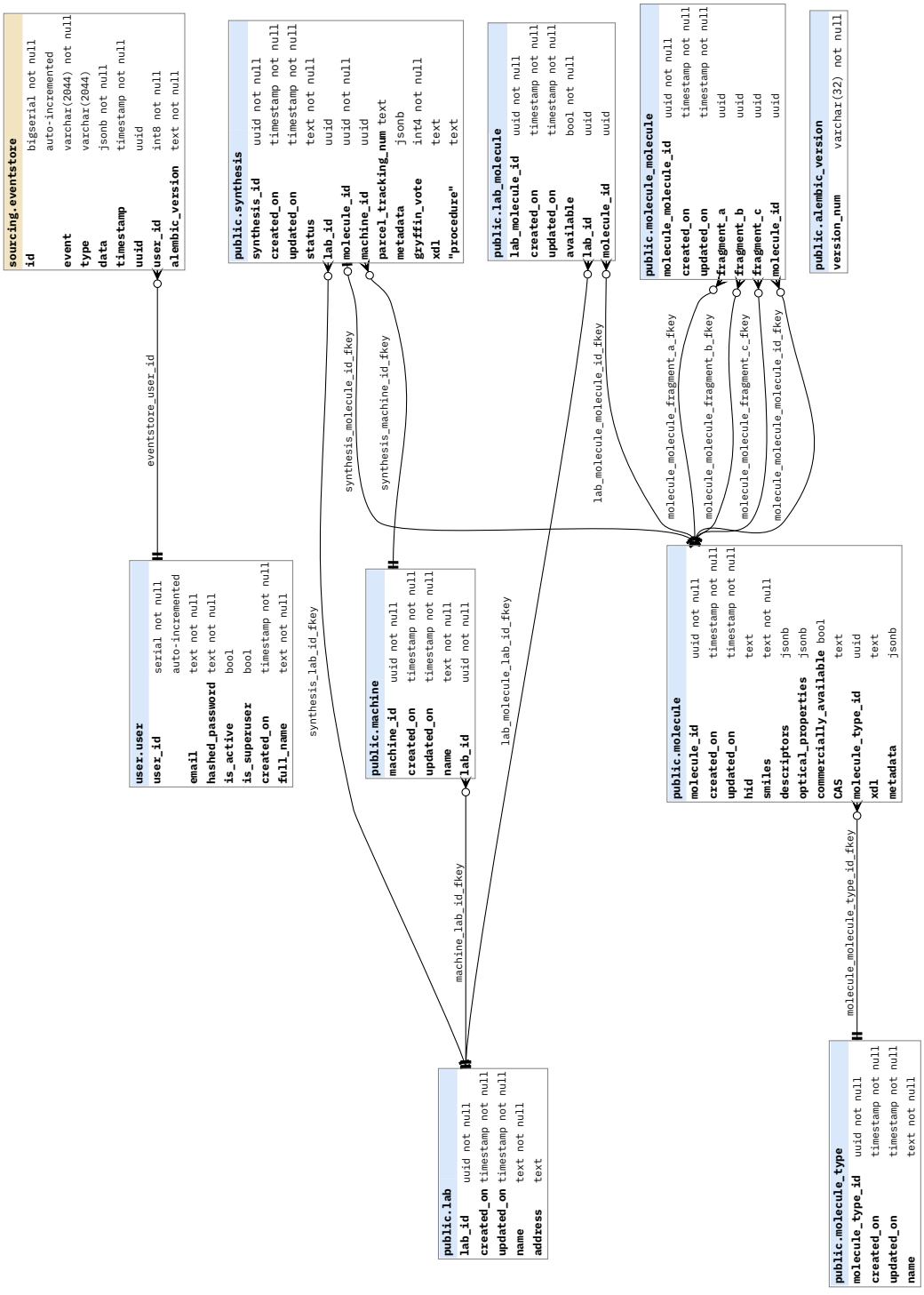


Figure 2.15: Database project for the organic laser molecule material campaign. Arrows depict relationships between tables. This figure was generated with the help of SchemaCrawler (https://www.schemacrawler.com/) and taken from [70].

familiar to most scientists using Python. The web interface was built using React [219]. This ensures that any researcher with even low technical skills can interact with ORGANIZER. From the web interface, it is possible to edit inventory, view the results of experiments, etc.

2.6.1.2 Closed-Loop Experiment Campaign

In our research, we utilized ORGANIZER as the central hub for experiment planning and data storage in a delocalized discovery campaign aimed at identifying novel gain materials for OSLDs. This campaign involved the collaboration of multiple experimental laboratories, including the University of Toronto(UoT), the University of Illinois Urbana-Champaign(UoIUC), the University of British Columbia Vancouver(UBCV), and the University of Glasgow(UoG).

To ensure efficient coordination and synchronization of the campaign, ORGANIZER was integrated with robotic platforms, compound inventories, and the Bayesian Optimization algorithm for experiment planning. By serving as the core unit of the closed-loop discovery workflow, ORGANIZER facilitated seamless communication and information sharing among the participating laboratories.

The primary objective of our campaign was to optimize the properties of organic pentamers belonging to the cap-bridge-core-bridge-cap family. These organic compounds were synthesized using an automated, two-step iterative Suzuki-Miyaura cross-coupling process. The availability of commercially accessible building blocks allowed us to explore a vast search space comprising approximately $200,000$ molecules. By leveraging the capabilities of ORGANIZER and the collaborative efforts of the participating laboratories, we aimed to expedite the discovery of high-performance gain materials for organic solid-state lasing devices. The centralized nature of ORGANIZER enabled efficient experiment planning, data storage, and analysis, fostering a streamlined and productive research environment.

The campaign would leverage the Bayesian optimizer that interacts with ORGANIZER through the Python API to obtain vital information regarding the campaign status. This includes retrieving previous experimental results and simulated molecular descriptors, which play a key role in building a surrogate model to capture the underlying relationship between molecular structure and lasing performance. One of the significant contributions of the ORGANIZER is its ability to provide real-time information on the availability of synthesis building blocks at different laboratory sites. This feature enables the construction of a dynamic search space, allowing for the efficient allocation of experimental resources. The Bayesian optimizer utilizes this information to generate new experimental recommen-

dations that are directly uploaded to ORGANIZER. By facilitating seamless communication and data exchange, ORGANIZER acts as a catalyst in driving the discovery process forward.

To execute the experiments, five robotic platforms were deployed across four different sites. These platforms encompass diverse functionalities, including Chemputers [224] at the UoBCV and the UoG which were dedicated to large-scale building block preparation and synthesis machines employed for the final material synthesis. The final OSLD gain materials were synthesized on a Chemspeed Swing platform at the UoT and with the use of two parallelized synthesis machines by Burke et al [141] (at the UoIUC and the UoT). Furthermore, to ensure accurate analysis and characterization of the experimental outcomes, the reaction mixture undergoes automated analysis using High Performance Liquid Chromatography (HPLC)-Mass Spectrometry (MS), followed by spectroscopic characterization of the target compound. The resulting experiment statuses, HPLC-MS data, and optical analysis results were processed and systematically uploaded to ORGANIZER through the Python API.

One important challenge encountered was the sharing of incompatible data types. Specifically, the ThermoFisher Scientific Vanquish HPLC-MS system at the UoT and the Agilent HPLC-MS system at the UoIUC lacks interoperability. This challenge was important because we required the exchange of chromatographic data beyond simple peak tables.

With the lack of standards in open data formats for HPLCs, there are many custom and proprietary formats to handle. Instrument manufacturers usually have some proprietary data formats. The mzML data format is a widespread format for mass spectrometric chromatography. Some tools exist which aim to bridge this gap; HappyTools [111] exports summaries of HPLC data in tabular formats, and ChromConverter [13] extracts data in the R language. Mocca [90] provides a handy jupyter notebook [124] interface for analyzing HPLC data from Agilent Chemstation. Mocca is also able to interface with the HDF5-based Allotrope Data Format [151]. However, none of these programs fit our need for Python-based raw data extraction from the ThermoFisher Scientific Agilent MassHunter proprietary formats and a lightweight open data specification for storage in molar.

We have developed an open-source data specification for HPLC-MS data, using JSON as the file format. The specification includes the storage of mass spectrometry data in the mzML format, as well as the extraction of raw HPLC diode array detector (DAD) data using a dictionary format compatible with numpy. Custom code and adaptations of existing tools were employed to extract data from different formats, including ThermoFisher Scientific and Agilent MassHunter DAD. The specification incorporates machine-readable metadata, providing information about gradients and columns (manufacturer, model, model

number, length in mm, inner diameter in mm, pore size in Å, and particle size). Live data extraction is achieved by monitoring specified directories, and to address the large size of HPLC-MS datasets, a tarball compression algorithm is used. This open data format can be extended or integrated with other hierarchical data formats such as HDF, Parquet, or the Allotrope Data Format, offering flexibility and compatibility for data storage and analysis.

2.7 Learning Long-Range Atomic Interactions

With the ever-increasing demand for new materials with improved properties, particularly in the context of energy storage, traditional experimental approaches alone have become unsustainable. High-throughput screening, coupled with machine learning algorithms, provides a powerful framework for accelerating the discovery and design of materials [81, 173, 182]. By leveraging large-scale databases (see Section 2.6), advanced computational models, and intelligent algorithms(Section 2.5.2), researchers are now able to rapidly explore and predict the properties of a vast range of materials, enabling targeted and efficient experimentation. This paradigm shift in materials discovery offers great promise for addressing critical energy challenges and propelling advancements in various other fields reliant on tailored materials.

While machine learning has shown great promise in materials discovery, it is not without its challenges. One of the primary difficulties lies in the representation of materials, which is crucial for the success of machine learning algorithms. Materials are complex entities with properties that emerge from the interactions of numerous atoms, and capturing this complexity in a form that a machine learning algorithm can understand is a nontrivial task [196].

Traditional descriptors or "fingerprints" used for molecules often fail to capture the complexity of materials, leading to inaccurate predictions. Furthermore, machine learning algorithms can fail when extrapolating to materials outside the training set, limiting their predictive power to only similar materials [185].

Tailored atomic representations, such as the Many-Body Tensor Representation (MBTR) [104] and Smooth Overlap of Atomic Positions (SOAP) [12], have been developed to address these issues. These representations capture the local atomic environment and the many-body interactions, providing a more accurate and comprehensive description of materials.

However, creating these representations is computationally intensive and requires a deep understanding of the material's structure and properties. Despite these challenges, the development of more accurate and efficient representations remains an active area of research, with the potential to significantly enhance the predictive power of ML in materials discovery.

2.7.1 Representations of Materials

The success of ML in materials design hinges on the ability to effectively represent materials in a manner that captures essential features and allows for accurate prediction of their properties. This chapter delves into various methods of materials representation used in ML for materials discovery, providing an overview of traditional descriptors, tailored atomic representations, and emerging techniques.

Traditional descriptors, such as composition and crystal structure, have been widely used in the early stages of ML for materials. **Compositional descriptors** are based on the types and proportions of elements present in a material. Examples include the atomic fraction of each element, the mean and range of atomic radii, electronegativity, and ionization energy. These descriptors can capture basic chemical properties of the material but often fail to account for the complex interactions between atoms [235]. **Structural descriptors** capture some aspects of the material's structure but lack in certain aspects. Examples are the coordination number (the number of nearest neighbours an atom has), bond lengths, bond angles, and the distribution of these quantities. For crystalline materials, descriptors such as the crystal system (e.g., cubic, hexagonal), space group, and lattice parameters are often used. These descriptors can capture some aspects of the material's local structure but don't account for more complex structural features.

Ideally, descriptors in materials science should encapsulate information that could distinguish between different atomic and crystal environments [73]. One of the earliest strategies for extracting quantitative representations of crystal materials involved the use of pairwise distance models to predict potential energy [17]. However, these models are limited in that they only work for a fixed number of atoms and do not provide a unique representation under the permutation of atoms. Some machine learning models in materials science have been developed that rely on datasets of compounds with the same stoichiometry or the same structure. These models are often used when the goal is to understand the variations in properties within a specific class of materials or to predict new materials within that class. For instance, models that rely on datasets of compounds with the same stoichiometry are often used in the study of alloys or intermetallic compounds. In these cases, the stoichiometry is fixed, but the arrangement of atoms can vary, leading to different crystal structures and properties. ML models can be trained on these datasets to predict the most stable crystal structure for a given stoichiometry or to predict the properties of new alloys with the same stoichiometry [210]. Similarly, models that rely on datasets of compounds with the same structure are often used in the study of crystal structure-property relationships. For example, perovskites are a class of materials that all share the same crystal structure but can have vastly different properties depending on

the specific elements present. Machine learning models can be trained on datasets of perovskites to predict the properties of new perovskites or to identify promising candidates for specific applications [178].

While these models can be quite effective within their specific domain, they are limited in that they cannot be easily generalized to other stoichiometries or structures. This has led to the development of more sophisticated models that can handle diverse compositions and structures. To fully encompass the diverse compositions and structures found in crystal materials, it is said that a representation invariant under rotation, translation, and scaling transformations is necessary.

Faber et al. proposed generalized Coulomb Matrix (CM) approaches to address this need [59]. The first approach considers full Coulomb interactions between two atoms in a lattice. The second approach models atomic electrostatic interactions in the unit cell and its nearest neighbour environment. In the third approach, the Coulomb interaction is replaced by a periodic potential with respect to the lattice vectors.

Schütt et al. constructed a model to predict the density of states at the Fermi energy based on a crystal representation called the partial radial distribution function. This representation is invariant under translation, rotation, and the choice of the unit cell [209].

Xie et al. proposed a novel approach to constructing features using a crystal graph convolutional neural network, as introduced in Section 2.5.4.7, which is invariant for unit cell choices and achieves high prediction accuracy for many properties [242]. However, neural networks are often considered "black boxes" due to their large number of parameters and it is sometimes difficult to interpret the final predictors physically. Therefore a need for representations inspired by the physical system remains.

In contrast, topology, which considers the global connectivity of various components in space, has been used to study isolated entities, rings, and higher dimensional faces [205]. Traditional topology often results in too much geometric reduction to provide a useful description of crystal structures. Persistent homology, however, bridges geometrical shape analysis and topological characterization by embedding multiscale geometric information into topological invariants [57]. This method has been applied to represent organic molecular and biomolecular properties, and its successful application in biomolecules has motivated its use in representing crystal compounds for predicting their physical properties [30, 31, 47, 241].

As the need for more detailed and specific representations of materials increased, descriptors have been divided into local and global descriptors [47, 148]. **Local descriptors** focus on the environment of individual atoms or atomic sites. They are particularly suited to materials where local variations can drastically impact material properties, such as alloys

or amorphous materials. Examples include Atom-Centred Symmetry Functions (ACSF), which encode the coordination environment of atoms [15], and SOAP descriptors, which represent **local** atomic environments by considering overlaps of atomic Gaussian distributions centred at neighbouring atoms [11].

SOAP is a sophisticated descriptor that provides a numerical representation of local atomic environments. It forms part of a larger family of atom-centred descriptors that are pivotal to accurately capturing atomic spatial configurations. The essence of SOAP lies in treating atomic environments as a form of "density" rather than discrete atomic positions. For each atom in the local environment, a Gaussian distribution is placed centred at the atom. As these Gaussian distributions overlap, a continuous representation of the atomic environment is created. This overlap encodes both the individual atomic positions and the aggregate distribution, capturing details of the local atomic topology. The representation of atomic environments in SOAP is more formally defined as a power spectrum[17] obtained from the expansion of atomic density in a series of radial and spherical harmonic functions. Mathematically, the atomic environment density of species i is given by:

$$\rho_i(\mathbf{r}) = \sum_{j:Z_j=i} e^{-(\mathbf{r}-\mathbf{r}_j)^2/2\sigma^2} \tag{2.75}$$

where $\mathbf{r}_j$ are the positions of atoms of species i, and σ is the width of the distribusion. The atomic density $\rho_i(\mathbf{r})$ is then expanded:

$$\rho_i(\mathbf{r}) = \sum_{nlm} c^i_{nlm} g_n(r) Y_{lm}(\theta, \phi) \tag{2.76}$$

where $g_n(r)$ are radial basis functions and $Y_{lm}(\theta, \phi)$ are spherical harmonics. Finally, the SOAP representation is given by the power spectrum of the coefficients c^i_{nlm}, defined as:

$$p(r) = \sum_{m=-l}^{l} c^i_{nlm} {c^{i'}_{n'lm}}^*. \tag{2.77}$$

This representation is rotationally invariant which ensures that the SOAP descriptor doesn't change with the rotation of atomic environments, a prerequisite for descriptors as material properties are agnostic to the rotation of environments. This blend of local atomic position and distribution encapsulation makes SOAP a robust descriptor capable of modelling intri-

[17] It essentially captures the frequencies at which variations occur in the atomic density, and their corresponding intensities.

cate short-range and medium-range atomic order, which are critical to predicting materials' properties accurately [52].

ACSF represents a powerful method for encoding atomic environments as well. This method was originally developed by Behler and Parrinello in the context of neural network potential energy surfaces. ACSFs create a unique fingerprint for each atom based on its local environment. They capture both radial and angular information, encapsulating both bond lengths and angles. As a result, they encode the essential geometric features required to model complex properties and behaviours in a variety of materials.

Mathematically, ACSFs consist of two types of functions: radial symmetry functions (RSFs) and angular symmetry functions (ASFs). RSFs are used to describe the pairwise interactions between atoms and are given by:

$$G_i^{\text{rad}} = \sum_{j \neq i} e^{-\eta(r_{ij} - r_c)^2} f_c(r_{ij}) \tag{2.78}$$

where r_{ij} is the distance between atoms i and j, r_c is a cutoff radius beyond which interactions are not considered, η is a parameter that determines the width of the Gaussian centred at r_c, and $f_c(r)$ is a smooth cutoff function that ensures G_i^{rad} goes to zero smoothly as r_{ij} approaches r_c.

ASFs, on the other hand, are used to describe the angular distribution of neighbours and are given by:

$$G_i^{\text{ang}} = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} (1 + \cos(\theta_{ijk}))^{\zeta} e^{-\eta[(r_{ij} - r_c)^2 + (r_{ik} - r_c)^2]} f_c(r_{ij}) f_c(r_{ik}) \tag{2.79}$$

where θ_{ijk} is the angle between atoms j, i and k, and ζ is a parameter that determines the width of the cosine function.

The combination of RSFs and ASFs provides a comprehensive descriptor of an atom's local environment, capturing both the distribution of neighbouring atoms and the angles between them. ACSFs are highly adaptable, allowing the parameters η, r_c and ζ to be tuned according to the specific requirements of the material or property being studied. Additionally, they are invariant under translation and rotation transformations.

On the other hand, **global descriptors** capture features of the material as a whole. They are often used to describe bulk properties that depend on the overall structure of the material, such as density, bandgap, or crystal symmetry. Notable examples include the Coulomb matrix, which represents the interactions between all pairs of atoms in a molecule

or a periodic cell [196], and the MBTR, which encapsulates the many-body correlations of atoms in a material [104].

The **CM** is a powerful global descriptor for molecules that encodes the nuclear charge and the inverse distance between all atom pairs in a molecule. It was originally proposed by Rupp et al. for quantum mechanical molecular atomization energy predictions using kernel ridge regression. As a descriptor, it is powerful because it exploits the key principles of quantum mechanics, namely the Coulomb interactions between charged particles, which are fundamental for understanding molecular properties.

The CM for a molecule with N atoms is a symmetric $N \times N$ matrix M, where the element M_{ij} is defined as follows:

- For $i = j$, it is 0.5 times the atomic number Z_i to the power of 2.4 (an empirical scalar). This is a representation of the atomic self-energy.
- For $i \neq j$, it is the atomic number Z_i times the atomic number Z_j divided by the Euclidean distance r_{ij} between atoms i and j. This reflects the pairwise Coulomb repulsion between atoms.

Mathematically, this can be represented as:

$$M_{ij} = \begin{cases} 0.5 Z_i^{2.4}, & \text{if } i = j \\ \frac{Z_i Z_j}{r_{ij}}, & \text{if } i \neq j. \end{cases} \tag{2.80}$$

One of the challenges with the CM is that it is not uniquely defined because the rows and columns can be permuted according to the atom indices. This leads to different CM representations for the same molecule, which can be a challenge for machine-learning models. A common solution to this issue is to sort the rows and columns of the CM by the norm, leading to what is known as the "eigenspectrum" or "sorted Coulomb matrix". Another challenge with the CM is that the dimension of the CM scales with the number of atoms in the molecule. This can make it more difficult to apply for larger molecules or materials. Despite these challenges, the CM has been successfully applied to a range of molecular property prediction tasks and has set the foundation for more advanced global descriptors [156, 184].

The Coulomb matrix can also be taken one step further, by computing the sorted eigenvalues of the CM known as the Coulomb Matrix Eigenvalue (CME) [196] representation. These provide a unique representation of the atomic structure, capturing the positions of

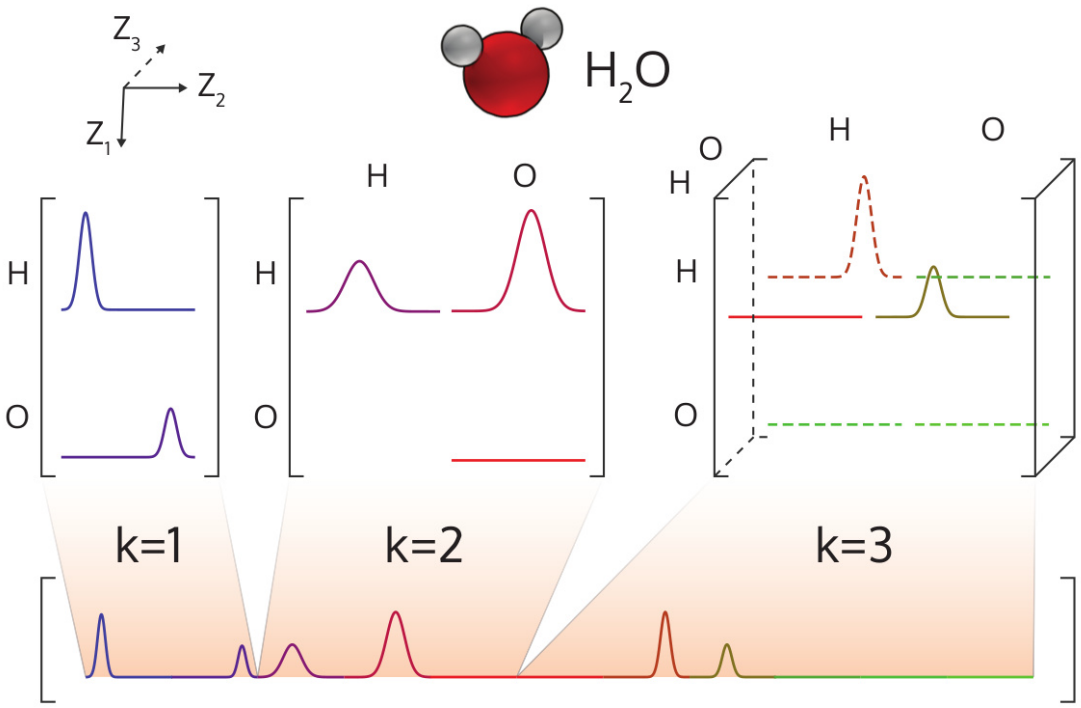


Figure 2.16: Depiction of the MBTR method. The different modalities ($k = 1, 2, 3$) are represented. $k = 1$ is the encoded atomic species, $k = 2$ the pairwise distances and $k = 3$ the bond-angles. This image was taken from the DScribe [133] documentation[18].

atoms in the materials. Given a sorted list of these eigenvalues, they serve as an invariant descriptor that can be used as the input to machine learning algorithms. The use of the eigenvalues, rather than the raw CM, provides several benefits, including invariance to permutations of the atom indices and robustness to small perturbations in the atomic positions. Additionally, it reduces the size of the representation, as an N atom system has N CMEs, rather than the N^2 entries in the full CM.

The **MBTR** is a significant development in materials representation that captures the many-body interactions among atoms and offers a more comprehensive representation of the material's structure [104]. The MBTR builds on the idea of atomic pair-wise distances introduced by the Coulomb matrix but extends it to include three-body terms, thus capturing the angle between atom triples and even many-body terms. The first "order" of the MBTR is a one-body term, which corresponds to the atom densities and is invariant

[18]This figure can be found at the following URL: https://singroup.github.io/dscribe/0.3.x/

under permutation and rotation. The second order corresponds to the two-body interactions, which give rise to pairwise distances between atoms. However, unlike the CM, which computes the potential energy between pairs of atoms, the second-order MBTR term integrates over all distances for a given pair of atomic species, effectively forming a histogram of distances [104].

The third order of MBTR represents the three-body interactions, which capture the angles between triples of atoms. Like the second-order term, the third-order term also forms a histogram but over all angles for a given triple of atomic species. The second and third-order terms are both invariant under rotation and permutation [104].

Higher-order terms can be included in MBTR, although in practice, they are rarely used due to their high computational cost and the decreasing significance of higher-order atomic interactions [104]. Equation 2.81 shows a general form of the MBTR terms.

$$K^{(n)}IJK(p) = \sum_{i\in I, j\in J, k\in K, \ldots} w_{ij}(r_{ij})w_{ik}(r_{ik})\ldots f(p; x_{ijk\ldots}), \tag{2.81}$$

where n denotes the order of the interactions, IJK are atomic species indices, $i, j, k, \ldots$ are atom indices, w_{ij} is a smooth cutoff function ensuring the continuity of the descriptor, r_{ij} is the distance between atoms i and j, $x_{ijk\ldots}$ is the many-body coordinate (e.g., distance or angle), and $f(p; x_{ijk\ldots})$ is a function generating histograms.

Despite the increased computational cost, the MBTR provides a robust, systematic way of including many-body interactions, making it a powerful descriptor for complex materials systems, such as amorphous materials, and has been employed in a variety of applications ranging from molecules to solid-state materials [38,104,181]. Its ability to capture complex structural information has shown great promise in enhancing the predictive accuracy of machine learning models for materials properties [104]. An illustrative representation of the MBTR descriptor is shown in Figure 2.16.

We now introduce the need for a new representation of long-range interactions.

2.7.2 The Splashdown Representation

The need for a new representation comes from the lack of attention to long-range interactions. Very little research has been put into training machine learning models for long-range physics. Long-range interactions in chemical systems contribute to numerous

`tutorials/mbtr.html`.

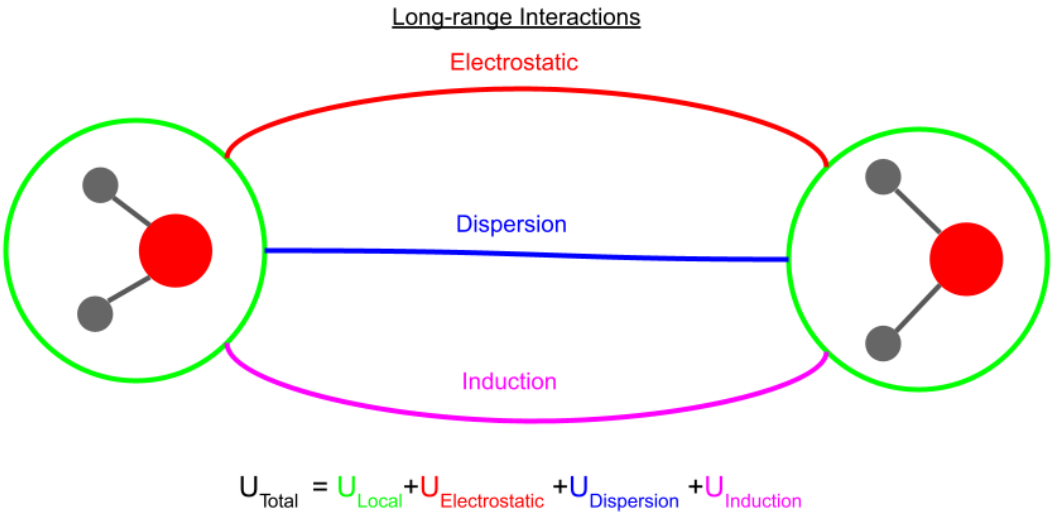
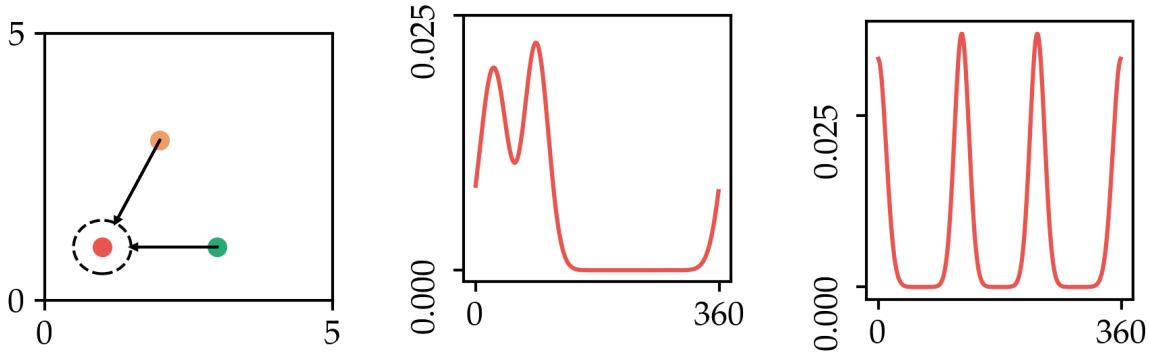


Figure 2.17: Summary of interactions studied for in Machine-Learned Interaction Potentials (MLIP)s for the total potential energy U_{Total}. U_{Local} consists of what general short-range MLIPs study, and $U_{\text{Electrostatic}}$, $U_{\text{Dispersion}}$ and $U_{\text{Induction}}$ compose what is referred to as long-range interaction energy contributions. This figure is adapted from Anstine et al. [3].

physical phenomena: e.g., thermodynamic phase behaviour [250], variations in conformer geometry [169], interfacial properties [160], permeation rates of molecules through porous materials [168], protein structure and dynamics [255], and the self-assembly or the directed assembly of macromolecular complexes [144]. Some are of the opinion that using a model with a cutoff function but simply extending it may work [79] however this comes at the cost of increasing the number of descriptor calculations. This may require very large computing power and the learning task might become unassailable due to the growing chemical space. Relying on local features has also been shown to complicate simulations of long-range features [16]. These features are summarized in Figure 2.17.

When considering non-local interactions, MLIPs can be split into three groups: some (a) ignore their contributions, (b) supplement these with conventional long-range functional forms, such as Coulomb's law, where parameters are dependent on the local environment, and (c) use MLIPs with long-range interactions that are influenced by the overall characteristics of the system [3, 11]. The selection of one of these three strategies is guided by physical principles. For instance, systems with a limited variety of elements and short screening distances can often be modelled using the first strategy with minimal loss of accuracy [6, 17, 142, 143]. In these situations, the use of the second or third strategy results



(a) The system of Hydrogen atoms. Colours only denote different atoms.

(b) The splashdown representation of the red atom in Figure 2.18(a). The x axis is split into 360 bins.

(c) The splashdown COM representation of the system in Figure 2.18(a).

Figure 2.18: The Splashdown Representation.

in additional computational costs and training complexities without a significant increase in accuracy for the intended application. Conversely, using the first strategy for systems exhibiting phenomena such as long-range polarization and electrostatics is likely to yield inaccurate predictions. The idea of selecting an appropriate model design for specific systems is a well-established concept in traditional molecular simulations and is also applicable to simulations conducted with MLIPs [3].

The *Splashdown* representation is now introduced in an effort to solve the problem of modelling long-range interactions. Starting with a system of atoms, the central goal is to be able to model the long-range interactions which tend to diminish rapidly in magnitude. We consider only two-dimensional systems for simplicity, however, an extension to three is straightforward. This representation is built by first selecting a central atom i. Then, one can imagine a counterclockwise sweep starting from the $+y$ axis (with the central atom located at the origin). A normal distribution $\mathcal{N}(\theta_j, \sigma_{ij}^2)$ is then computed for all atoms $j \neq i$ with $\sigma_{ij} = \mu r_{ij}$. Here, μ is a scaling factor, and r_{ij} is the Euclidean distance between atoms i and j. We call this step the "projection" of atom j onto i. These projections are then summed to make a single distribution representing the system. This can be visualized in Figure 2.18.

This representation can be described mathematically by the following relation

$$\mathcal{S}_i = \sum_{j \in s, j \neq i} \mathcal{B}(\mathcal{N}(\theta_j, \sigma_{ij}^2)) \tag{2.82}$$

where $\mathcal{S}_i$ is the Splashdown representation of atom i, s is the set of atoms in the system, $\mathcal{B}$ is some binning function and $\mathcal{N}$ is a normal distribution with mean θ_j and variance σ_{ij}^2.

The Splashdown representation, as discussed, offers a unique approach to capturing both short-range and long-range interactions in atomic systems. However, its initial formulation is centred on individual atoms, which may limit its ability to fully capture the global characteristics of the system. To address this limitation, we propose a modification to the Splashdown representation: the Splashdown Center of Mass ($\mathcal{S}_{\text{cm}}$) representation. In this approach, instead of generating the Splashdown representation at each individual atom, we generate it at the center of mass of the system. The center of mass, being a point that reflects the overall distribution of mass in the system, provides a natural choice for a global representation. This modification allows the Splashdown representation to capture the global properties of the system, making it even more powerful for modelling long-range interactions and global phenomena.

The mathematical formulation of the $\mathcal{S}_{\text{cm}}$ representation would be similar to the original Splashdown representation, but with the central point being the center of mass. The center of mass r_{cm} is calculated as the weighted average of the positions of the atoms, with the weights being the masses of the atoms:

$$r_{\text{cm}} = \frac{\sum_i m_i r_i}{\sum_i m_i} \tag{2.83}$$

where m_i is the mass of atom i and r_i its position. The "projection" of atom i onto the center of mass can then be represented as a Gaussian distribution centred at r_{cm}. σ_i remains the standard deviation of the distribution, which is proportional to the distance between the center of mass and atom i. The total "projection" onto the center of mass is then the sum of the projections from all atoms. Equation (2.82) therefore becomes

$$\mathcal{S}_{\text{cm}} = \sum_i \mathcal{B}(\mathcal{N}(\theta_i, \sigma_{\text{cm}-j}^2)) \tag{2.84}$$

This equation represents the Splashdown Center of Mass representation of the atomic system. This can be visualized in Figure 2.18(c).

The Splashdown representation, as shown in Figure 2.18(b) and Figure 2.18(c), present several unique advantages over other atomic representations, making compelling choices for modelling atomic systems. One of the primary strengths of the Splashdown COM representation lies in its inherent ability to capture long-range interactions. By considering all atoms in the system and encoding distances, it is able to account for the effects of

distant atoms, thereby providing a more comprehensive picture of the system. This is particularly beneficial in systems where long-range electrostatic or van der Waals forces play a significant role.

The Splashdown representation also includes a few other advantageous properties. It provides a continuous and differentiable representation of each atom through the use of a Gaussian distribution, a feature that is particularly beneficial for machine learning algorithms which often require smooth input spaces. Furthermore, it is invariant to translations and rotations of the system, a crucial property for any atomic representation, as it only depends on the relative positions of the atoms. The scalability of the Splashdown representation (which is linear with the number of atoms) ensures its computational efficiency for large systems. Lastly, its interpretability provides a physically intuitive picture of the atomic environment, with the "height" of the splashdown at a given point indicating the influence of the atoms at that point. Furthermore, Splashdown is easily be extended to many-atom systems by simply including more channels for each atom. This can be imagined as a different colour, alloying the encoding of any number of atoms, however, this would increase the space complexity of this representation.

By providing a more accurate description of the atomic environment through the inclusion of long-range interactions, the Splashdown representation can lead to more accurate predictions of material properties. This is especially relevant for systems with long-range order, such as crystals, or systems where long-range forces play a significant role, such as in the case of polar molecules or charged species. It is important to note, however, that these advantages are based on the description provided, and the actual benefits may vary depending on the specific implementation and application of the Splashdown representation. In Section 3.4 we test Splashdown in various environments and compare it to other well-known representations.

Chapter 3

Results

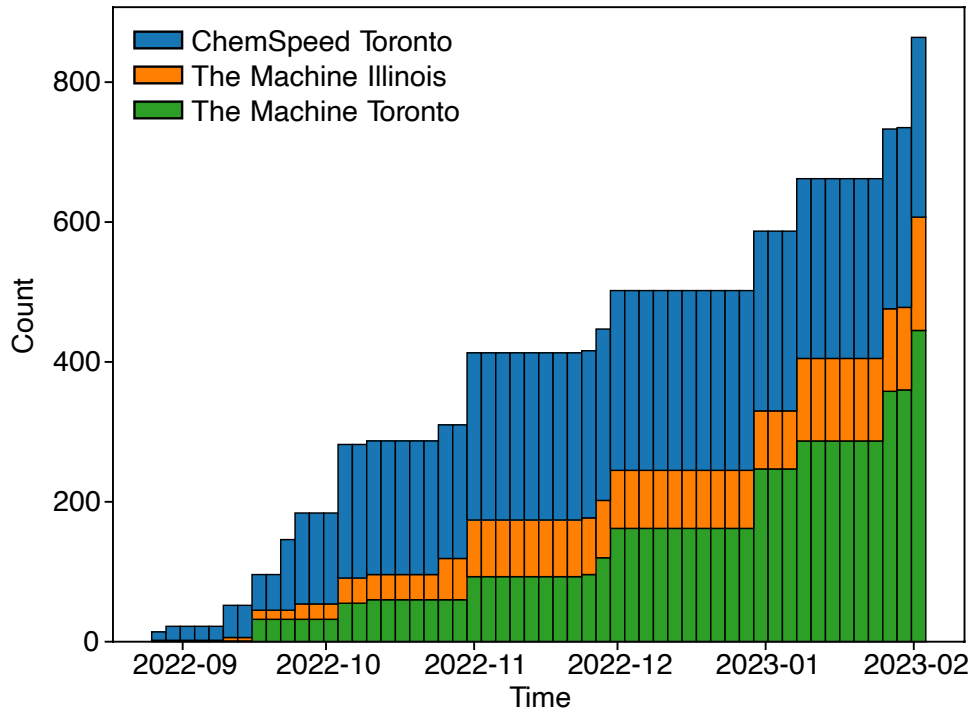
In the following sections, we denote the key results found in the projects worked on. The work done will be denoted clearly in each section however for the global picture, most results from the papers are included.

3.1 Organizer

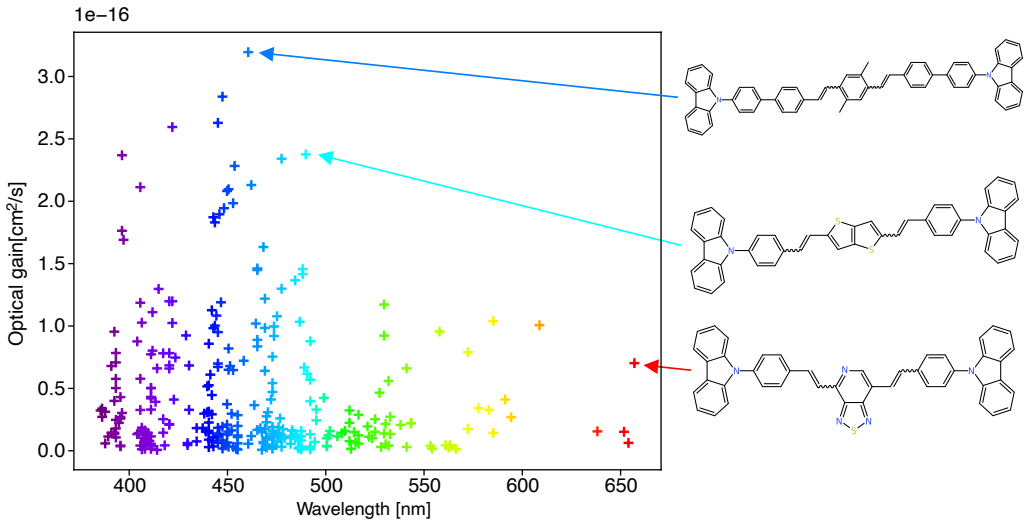
The main objectives of the `ORGANIZER` project were to efficiently and asynchronously integrate an automated workforce across different laboratories using Molar as a backend database for data storage and sharing. The capabilities of `ORGANIZER` and Molar to work in a closed-loop environment were demonstrated by the OSLD gain materials discovery. We enabled the collaboration of multiple labs with differing equipment all with the same goal. As far as we know, `ORGANIZER` is the first delocalized chemistry laboratory.

The results can be seen in Figure 3.1. In Figure 3.1(a) we show the increasing number of unique experiments over time for various machines used in this project. This underlines the capability of `ORGANIZER` to coordinate collaboration between remote groups and minimize replicated efforts. Plateaus in each colour group denote maintenance or the development of new features.

In Figure 3.1(b) we see the broad spectrum of laser gain materials obtained within this campaign. We show the measured emission gain cross-section (denoting lasing performance) and the emission wavelengths of the top 300 candidate molecules obtained. We were able to discover a potential gain material with an emission gain cross-section of



(a) Cumulative number of syntheses on three different machines over time.



(b) Optical gain vs the wavelength for the 300 brightest organic laser molecules. The colours correspond to the wavelength the laser molecule would emit

Figure 3.1: Results for the `ORGANIZER` project.

3.2×10^{-16}cm^2 in solution. This outperforms the current state-of-the-art material by a factor of 2. The new material has now been isolated on a larger scale and its characterization in actual lasing devices is ongoing. These discoveries illustrate the potential of delocalized self-driving labs for materials research - enabling the discovery of new molecular materials within a matter of months.

3.2 aMC

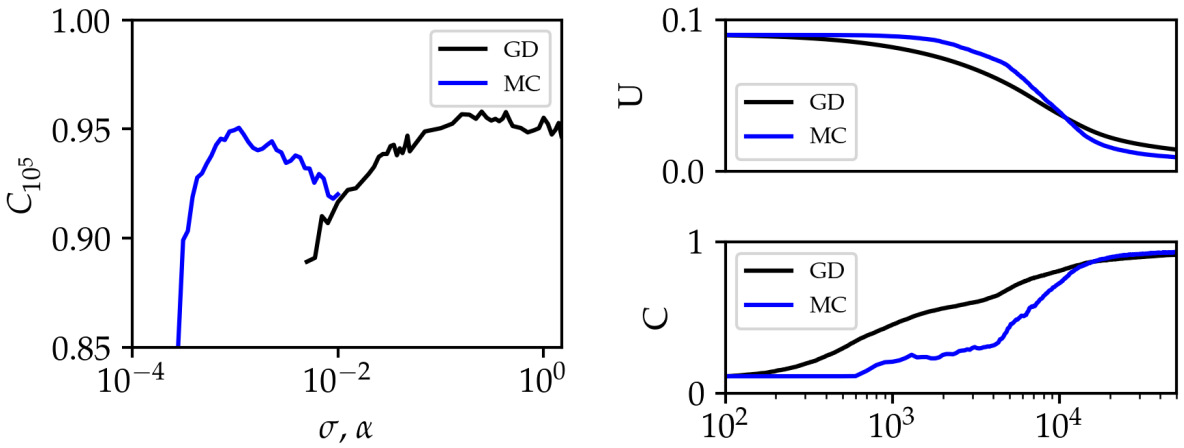
For original results/discussion of the following section see [238]. The following only summarizes a few key points of the Whitelam et al. paper. My contribution to these results is mainly the first section Section 3.2.1 below, where I developed and trained the required models to achieve these results and in Section 3.2.4 where I helped run and develop the RNNs. This work was led by Stephen Whitelam at the Lawrence Berkeley National Lab and my supervisor Isaac Tamblyn at the University of Ottawa. Their guidance throughout was greatly appreciated.

3.2.1 Metropolis MC and Gradient Descent

We started by training a well-known database with a simple neural network and a benchmark optimizer to show the equivalence connection between it and the Metropolis Monte Carlo optimizer. Given the success of gradient-based methods, we hypothesized that the MC algorithm would also be able to train neural networks successfully, and on par with GD. The results obtained can be seen in Figure 3.2.

To run these tests, we used the standard MNIST [137] image dataset using a fully connected, two-layer neural net. The net had 784 inputs (which corresponds to the flattened input image size), 16 neurons in each hidden layer and an output layer of 10 neurons, the number of classes in the dataset. The hidden neurons have the hyperbolic tangent activation function, and the output neurons include a softmax function so the outputs are interpretability (as probabilities of each class). In total, this amounts to 13002 learnable parameters. We ran batch learning (meaning the dataset was not split into batches, but trained as a single large batch), with the loss function being the mean-squared error on the MNIST training set of size 6×10^4 (we considered the ground truth for each training example to be a 1-hot encoding[1] of the class label). The model weights were always initialized to a standard normal distribution.

[1]One-hot encoding is a data transformation technique where categorical variables are represented as



(a) MNIST test-set accuracy C_{10^5} after 10^5 epochs of batch learning as a function of MC step size σ (blue) or GD learning rate α (black).

(b) Training loss U and test-set accuracy C versus epoch n for MC step size $\sigma = 2\times10^{-1}$ (blue) and GD learning rate $\alpha = 4.5 \times 10^{-2}$ (black).

Figure 3.2: Comparison of zero-temperature Metropolis Monte Carlo (MC) and gradient descent (GD) used to train a neural network to minimize the mean-squared error U on the MNIST training set.

In Figure 3.2(a), we present the results of our experiments on the MNIST test set. We measure the classification accuracy, denoted as C_{10^5}, after training for 5×10^5 epochs. We compare two algorithms: MC (shown in blue) and GD (shown in black). We vary the step size σ and the learning rate α for MC and GD respectively.

Both algorithms exhibit similar behaviour. They have a specific range of their single parameter where they perform effectively and demonstrate a peak performance at a particular value of that parameter. The maximum accuracy achieved by GD is slightly higher than that of MC, approximately 96% compared to 95%. Moreover, GD achieves near-maximal results over a broader range of its single parameter compared to MC.

In Figure 3.2(b), we depict the loss function U and classification accuracy C as functions of the training epoch for two examples from Figure 3.2(a). Initially, GD trains faster, but towards the end of the learning process, the results converge. It's important to note that the learning dynamics of these algorithms differ. When the step size is very small, the

binary vectors, with each vector element indicating the presence or absence of a particular category. For example, if we had two classes "dog" and "cat", we can consider the vector $[1, 0]$ to represent "dog" and $[0, 1]$ to represent "cat".

zero-temperature Metropolis algorithm approximates normalized or clipped GD, not the standard GD. However, this equivalence can only be observed by appropriately scaling the horizontal axis.

Despite these differences, MC can effectively "sense" the gradient and take downhill steps in loss as long as the step size remains relatively small. For this specific problem, the range of suitable step sizes for MC is smaller compared to the effective step size of GD. Consequently, GD trains faster, but MC exhibits a similar capacity for learning. In terms of computational cost per epoch, both algorithms are of a similar order. However, MC is more cost-effective per epoch for batch learning since each MC step only requires a forward pass through the data, while each GD step involves a forward and backward pass.

We note that no preprocessing was done to these data and a very basic neural net was used. Standardizing the dataset, for example, or using a more complex neural network would lead to improved results. We show this in a later figure in [238]. However, this all came to show that MC can get comparable accuracy to GD on problems where gradients are available. Some caveats include the fact that GD trains faster and that for select MC step sizes (σ) it may appear that MC does not train at all. These need to carefully be selected.

3.2.2 Metropolis acceptance rate as a function of net size

We considered MC for coping with a large number of parameters. Vanilla MC was impaired by network heterogeneity and we therefore tested the aMC analog. We investigate the efficiency of the Metropolis algorithm in relation to neural network size and architecture. We focus on a supervised learning problem where a neural net is trained to express the sine function on a specific interval. The loss is measured using mean-squared error at evenly-spaced points on the interval.

We explore the impact of varying the width and depth of the neural net. Varying the width, we set the depth to 2 and the width ranges from 10 to 10^3, corresponding to largely different parameter counts. Varying the depth, we set the width to 25 and the depth ranges from 2 to 10, corresponding to different parameter counts as well. These can be seen in Figure 3.3 (a) and (b).

We showed the loss as a function of the Metropolis acceptance rate for different neural net widths. The acceptance rates indicated the fraction of directions that lead to a decrease in loss. It decreased slowly with increasing net size, showing a slight decline of about 1 order of magnitude when the net size increases by 4 orders of magnitude. Similarly, the

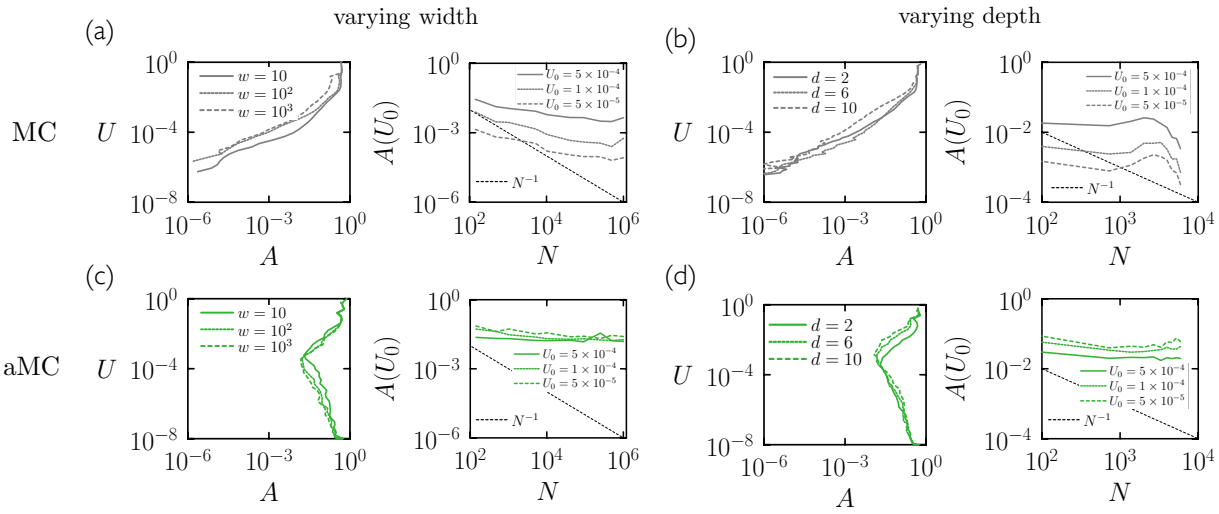


Figure 3.3: Acceptance rate of Metropolis MC with net size. For a particular network in a supervised learning task, the acceptance rate decreases with loss but not the model size. (a) Left: Loss (U) vs. acceptance rate A for MC for a deep NN of varying widths. Right: Acceptance rate A(U_0) at fixed loss U_0 for models of different widths, as a function of the number of model parameters. (b) Analog of (a) but with increasing depth instead of width. (c) Analog of (a) but with aMC. (d) Analog of (b) but with aMC. aMC hyperparameters $(\sigma_0, \epsilon, n_s, \text{signal norm}) = (10^{-2}, 10^{-2}, 10^2, \text{on})$. Image taken from [238]. See the original paper for more thorough discussion.

acceptance rate declines sharply with loss and slightly with the number of parameters, but not as rapidly as a N^{-1} scaling.

These observations suggested that the Metropolis algorithm can handle large numbers of parameters effectively, and there are no fundamental limitations. However, network heterogeneity can affect the algorithm's training capability. To address this, aMC was tested as well (see Figure 3.3 (c) and (d)). The acceptance rates of aMC remained large and essentially constant with loss or model size across the parameter range considered, neutralizing the trends observed in the original Metropolis algorithm.

3.2.3 Adaptivity speeds learning, particularly of high-frequency features

In this section, we showed that aMC parameter ϵ (Equation (2.73)) can speed the rate at which a neural network can learn high-frequency features. This is analogous to gradient-based algorithms. We used aMC to train a model to predict a function with low and high frequencies; the two-dimentional Rosenbrock function

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{3.1}$$

which is often used to test optimization methods [193, 213]. This is because of its complex topology, including a long valley with steep slopes on either sides that contains the minimum (at $(1, 1)$). It turns out that Adam learned the high-frequencies more rapidly than GD, but that aMC is also able to learn these features comparably to Adam. aMC is shown to train faster than pure GD. Ee conducted three simulations with varying ϵ values. Notably, larger ϵ values significantly accelerated the convergence to the global minimum, highlighting the effectiveness of adaptivity. This adaptivity effect is comparable to the impact of momentum in traditional gradient descent.

Expanding our scope to neural network training, we found that the ϵ parameter in aMC plays a crucial role. It influences the rate at which neural networks can learn high-frequency features, a factor dependent on the specific application's requirements. For instance, in scenarios involving training data with high-frequency noise, it might be beneficial to mitigate the network's capacity to learn this noise in favor of improved generalization [195].

To substantiate our findings, we conducted experiments using a neural network tasked with learning a function comprising both low-frequency (logarithm) and high-frequency (sine) components. The neural network architecture included one input neuron receiving the input θ, one output neuron delivering the result $f(\theta)$, and a hidden layer with 100 neurons activated by the hyperbolic tangent function. The network parameters were initialized with random values.

Our experimental results showcased the training loss and pseudo-loss for various optimization methods, including Gradient Descent (GD), Adam, and aMC, with different ϵ values. The pseudo-loss measures the deviation between the network output and the low-frequency component of the target function.

Notably, positive ϵ values in aMC expedited the learning of high-frequency features, while negative ϵ values allowed treating high-frequency components as noise, with the pseudo-loss serving as a gauge of generalization error. These results can be seen in Figure 3.5

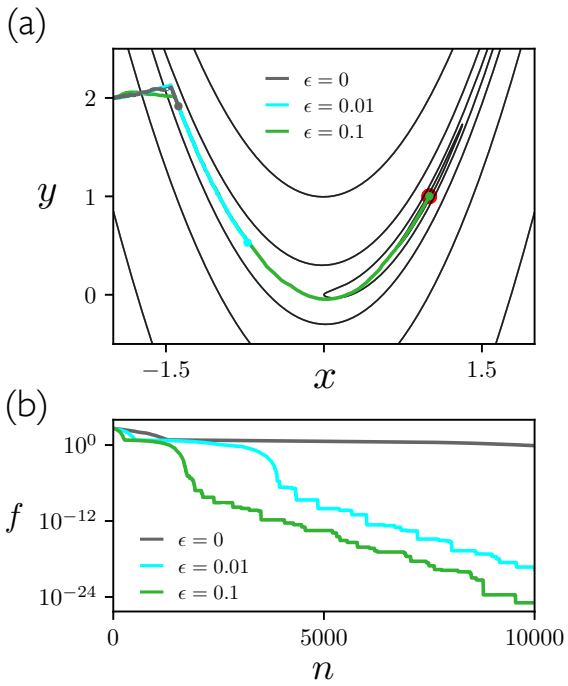


Figure 3.4: Adaptivity speeds convergence. We tested the convergence of aMC with three values of ϵ to simulate a particle moving on the Rosenbrock function Equation (3.1). (a) shows a 2000-step simulation for each of the ϵ values. The red dot is at the global minimum $(1, 1)$. (b) shows the value of the function along each trajectory when ran for 10^4 steps. aMC hyperparameters were $(\sigma_0, n_s) = (10^{-3}, 20)$.

3.2.4 Monte Carlo algorithms can train a neural network even when gradients are unreliable

This section demonstrates the ability of MC methods to cope with vanishing and exploding gradients enabling them to train RNNs, which are gradient-based methods notoriously struggle with. RNNs are designed for sequential data processing but face challenges when sequences get long and contain long-term dependencies. Gradient-based methods struggle to train simple RNNs in such cases, requiring more complex architectures like long short-term memory or gated recurrent units. However, MC methods provide a solution to this problem by successfully training RNNs without these explosive gradients. Here, the results of training a simple RNN on the MNIST dataset using aMC and gradient-based optimizers were shown, and aMC achieves lower loss values and higher accuracy compared to gradient-

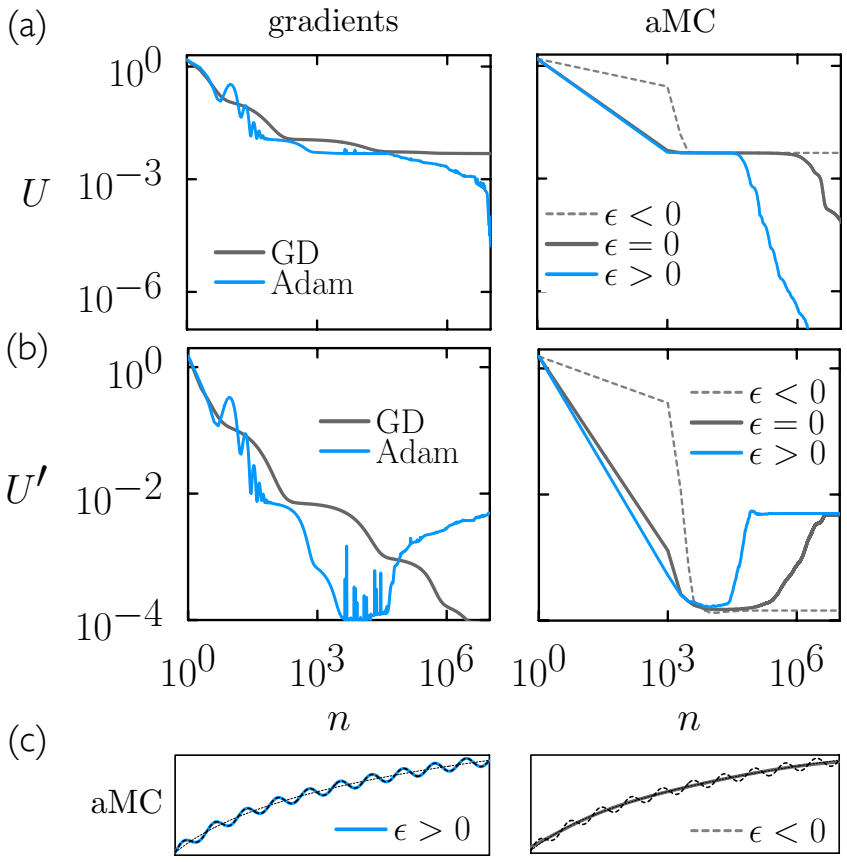


Figure 3.5: Adaptivity speeds learning of high-frequency features. We trained a neural network to learn the function $f_0(\theta) = \ln(1 + 5\theta) + \frac{1}{10}\sin(20\pi\theta)$ using three optimization methods: GD, Adam, and aMC with varying ϵ values. Our analysis includes: (a) Loss (U) plotted against training time (n). (b) Pseudo-loss (U'), measuring the deviation between the network output and the low-frequency logarithmic component. (c) The output of neural networks trained with aMC at specific training intervals, represented by colored lines. The thicker and thinner black lines denote the function and its low-frequency component, respectively. For GD and Adam, we used a learning rate (α) of 5×10^{-3}, while aMC was configured with hyperparameters (α_0, n_s) set to $(0.1, 50)$. This figure was taken from [238].

based methods.

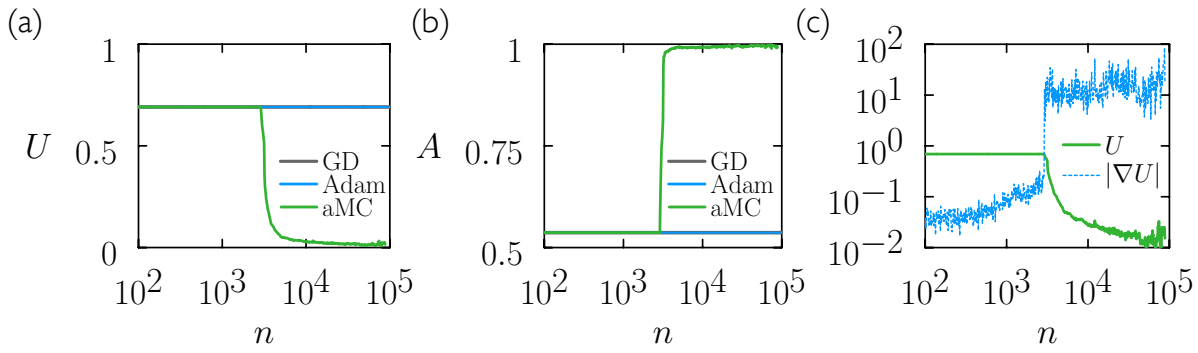


Figure 3.6: aMC is capable of training even when gradients get large or small - a common failing point of gradient-based methods. We show (a) training set loss U and (b) accuracy A for training a simple RNN to predict MNIST 1's and 0's. We show GD in grey, Adam in blue and aMC in green. The gradient-based methods fail entirely to train this whereas aMC succeeds to an accuracy of 99.8%. In (c) we see the size of the gradient $|\nabla U|$ and the loss from (a). This is to show that aMC can train even when the gradient is large or very small. This image is taken from the aMC paper [238].

The findings in Figure 3.6 highlight the potential of MC methods to train RNNs effectively, allowing for wider utilization of these architectures. MC overcomes many obstacles faced by gradient-based methods and can handle both small and large gradients as well as models. Additionally, the memory cost of MC does not depend on the sequence length, unlike gradient-based methods, which scale linearly with sequence length.

3.3 Discovery of HEA Electrocatalysts using ML-informed similarity Analysis

The following results are from an article currently being reviewed. It was written in collaboration between myself, Ian, Christoff Reimer at the University of Ottawa and Hitarth Choubisa, Daojin Zhou, Xiao-Yan Li, Pengfei Ou at the University of Toronto. It was supervised by Dr Isaac Tamblyn and Dr. Edward H. Sargent at the Universities of Ottawa and Toronto respectively. The contributions to this paper consisted of analyzing the similarity matrices and designing, running and optimizing the machine learning sections.

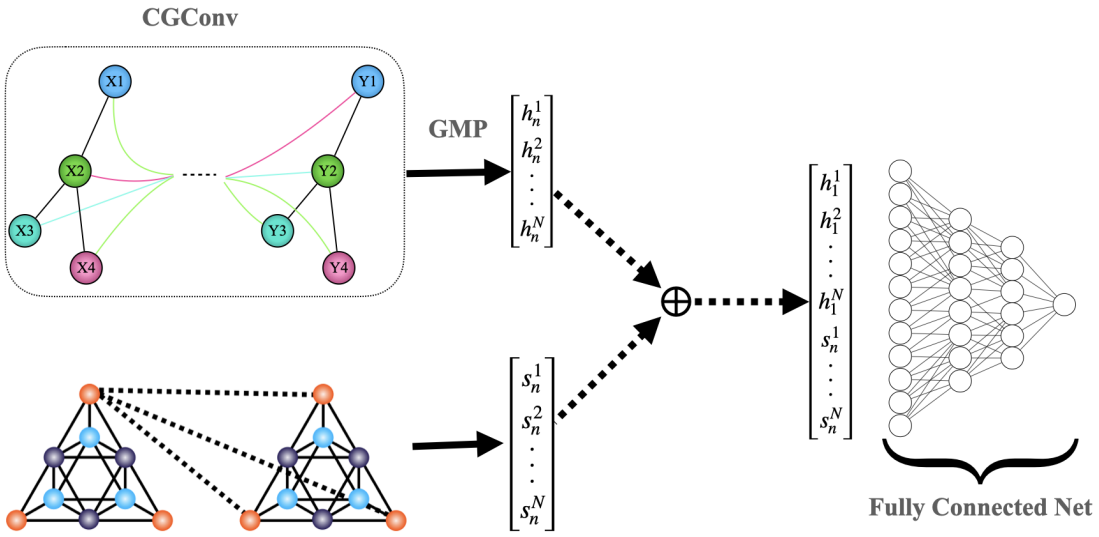
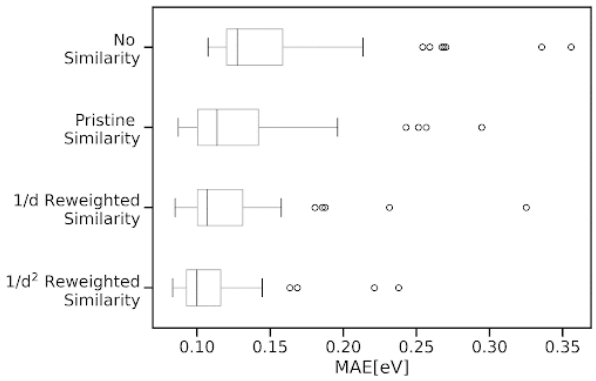


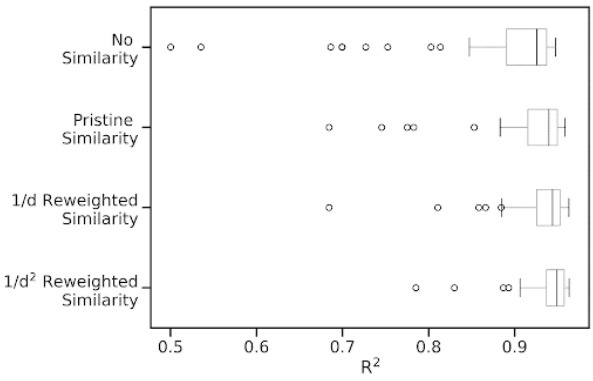
Figure 3.7: CGCNN augmented model. Here we show the CGCNN model used in training DFT-calculated HEA adsorption energies with OH* and H* adsorption sites, the added similarity matrices and how they were combined. GMP denotes the Global Max Pooling operation to extract features from the resulting graphs after multiple layers of convolution and $\bigoplus$ denotes the concatenation operation. h_n^i are the n learned features for graph i and s_n^i are the n similarity matrix features for atom i. After concatenation, we predict adsorption energies using a fully connected net with ReLU activation.

In summary, we discuss the use of a machine-learning model to speed up the discovery of new electrocatalysts. A model was trained using DFT data, with the addition of a measure of similarity between potential catalyst sites. This similarity is defined by the largest common subgraph between two adsorption graphs [99]. This augmented graph model reduced the amount of training data needed by half, from 1600 to 800. As a result, our model was able to identify new HEA catalysts that have an expected 40mV lower overpotential than those discovered using a DFT-only search. When the model was tested with a random set of HEA structures, it discovered a catalyst that had an overpotential of 0.24V, compared to 0.28V found with the same amount of DFT data alone. The reduction in training dataset sizes and time spent with training and data collection can significantly speed up the discovery process of HEAs and materials in general.

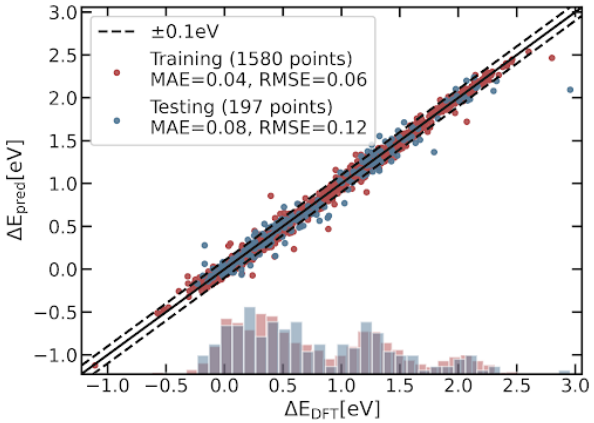
We started with surface characterization by randomly creating HEA structures and establishing OH* and O* adsorption sites, along with their related graphs, to use as our



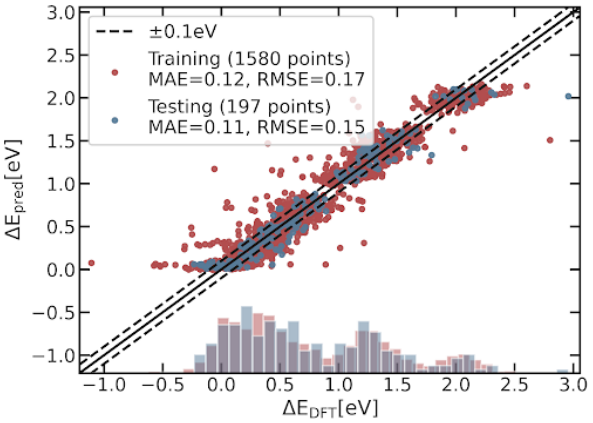
(a) Box and whisker plot of MAE over various hyperparameters.



(b) Box and whisker plot of R^2 over various hyperparameters.

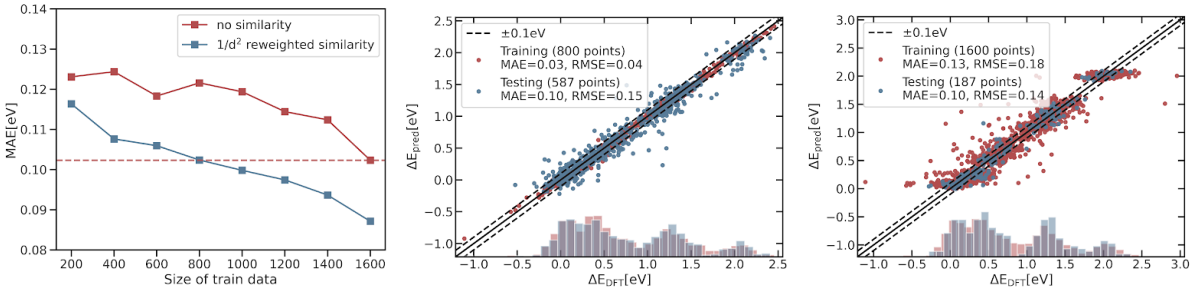


(c) The plot of predicted vs. true adsorption energies in eV for the optimized model with $1/d$ reweighted similarity feature.



(d) The plot of predicted vs. true adsorption energies in eV for the optimized model with no similarity feature.

Figure 3.8: Performance comparison of ML models with fixed-size training data. Note the improvement in results when including the similarity feature vs without ("No Similarity" vs the rest). "Pristine Similarity" refers to a simple heuristic-based similarity approach only considering the number of common atoms between two structures. "$1/d$ Reweighted Similarity" and "$1/d^2$ Reweighted Similarity" refer to similarity measures considering both the number of common atoms and their distance to surface atoms and the adsorbate, reweighted as indicated. Overlayed onto plots Figure 3.8(c) and Figure 3.8 are a $y = x$ line (solid black) and ±0.1 eV (dotted black line) as a visual aid and histograms corresponding to the distribution of the data.



(a) MAE vs. training size with and without the inclusion of a similarity feature.

(b) Predicted vs. true adsorption energies in eV for the optimized model trained using 800 data with the $1/d^2$ reweighted similarity feature.

(c) Predicted vs. true adsorption energies in eV for the optimized GNN model trained using 1600 graphs without the inclusion of any similarity feature.

Figure 3.9: Performance comparison of machine learning models with varying training set sizes. Overlayed on Figure 3.9(b) and Figure 3.9(c) are a $y = x$ line (solid black) and ± 0.1 eV (black dotted lines) as a visual aid. Overlayed are also histograms corresponding to the distribution of the data.

training dataset. Then, we ran similarity assessment and data diversification for DFT calculations by evaluating the likeness between each pair of graphs within the training dataset. This was followed by data diversification based on similarity to ensure a varied training environment, thereby enhancing the precision of our model which was achieved by choosing sites that minimized the total of all similarity values. This showed very low impact when compared to training randomly selected sites so we decided to simply incorporate the similarity values into training.

For the ML model, we used the CGConv model incorporating both graphs and the similarity matrix as integral features. A diagram of the model employed can be found in Figure 3.7. When we used similarity as the feature for model training Figure 3.8(a), the MAE improved from 0.11 eV (only graphs) to 0.08 eV (incorporating similarity into training). This shows the gain in performance when incorporating these similarity matrices. To learn how much data is needed to achieve an MAE of 0.10 eV (as a standard), we trained our model using different dataset sizes and concluded that only 800 adsorption cite graphs were needed. This reduces the number of DFT calculations needed vastly for researchers looking to predict adsorption energies. These results can be seen in Figure 3.9.

After successfully building a model which is able to predict DFT adsorption energies

at a low error rate, we predicted new candidate catalysts using the similarity measures. We generated a list of random structures, computed the similarity measure and then predicted the adsorption energy for each slab, selecting only ones that would be favourable. A candidate found is $Fe_{0.125}Co_{0.125}Ni_{0.229}Ir_{0.229}Ru_{0.292}$ adsorbing at the Ni site, with an overpotential of 0.27V. The DFT calculated overpotential for this structure is 0.24V showing close agreement with our prediction.

3.4 Splashdown

In order to test the Splashdown representation, we decided to compare it to a well-studied representation on an easily conceptualized potential. To this end, we benchmark it with the MBTR, using the electrostatic potential estimated by Equation (2.80).

The electrostatic force is a fundamental interaction in atomic systems, especially in materials with charged species or polar molecules. It is a long-range force, meaning it can have significant effects even at large distances, and it plays a crucial role in determining the structure, stability, and properties of many materials. Therefore, accurately predicting the electrostatic force is a key test for any atomic representation.

The MBTR is a well-established method for representing atomic systems in materials science. It is designed to capture the many-body nature of physical interactions in materials, and it has been shown to perform well in a variety of applications. However, like many traditional atomic representations, MBTR primarily focuses on short-range interactions and may not fully capture the effects of long-range electrostatic forces.

To test the Splashdown representation, we can use it to predict the electrostatic force in a variety of atomic systems and compare the results with those obtained using the MBTR representation. This can be done by training machine learning models using both representations and evaluating their performance on a test set of atomic configurations. The models can be trained to predict the electrostatic force based on the atomic representation, and their predictions can be compared with the true forces calculated using quantum mechanical methods.

To properly benchmark these representations, we use three models to evaluate performance. A basic multilayer perceptron, XGBoost [40] and a kernel ridge regressor. These were used due to a combination of their expressive power and simplicity.

The comparison can be made in terms of both accuracy and computational efficiency. The accuracy can be measured using metrics such as the mean absolute error (MAE), the

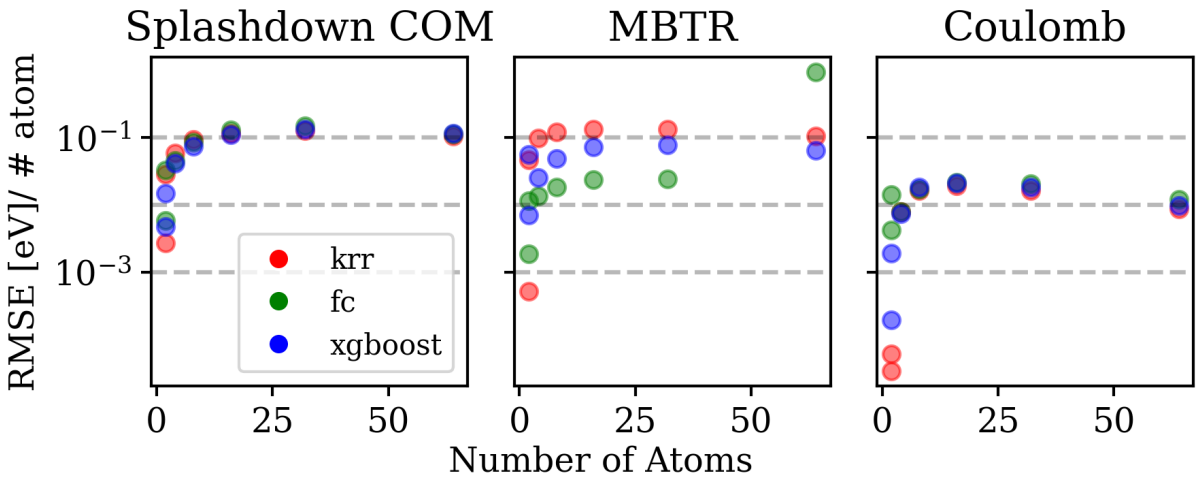


Figure 3.10: Splashdown COM representation testing results compared with CM and MBTR. A dataset of 10k random systems of Hydrogen atoms was generated on 10×10Å grids. The three models tested are the KRR ("krr", in red), a fully connected network ("fc", in green) and XGBoost (in blue). **Left** MBTR representation results. **Center** The Splashdown COM representation results and **Right** the coulomb representation. Select training plots (loss curves, true vs predicted for training and test sets) can be found in Figure A.1. Training parameters can be found in Appendix A.1.

mean squared error (MSE) or the root mean square error (RMSE)[2] between the predicted and true forces. The computational efficiency can be evaluated by comparing the time it takes to compute the representations and to train and use the models.

3.4.1 Results

The question we would like to answer is whether or not Splashdown is a good representation for capturing long-range interactions. The MLP has varying input sizes (for the different representations) however it has hidden layers of size 720, 360, 90 and finally a single output representing the computed energy. After the first layer, we apply a 20% dropout and batch normalization at each following layer. A small grid search over the alpha and gamma parameters was run for KRR. The ranges were the logspace from 1×10^{-3} to 1×10^{3} with 5 fold cross-validation for model selection.

[2]RMSE is sometimes preferred as it preserves the units of the predicted value whereas MSE has squared units.

Figure 3.10 compares the Splashdown COM representation with MBTR and CME. We see the splashdown features are good for predicting coulomb energies with error rates on par with other state-of-the-art descriptors in a long-range setting.

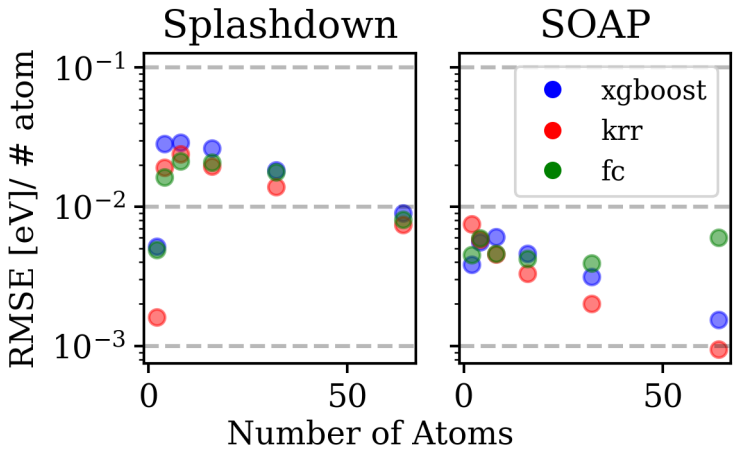


Figure 3.11: Splashdown local benchmark. We compare the splashdown representation to SOAP with three models as in Figure 3.10. We generate 10×10Å grids and generate the SOAP representation with a 5Å cutoff. We see that these perform similarly however SOAP seems to perform better than Splashdown for certain system sizes.

To study the local version of Splashdown (Equation (2.82)), we can compare it to the local representation SOAP. The latter seems to outperform Splashdown (Figure 3.11) using most models, showing a shortcoming of our representation. We hypothesize that for larger grid sizes, however, Splashdown would overcome this gap.

To further learn about Splashdown's capability in predicting material properties, we run a test comparing the capabilities of the different representations in extrapolating. These results are tabulated in Table 3.1. Although the Splashdown representation does not outperform the Coulomb representation for all models, it is able to achieve comparable results which shows its capability.

With these results, we have learned the power of Splashdown as a global representation. Although it does not outperform other local representations, we have shown its potential to predict long-range interactions as it can accurately predict the coulomb interaction. Firstly, more testing should be done to train and predict on benchmark datasets, with DFT data. To test Splashdown in more depth, increasing the number of bins could aid performance as the representation would include finer details of the system. We do not

Representation	Model	RMSE Mean	Std
Coulomb Eigenval	FC	0.05	0.01
	KRR	0.03	0.01
	XGBoost	0.01	0.00
MBTR	FC	0.04	0.02
	KRR	0.13	0.09
	XGBoost	0.05	0.01
Splashdown COM	FC	0.10	0.02
	KRR	0.18	0.01
	XGBoost	0.02	0.01

Table 3.1: Splashdown extrapolation results. We learn on 5k systems of $2-5$ atoms and predict on systems with $6-10$ atoms. These results show that on average, the coulomb eigenvalue representation is better capable of extrapolating to a higher number of atoms, however, the splashdown representation achieves comparable results with the XGBoost model. These results are averaged over 10 runs, with the hyperparameters outlined in Appendix A.1.

believe the Splashdown representation is ready to replace other representations as of yet, however, we have successfully demonstrated its simplicity and capability as a materials representation for machine learning.

Chapter 4

Conclusion

Material science is the foundation of virtually every field of technology; it's fundamental to the development of everything from safer and more efficient vehicles to advanced medical devices and renewable energy technologies. As we continually strive to improve existing materials and design new ones with unprecedented properties, the need for accurate material predictions has never been more pronounced.

Traditionally, Density Functional Theory DFT has been the go-to computational method for studying the electronic properties of materials. It's widely used for understanding, predicting, and designing materials from first principles. DFT provides a way to compute the ground state of a many-body quantum system, effectively predicting the behaviour of atoms in molecules and solids. While immensely valuable, DFT simulations can become intractable due to their computational cost as the size and complexity of the system increase. The computational expense rises steeply with the number of electrons in the system, making it challenging to apply DFT to large, complex, and dynamic systems.

To address this, my thesis explores the potential of ML models for predicting material properties, with a particular focus on adsorption energy prediction. I successfully trained ML models to do so with the inclusion of a similarity matrix, opening a new avenue for fast and accurate discovery of HEA electrocatalysts. The trained model was able to predict new potential HEAs for catalysis with an overpotential of 0.27eV, a breakthrough with strong implications in energy storage solutions.

Second, my work on `ORGANIZER` has led to the creation of a novel database management system, specially designed to aid developers and scientists with little technical expertise. This state-of-the-art system has facilitated the discovery of a new potential laser gain

material, surpassing current industry standards, by promoting inter-laboratory collaboration and data sharing. The success in discovering a new organic laser molecule using a self-driving lab approach exemplifies the enormous potential of melding interdisciplinary knowledge with advanced computing tools.

In the development of the aMC project, my involvement was instrumental in testing and implementing this new Monte Carlo-based optimizer. This tool effectively eliminates the need for costly gradient computations and has shown performance comparable to gradient-based optimizers.

Lastly, my work developing the splashdown representation resulted in a novel method of material representation for machine learning, which stands on par with other advanced representations in learning long-range interactions. The collective findings from these projects demonstrate the transformative role of machine learning in computational physics and materials science, pushing the boundaries of current capabilities and providing new pathways for scientific discovery and advancing toward a future with greener energy.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[3] Dylan M Anstine and Olexandr Isayev. Machine learning interatomic potentials and long-range physics. *The Journal of Physical Chemistry A*, 127(11):2417–2431, 2023.

[4] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

[5] F. Appel and R. Wagner. Intermetallics: Titanium aluminides. In K.H. Jürgen Buschow, Robert W. Cahn, Merton C. Flemings, Bernhard Ilschner, Edward J. Kramer, Subhash Mahajan, and Patrick Veyssière, editors, *Encyclopedia of Materials: Science and Technology*, pages 4246–4264. Elsevier, Oxford, 2001.

[6] Nongnuch Artrith and Alexie M Kolpak. Understanding the composition and activity of electrocatalytic nanoalloys in aqueous solvents: a combination of dft and accurate neural network potentials. *Nano letters*, 14(5):2670–2676, 2014.

[7] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11:501–528, 2020.

[8] Allen J Bard, Larry R Faulkner, and Henry S White. *Electrochemical methods: fundamentals and applications*. John Wiley & Sons, 2022.

[9] Allen J Bard and Marye Anne Fox. Artificial photosynthesis: solar splitting of water to hydrogen and oxygen. *Accounts of Chemical Research*, 28(3):141–145, 1995.

[10] Christopher J Bartel, Amalie Trewartha, Qi Wang, Alexander Dunn, Anubhav Jain, and Gerbrand Ceder. A critical examination of compound stability predictions from machine-learned formation energies. *npj computational materials*, 6(1):97, 2020.

[11] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[12] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010.

[13] Ethan Bass. ethanbass/chromconverter: v0.4.1, June 2023.

[14] Thomas AA Batchelor, Jack K Pedersen, Simon H Winther, Ivano E Castelli, Karsten W Jacobsen, and Jan Rossmeisl. High-entropy alloys as a discovery platform for electrocatalysis. *Joule*, 3(3):834–845, 2019.

[15] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of chemical physics*, 134(7), 2011.

[16] Jörg Behler and Gábor Csányi. Machine learning potentials for extended systems: a perspective. *The European Physical Journal B*, 94:1–11, 2021.

[17] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

[18] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[19] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[20] Peter Blaha, Karlheinz Schwarz, Fabien Tran, Robert Laskowski, Georg KH Madsen, and Laurence D Marks. Wien2k: An apw+ lo program for calculating the properties of solids. *The Journal of chemical physics*, 152(7):074101, 2020.

[21] Daniel J Blair, Sriyankari Chitti, Melanie Trobe, David M Kostyra, Hannah MS Haley, Richard L Hansen, Steve G Ballmer, Toby J Woods, Wesley Wang, Vikram Mubayi, et al. Automated iterative c sp 3–c bond formation. *Nature*, 604(7904):92–97, 2022.

[22] Sophie Boehm, Louise Jeffery, Kelly Levin, Judit Hecke, Clea Schumer, Claire Fyson, Aman Majid, Joel Jaeger, Anna Nilsson, Stephen Naimoli, et al. State of climate action 2022. 2022.

[23] Silvia Bonfanti and Walter Kob. Methods to locate saddle points in complex landscapes. *The Journal of chemical physics*, 147(20), 2017.

[24] Phelim P Boyle. Options: A monte carlo approach. *Journal of financial economics*, 4(3):323–338, 1977.

[25] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[26] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[27] Richard L Burden. *Numerical analysis*. Brooks/Cole Cengage Learning, 2011.

[28] Laurie J Butler. Chemical reaction dynamics beyond the born-oppenheimer approximation. *Annual review of physical chemistry*, 49(1):125–171, 1998.

[29] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[30] Zixuan Cang and Guo-Wei Wei. Analysis and prediction of protein folding energy changes upon mutation by element specific persistent homology. *Bioinformatics*, 33(22):3549–3557, 2017.

[31] Zixuan Cang and Guo-Wei Wei. Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. *International journal for numerical methods in biomedical engineering*, 34(2):e2914, 2018.

[32] Brain Cantor, ITH Chang, Peter Knight, and AJB Vincent. Microstructural development in equiatomic multicomponent alloys. *Materials Science and Engineering: A*, 375:213–218, 2004.

[33] Richard Catlow. *Defects and disorder in crystalline and amorphous solids*, volume 418. Springer Science & Business Media, 2012.

[34] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[35] David M Ceperley and Berni J Alder. Ground state of the electron gas by a stochastic method. *Physical review letters*, 45(7):566, 1980.

[36] Albert Aarón Cervera-Uribe and Paul Erick Méndez-Monroy. U19-net: a deep learning approach for obstacle detection in self-driving cars. *Soft computing (Berlin, Germany)*, 26(11):5195–5207, 2022.

[37] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23(89):1–64, 2022.

[38] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[39] Chi Chen, Yunxing Zuo, Weike Ye, Xiangguo Li, and Shyue Ping Ong. Learning properties of ordered and disordered materials from multi-fidelity data. *Nature Computational Science*, 1(1):46–53, 2021.

[40] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[41] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[42] Melodie Christensen, Lars PE Yunker, Folarin Adedeji, Florian Häse, Loïc M Roch, Tobias Gensch, Gabriel dos Passos Gomes, Tara Zepel, Matthew S Sigman, Alán Aspuru-Guzik, et al. Data-science driven autonomous process optimization. *Communications Chemistry*, 4(1):112, 2021.

[43] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[44] Marvin L Cohen and James R Chelikowsky. *Electronic structure and optical properties of semiconductors*, volume 75. Springer Science & Business Media, 2012.

[45] YL Cun, L Bottou, G Orr, and K Muller. Efficient backprop, neural networks: tricks of the trade. *Lecture notes in computer sciences*, 1524:5–50, 1998.

[46] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International conference on machine learning*, pages 2260–2268. PMLR, 2020.

[47] James Damewood, Jessica Karaguesian, Jaclyn R Lunger, Aik Rui Tan, Mingrou Xie, Jiayu Peng, and Rafael Gómez-Bombarelli. Representations of materials for machine learning. *Annual Review of Materials Research*, 53, 2023.

[48] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

[49] Mark K Debe. Electrocatalyst approaches and challenges for automotive fuel cells. *Nature*, 486(7401):43–51, 2012.

[50] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[51] Rui Ding, Shiqiao Zhang, Yawen Chen, Zhiyan Rui, Kang Hua, Yongkang Wu, Xiaoke Li, Xiao Duan, Xuebin Wang, Jia Li, et al. Application of machine learning in optimizing proton exchange membrane fuel cells: A review. *Energy and AI*, page 100170, 2022.

[52] Daniele Dragoni, Thomas D Daff, Gábor Csányi, and Nicola Marzari. Achieving dft accuracy with a machine-learning interatomic potential: Thermomechanics and defects in bcc ferromagnetic iron. *Physical Review Materials*, 2(1):013808, 2018.

[53] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[54] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13, 2000.

[55] Richard Durrett and R Durrett. *Essentials of stochastic processes*, volume 1. Springer, 1999.

[56] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[57] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

[58] Tov Elperin, I Gertsbakh, and Michael Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, 1991.

[59] Felix A Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S Schoenholz, George E Dahl, Oriol Vinyals, Steven Kearnes, Patrick F Riley, and O Anatole Von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid dft error. *Journal of chemical theory and computation*, 13(11):5255–5264, 2017.

[60] Martha M Flores-Leonar, Luis M Mejía-Mendoza, Andrés Aguilar-Granda, Benjamin Sanchez-Lengeling, Hermann Tribukait, Carlos Amador-Bedolla, and Alán Aspuru-Guzik. Materials acceleration platforms: On the way to autonomous experimentation. *Current Opinion in Green and Sustainable Chemistry*, 25:100370, 2020.

[61] David B Fogel and Lauren C Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32(3):171–182, 1994.

[62] Stephanie Forrest. Genetic algorithms. *ACM computing surveys (CSUR)*, 28(1):77–80, 1996.

[63] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

[64] Nathan C Frey, Deji Akinwande, Deep Jariwala, and Vivek B Shenoy. Machine learning-enabled design of point defects in 2d materials for quantum and neuromorphic information processing. *ACS nano*, 14(10):13406–13417, 2020.

[65] Rudiger Frey and Paul Embrechts. *Quantitative Risk Management*. Princeton University Press, 2010.

[66] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[67] Yuqing Gao and Khalid M. Mosalam. Deep transfer learning for image-based structural damage recognition. *Computer-aided civil and infrastructure engineering*, 33(9):748–768, 2018.

[68] Hubert A Gasteiger, Shyam S Kocha, Bhaskar Sompalli, and Frederick T Wagner. Activity benchmarks and requirements for pt, pt-alloy, and non-pt oxygen reduction catalysts for pemfcs. *Applied Catalysis B: Environmental*, 56(1-2):9–35, 2005.

[69] Hubert A Gasteiger and Nenad M Marković. Just a dream—or future reality? *science*, 324(5923):48–49, 2009.

[70] Theophile Gaudin. *ALGORITHMS FOR SELF-DRIVING LABS*. PhD thesis, 2023.

[71] Christopher R Genovese, Christopher J Miller, Robert C Nichol, Mihir Arjunwadkar, and Larry Wasserman. Nonparametric inference for the cosmic microwave background. 2004.

[72] Charles J Geyer. Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022:45, 2011.

[73] Luca M Ghiringhelli, Jan Vybiral, Sergey V Levchenko, Claudia Draxl, and Matthias Scheffler. Big data of materials science: critical role of the descriptor. *Physical review letters*, 114(10):105503, 2015.

[74] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.

[75] Michael B Giles. Multilevel monte carlo path simulation. *Operations research*, 56(3):607–617, 2008.

[76] Eric P Gillis and Martin D Burke. A simple and modular strategy for small molecule synthesis: iterative suzuki- miyaura coupling of b-protected haloboronic acid building blocks. *Journal of the American Chemical Society*, 129(21):6716–6717, 2007.

[77] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[78] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

[79] Zachary L Glick, Derek P Metcalf, Alexios Koutsoukas, Steven A Spronk, Daniel L Cheney, and C David Sherrill. Ap-net: An atomic-pairwise neural network for smooth and transferable interaction potentials. *The Journal of Chemical Physics*, 153(4), 2020.

[80] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 347–352. IEEE, 1996.

[81] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[82] Xavier Gonze, Bernard Amadon, Gabriel Antonius, Frédéric Arnardi, Lucas Baguet, Jean-Michel Beuken, Jordan Bieder, François Bottin, Johann Bouchet, Eric Bousquet, Nils Brouwer, Fabien Bruneval, Guillaume Brunin, Théo Cavignac, Jean-Baptiste Charraud, Wei Chen, Michel Côté, Stefaan Cottenier, Jules Denier, Grégory Geneste, Philippe Ghosez, Matteo Giantomassi, Yannick Gillet, Olivier Gingras, Donald R. Hamann, Geoffroy Hautier, Xu He, Nicole Helbig, Natalie Holzwarth, Yongchao Jia, François Jollet, William Lafargue-Dit-Hauret, Kurt Lejaeghere, Miguel A. L. Marques, Alexandre Martin, Cyril Martins, Henrique P. C. Miranda, Francesco Naccarato, Kristin Persson, Guido Petretto, Valentin Planes, Yann Pouillon, Sergei Prokhorenko, Fabio Ricci, Gian-Marco Rignanese, Aldo H. Romero, Michael Marcus Schmitt, Marc Torrent, Michiel J. van Setten, Benoit Van Troeye, Matthieu J. Verstraete, Gilles Zérah, and Josef W. Zwanziger. The abinit project: Impact, environment and recent developments. *Comput. Phys. Commun.*, 248:107042, 2020.

[83] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[84] Christoph Gorgulla, AkshatKumar Nigam, Matt Koop, Suleyman S Cinaroglu, Christopher Secker, Mohammad Haddadnia, Abhishek Kumar, Yehor Malets, Alexander Hasson, Roni Levin-Konigsberg, et al. Virtualflow 2.0-the next generation drug discovery platform enabling adaptive screens of 69 billion molecules. *bioRxiv*, pages 2023–04, 2023.

[85] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[86] Martin A Green, Anita Ho-Baillie, and Henry J Snaith. The emergence of perovskite solar cells. *Nature photonics*, 8(7):506–514, 2014.

[87] RL Greenaway, V Santolini, MJ Bennison, BM Alston, CJ Pugh, MA Little, M Miklitz, EGB Eden-Rump, R Clowes, A Shakil, et al. High-throughput discovery of organic cages and catenanes using computational screening fused with robotic synthesis. *Nature communications*, 9(1):2849, 2018.

[88] PostgreSQL Global Development Group, Jun 2023.

[89] James E Gubernatis. Marshall rosenbluth and the metropolis algorithm. *Physics of plasmas*, 12(5):057303, 2005.

[90] Christian P Haas, Maximilian Lubbesmeyer, Edward H Jin, Matthew A McDonald, Brent A Koscher, Nicolas Guimond, Laura Di Rocco, Henning Kayser, Samuel Leweke, Sebastian Niedenfuhr, et al. Open-source chromatographic data analysis for reaction optimization and screening. *ACS Central Science*, 9(2):307–317, 2023.

[91] DR Hamann, M Schlüter, and C Chiang. Norm-conserving pseudopotentials. *Physical Review Letters*, 43(20):1494, 1979.

[92] James D Hamilton. *Time series analysis*. Princeton university press, 2020.

[93] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[94] Douglas R Hartree. The wave mechanics of an atom with a non-coulomb central field. part i. theory and methods. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24, pages 89–110. Cambridge university press, 1928.

[95] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, and Alán Aspuru-Guzik. Gryffin: An algorithm for bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3):031406, 2021.

[96] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[97] QF He, ZY Ding, YF Ye, and Y Yang. Design of high-entropy alloy: a perspective from nonideal mixing. *Jom*, 69:2092–2098, 2017.

[98] Frank Herman, John P Van Dyke, and Irene B Ortenburger. Improved statistical exchange approximation for inhomogeneous many-electron systems. *Physical Review Letters*, 22(16):807, 1969.

[99] Maritza Hernandez, Arman Zaribafiyan, Maliheh Aramon, and Mohammad Naghibi. A novel graph-based approach for determining molecular similarity. *arXiv preprint arXiv:1601.06693*, 2016.

[100] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[101] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[102] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

[103] Alois Huning. Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution, 1976.

[104] Haoyan Huo and Matthias Rupp. Unified representation for machine learning of molecules and crystals. *arXiv preprint arXiv:1704.06439*, 13754, 2017.

[105] Giuseppe Iadonisi, Giovanni Cantele, and Maria Luisa Chiofalo. *Introduction to Solid State Physics and Crystalline Nanostructures*. Springer, 2014.

[106] Richard J Ingham, Claudio Battilocchio, Daniel E Fitzpatrick, Eric Sliwinski, Joel M Hawkins, and Steven V Ley. A systems approach towards an intelligent and self-controlling platform for integrated continuous reaction sequences. *Angewandte Chemie*, 127(1):146–150, 2015.

[107] Robert A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988.

[108] Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino. Food discovery with uber eats: Using graph learning to power recommendations. *Accessed March*, 1:2021, 2019.

[109] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.

[110] Jidon Jang, Geun Ho Gu, Juhwan Noh, Juhwan Kim, and Yousung Jung. Structure-based synthesizability prediction of crystals using partially supervised learning. *Journal of the American Chemical Society*, 142(44):18836–18843, 2020.

[111] Bas Cornelis Jansen, Lise Hafkenscheid, Albert Bondt, Richard Andrew Gardner, Jenifer Lynn Hendel, Manfred Wuhrer, and Daniel Ian Richard Spencer. Happytools: A software for high-throughput hplc data processing and quantitation. *PloS one*, 13(7):e0200280, 2018.

[112] Kun Ji, Chuanbin Zhu, Saman Yaghmaei-Sabegh, Jianqi Lu, Yefei Ren, and Ruizhi Wen. Site classification using deep-learning-based image recognition techniques. *Earthquake engineering & structural dynamics*, 2022.

[113] Yi Jiang, Yuan-Yuan Liu, Xu Liu, He Lin, Kun Gao, Wen-Yong Lai, and Wei Huang. Organic solid-state lasers: A materials view and future development. *Chemical Society Reviews*, 49(16):5885–5944, 2020.

[114] YEH Jien-Wei. Recent progress in high entropy alloys. *Ann. Chim. Sci. Mat*, 31(6):633–648, 2006.

[115] Michael Jones, Brain Campbell, and Chuck Mortimore. Json web token (jwt) profile for oauth 2.0 client authentication and authorization grants. Technical report, 2015.

[116] Peter Bjørn Jørgensen, Estefanía Garijo del Río, Mikkel N Schmidt, and Karsten Wedel Jacobsen. Materials property prediction using symmetry-labeled graphs as atomic position independent descriptors. *Physical Review B*, 100(10):104114, 2019.

[117] Nitin Kanagaraj, David Hicks, Ayush Goyal, Sanju Tiwari, and Ghanapriya Singh. Deep learning using computer vision in self driving cars for lane and traffic sign detection. *International journal of system assurance engineering and management*, 12(6):1011–1025, 2021.

[118] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[119] Dipl Ing Karl Heinz Kellermayer. Numerische optimierung von computer-modellen mittels der evolutionsstrategie hans-paul schwefel birkhäuser, basel and stuttgart, 1977 370 pages hardback sf/48 isbn 3-7643-0876-1. *Cybernetics and System*, 7(3-4):319–320, 1977.

[120] Sang-Heon Kim, Hansoo Kim, and Nack J Kim. Brittle intermetallic compound makes ultrastrong low-density steel with large ductility. *Nature*, 518(7537):77–79, 2015.

[121] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[122] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[123] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[124] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[125] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

[126] Georg Kresse and Jürgen Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational materials science*, 6(1):15–50, 1996.

[127] Georg Kresse and Jürgen Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical review B*, 54(16):11169, 1996.

[128] Georg Kresse and Jürgen Hafner. Ab initio molecular dynamics for liquid metals. *Physical review B*, 47(1):558, 1993.

[129] Georg Kresse and Jürgen Hafner. Ab initio molecular-dynamics simulation of the liquid-metal–amorphous-semiconductor transition in germanium. *Physical Review B*, 49(20):14251, 1994.

[130] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[131] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[132] John K Kruschke. Bayesian data analysis. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(5):658–676, 2010.

[133] Jarno Laakso, Lauri Himanen, Henrietta Homm, Eiaki V. Morooka, Marc O. J. Jäger, Milica Todorović, and Patrick Rinke. Updates to the dscribe library: New descriptors and derivatives, 2023.

[134] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*, volume 3. Mcgraw-hill New York, 2007.

[135] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[136] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[137] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[138] Chengteh Lee, Weitao Yang, and Robert G Parr. Development of the colle-salvetti correlation-energy formula into a functional of the electron density. *Physical review B*, 37(2):785, 1988.

[139] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254):10, 2012.

[140] Steven D Levitt and Stephen J Dubner. *Freakonomics*. B DE BOOKS, 2014.

[141] Junqi Li, Steven G Ballmer, Eric P Gillis, Seiko Fujii, Michael J Schmidt, Andrea ME Palazzolo, Jonathan W Lehmann, Greg F Morehouse, and Martin D Burke. Synthesis of many different types of organic small molecules using one automated process. *Science*, 347(6227):1221–1226, 2015.

[142] Qing-Jie Li, Emine Küçükbenli, Stephen Lam, Boris Khaykovich, Efthimios Kaxiras, and Ju Li. Development of robust neural-network interatomic potential for molten salt. *Cell Reports Physical Science*, 2(3), 2021.

[143] Wenwen Li, Yasunobu Ando, Emi Minamitani, and Satoshi Watanabe. Study of li atom diffusion in amorphous li3po4 with neural network potential. *The Journal of chemical physics*, 147(21), 2017.

[144] Jean-Francois Lutz, Jean-Marie Lehn, EW Meijer, and Krzysztof Matyjaszewski. From precision polymers to complex materials and systems. *Nature Reviews Materials*, 1(5):1–14, 2016.

[145] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA, 2013.

[146] Benjamin P MacLeod, Fraser GL Parlane, Thomas D Morrissey, Florian Häse, Loïc M Roch, Kevan E Dettelbach, Raphaell Moreira, Lars PE Yunker, Michael B Rooney, Joseph R Deeth, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020.

[147] Richard M Martin. Electronic structure. *Physik Journal*, 8(4):54, 2009.

[148] Bryce Meredig, Ankit Agrawal, Scott Kirklin, James E Saal, Jeff W Doak, Alan Thompson, Kunpeng Zhang, Alok Choudhary, and Christopher Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[149] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2.

[150] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[151] Todd Millecam, Austin J Jarrett, Naomi Young, Dana E Vanderwall, and Dennis Della Corte. Coming of age of allotrope: proceedings from the fall 2020 allotrope connect. *Drug Discovery Today*, 26(8):1922–1928, 2021.

[152] Inc. MinIO. Minio| high performance, kubernetes native object storage. *MinIO.*, 2021.

[153] Daniel B Miracle, Jonathan D Miller, Oleg N Senkov, Christopher Woodward, Michael D Uchic, and Jaimie Tiley. Exploration and development of high entropy alloys for structural applications. *Entropy*, 16(1):494–525, 2014.

[154] Daniel B Miracle and Oleg N Senkov. A critical review of high entropy alloys and related concepts. *Acta Materialia*, 122:448–511, 2017.

[155] David J Montana, Lawrence Davis, et al. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

[156] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, 2013.

[157] Klaus Müller and Leo D Brown. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoretica chimica acta*, 53:75–93, 1979.

[158] Subramanian Nellaiappan, Nirmal Kumar Katiyar, Ritesh Kumar, Arko Parui, Kirtiman Deo Malviya, KG Pradeep, Abhishek K Singh, Sudhanshu Sharma, Chandra Sekhar Tiwary, and Krishanu Biswas. High-entropy alloys as catalysts for the co2 and co reduction reactions: Experimental realization. *ACS Catalysis*, 10(6):3658–3663, 2020.

[159] Mark Newman. *Networks*. Oxford university press, 2018.

[160] Samuel P Niblett, Mirza Galib, and David T Limmer. Learning intermolecular forces at liquid–vapor interfaces. *The Journal of chemical physics*, 155(16), 2021.

[161] Juhwan Noh, Geun Ho Gu, Sungwon Kim, and Yousung Jung. Uncertainty-quantified hybrid machine learning/density functional theory high throughput screening method for crystals. *Journal of Chemical Information and Modeling*, 60(4):1996–2003, 2020.

[162] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[163] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[164] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[165] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, Ricardo Guerrerro Moreno, Ben Glocker, and Daniel Rueckert. Spectral graph convolutions for population-based disease prediction. In *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20*, pages 177–185. Springer, 2017.

[166] Cheol Woo Park, Mordechai Kornbluth, Jonathan Vandermause, Chris Wolverton, Boris Kozinsky, and Jonathan P Mailoa. Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture. *npj Computational Materials*, 7(1):73, 2021.

[167] Cheol Woo Park and Chris Wolverton. Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery. *Physical Review Materials*, 4(6):063801, 2020.

[168] Ho Bum Park, Jovan Kamcev, Lloyd M Robeson, Menachem Elimelech, and Benny D Freeman. Maximizing the right stuff: The trade-off between membrane permeability and selectivity. *Science*, 356(6343):eaab0530, 2017.

[169] Behnam Parsaeifard, Deb Sankar De, Jonas A Finkler, and Stefan Goedecker. Fingerprint-based detection of non-local effects in the electronic structure of a simple single component covalent system. *Condensed Matter*, 6(1):9, 2021.

[170] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[171] Jean-Francois Paul and Edmond Payen. Vacancy formation on mos2 hydrodesulfurization catalyst: Dft study of the mechanism. *The Journal of Physical Chemistry B*, 107(17):4057–4064, 2003.

[172] Jack K Pedersen, Thomas AA Batchelor, Alexander Bagger, and Jan Rossmeisl. High-entropy alloys as catalysts for the co2 and co reduction reactions. *Acs Catalysis*, 10(3):2169–2176, 2020.

[173] Jiayu Peng, Daniel Schwalbe-Koda, Karthik Akkiraju, Tian Xie, Livia Giordano, Yang Yu, C John Eom, Jaclyn R Lunger, Daniel J Zheng, Reshma R Rao, et al. Human-and machine-centred designs of molecules and materials for sustainability and decarbonization. *Nature Reviews Materials*, 7(12):991–1009, 2022.

[174] John P Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical review letters*, 77(18):3865, 1996.

[175] John P Perdew, Stefan Kurth, Aleš Zupan, and Peter Blaha. Accurate density functional with correct formal properties: A step beyond the generalized gradient approximation. *Physical review letters*, 82(12):2544, 1999.

[176] John P Perdew and Yue Wang. Accurate and simple analytic representation of the electron-gas correlation energy. *Physical review B*, 45(23):13244, 1992.

[177] John P Perdew and Alex Zunger. Self-interaction correction to density-functional approximations for many-electron systems. *Physical Review B*, 23(10):5048, 1981.

[178] Ghanshyam Pilania, Arun Mannodi-Kanakkithodi, BP Uberuaga, Rampi Ramprasad, JE Gubernatis, and Turab Lookman. Machine learning bandgaps of double perovskites. *Scientific reports*, 6(1):19375, 2016.

[179] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[180] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[181] Wiktor Pronobis, Alexandre Tkatchenko, and Klaus-Robert Muller. Many-body descriptors for predicting molecular properties with machine learning: Analysis of pairwise and three-body interactions in molecules. *Journal of chemical theory and computation*, 14(6):2991–3003, 2018.

[182] Edward O Pyzer-Knapp, Changwon Suh, Rafael Gómez-Bombarelli, Jorge Aguilera-Iparraguirre, and Alán Aspuru-Guzik. What is high-throughput virtual screening? a perspective from organic materials discovery. *Annual Review of Materials Research*, 45:195–216, 2015.

[183] Hua-Jun Qiu, Gang Fang, Jiaojiao Gao, Yuren Wen, Juan Lv, Huanglong Li, Guoqiang Xie, Xingjun Liu, and Shuhui Sun. Noble metal-free nanoporous high-entropy alloys as highly efficient electrocatalysts for oxygen evolution reaction. *ACS Materials Letters*, 1(5):526–533, 2019.

[184] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Big data meets quantum chemistry approximations: the δ-machine learning approach. *Journal of chemical theory and computation*, 11(5):2087–2096, 2015.

[185] Rampi Ramprasad, Rohit Batra, Ghanshyam Pilania, Arun Mannodi-Kanakkithodi, and Chiho Kim. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials*, 3(1):54, 2017.

[186] Federico Rastrelli and Alessandro Bagno. Predicting the nmr spectra of paramagnetic molecules by dft: Application to organic free radicals and transition-metal complexes. *Chemistry–A European Journal*, 15(32):7990–8004, 2009.

[187] Asha Raveendran, Mijun Chandran, and Ragupathy Dhanusuraman. A comprehensive review on the electrochemical parameters and recent material development of electrochemical water splitting electrocatalysts. *RSC advances*, 13(6):3843–3876, 2023.

[188] Ali Raza, Arni Sturluson, Cory M Simon, and Xiaoli Fern. Message passing neural networks for partial charge assignment to metal–organic frameworks. *The Journal of Physical Chemistry C*, 124(35):19070–19082, 2020.

[189] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

[190] LM Rasdi Rere, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72:137–144, 2015.

[191] Loïc M Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars PE Yunker, Jason E Hein, and Alán Aspuru-Guzik. Chemos: Orchestrating autonomous experimentation. *Science Robotics*, 3(19):eaat5559, 2018.

[192] Marshall N Rosenbluth. Genesis of the monte carlo algorithm for statistical mechanics. In *AIP Conference Proceedings*, volume 690, pages 22–30. American Institute of Physics, 2003.

[193] HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The computer journal*, 3(3):175–184, 1960.

[194] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

[195] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34, 1995.

[196] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[197] Marcel Salathé and James H Jones. Dynamics and control of diseases in networks with community structure. *PLoS computational biology*, 6(4):e1000736, 2010.

[198] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[199] Donald E Sands. *Introduction to crystallography*. Courier Corporation, 1993.

[200] Y Sarada Devi, S Sathvik, P Ananya, P Tharuni, and N Naga Krishna Vamsi. Vision-based obstacle detection and collision prevention in self-driving cars. *Journal of physics. Conference series*, 2335(1):12019–, 2022.

[201] Hamed Sarvari, Ehab Abozinadah, Alex Mbaziira, and Damon McCoy. Constructing and analyzing criminal networks. In *2014 IEEE security and privacy workshops*, pages 84–91. IEEE, 2014.

[202] Vishwas Satgar. *The Climate Crisis*. Wits University Press, Johannesburg, 2018.

[203] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[204] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998.

[205] Sara Scaramuccia, Federico Iuricich, Leila De Floriani, and Claudia Landi. Computing multiparameter persistent homology through a discrete morse-based approach. *Computational Geometry*, 89:101623, 2020.

[206] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[207] Tom Schaul, Ioannis Antonoglou, and David Silver. Unit tests for stochastic optimization. *arXiv preprint arXiv:1312.6055*, 2013.

[208] Jonathan Schmidt, Love Pettersson, Claudio Verdozzi, Silvana Botti, and Miguel AL Marques. Crystal graph attention networks for the prediction of stable materials. *Science advances*, 7(49):eabi7948, 2021.

[209] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):13890, 2017.

[210] Atsuto Seko, Tomoya Maekawa, Koji Tsuda, and Isao Tanaka. Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single-and binary-component solids. *Physical Review B*, 89(5):054303, 2014.

[211] Oleg N Senkov, GB Wilks, JM Scott, and Daniel B Miracle. Mechanical properties of nb25mo25ta25w25 and v20nb20mo20ta20w20 refractory high entropy alloys. *Intermetallics*, 19(5):698–706, 2011.

[212] Randall S Sexton, Robert E Dorsey, and John D Johnson. Beyond backpropagation: using simulated annealing for training neural networks. *Journal of Organizational and End User Computing (JOEUC)*, 11(3):3–10, 1999.

[213] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.

[214] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[215] Pei Kang Shen. *Electrochemical oxygen reduction: fundamental and applications*. Springer Nature, 2021.

[216] John C Slater. A simplification of the hartree-fock method. *Physical review*, 81(3):385, 1951.

[217] Edwin Soedarmadji, Helge S Stein, Santosh K Suram, Dan Guevarra, and John M Gregoire. Tracking materials science data lineage to manage millions of materials experiments and analyses. *npj Computational Materials*, 5(1):79, 2019.

[218] Alan Sokal. Monte carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration: Basics and applications*, pages 131–192. Springer, 1997.

[219] Meta Open Source. React. the library for web and native user interfaces.

[220] Alessandro Sperduti. Encoding labeled graphs by labeling raam. *Advances in Neural Information Processing Systems*, 6, 1993.

[221] SQLAlchemy. Alembic.

[222] TS Srivatsan and Manoj Gupta. High entropy alloys: innovations, advances, and applications. 2020.

[223] Vojislav R Stamenkovic, Ben Fowler, Bongjin Simon Mun, Guofeng Wang, Philip N Ross, Christopher A Lucas, and Nenad M Markovic. Improved oxygen reduction activity on pt3ni (111) via increased surface site availability. *science*, 315(5811):493–497, 2007.

[224] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław M Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip J Kitson, Davide Angelone, et al. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211, 2019.

[225] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[226] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[227] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[228] Rohun Tripathi and Bharat Singh. Rso: a gradient free sampling based approach for training deep neural networks. *arXiv preprint arXiv:2005.05955*, 2020.

[229] Christian Tuma, A Daniel Boese, and Nicholas C Handy. Predicting the binding energies of h-bonded complexes: A comparative dft study. *Physical Chemistry Chemical Physics*, 1(17):3939–3947, 1999.

[230] David Vanderbilt. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. *Physical review B*, 41(11):7892, 1990.

[231] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[232] David J Wales. Rearrangements of 55-atom lennard-jones and (c60) 55 clusters. *The Journal of chemical physics*, 101(5):3750–3762, 1994.

[233] Binhang Wang, Tianyu Wu, Guangrui Chen, Xinyao Liu, Wen Li, Qingxia He, Dong-Sheng Li, Bu Yuan Guan, and Yunling Liu. General synthesis of hierarchically macro/mesoporous fe, ni-doped cose/n-doped carbon nanoshells for enhanced electrocatalytic oxygen evolution. *Inorganic Chemistry*, 60(9):6782–6789, 2021.

[234] Zhilong Wang, Yanqiang Han, Junfei Cai, Sicheng Wu, and Jinjin Li. Deeptmc: A deep learning platform to targeted design doped transition metal compounds. *Energy Storage Materials*, 45:1201–1211, 2022.

[235] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016.

[236] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, pages 2022–12, 2022.

[237] Christian Samuel Weiss. *über eine verbesserte Methode für die Bezeichnung der verschiedenen Flächen eines Krystallisationssystemes*. 1819.

[238] Stephen Whitelam, Viktor Selin, Ian Benlolo, Corneel Casert, and Isaac Tamblyn. Training neural networks using metropolis monte carlo and an adaptive variant. *Machine Learning: Science and Technology*, 3(4):045026, 2022.

[239] Stephen Whitelam, Viktor Selin, Sang-Won Park, and Isaac Tamblyn. Correspondence between neuroevolution and gradient descent. *Nature communications*, 12(1):6317, 2021.

[240] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[241] Kelin Xia, Xin Feng, Yiying Tong, and Guo Wei Wei. Persistent homology for the quantitative prediction of fullerene stability. *Journal of computational chemistry*, 36(6):408–422, 2015.

[242] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[243] Yue Xin, Shuhui Li, Yayang Qian, Wenkun Zhu, Haibo Yuan, Pengyan Jiang, Ruihan Guo, and Liangbing Wang. High-entropy alloys as a platform for catalysis: progress, challenges, and opportunities. *Acs Catalysis*, 10(19):11280–11306, 2020.

[244] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[245] J-W Yeh, S-K Chen, S-J Lin, J-Y Gan, T-S Chin, T-T Shun, C-H Tsau, and S-Y Chang. Nanostructured high-entropy alloys with multiple principal elements: novel alloy design concepts and outcomes. *Advanced engineering materials*, 6(5):299–303, 2004.

[246] Jien-Wei Yeh. Alloy design strategies and future trends in high-entropy alloys. *Jom*, 65:1759–1771, 2013.

[247] Jien-Wei Yeh. Physical metallurgy of high-entropy alloys. *Jom*, 67(10):2254–2261, 2015.

[248] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[249] Junhua You, Ruyue Yao, Wuren Ji, Yao Zhao, and Zhaoyu Wang. Research of high entropy alloys as electrocatalyst for oxygen evolution reaction. *Journal of Alloys and Compounds*, page 164669, 2022.

[250] Shuwen Yue, Maria Carolina Muniz, Marcos F Calegari Andrade, Linfeng Zhang, Roberto Car, and Athanassios Z Panagiotopoulos. When do short-range atomistic machine-learning models fall short? *The Journal of Chemical Physics*, 154(3), 2021.

[251] Kai Zeng and Dongke Zhang. Recent progress in alkaline water electrolysis for hydrogen production and applications. *Progress in energy and combustion science*, 36(3):307–326, 2010.

[252] Ye Zeng, Mengting Zhao, Zihao Huang, Weijie Zhu, Jiaxian Zheng, Qiu Jiang, Zhoucheng Wang, and Hanfeng Liang. Surface reconstruction of water splitting electrocatalysts. *Advanced Energy Materials*, 12(33):2201713, 2022.

[253] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[254] Yong Zhang, Ting Ting Zuo, Zhi Tang, Michael C Gao, Karin A Dahmen, Peter K Liaw, and Zhao Ping Lu. Microstructures and properties of high-entropy alloys. *Progress in materials science*, 61:1–93, 2014.

[255] Huan-Xiang Zhou and Xiaodong Pang. Electrostatic interactions in protein structure, folding, binding, and condensation. *Chemical reviews*, 118(4):1691–1741, 2018.

[256] Han Zhu, Zhenfeng Zhu, Jiace Hao, Shuhui Sun, Shuanglong Lu, Chan Wang, Piming Ma, Weifu Dong, and Mingliang Du. High-entropy alloy stabilized active ir for highly efficient acidic oxygen evolution. *Chemical Engineering Journal*, 431:133251, 2022.

[257] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

APPENDICES

Appendix A

Appendix

A.1 Splashdown

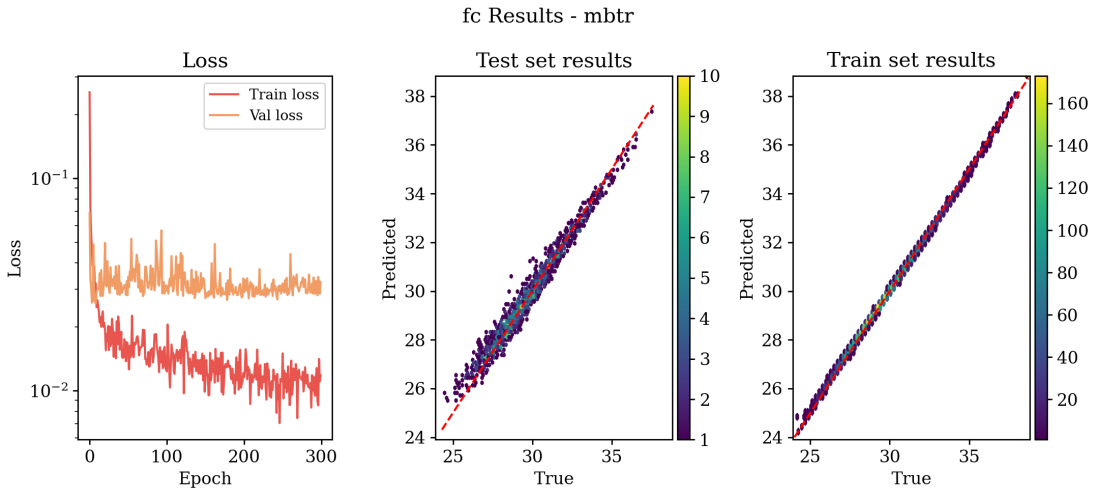
A.1.1 Training Hyperparameters

For all Splashdown runs, we use the following hyperparameters with a squared error loss function.

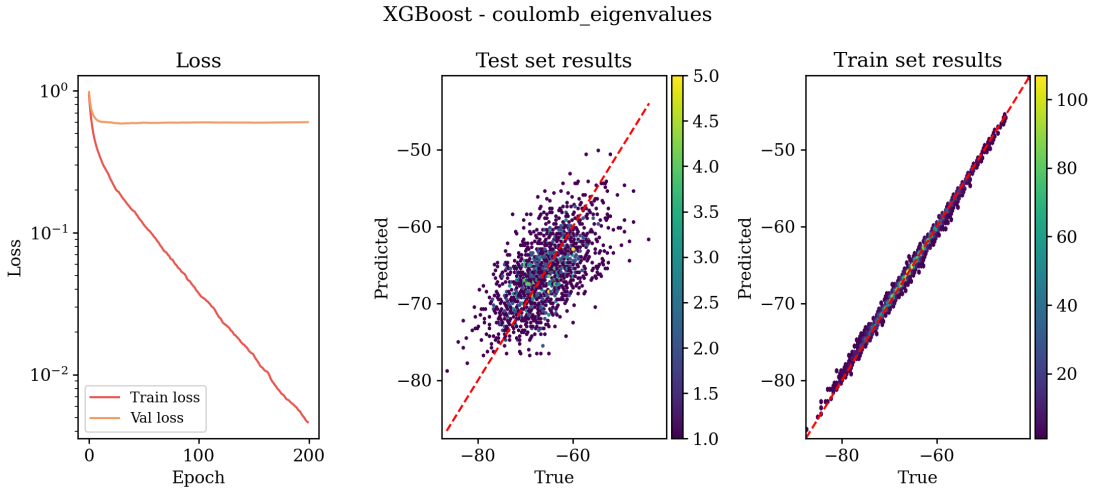
- **Fully-Connected Net**. For the MLP runs, we use a fully connected net with 3 hidden layers. The input of size N ($N = 360$ for the splashdown representation) to hidden layers of sizes 720, 360 and finally 90 neurons. The output layer is of unit size. Each layer (less the final one) have a SeLU activation, and the first layer uses a 20% dropout. These were trained with the use of a t4v2 gpu. The optimizer used was the Adam optimizer with a learning rate of $1e-3$.
- **XGBoost**. The gradient-boosted tree method uses 200 estimators and is trained with GPU as well.
- **Kernel Ridge Regression**. The KRR model was run with a 5-fold cross-validation and a grid search parameter sweep over $\alpha,\ \gamma = [1e-03, 1e-02, 1e1, 1, 1e1, 1e2, 1e3]$

A.1.2 Training Plots

In Figure A.1 we show training plots for a few splashdown runs.



(a) Training plot for the fully connected network with the MBTR representation. This training set contains 16 hydrogen atoms per system.



(b) XGBboost training example for a run using the CM representation. This training set consisted of 64 atom systems. We notice overfitting here as the training set decreases steadily however the validation set loss stops decreasing early. This translated to poor predictions on the test set.

Figure A.1: Some example training plots for the Splashdown benchmarking runs. Datasets consisted of 10k systems split into training, testing and validation sets with a 70/15/15 split respectively.

A.2 Data

	dtemp	final_length
0	86.437132	8.604352
1	81.119884	8.391744
2	41.676899	4.674493
3	36.945464	4.264215
4	47.433241	5.459050
5	55.764610	6.210544
6	88.963731	9.018799
7	57.671267	6.132602
8	41.166006	4.410357
9	79.004828	8.434166
10	31.680556	3.972634
11	87.563798	8.969953
12	57.206068	6.321179
13	69.717097	7.089026
14	66.695787	6.974062
15	76.762948	7.795214
16	49.368632	5.329521
17	84.262923	8.359585
18	41.745113	4.523680
19	30.906396	3.799927
20	61.765105	6.800428
21	39.869380	4.822165
22	60.058977	6.278803
23	54.898349	6.064562
24	47.202151	5.230793
25	86.757190	8.810087
26	41.296262	4.586234
27	83.648311	8.646468
28	37.472184	4.190874
29	49.671637	5.302489
30	51.883720	5.759951
31	46.404042	4.971677
32	53.535369	5.737427

33	45.455733	5.030863
34	64.101316	6.438753
35	88.061422	8.980661
36	49.947621	5.845035
37	31.970885	4.191903
38	58.412066	6.349729
39	72.355837	7.615527
40	44.261671	4.721758
41	71.322570	7.415449
42	49.663088	5.561402
43	54.542565	5.953140
44	73.397405	8.038000
45	63.317362	7.047392
46	74.196914	7.628191
47	62.078388	6.421451
48	40.701324	4.460888
49	52.547248	5.623986

Table A.1: Dummy data used for linear regression example. dtemp represents the change in temperature and final_length is the final length observed.

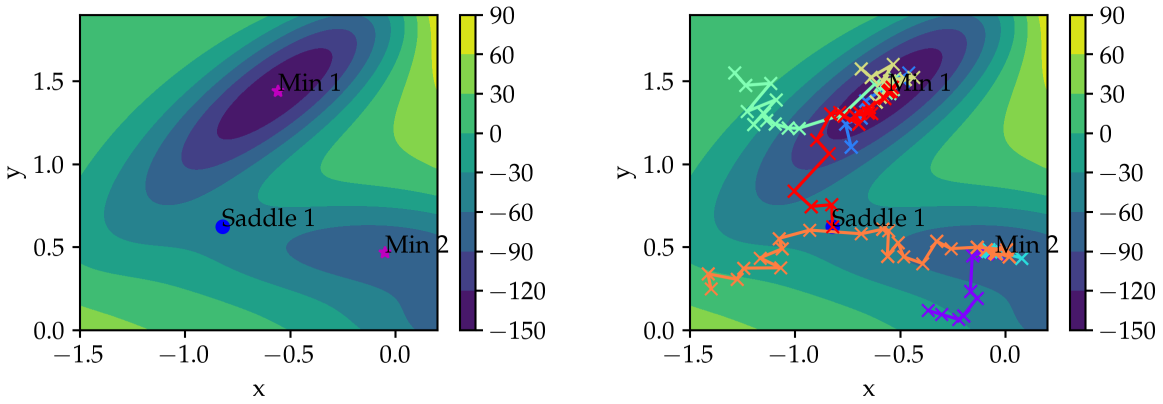
A.3 Diversity of Trajectories and MC Step Size

Here we conduct a small study on how σ, the move proposal variance, affects the diversity of MC optimization trajectories. We begin by defining a double-well potential; the Müller Brown potential which is a model which was introduced to describe a simple potential energy landscape and whose properties have been studied extensively to test the performance of various algorithms aimed at finding trajectories [23, 157, 232]. It is the sum of four gaussians and is defined by

$$V_{\mathrm{MB}}(x,y) = \sum_{i=1}^{4} A_i \exp\left[a_i\,(x-\bar{x}_i)^2 + b_i\,(x-\bar{x}_i)\,(y-\bar{y}_i) + c_i\,(y-\bar{y}_i)^2\right] \tag{A.1}$$

where

$$\begin{aligned} &A = (-200,-100,-170,15); \quad a = (-1,-1,-6.5,0.7) \\ &b = (0,0,11,0.6); \quad c = (-10,-10,-6.5,0.7) \\ &\bar{x} = (1,0,-0.5,-1); \quad \bar{y} = (0,0.5,1.5,1). \end{aligned} \tag{A.2}$$



(a) Contour plot of the Müller Brown potential.

(b) Five MC trajectories on the Müller Brown potential. These were run for $n = 1000$ steps with a move proposal variance of $\sigma = 10^{-2}$.

Figure A.2: Contour plots of the Müller Brown potential and five trajectories. Indicated are the saddle point at the two minimums at (around) $m1 \approx (-0.56, 1.44)$, $m2 \approx (-0.05, 0.467)$ and the saddle point at $s1 \approx (-0.82, 0.62)$.

We can use the MC algorithm Equation (2.70) to minimize this surface. Figure A.2(a) shows a contour plot of this surface and Figure A.2(b) shows five trajectories ran on this surface. Note that there are two minima on this surface and here we are using zero-temperature MC therefore there can not be any exploration between minima as there are no "uphill" moves allowed.

We showed that MC is able to minimize a complex landscape but now we explore how the trajectories' nature changes with different σ. By nature here we refer to how "diverse" are the runs, in that they explore more of the space. To study this we look at autocorrelation [92] which is a statistical technique used to measure the linear relationship between a time series and its lagged values. It measures how a data point at a specific time relates to data points at previous time steps. The autocorrelation function (ACF) at lag k, denoted as $\rho(k)$, is calculated using

$$\rho(k) = \frac{\text{Cov}(X_t, X_{t-k})}{\sqrt{\text{Var}(X_t) \cdot \text{Var}(X_{t-k})}} \tag{A.3}$$

Where: X_t is the data point at time t, $\text{Cov}(X_t, X_{t-k})$ is the covariance between X_t and X_{t-k} and $\text{Var}(X_t)$ is the variance of X_t.

The ACF provides insights into the temporal dependencies within a time series. A significant autocorrelation at lag k suggests that values at time t are more highly influenced by values at time $t-k$, indicating a pattern or seasonality in the data. Autocorrelation is a valuable tool in time series analysis for identifying trends, periodicities, and appropriate models for forecasting or analysis.

We use it here to study the temporal dependance of our trajectories and how they evolve with varying σs. To do so, we ran 1000 trajectories for 100 steps with $\sigma = [2, 10^{-1}, 5 \times 10^{-1}, 10^{-2}, 5 \times 10^{-2}, 10^{-3}, 5 \times 10^{-1}, 10^{-4}]$ and computed the first $M = 11$ lags. A summary of these aucorrelations is then estimated using $\hat{\tau}(M) = 1 + 2\sum_{i=1}^{M} \hat{\rho}(k)$, known as the integrated autocorrelation time [218]. As long as $M << N$ holds, where N is the sequence lengt then this estimator has a smaller variance.

The autocorrelation plots across varying sigma values in Figure A.3 show a trend about the behavior of the MC optimization. For smaller σ values, such as $\sigma = 0.001$, the distribution of autocorrelation times is sharp and narrow, indicating a consistent behavior. This suggests that with limited "noise" or randomness in trajectories, their patterns become more predictable. As σ values increase, the distribution broadens and flattens, as shown in $\sigma = 0.05$ to 0.5. This indicates increased variance in the autocorrelation times, suggesting more diverse paths through the landscape due to the introduction of greater randomness with the step sizes. For even larger sigma values, such as $\sigma = 2.0$, there's a reconvergence of behaviors with a more pronounced peak. This could be a manifestation of excessive noise pushing trajectories to behave more similarly because of fewer accepted moves (due to the zero-temperature limitation). In essence, while minimal randomness produces trajectories with consistent behaviors, increasing this randomness introduces variability, which might eventually stabilize at extremely high values, leading to a reconvergence towards less diverse trajectories.

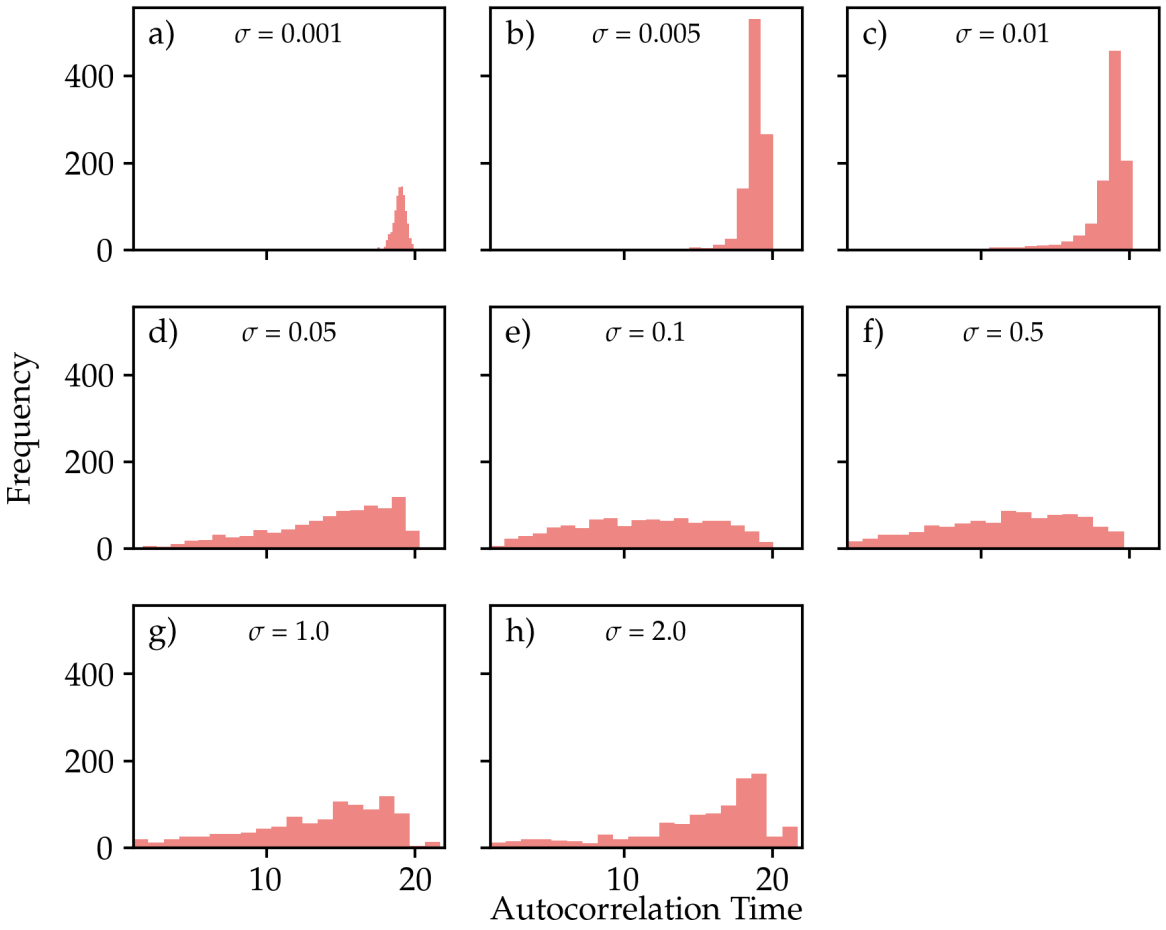


Figure A.3: Autocorrelation time distributions for various σ values. The behavior of MC trajectories on the Müller Brown potential is studied with the autocorrelation time distribution plotted for $\sigma = [2, 10^{-1}, 5 \times 10^{-1}, 10^{-2}, 5 \times 10^{-2}, 10^{-3}, 5 \times 10^{-1}, 10^{-4}]$.