Norges miljø- og biovitenskapelige universitet

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology

Training machine learning force fields for simulations of a hybrid organic-inorganic perovskite system

Helge Helø Klemetsdal
Environmental Physics and Renewable energy

Acknowledgements

This master's thesis concludes six eventful years as a student at NMBU. The years have broadened my knowledge and interests in the fields of physics, informatics, and machine learning, and I am truly inspired to learn more within these fields moving forward.

Firstly, I want to forward a big thank you to my supervisor Kristian Berland for his help during my work on this thesis. You are immensely creative, and the ideas and help you have given me have made this work a great and challenging experience, from which I have learned a lot. I also want to thank my co-supervisor Elin Dypvik Sødahl for the great discussions, general help, and proofreading. Also thanks to Rasmus André Tranås for creating data and providing me with the necessary help needed in the beginning phases of the work.

I also want to thank my mother, father, and brother for their continuous support throughout the years. This thesis would not have been possible without you. Finally, a big thanks to my friends and fellow students at NMBU. You have made the years incredibly fun, which gave me motivation to work, even in times of lockdown.

Ås, June 14, 2023
Helge Helø Klemetsdal

Abstract

Machine learning force fields (MLFF) have become gradually more popular within the field of material science as of late. Especially within molecular dynamics (MD) simulations have MLFFs seen prominent results, with both accuracy and efficiency comparable to traditional methods. In this study, a MLFF software called NeuralIL has been used to calculate the interatomic forces of a hybrid organic-inorganic perovskite material (DMMgF). Various NeuralIL architectures were explored, and several models showing promising results were selected for flexible cell MD simulations in the functional code Jax-MD. Finally, the accuracy of the NeuralIL-based MD was assessed by investigating whether the simulations could reproduce the phase transition of DMMgF in accordance with experimental data.

The interatomic force calculations provided by NeuralIL showcased the model's high capability to reproduce ab initio levels of accuracy. A mean absolute error of 0.020 eV/Å was the lowest seen in test sets that had configurations with ground truth forces calculated from density functional theory.

From a variety of NeuralIL architectures explored, a selection was chosen for Jax-MD simulations. Stability in the volume fluctuations was achieved for a single model, with the rest crashing or showing un-physical results. The results shed light on challenges related to training data and overfitting when using MLFF in MD simulations.

The phase transition of DMMgF was not shown in accordance with experimental data, although indications of structural changes concurrent with expectations were evident in some simulations. Possible weaknesses in the methodology are discussed as reasons, with special emphasis on the diversity of the training data.

Sammendrag

Maskinlærte kraftfelter (MLFF) har blitt stadig mer populære i materialteknologi den siste tiden. Spesielt innenfor molekylær dynamikk (MD) simuleringer har MLFFer sett gode resultater, der metodene har hatt både sammenlignbar nøyaktighet og effektivitet med tradisjonelle metoder. I dette studiet har en MLFF programvare kallt NeuralIL blitt brukt til å predikere interatomære krefter på et hybrid-organisk-inorganisk-perovskitt material (DMMgF). Forskjellige NeuralIL arkitekturer ble utforsket, hvor spesifikke modeller som viste antydninger til gode resultater ble valgt for simuleringer i den funksjonale koden Jax-MD med fleksible celler. Til slutt ble nøyaktigheten til NeuralIL-baserte MD vurdert ved å undersøke om simuleringene kunne reprodusere en faseovergang i DMMgF som stemte overens med experimentelle data.

De beregnede interatomiske kraftprediksjonene viste at NeuralIL har høy evne til å reproducere ab initio nøyaktighet. Gjennomsnittlig absolutte feilverdier på 0.020 eV/Å ble sett i testsett med konfigurasjoner som hadde sannhetsverdier beregnet av tetthetsfunksjonalteori.

Fra et utvalg av NeuralIL-arkitekturer ble det valgt ut en rekke modeller for Jax-MD simuleringer. Stabilitet i volumfluktuasjonene ble oppnådd for én modell, mens de resterende kræsjet eller viste ufysiske resultater . Resultatene belyser utfordringer knyttet til treningsdata og overtilpasning ved bruk av MLFF i MD-simuleringer.

Faseovergangen til DMMgF ble ikke vist i samsvar med eksperimentelle data, selv om indikasjoner på strukturelle endringer som samsvarte med forventninger var tydelige i noen simuleringer. Mulige svakheter i metoden ble diskutert som årsaker, med særlig vekt på variasjonen i treningsdataene.

Contents

Nomenclature

DFT Density functional theory

$DMMgF$ Hybrid organic-inorganic-perovskite system

MAE Mean absolute error

MD Molecular dynamics

$MLFF$ Machine learning force field

NN Neural Network

NPT Isobaric-isothermic ensemble

NVE Microcanonical ensemble

NVT Canonical ensemble

$ResNet$ Residual neural network

$RMSE$ Root mean square error

SBD Spherical bessel descriptor

VJP Vector Jacobian Product

Chapter 1

Introduction

Materials have been essential to human life in all stages of our history. In fact, most of humanity's technological advancement can be connected to the discovery and application of newfound materials [1], as is evident by the terms Bronze Age and Iron Age. Today, the need for new and better materials is no less important than before. Many modern technologies, like computers and phones, rely on materials with specific electronic properties. Another example is the semiconductor materials used in solar panels, where improvements and new material combinations have steadily increased their efficiency over the last decades [2]. With an increase in efficiency, the cost of solar panels has also dropped significantly [2]. Other fields, like refrigeration, has huge problems with global warming potential through the use of hydrofluorocarbons [3]. Replacing these materials with climate-friendly alternatives is vital, and barocalorics is a front-runner in this field [4]. Regardless of advancement in research of climate-friendly materials, the world's energy production is dominated by fossil fuels, by as much as 82.3% in 2021 [5], and hydrofluorcarbons are still used in refrigeration [3]. The resulting greenhouse emissions come with devastating effects, as seen in the recent IPCC report [6]. Further advancements are still needed, and effective methods for finding materials with favorable properties lie central to achieving them.

Ensuring that a material is a good fit for an application is no simple task. As the field of material science has seen much development in the 20th century, most materials used are already refined with regards to their field of use [1]. Finding candidate materials that are a better fit requires a detailed study of their properties. One way to gain insight into material properties is by simulating how the atoms of a material move in time, called *molecular dynamics* (MD). MD simulations are based on a numerical scheme, updating the positions of the atoms stepwise through an iterative algorithm. To be able to move the atoms in each step, the forces that the atoms experience have to be calculated [7]. A usual way to do this is by the use of *density functional theory* (DFT).

DFT is a quantum mechanical method for calculating the electronic structure properties of materials. DFT calculations are approximations of the many-body Schrödinger equation. Although accurate approximations can be achieved with DFT, the accuracy can come with significant cost [8]. Due to this cost, MD simulation with accurate DFT computed forces is often regarded as costly compared to other MD methods [9]. Less costly alternatives are necessary, especially should larger and longer simulations be computationally feasible.

One way to get a less costly MD simulation than with DFT computed forces is through *classical-MD*. Classical-MD uses forces calculated by specifically parameterized analytical expressions, named *classical force fields*. The parameters of the force fields need to be tuned based on knowledge of the material's chemistry for the force fields to be accurate [10]. Force calculations from the force fields are less costly compared to that of DFT. However, developing force fields that accurately calculate the forces for a material can be challenging due to the complexity of describing its inter-atomic chemistry, often rendering the parametrizations inaccurate or nontransferable to other systems [10]. A better solution is needed, that avoids the difficult development of the classical force fields in classical-MD, and the large cost associated with DFT-based MD.

In recent years, machine learning has become a new way to calculate interatomic forces, giving way to machine learning force fields (MLFF). MLFFs give a way to learn the statistical relation between the material and the forces experienced by the atoms without needing much pre-conceived knowledge about the material's chemistry [11]. This avoids the parametrization challenges associated with classical force fields while providing force calculations at a higher efficiency than with DFT. Given these advantages, the question is whether MLFFs reproduce the accuracy of ab initio calculations made by DFT, and if the forces can be used to perform MD simulations with comparable accuracy to the traditional methods. Many studies have already explored the use of MLFFs, where kernel-based Gaussian approximation [12] [13][14], and deep learning using neural networks (NN) [15][16][17] have shown promising results.

In this study, NeuralIL [16] will be used to compute the forces on the hybrid organic-inorganic perovskite system,$(CH3)_2NH_2Mg(HCOO)_3$ [18], hereafter referred to as *DMMgF*. Due to its characteristic phase transition, DMMgF has a high barocaloric effect, making it potential material for use within cooling applications like refrigeration [18]. Figure 1.1 shows the DMMgF structure used in this study.

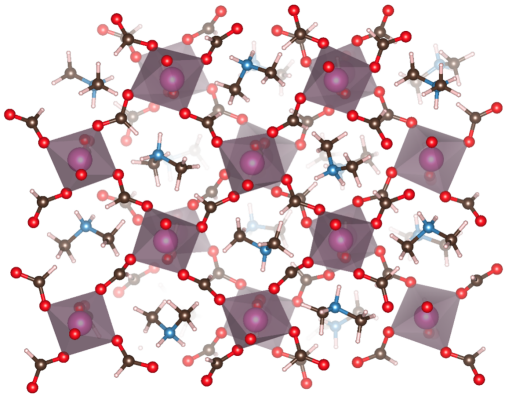


Figure 1.1: Illustration of the perovskite system, made by Kristian Berland. Mg atoms are shown in pink, carbons in brown, nitrogen in blue, oxygen in red, and hydrogen in white.

There are three main goals of this study using NeuralIL and DMMgF.

- Compare NeuralIL computed forces with first principle force calculations made by DFT.
- Explore the possibility of performing stable MD simulations by use of NeuralIL computed

forces.

- Investigate a range of simulation temperatures for the DMMgF system using NeuralIL-based MD. The simulations will be inspected to see if NerualIL-based MD can reproduce the characteristic phase transition as shown in a study by Szafrański et. al. [18]. In their study, the authors found an experimental phase transition from the low-temperature phase to the high-temperature phase of DMMgF occurring at temperatures between 262K and 264K. This thesis will investigate temperatures around this region.

The following chapter will provide a thorough explanation of the theory behind NeuralIL and the data that is fed to in training. A method chapter will then be provided, explaining how NeuralIL is trained to predict forces, and how the MD-simulations are performed with trained models. The results will be presented with a series of figures before they are discussed. Finally, the work will be briefly concluded.

Chapter 2

Theory

This chapter provides the fundamental theory needed to understand NeuralIL and its usage in MD simulations. Firstly, some context about the forces that NeuralIL predicts will be presented. The principles behind molecular dynamics (MD) simulations will follow, before an explanation of how atomic configurations from MD trajectories can be converted to numerical data for the machine learning force field (MLFF) to train on. Then, general machine learning and neural network (NN) theory will be introduced before NeuralIL is presented in light of this theory.

2.1 Forces in atomic configurations

Performing MD simulations of materials requires the atoms of the material to be placed in an *atomic configuration*. Atomic configurations have a finite collection of atoms, where the atom's placements and species are defined by the spatial extent of a *unit cell*. Given a temperature above absolute zero, each atom in an atomic configuration moves and interacts with its neighboring atoms. The interactions result in all atoms experiencing a force, determined by interactions like the electrostatic forces, bonds, angles, distances between atoms, and long-range interactions like van der Waals interactions Furthermore, these interactions define the configurations *interatomic potential*, also thought of as the expressions that define the configuration's total potential energy [19]. This interatomic potential can be can generally be expressed as a sum of all pairwise interactions [19] as in (2.1).

$$U_{\text{total}} = \sum_{i<j} U(r_{ij}). \tag{2.1}$$

Here, $U(r_{ij})$ is the potential energy from interactions between an atom i and j. Each force experienced by every atom in the configuration can be calculated as the partial derivative of the interatomic potential, with regard to the individual atom positions, as expressed in (2.2).

$$F_j = -\frac{\partial U_{\text{total}}}{\partial r_j}. \tag{2.2}$$

Here, r_j is the Cartesian coordinates for an arbitrary atom j within the atomic configuration. Computing the forces on the atoms within a configuration is dependent on a calculation of the configuration's interatomic potential. This potential energy field can be calculated in several ways, and one of the most frequently used is density functional theory (DFT).

2.1.1 Density functional theory

DFT is a popular quantum-mechanical numerical method, commonly used in materials science and chemistry to calculate the electronic structure properties of materials. DFT calculations are based on the fact that the ground state energy of a system can be expressed as a functional of the system's electron density [8, p 11]. While a general functional is not known, an approximation to the ground state energy is found by variation of the electron density until an energy minimum is reached [8, p 11]. As with other optimization problems, the calculation has to be stopped at the point where the solutions are self-consistent, meaning further iterations not improving the result significantly. How to chose this stopping criterion results in a trade-off between accuracy and computational cost. In short, there are two main outtakes to keep in mind when using DFT to calculate the forces on individual atoms in a configuration:

- Computing forces with DFT do not require extensive parameterization and can give accurate results due to its rigorous treatment of materials' electronic properties.
- DFT calculations contain a trade-off between numerical accuracy and computational cost as a result of it being an approximate solution to the many-body Schrödinger equation. An accurate calculation can therefore be significantly costly for complex materials.

2.1.2 Classical force fields

Computing the interatomic potential in equation (2.1) for any material without using the numerical scheme of DFT requires equations specifically based on the interactions between the atoms in the material. Equations developed from this knowledge are often called *classical force fields*, and can routinely be used for computation of forces in an atomic configuration. An example of a simple classical force field is the *Lennard-Jones potential* [20], which can be expressed as

$$U(r) = \frac{A}{r^{12}} - \frac{B}{r^6}. \tag{2.3}$$

The potential describes the interaction between two neighboring atoms within a configuration, separated by a distance r, with parameters A and B. Using the Lennard-Jones potential would require a fitting of these parameters with respect to the material in question, often based on knowledge about interactions in the material. Since most materials contain different chemical structures, different parametrizations are needed for the interatomic potential to accurately describe the energy interactions within the material's atomic environment [10]. Furthermore, relatively simple potentials like the Lennard-Jones potential might not be accurate enough for more complex materials structures. While many different classical force fields have been developed, many lack specific parametrizations or are just inaccurate for certain materials [10]. An example can be seen in a NN-backed MD study of hafnia (HfO_2) by Sebastian Bichelmeier [21]. Here, Bichelmeier found that existing classical force field parametrizations for the material were sparse, where the few that existed were inaccurate, or specifically tuned towards the amorphous phase of the material [21]. Avoiding the work connected to re-parametrizing these force fields accurately, Bichelmaier instead chose to use MLFFs. In this study, we chose the same approach.

2.1.3 Machine learning force fields

The key idea of a MLFF is learning the general relationship between a data set of atomic configurations, and the corresponding properties to predict, like the forces and energies of

the atoms. Since MLFF algorithms implicitly learn the statistical relations in the data, they avoid the potentially difficult parametrization problem of classical force fields. Furthermore, they have significant advantages in efficiency compared to DFT, where an accurate calculation comes at the expense of cost. MLFFs seek to bridge the gap between inaccurate classical force fields and costly DFT calculations of atomic forces, providing a more computationally efficient alternative for performing MD simulations while maintaining a reasonable level of accuracy [11].

2.2 Molecular Dynamics

How molecules and atoms move within the atomic configurations can provide important insight into the properties of a material. The simulation of these movements with MD has become increasingly popular in various fields of material science, following the gradual increase in available computational resources [7, p. 96]. By definition, MD is an iterative method to simulate the trajectories of individual atoms numerically through time. The different atomic configurations seen in the time steps of this numerical simulation provide the basis for the numerical data that the MLFF model needs to train on in order to predict atomic forces. This section will explain the fundamental methods of how MD simulations are performed and uses *Statistical Mechanics: Theory and Molecular Simulation* [7] by Mark Tuckerman, and *Density functional theory: A practical introduction* [8] by Sholl and Steckel as references, if not else stated.

2.2.1 Simulation environments

When simulating the movements of a microscopical system of atoms, some important assumptions have to be made about the simulation environment. Firstly, it's central to keep some thermodynamic variables of your system under control. Typically, these control variables are the number of atoms N, and some thermodynamic variables, like the volume V, and energy E. Which of the control variables are held fixed characterizes the *ensemble*, determining the thermodynamic state of the system. Figures 2.1(a)-2.1(c) shows the microcanonical (NVE), canonical (NVT), and isobaric-isothermic ensemble (NPT).

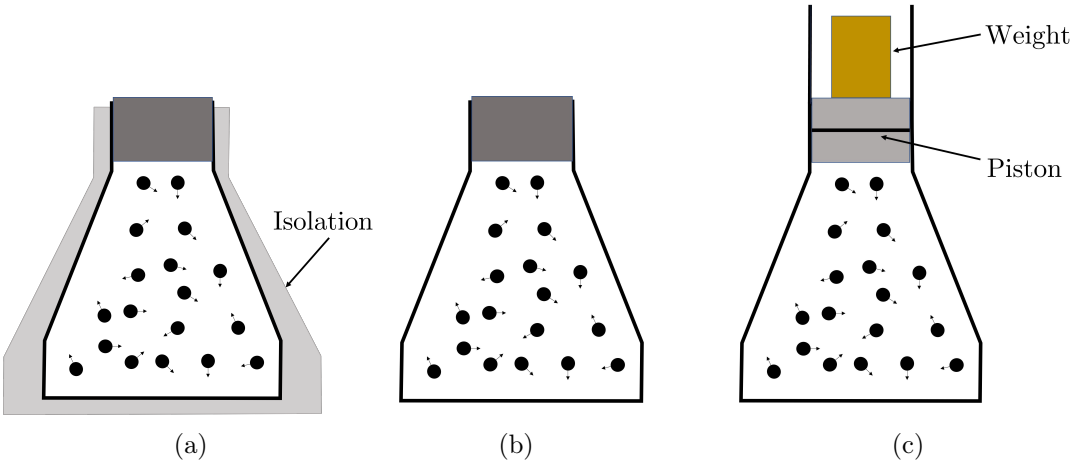


Figure 2.1: (a) Illustration of the NVE ensemble holding the number of atoms (N), volume (V), and energy (E) constant. An isolation layer is drawn around the ensemble illustrating the heat exchange to be zero in the ensemble. (b) Illustration of the NVT ensemble holding the number of atoms (N), Volume (V), and temperature (T) constant. (c) Illustration of the NPT ensemble holding the number of atoms (N), pressure (P), and temperature (T) constant. The illustration depicts the pressure being held constant by a piston and weight, and letting the volume vary.

The ensembles shown in figure 2.1 have the number of atoms(N), and either the volume (V), energy (E), temperature (T), or pressure (P) held constant, uniquely defining the ensemble names. Each of the ensembles has its advantages with respect to the purpose of the simulation, and which properties are being analyzed. Careful consideration of the ensemble is therefore an important aspect to consider before running an MD simulation.

In addition to the ensemble, there are other central parameters that can be set for an MD simulation. One of the more important of these is the lattice constant, defining the spacing between the atoms in the configuration. A larger lattice constant corresponds to a larger volume in the unit cell, directly affecting the cell environment. The number of atoms used in the simulation is also central, defined by the *supercell* used in the simulation [Chapter. 2.1] [8]. In NVE and NVT simulation, the lattice constants are set, while in the NPT ensemble, they can vary along with the volume.

2.2.2 Equations of motion

Updating the positions and velocities of atoms iteratively requires information about the forces that act on them in a given timestep. Treating the nuclei classically, a set of differential equations of motion can be derived from classical mechanics. Beginning with Newton's second law, the force on a nucleus j is

$$F_j = m_j a_j = m_j \frac{dv_j}{dt}, \tag{2.4}$$

where m_j and a_j are the mass and acceleration of the nucleus, respectively. The acceleration can also be expressed as the derivative of the velocity v_j with respect to time, resulting in the last part in equation (2.4). Combining equation (2.4) and the previously defined equation for

the interatomic forces in (2.2), one gets

$$\frac{dv_j}{dt} = -\frac{1}{m_j}\frac{\partial U(\mathbf{R})}{\partial r_j}. \tag{2.5}$$

Here, $\mathbf{R}$ is the vector of all atomic positions in the atomic configuration. Writing the velocities of the nuclei as the derivative of their position with respect to time gives

$$\frac{dr_j}{dt} = v_j. \tag{2.6}$$

Equation (2.5) and (2.6) are the equations of motion for nuclei j in the NVE ensemble. For the NVT and NPT ensemble, additional terms are added to these equations to ensure the thermodynamic control variables of the system are held constant. In the NVT ensemble, this is done through the use of thermostats, which adjusts the velocities of the atoms, thereby keeping the temperature of the ensemble constant [8, p. 197]. Similarly, a barostat can be utilized in the NPT ensemble, letting the volume vary keeping the average internal pressure equal to the applied external pressure [7, p. 233]. Regardless of which ensemble is used in the simulation, the equations of motion constitute a set of differential equations for each atom in the configuration, which can be solved numerically.

2.2.3 The Verlet algoritm

Developing a numerical method for solving the equations of motion starts with a Taylor series with respect to the positions, velocities, and accelerations to solve for the atomic positions at time $t + \Delta t$:

$$r_j(t + \Delta t) \approx r_j(t) + \frac{dr_j}{dt}\Delta t + \frac{d^2 r_j}{dt^2}\Delta t^2 \tag{2.7}$$

where Δt is the step size. All higher-order terms have been left out of this equation as they will be canceled out in a later step of the derivation. Using equation (2.4) and (2.5) rewrites (2.7) to (2.8).

$$r_j(t + \Delta t) \approx r_j(t) + v_j(t)\Delta t + \frac{\Delta t^2}{2m_j}F_j(t) \tag{2.8}$$

Expressing equation (2.8) with a negative timestep $r(t - \Delta t)$ gives (2.9).

$$r_j(t - \Delta t) \approx r_j(t) - v_j(t)\Delta t + \frac{\Delta t^2}{2m_j}F_j(t) \tag{2.9}$$

Summing the positive and negative timestep gives (2.10).

$$r_j(t + \Delta t) = 2r_j(t) - r_j(t - \Delta t) + \frac{\Delta t^2}{m_j}F_j(t) \tag{2.10}$$

The velocities for any step in the simulation can then be extracted from equation (2.10) resulting in (2.11).

$$v_j(t + \Delta t) = v_j(t) + \frac{\Delta t}{2m_j}\left[F_j(t) + F_j(t + \Delta t)\right] \tag{2.11}$$

This is known as *the velocity Verlet algorithm*, from where the positions of the atoms can be updated, given a set of initial conditions for the positions and velocities. The resulting list of calculated coordinates for each timestep, usually referred to as a *trajectory*, describes the movements of all the atoms of the configuration over the simulation time. As seen by the velocity

Verlet algorithm, the forces are a function of the timestep t, meaning it has to be computed for each iterative step in the MD simulation.

How the forces are computed determines the category of MD simulation used, where DFT-based MD and classical-MD are seen as traditional methods. Referring to the section on DFT (2.1.1), DFT-based MD would need a self-consistent DFT calculation in each step of an MD simulation to compute the forces. As a result, the DFT-based MD becomes a nested loop of numerical calculations, and can quickly become very costly [9]. In contrast to this, classical MD provides an almost instantaneous calculation of forces but can lack accuracy. Using MLFF gives way to MLFF-based MD, a simulation possibly containing both the speed of classical-MD with force calculations of comparable accuracy to DFT. Performing MLFF-based MD requires a MLFF to be trained. The next section will explain how a MLFF can be trained on a data set of atomic configurations using atomic-centered descriptors.

2.3 Training data

2.3.1 Atomic centered descriptors

In general, machine learning models need to be trained on numerical data. For a MLFF this implies that the different atomic configurations the machine should predict forces on have to be represented by numbers. The question is how we can effectively capture all the information in the configurations without losing relevant physical information about the system. The traditional choice would be using the global Cartesian position of the atoms. However, using the Cartesian would mean that the numerical descriptions would change if the configuration is rotated or translated in space, although the underlying properties of the system would remain unchanged.

The fact that the global Cartesian positions are sensitive to these variances makes them unsuitable for describing atomic configurations for an MLFF [22]. Furthermore, the representations are required to be invariant with regard to the atomic indices. This makes the descriptors produce similar numerical representations regardless of configurations where the structure is similar but the atoms are ordered differently.

The descriptors are ensured to be invariant by using a set of local coordinate systems, centered around each of the atoms within a configuration. These *local environments* are spheres with a defined cut-off radius, r_{cut}, defining the size of the local environments of the atoms, and the amount of neighboring atoms contained within the environment. Figure 2.2 illustrates an example of a local environment.

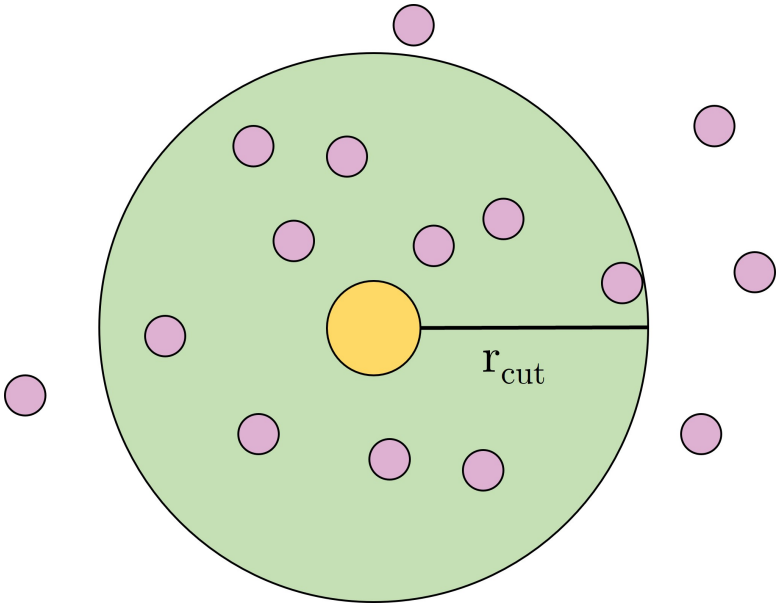


Figure 2.2: An illustration of a local environment around an atom in a configuration. The atom in yellow is placed at the center of the local environment, of defined size r_{cut}. The local environment is shown in green, the central atom in yellow, and other atoms in pink color.

The cut-off radius directly affects the number of neighboring atoms in the environments, making it a central parameter in defining numerical representations of the system. To attain the numerical representations of the local environments, mathematical functions that describe the density of neighboring atoms in the environments are used. The resulting real values are called *descriptors*, which serve as the input for an MLFF to train on. Among known descriptors in the literature, the smooth overlap of atomic positions (SOAP) [23] and Zernike descriptors [24] are commonly used, but in NeuralIL, a newer type of descriptor is chosen: The *Spherical bessel descriptors* (SBD).

2.3.2 Spherical Bessel Descriptors

The section explaining SBD will follow the approach of Kocer, Mason, and Erturk [25]. As previously mentioned, descriptors are formed by density functions from local environments wrapped around each individual atom in the configuration. The neighboring density functions of the atoms in the local environments are defined as in (2.12).

$$\rho_i(r) = \sum_j \delta(r - r_{ij}) \ , \ r < r_{cut} \tag{2.12}$$

Here the density is evaluated at a distance r away from the atom i at the center of the local environment up to the defined cut-off radius r_{cut}. The indices j denotes the neighboring atom found at a relative distance, r_{ij} away from atom i. $\delta(r - r_{ij})$ are Dirac-delta functions. The density function is projected onto a set of orthonormal basis functions on the sphere formed by the local environment shown in figure 2.2. The basis functions are defined as

$$B_{lmn}(r) = g_{n-l,l}(r) Y_l^m(\theta, \phi), \tag{2.13}$$

containing a radial part, $g_{n-l,l}(r)$, shown in figure 2.3, and an angular part $Y_l^m(\theta, \phi)$ by spherical harmonics. Examples of the spherical harmonics are shown in figure 2.6. The variables r, θ, and

ϕ are spherical coordinates. The numbers n, l, and m are the principal, angular, and magnetic quantum numbers, respectively. The quantum numbers are constrained as follows:

- $0 \leq n \leq n_{max}$
- $0 \leq l \leq n$
- $-l \leq m \leq l$

The role of the parameter $n_{\max}$ on the descriptors will be discussed in detail in section 2.3.2. The projections onto the sphere in figure 2.2 yield an expansion of the form in (2.14).

$$\rho(r) \approx \sum_{n=0}^{n_{max}} \sum_{n=0}^{n} \sum_{l=0}^{l} c_{nlm} g_{n-l,l}(r) Y_l^m(\theta, \phi). \tag{2.14}$$

The expansive coefficients

$$c_{nlm} = \sum_j g_{n-l,l}(r_{ij}) Y_l^{m*}(\theta_{ij}, \phi_{ij}), \tag{2.15}$$

are calculated from the atoms' relative spherical coordinates denoted $r_{ij}, \theta_{ij,\phi_{ij}}$. To get a complete description of the SBD, one has to know the full form of the radial part of the basis functions. Although the functions are relatively complex, the equations along with a derivation will be presented for the interested reader. See [25] and its supplementary material for more details on the derivations. The radial basis functions are built starting from a linear combination:

$$f_{nl}(r) = a_{nl} \cdot j_l \left(r \frac{u_{ln}}{r_c} \right) + b_{nl} \cdot j_l \left(r \frac{u_{l,n+1}}{rc} \right), \tag{2.16}$$

where a_{nl} and b_{nl} are constants, j_l is the l'th spherical Bessel function of the first kind, signified by its continuity at the origin. The function u_{ln} is the $(n+1)$'th root of j_l. Choosing the constants so that the radial part is twice differentiable at the cut-off radius, along with normalization leads to (2.17).

$$f_{nl}(r) = \left(\frac{1}{r_c^3} \frac{2}{u_{ln} + u_{ln+1}} \right)^{\frac{1}{2}} \times \left[\frac{u_{l,n+1}}{j_{l+1}(u_{ln})} j_l \left(r \frac{u_{ln}}{r_c} \right) - \frac{u_{ln}}{j_{l+1}(u_{l,n+1})} j_l \left(r \frac{u_{l,n+1}}{r_c} \right) \right]. \tag{2.17}$$

Executing a Gram-Schmidt orthogonalization procedure on $f_{nl}(r)$ for $0 < n < n_{max}$ gives the final radial basis functions $g_{\mathrm{n-l,l}}$. Figure 2.3 shows examples of $g_{\mathrm{n-l,l}}$ for distances up to a set cut-off radius of $r_{cut} = 3.5$ Å.

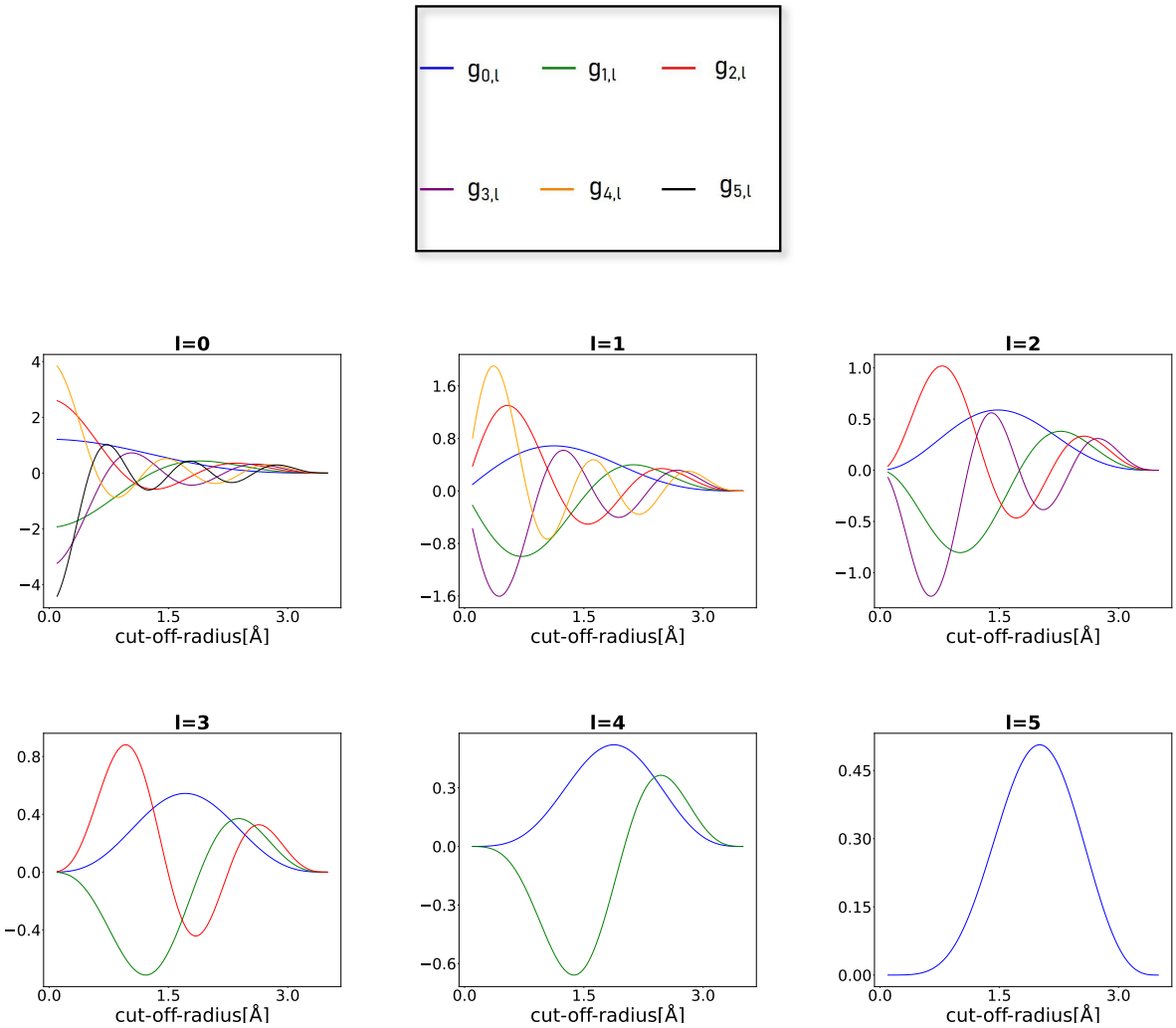


Figure 2.3: Example plots of radial basis functions $g_n - l, l$ for values of a cut-off-radius up to 3.5 Å.

The generalized power spectrum is used to extract the SBDs and is defined as in (2.18).

$$p_{nl} = \sum_{m=-l}^{l} c^*_{nlm} c_{nlm}, \tag{2.18}$$

Inserting the expressions for the expansion coefficients c_{nlm}, and utilizing the spherical harmonic addition theorem gives the final SBDs in (2.19).

$$p_{nl} = \frac{2l+1}{4\pi} \sum_j \sum_k g_{n-l,l}(r_j) g_{n-l,l}(r_k) P_l \cos(\gamma_{jk}) \tag{2.19}$$

The subscript i, for the central atom, is left out in this formula for simplicity. The subscript k, and j are neighboring atoms to the central atom, P_l is the l'th order Legendre polynomial and γ_{jk} is the angle between the three atoms. Equation (2.19) gives the real-valued SBDs that serve as training data for NeuralIL. Figures 2.4 and 2.5, show the SBDs for a selected atom of each type in the DMMgF system.

Equation (2.15) gives a set of n_p components for every atom:

$$n_p = n_B \cdot n_{el} \frac{(n_{el} + 1)}{2} \tag{2.20}$$

This number is dependent on the number of distinct elements, n_{el} in the system to be considered, and the number of Bessel functions determined by

$$n_B = (n_{\max} + 1) \cdot \frac{(n_{\max} + 2)}{2} \tag{2.21}$$

For each configuration, the resulting number of descriptors is determined by

$$n_{\text{descriptors}} = n_p \cdot n_{\text{atoms}} \tag{2.22}$$

where n_{atoms} is the number of atoms in the configuration.

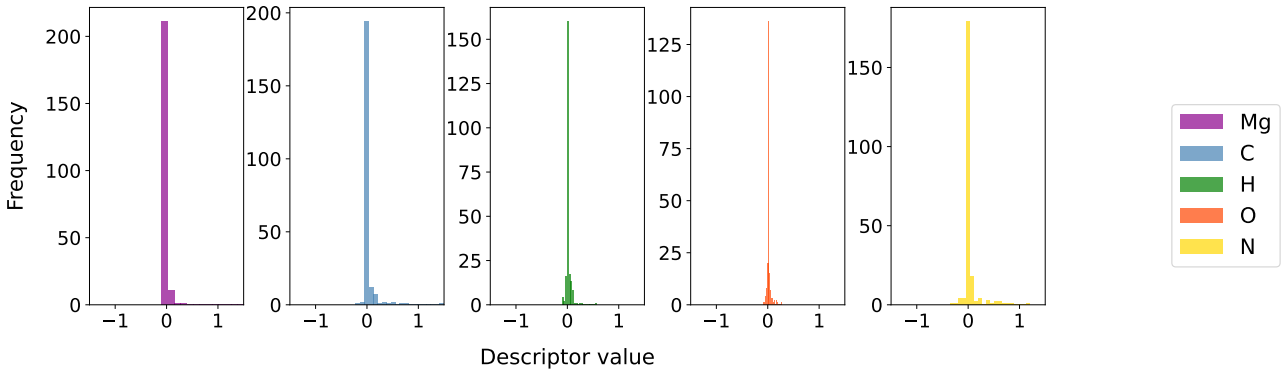


Figure 2.4: Descriptors for different atom types $r_{cut} = 3.5$ and $n_{max} = 4$ truncated to values between -1.5 and 1.5.

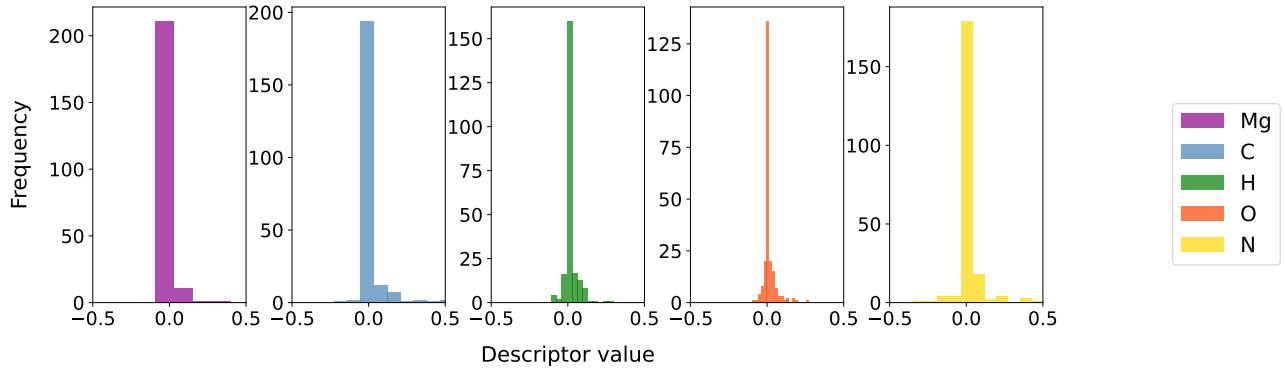


Figure 2.5: Examples of descriptors for different atom types in DMMgF. The descriptors are generated with parameters $r_{cut} = 3.5$ and $n_{max} = 4$ truncated to values between -0.5 and 0.5

As can be seen from figure 2.4 and 2.5, the descriptor vectors are sparse, and centered around 0, with a few outliers. These real-valued numbers serve as the numerical data NeuralIL is trained on.

The n_{max} parameter

The role of the parameter n_{max} is central when defining the detail of the local environments. Figure 2.6 shows the effect an increase of n has on the spherical harmonics.

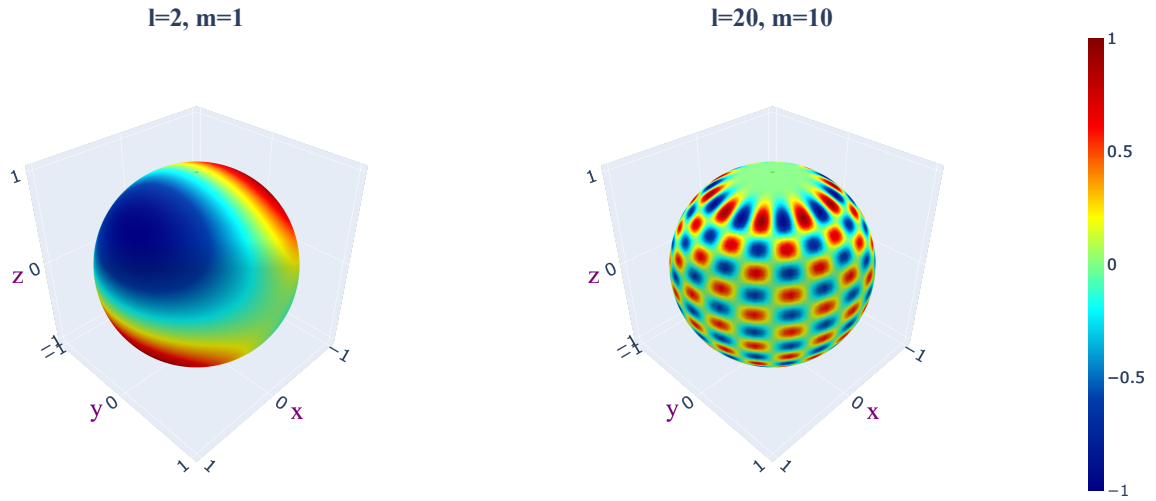


Figure 2.6: Illustration of spherical harmonics with quantum numbers l=2,m=*1*, on the left, and l=20, m=10, on the right. The spheres each show distinct areas of colored areas, corresponding to a range of different real numbers. The figures show an increase in the number of colored areas when the quantum numbers are increased.

Figure 2.6 shows how increasing the value of n and l results in a greater number of distinct colored areas on the surface, where each area takes different real-valued numbers on the sphere. In the event that certain atoms are bonded closely to each other, the descriptors might not accurately represent these atoms with their distinct values if the colored areas of the spherical harmonics are too large, as seen by the left plot in figure 2.6. If the value of n is too large, however, the detail of the description might become exaggerated, having effects on the computational cost associated with evaluating the SBDs. While it is n that defines the detail of the spherical harmonics directly, n_{max} lets the value of n take higher values, as seen by the constraints in section 2.3.2. The parameter n_{max} is therefore central when evaluating the SBDs. Moreover, letting the n_{max} be large directly increases the number of descriptors as can be seen by its quadratic relation to the amount of Bessel functions in equation (2.21). Figure 2.7 shows this relation for a DMMgF configuration of 384 atoms:

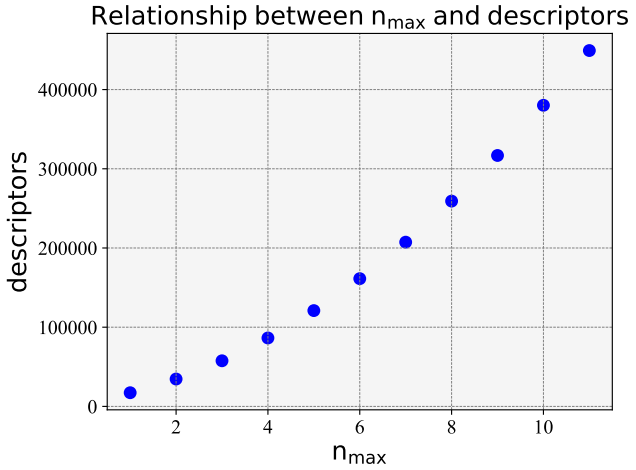


Figure 2.7: Plot of the relation between the n_{max} parameter and the number of descriptors for a DMMgF configuration containing 384 atoms.

The increase of descriptors increases the amount of numbers used to describe each atom in the local environments, having significant effects on the computational cost of the descriptors. Consequently, setting the n_{max} alongside r_{cut} should be seen as a trade-off between detail and cost, and should be given thought when training NeuralIL with SBDs.

Properties of Spherical Bessel Descriptors

The way the SBDs are constructed gives them some preferable properties compared to other descriptors in the literature. Other than the fact that the descriptors ensure rotational and translational invariance, it also has significant effects on computational cost [25]. Compared to the Gaussian density functions used to derive the SOAP descriptors, the superposition of Dirac-delta functions in eq (2.12) avoids computationally costly integrals while evaluating the descriptors [25]. In fact, the descriptors prove to be optimally complete, reducing the number of descriptors needed to a minimum, and therefore also the amount of redundant information in the numerical representations [25]. Furthermore, this does not come at a significant loss in accuracy compared to the other descriptors, making the SBD both efficient and accurate numerical representations for an MLFF to train on [25].

2.3.3 Embeddings

As the descriptors are only based on the atomic positions in the configurations, the training data lacks any information about the chemical species in the configuration. To include this, an embedding vector can be included in the training data. NeuralIL includes an embedding vector of predefined length n_{emb} that is given to the model as training data alongside the descriptors. The embedding coefficients are in this model only dependent on the chemical species of the atom located at the center of the given local environment.

2.4 Machine learning

This chapter has previously introduced methods for how training data is computed from atomic configurations. This section will lay the foundation of the machine learning theory needed to

understand how a MLFF predicts forces from this training data. The section will introduce key machine learning concepts, before explaining NN and residual neural networks (ResNets).

2.4.1 Key concepts and terminology

Before explaining the workings of NNs, it's necessary to understand some key concepts of ML. One of the most important concepts is identifying which type of machine learning problem one is working with. One of the most common branches of machine learning is supervised learning. Supervised learning is defined as all problems where the model seeks to predict a reality based on some labeled data [p. 3] [26]. Within supervised learning, the problems found can most often be categorized as either *regression* or *classification*. Classification tasks relate to the process of determining whether a set of labeled training data corresponds to a set of classes. A simple example is whether an email should be classified as spam or not. This example is shown in figure 2.8.

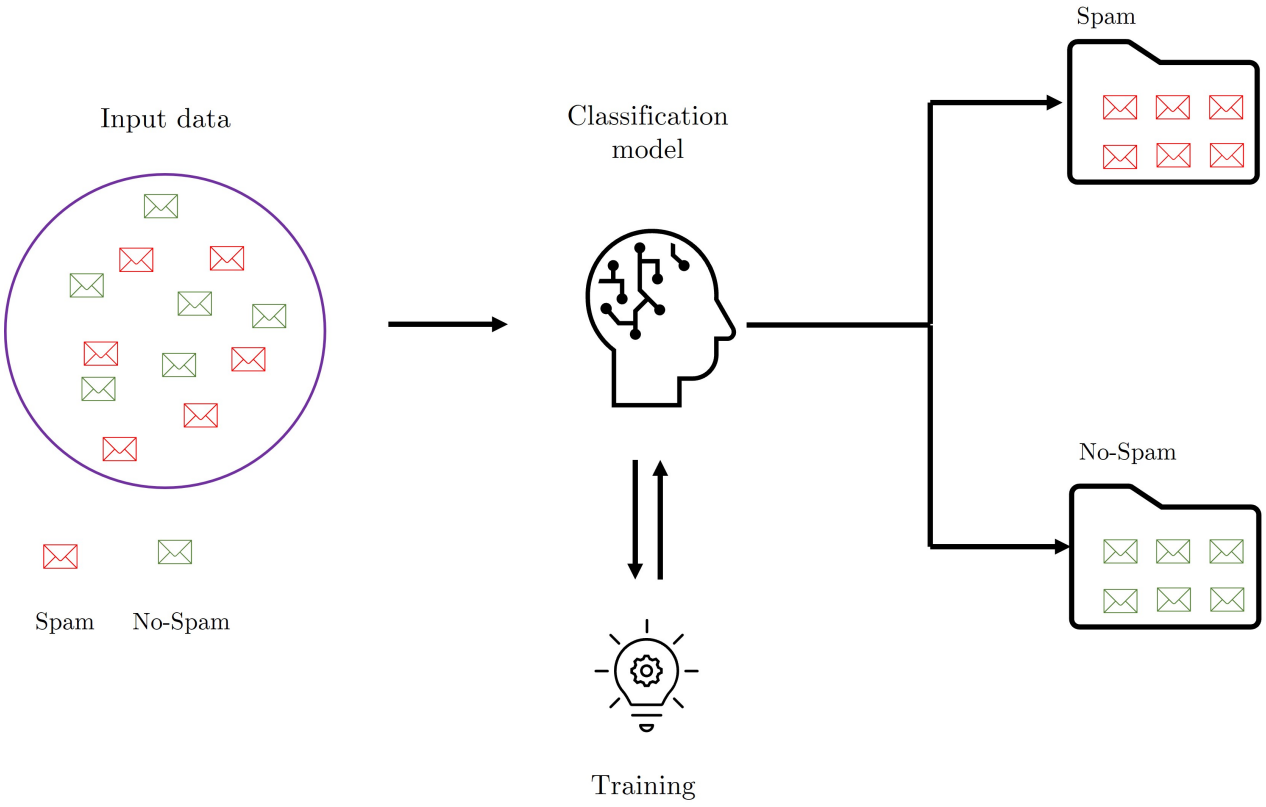


Figure 2.8: Illustration of a classification task, where a classification model is trained to determine whether an email is spam or not.

In contrast to classification, regression tasks make the model predict continuous outcomes, like the number of emails received on a single day. The example is illustrated in figure 2.9.

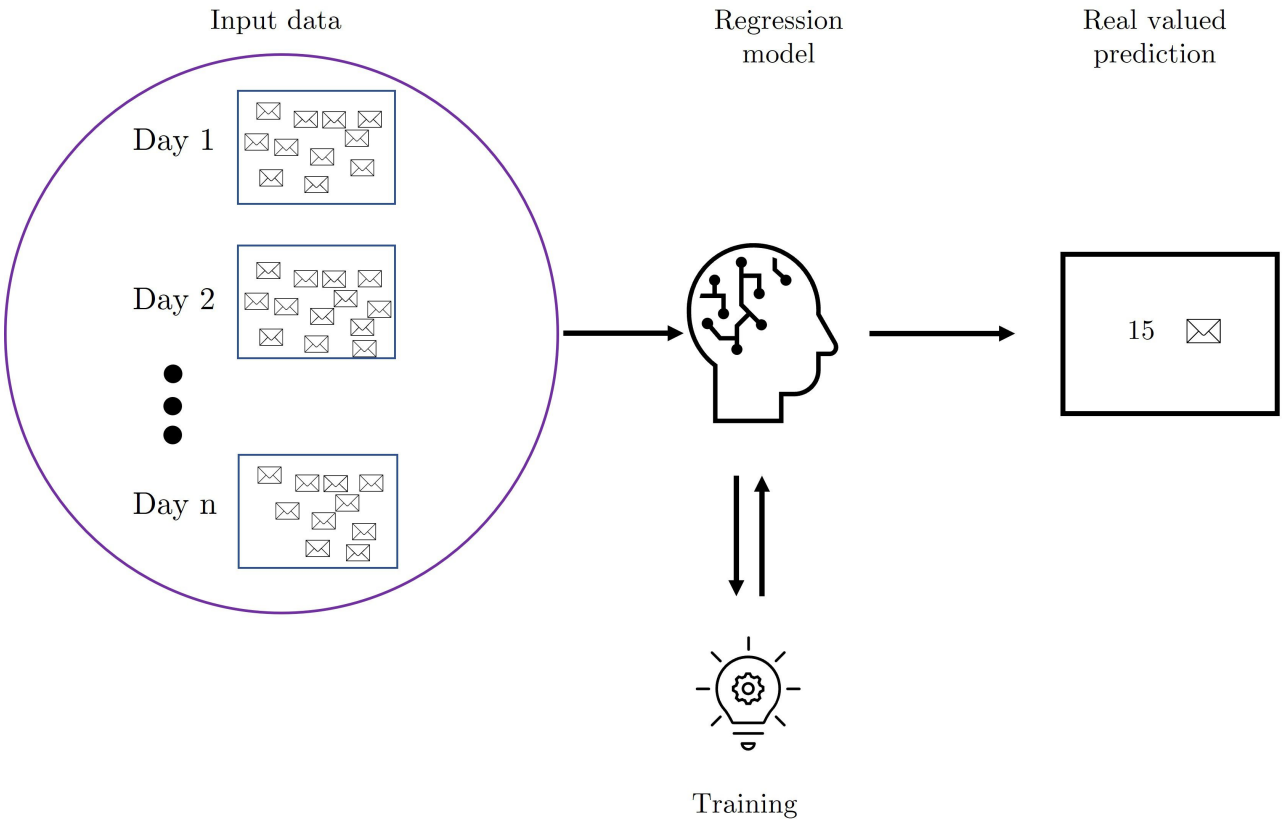


Figure 2.9: A regression task predicting the number of emails received on a specific day to be 15, based on a labeled input data set of a variable amount of received emails for n amount of days.

Central to these problems is how the input data is labeled. In the context of the regression example, the input data needs to be labeled according to what the machine should predict. These labels are often called *ground truths*. For MLFFs, the ground truths can be the atomic forces and total energies of an atomic configuration, which in this study is calculated by DFT. Therefore, our ground truths themselves have an associated error, due to DFT being an approximate solution to the many-body Schrödinger equation. The regression problem thus boils down to being able to reproduce the forces from DFT calculations, with as small an error as possible.

Other than the type of problem, there are other central concepts in machine learning and NN theory that are central to understanding how NNs operate:

- **Neurons, inputs and activations**: A neuron is a single node in a NN. Each neuron sends out an activation, based on the inputs received by other neurons. The activation output is determined by a set activation function.
- **Layer**: A layer of a NN is a column of multiple neurons working in parallel.
- **Loss**: The loss is the estimated error between a prediction from a model and the corresponding ground truth. The loss is calculated by a loss function, which can be defined

by the users. A common loss function in the literature is the mean absolute error:

$$MAE = \frac{1}{n} \sum |y_{\text{ground truth}} - y_{\text{predicted}}|. \tag{2.23}$$

Here n is the number of samples in the training data, $y_{\text{ground truth}}$ are the ground truth labels in the training data, and $y_{\text{predicted}}$ is the model prediction. For details on the workings of the loss function, see section 2.4.3.

- **Hyperparameters**: Parameters of a machine learning model that can be varied are often referred to as hyperparameters. Examples of these parameters are the number of neurons in the layers, the activation functions, and the loss function.

2.4.2 Over and underfitting

In addition to the concepts above, an important concept is under and overfitting. A machine learning algorithm can be trained for an arbitrarily long time, with a wide range of architectures and parameters. The choice of model and training time has a big effect on the model's capacity to learn the relationship between the training data and the ground truths. If the data is trained too little, it cannot replicate the truth it wants to predict, and the model is *underfitted*. Figure 2.10 shows the concept of underfitting.

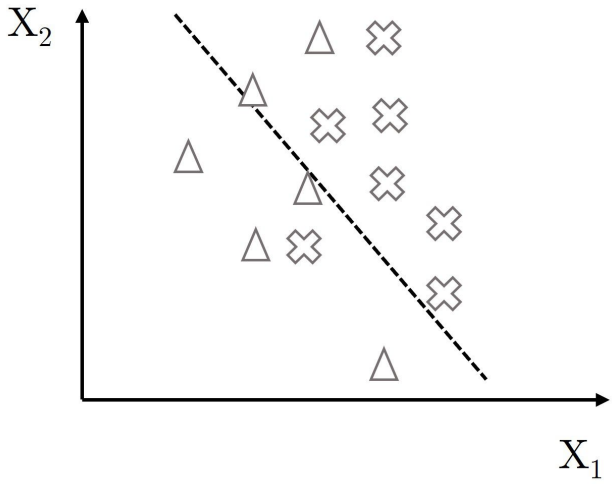


Figure 2.10: Illustration of a machine learning underfitting on a nonlinear decision boundary. The model has a high bias. Illustration adapted from figures in Raschka and Mirjalili [26, p. 76].

An underfitted model usually has a specific bias, meaning how far off a prediction is from the ground truths in general [26, p.76]. In contrast, if the model has learned the relationship too well, it will not be able to predict unseen data that contain differences from the training set, and the model is *overfitted*. Figure 2.11 shows a model that is overfitted on the training data.

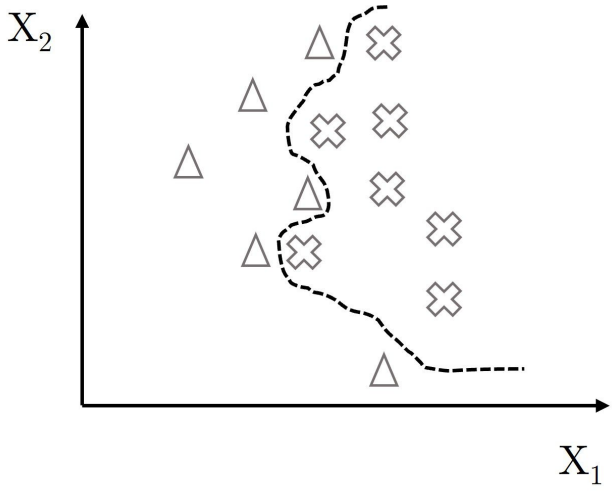


Figure 2.11: Illustration of a machine learning algorithm overfitting on a nonlinear decision boundary. The model has high variance. Illustration adapted from figures in Raschka and Mirjalili [26, p.76].

If a model has been trained in such a way, it usually is said to have high variance, as it has learned all intricate details contained in the training data. The goal of training a machine learning model is finding the correct bias-variance trade-off, giving the model optimal performance when it should generalize or extrapolate to new unseen data [26, p.76]. Figure 2.12 illustrates how such a trade-off can look in the context of the previously presented nonlinear decision boundary.

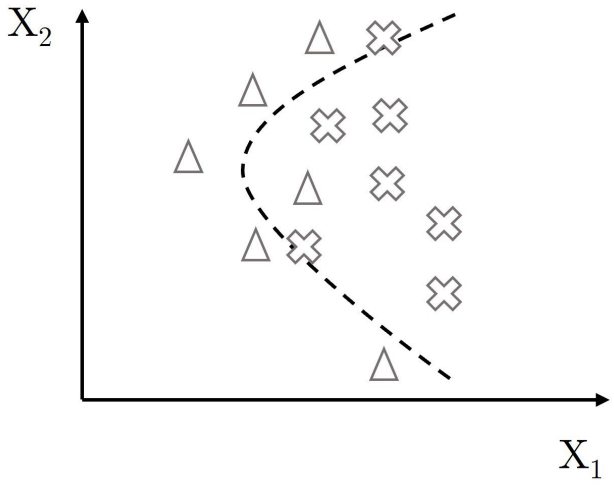


Figure 2.12: Illustration of a machine learning algorithm having a good compromise between bias and variance on a nonlinear decision boundary. Illustration adapted from figures in Raschka and Mirjalili [26, p.76].

2.4.3 Neural networks

The idea of creating artificial intelligence with NNs has existed for a long time, with the first deep learning models emerging in the 1960s [27]. Although the use of NN was constrained by

computational resources at this time, their impact on science and society has grown over time, reaching new levels of significance in the recent advance of natural language processing models like *ChatGPT* [28]. NNs are constructed by artificial neurons connected in a set of layers, mimicking the complex system of neuron connections in the human brain [26, p. 384].

Architecture

The general structure of a fully-connected NN is built up of a set of fully-connected layers, also known as dense layers. The simplest form of a fully-connected NN has the following structure [26, p. 388]:

- **The input layer**: A set of neurons taking in the numerical training data
- **Hidden layers**: A set of dense layers constituting the core of the NN. The layers usually contain variable amounts of neurons.
- **Output layer**: The final layer in a NN that generates the output prediction of the model. The layer can contain single, or multiple neurons, uniquely determined by the problem type. These neurons are usually called output neurons.

The general structure is shown in figure (2.13) with example layer widths of 5-5-3-1, corresponding to the number of neurons in each layer, respectively.

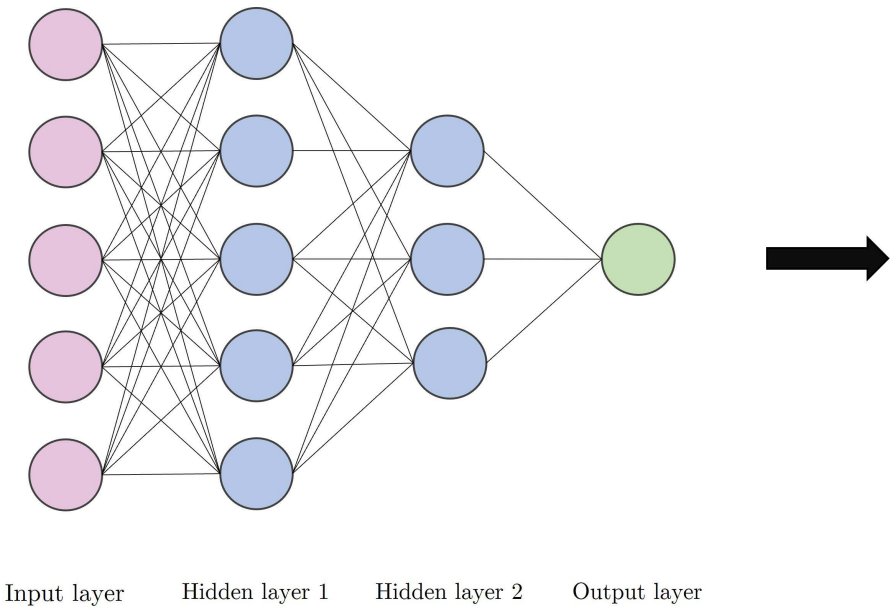


Figure 2.13: General structure of a fully-connected NN with dense layers with layer widths of 5-5-3-1 neurons, respectively.

Forward propagation

The final prediction from an output layer of a NN comes as a result of a scheme called *forward-propagation*. Forward propagation is the process by which activations are propagated from the input layer to the output layer in the NN, through weighted connections [26, p. 391]. Looking at figure 2.13, each neuron in each layer is connected to all the neurons in the next. The input received by a neuron i in a given layer, from a neuron j, can be written as

$$\text{Input}_{ji} = w_{ji} \cdot a_j \tag{2.24}$$

where w_{ji} is the weighting between neuron the neurons, and a_j is the activation from neuron j. Summing up all the inputs, a *net input* for a single neuron in a fully-connected network can be defined as a linear combination of all the inputs received from the neurons in the previous layer [26, p. 391]:

$$\text{Net input}_i = \sum_j^n \text{Input}_{ji} + b_i \tag{2.25}$$

Here, n is the number of neurons connected to neuron i from the previous layer, b_i is the associated bias for neuron i. Figure 2.14 shows the net input to an output neuron from three arbitrary neurons in a previous layer. The calculation of its activation is illustrated by an activation function O.

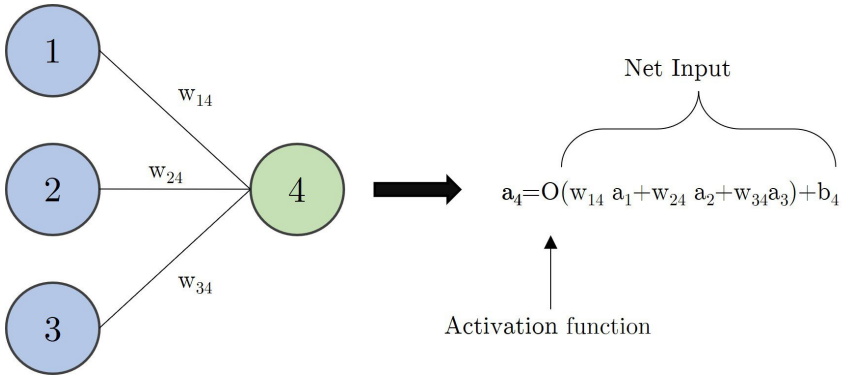


Figure 2.14: Illustration of the activation a_4 of a single neuron, given the input from three other neurons, 1,2 and 3, with activations a_1,a_2 and a_3, respectively. The activation function is illustrated by the letter O.

After the activations are calculated for all the neurons in a given layer, the process is repeated, where these new activations serve as inputs for the next layer in the network. In this sense, the signals are propagated through the network until they reach the output layer, where a final activation will serve as a prediction from the model.

Back propagation

Similar to how the neuronal connections in the human brain adapt to the information received, the weights in the connections between neurons and their biases need to be updated for the NN to predict precisely. If the weights and biases are not updated, the activations propagated through the network in the forward propagation will be non-optimal, giving an accumulation of errors, and bad model performance. The weights and biases are updated through a scheme called *back propagation* [26, p. 416-417].

The main goal for a NN used for regression is minimizing the error, meaning the difference between the model predictions and ground truths. Minimizing this estimate depends on updating the weights and biases according to the problem at hand. In short, the updates are done through the minimization of a *loss function* that determines the loss (error), between the predictions and ground truths [26, p. 391]. One popular method of minimizing the error is through a process called *gradient descent*. The gradient descent algorithm calculates the direction in space that minimizes the loss function according to the given weight. Essentially,

this means that the algorithm finds the weights that give the smallest value of the loss through a mathematical minimization approach. The process is illustrated in figure 2.15.

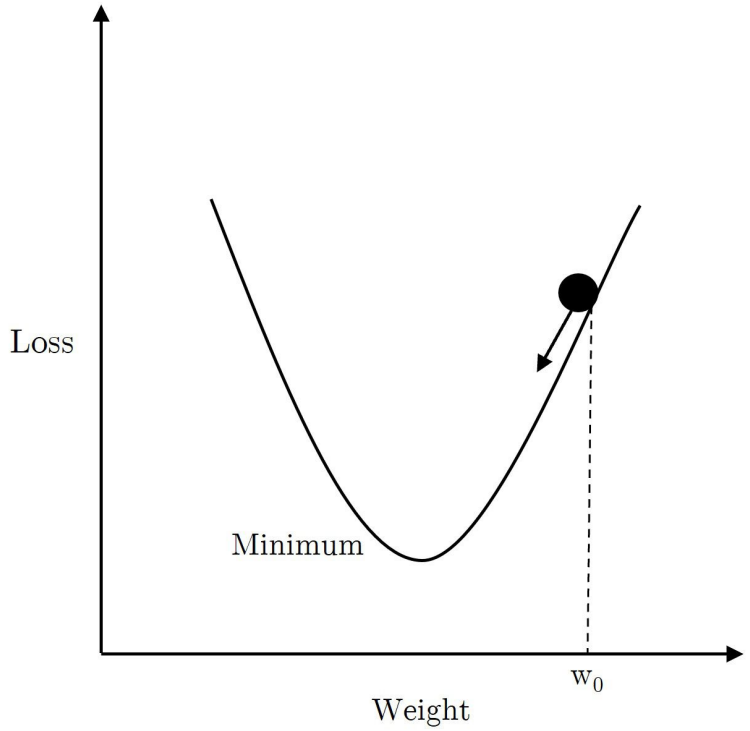


Figure 2.15: Illustration of the minimization approach. The loss function is reduced towards a minimum from a starting weight w_0. Adapted from Raschka and Mirjalili [26, p. 38].

Training

The full process of training a NN corresponds to a set of cycles with forward propagation of activations, and weight updates in back propagation. The weights need to be updated a number of times before the error function can reach its minimum, where convergence in the model training is reached. One of these cycles is called an *epoch* [26, p. 391], which is a central integer that needs to be defined when training a NN. The general scheme for training NN is visualized in figure (2.16).

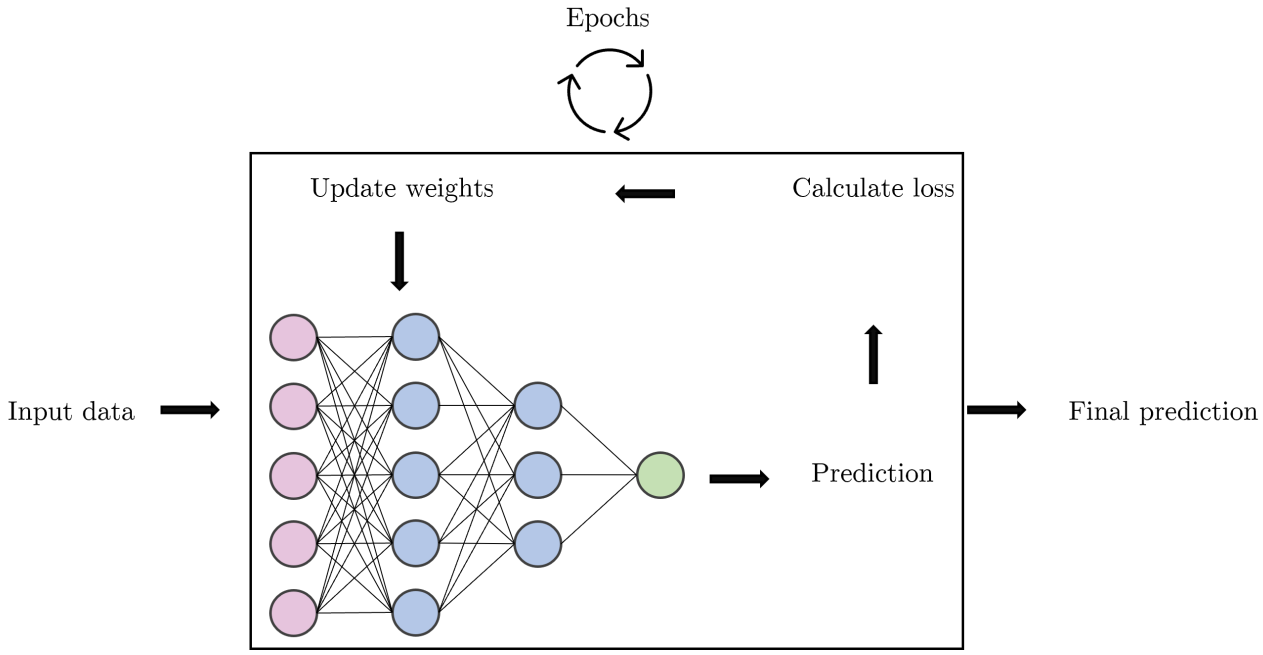


Figure 2.16: Illustration of the general training of a NN.

Another central idea of machine learning training is how the model performance is *validated*. The usual practice of validation is to hold a subset of the training data away from the training and validate the model on this subset after each epoch [26, p. 196]. By the results of this validation set, the hyperparameters of the model can be updated and optimized to achieve lower validation errors. However, there might be cases where the model performs well on the validation set but still does not generalize well to new unseen data. This comes as a consequence of possible similarities between the data and the extracted validation set, or that the model has generally learned which weights correspond to a good validation performance. To ensure good generalized performance, models are usually tested on *test sets* after training [26, p. 196]. Test sets are withheld from the training and validation of the model, and are extracted either from the same data set or a separate one. Test sets can also serve as a way to test model extrapolation, by checking if the model can have a good generalized performance on data with slightly different patterns and parameters than it was originally trained on.

Training optimization

The theory above roughly outlines the main training scheme for a NN. However, there are many improvements that are often implemented in the training that needs to be mentioned. Among many are the ones listed below.

- **Optimizers**: An optimizer is often implemented to enhance the efficiency of the weight updates in training. Essentially, the optimizers increase the performance of the gradient descent, making the NN find the optimal weights and biases quicker [29, Chapter. 8.3-8.5].
- **Mini-batching** The full set of training data is split into small batches containing a few samples of the training set. The batches are passed in a full training epoch, meaning the weights are updated after each mini-batch is passed. The process has been shown to have advantages in training and data memory [29, Chapter. 8.1.4].
- **Normalization** is done by scaling and centering inputs (activations and biases) either

along batches [30], or over a whole layer [31]. The process is shown to increase speed in training as a consequence of minimizing the variance of training inputs.

NeuralIL was originally built following the architecture of a fully-connected NN [16], however, there have been updates to the model since this version was published [32]. Due to increased performance, the model has now incorporated a ResNet architecture, with inspiration from the study of Chen et al. [33]. ResNets have proven to improve the predictive performance of deeper networks in image recognition previously [34], and the architecture has here been adapted to suit a regression problem.

2.4.4 Residual neural networks for regression

This section uses the newest NeuralIL article, along with the article introducing the ResNet architecture by Chen et. al. [33], if not else stated. Not much different from the fully-connected NN introduced earlier, the ResNets architecture introduces an important feature to the layer connections in the architecture of the model; *Skip connections*. A skip connection lets a set of layer activations bypass a number of dense hidden layers before it is summed with the activations of a later layer of choice. Looking at figure 2.17, the skip connection is visualized between the input and output layers of the block, skipping a set of two dense layers in the process. Then it is summed with the activations from the second skipped dense layer.

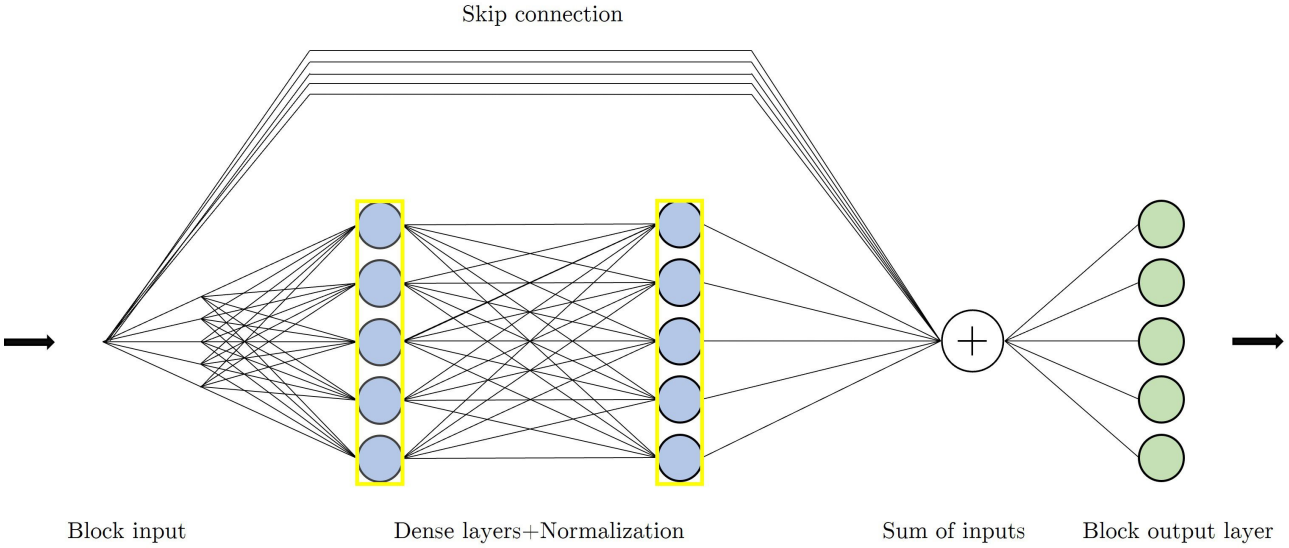


Figure 2.17: An illustration of an identity block. The block input is propagated through the network before it is summed together as an input to the block output layer. Normalization acts on the inputs to the layers, visualized with yellow borders

The number of skipped dense layers is two in figure 2.17, but can be more within the block. Normalizations of the dense layer activations are included as yellow borders around the neurons for easier visualization. The architecture shown in figure 2.17 is named an *Identity block*. Central to the identity block is that the activations sent through the skip connection contain the same dimensions as the block output layer, hence the name *identity*. In figure 2.17 the block input corresponds to activations from a 5-neuron layer, which equals the amount in the block output layer. Different from the identity block, figure 2.18 displays a *dense block* where the skip connections map a set of activations with an input dimension not equaling the dimension of

the output layer. These blocks need an additional dense layer in the skip connection to adjust for the difference in dimensions.

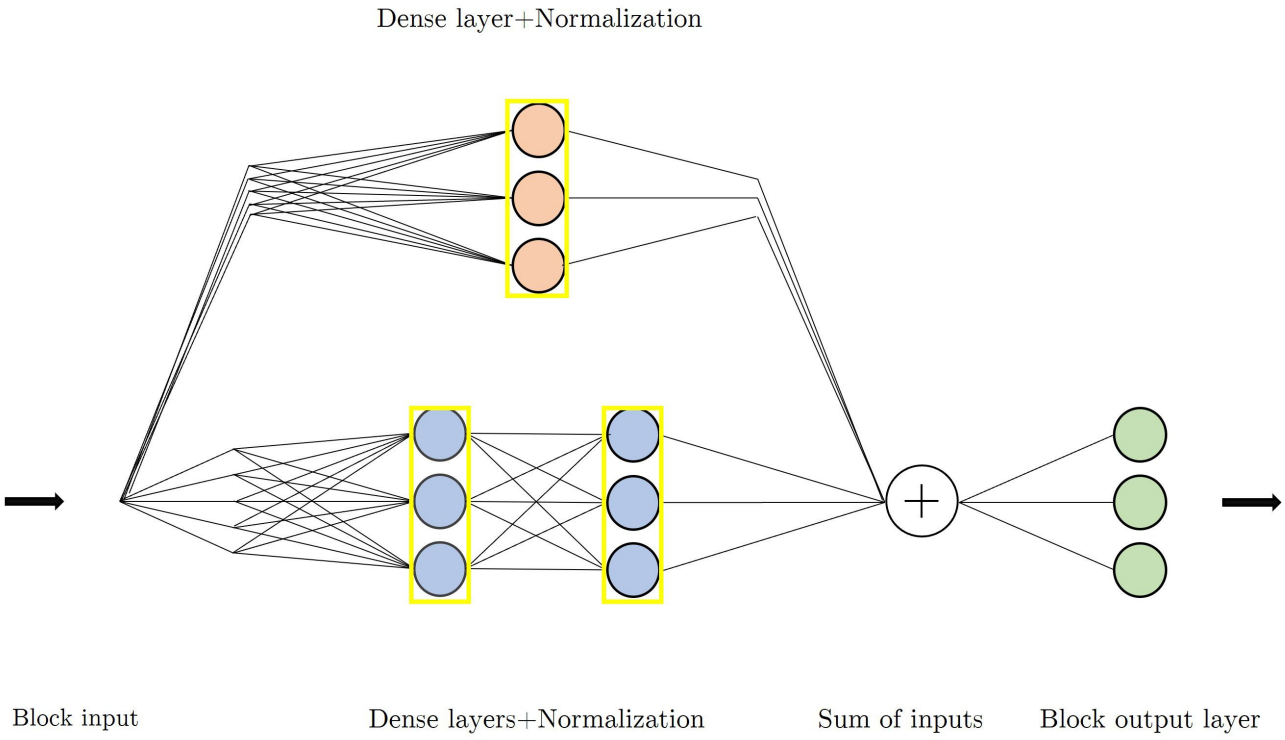


Figure 2.18: An illustration of a dense block. The block input is propagated through the network before it is summed together as an input to the block output layer. Normalization acts on the activations of the networks and is visualized with yellow borders. The skip-connection layer is visualized with orange neurons.

Figure 2.18, shows how a dense block maps a 5-dimensional input down to a 3-dimensional input layer through 3-neuron dense layers. The skip-connection needs to have a dense layer with 3 neurons as well to account for the change in dimension.

A full ResNet architecture essentially incorporates the same structure as fully-connected NNs in figure 2.13, just with the normal layers replaced by dense and identity blocks. Figure 2.19 shows an example architecture, containing arbitrary widths of 5-3-3-1 within the blocks, mapping an input of 5 dimensions down to a 1-dimensional output. These widths denote the layer widths within the blocks as explained in figures 2.17 and 2.18.

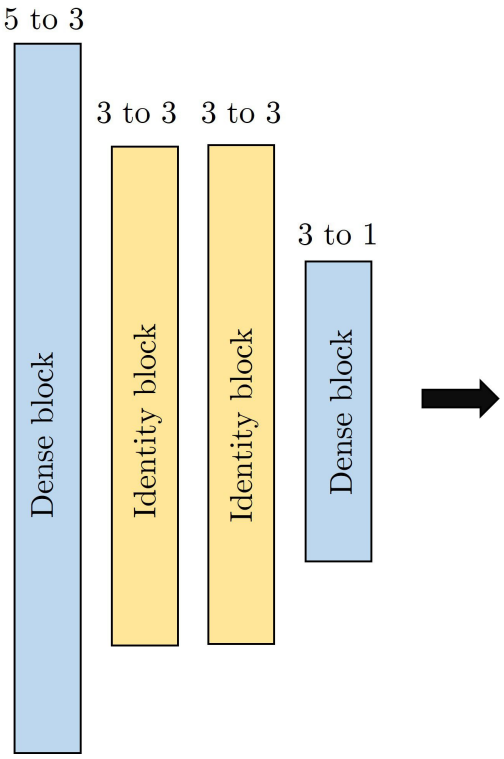


Figure 2.19: An example of a ResNet architecture with block dimensions of 5-3-3-1, corresponding to the layer widths within the blocks. The identity blocks contain layers with 3 neurons, while the dense blocks have layer widths corresponding to the output dimension. The identity blocks are shown in yellow, while the dense blocks are shown in blue.

Here, the final dense block will usually have an output layer containing a linear activation function, making the final regression prediction of the model.

2.5 NeuralIL

Originally developed for a use case on ionic liquids, NeuralIL is a MLFF software, containing a robust ResNet architecture, incorporating SBD and embeddings as training data to predict total energy and forces. Since the architecture is not specifically built for use on liquids, the model can be used on other material states. This section will explain the architecture and parameters of the model based on the original paper [16], and the newer published article [32].

Architecture

The architecture of NeuralIL contains the input of SBD and embeddings into a core ResNet model, containing a set of identity and dense blocks of chosen widths, similar to the model illustrated in figure 2.19. The core is followed by a single linear layer, constituting a prediction of single-atom energies. This layer accounts for possible offsets in the prediction, alongside getting the correct scale of the single-atom energies. The architecture is concluded with a "sum over atoms", summing the single atomic energies, resulting in the prediction of the total energy for the atomic configuration, denoted $E_{pot,predicted}$. This step essentially destroys the order of the inputs to the model and enforces the physical invariances of labeling in the model. The architecture is shown in figure 2.20.

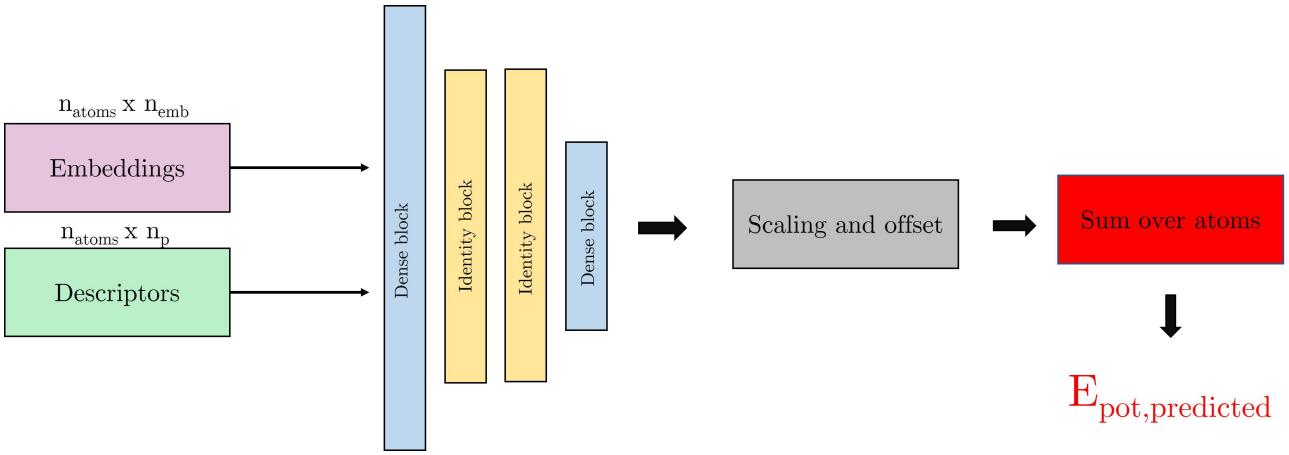


Figure 2.20: An illustration of NeuralIL predicting the interatomic potential of an atomic configuration. An input tensor consisting of embeddings and SBDs is passed to an input core Resnet model. The first input dense block has the same dimensionality as the input tensor. The activations from the core model are scaled and offset by a linear layer before the last sum over atoms layer, returning the interatomic potential $E_{pot,predicted}$.

Each block in the architecture is similar to the blocks visualized in figures 2.17 and 2.18, where the skip-connection skips two dense layers. However, their blocks also add the linear scaling layers and the sum-over-atoms layer in the blocks. The figures 2.21 and 2.22 show these slightly more complex block architectures:

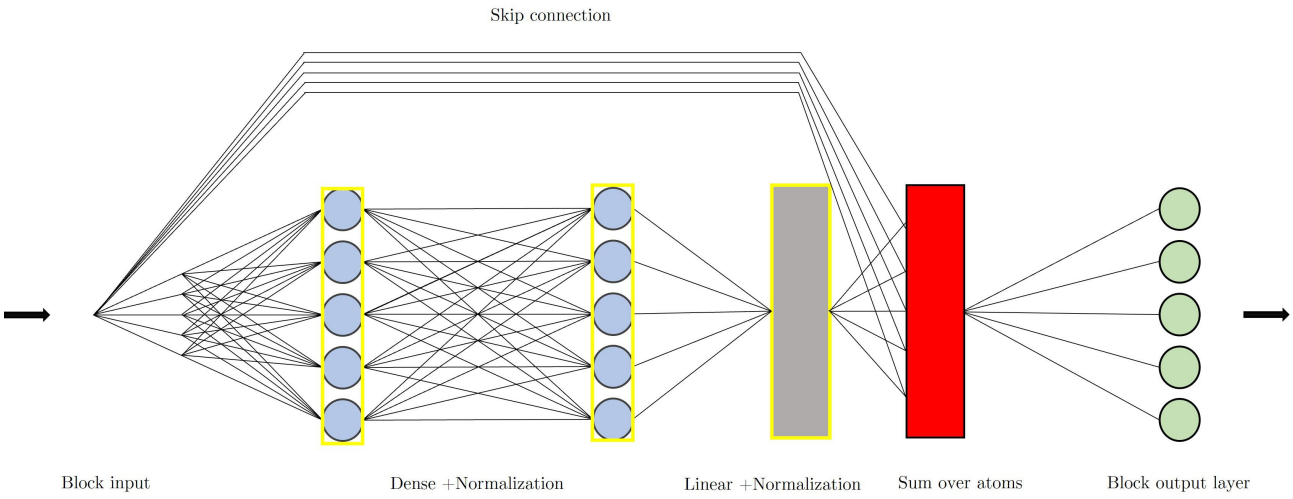


Figure 2.21: The identity block used in NeuralIL. Normalization layers covering the neuron layers, visualized in yellow, act on the net input to the respective layers.

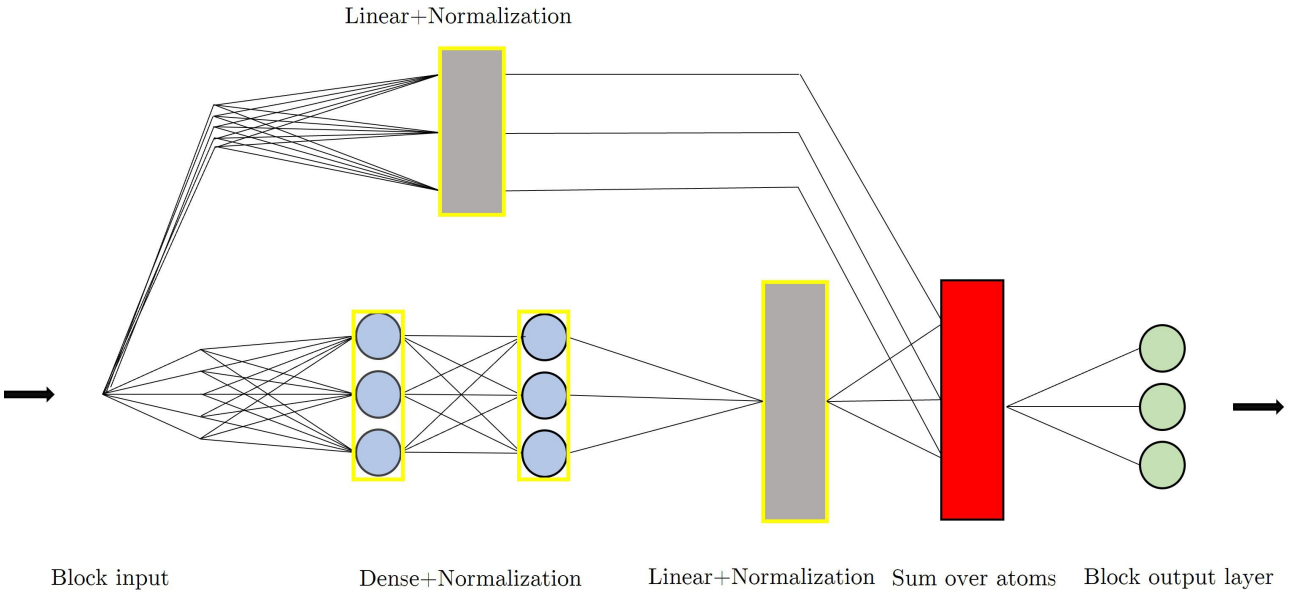


Figure 2.22: The dense block used in NeuralIL. Normalization layers covering the neuron layers act on the net input to the respective layers.

2.5.1 Weights and hyperparameters

Weights

The weights of NeuralIL are randomly initiated according to a truncated Gaussian distribution using the LeCun normal initialization [35]. This normalization initializes the weights according to a Gaussian distribution, based on the layer's own dimensions, which has been shown to have effective results in NN training [35]. The biases are initialized at zero.

Loss function

The loss function is dependent on the model prediction, as well as the ground truths. In the case of NeuralIL, the ground truths are the forces and total energies of the atomic configurations in the training data. The loss between the model prediction and ground truths in NeuralIL is calculated by the following loss function:

$$\begin{aligned} L = &\frac{1}{2}\left\langle \frac{0.2\,\mathrm{eV\AA^{-1}}}{n_{\mathrm{atoms}}} \sum_{i=1}^{n_{\mathrm{atoms}}} \log\left[\cosh\left(\frac{\|f_{\mathrm{i,predicted}} - f_{\mathrm{i,reference}}\|_2}{0.2\,\mathrm{eV^{-1}}}\right]\right\rangle \right. \\ &+ \frac{1}{2}\left\langle 0.02\log\left[\cosh\left(\frac{\mathrm{E_{pot,predicted}} - \mathrm{E_{pot,reference}}}{n_{\mathrm{atoms}} \cdot 0.02\,\mathrm{eVatom^{-1}}}\right]\right\rangle\right. . \end{aligned} \tag{2.26}$$

Here, the $< . >$ brackets denote an average over the atomic configurations in the current training batch. The variables $\mathrm{E_{pot,predicted}}$,$f_{\mathrm{i,predicted}}$,$\mathrm{E_{pot,reference}}$, $f_{\mathrm{i,reference}}$ are the ground truth energies and forces, and predicted energies and forces, respectively. The constants 0.2 and 0.02 are the loss functions force and energy parameters, respectively. These parameters can be tuned, where larger values directly affect the influence either the forces or the energy has on the loss function, and therefore also the model training. Previous studies using NeuralIL have excluded the energy in the loss, as it has been shown to have little effect on model performance

[16], however, this means an adjustment of the energies has to be completed at the end of the training. Including the energies as a part of the loss neglects this step [32]. The log(cosh) function has the property of *gradient clipping*, meaning it will reduce the influence of possible outliers in training. Figure 2.23 shows the log(cosh) function for different scaled values of x.

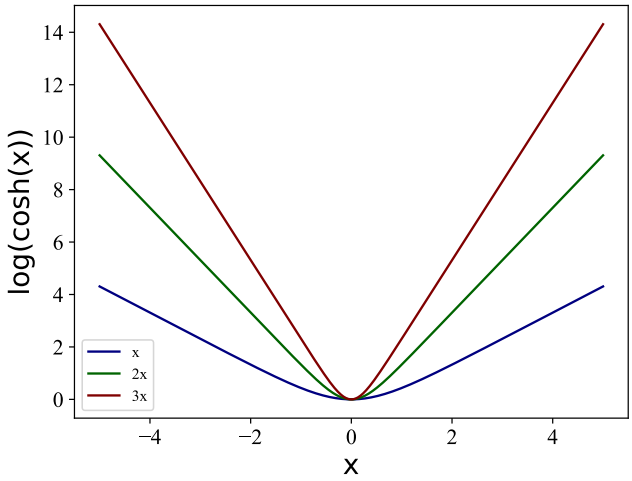


Figure 2.23: Log cosh function for different scaled values of x.

Velo optimiser

The loss function shown in 2.26 is minimized by the *VELO* algorithm [36], a versatile optimizer, which itself is a machine learning model trained on a wide variety of different optimization problems [36]. This stands in contrast to other well-known optimizers like ADAM [37] and RMSPROP [38] [39]. An advantage of the VELO optimizer is it not requiring the setting of a learning rate, which is a hyperparameter that has been subject to many studies of how to use correctly in machine learning [40]. The Velo optimizer is shown to reach higher performance compared to the other models in the literature [36].

Normalization

NeuralIL uses *Layernorm* [31] to normalize the inputs after every layer. LayerNorm normalizes based on the mean and variance of each feature in the input.

Activation functions

The activation functions in the hidden layers are non-linear swish-1 activation functions given by

$$s_1(x) = \frac{x}{1 + e^{-x}}, \tag{2.27}$$

where x is the net input to the neuron. The swish-1 activation function avoids the *vanishing gradient problem* [41], by which the training would be greatly slowed in the back-propagation scheme. Figure 2.24 shows the activation function:

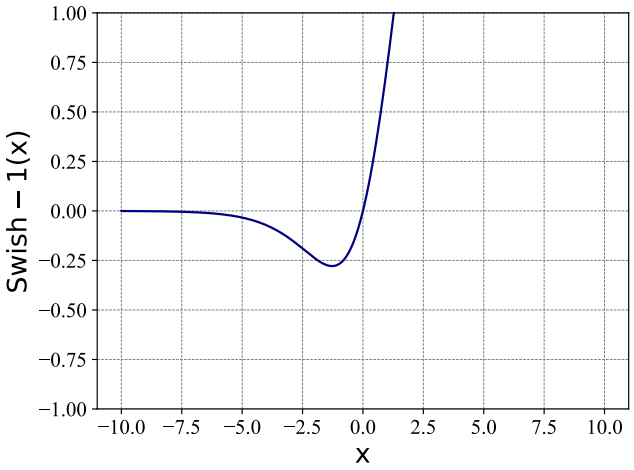


Figure 2.24: Illustration of the swish-1 activation function.

2.5.2 Jax and Flax

NeuralIL is based on Google's flexible machine learning framework JAX [42], with sub-packages FLAX [43] and OPTAX [44]. JAX is an efficient Python library specifically designed for high-performance computing, especially useful for machine learning purposes [45]. JAX and its sub-packages have been smoothly integrated into the code of NeuralIL, making it achieve some essential properties that come with the software:

- **Automatic differentiation**: JAX supports forward and reverse mode automatic differentiation, which is an efficient exact method for evaluation of derivatives [46].
- **Vectorization**: JAX implements automatic vectorization via a vector map. This makes it easy to apply functions to a large set of values, which is usually seen in machine learning, for example through the calculation of the loss function.
- **JIT-compilation**: Through the use of XLA (Accelerated Linear Algebra) [47] JAX's just-in-time (JIT) compilation allows for the software to run on GPUs, and cloud-based TPU accelerators, alongside the python package *Numpy's* efficient APIs.

2.5.3 Forces and automatic differentiability

An important feature of NeuralIL is the ability to train on forces. As each configuration contain $3 \cdot n_{atoms}$ amount of forces, the training data is much larger than if the model only trained on energies. Since the forces are defined as the gradient of the potential energies predicted by the model, it's essential to calculate these derivatives efficiently. NeuralIL does this through automatic differentiability, via a Vector-Jacobian product(VJP) shown in (2.28).

$$VJP_i\left(r, \frac{\partial E_{\text{pot,predicted}}}{\partial p_\alpha}\right) = \sum\nolimits_\alpha \left(\frac{\partial E_{\text{pot,predicted}}}{\partial p_\alpha}\right)_\alpha \cdot \frac{\partial p_\alpha}{\partial r_i}. \tag{2.28}$$

Here, α is an index running over all the descriptors for all atoms in configuration i, p_α are all the descriptors for all atoms in the configuration, and $E_{\text{pot,predicted}}$ is the potential energy predicted by the model. The VJP computes all the forces in the configuration by a single call of the potential energy, and can also provide higher order derivatives like the stress tensor [32].

2.5.4 Jax-MD and flexible cell simulations

After a model is trained to convergence, its parameters can be loaded and used for new force predictions. As mentioned in section 2.2, configurations are generated in MD by calculating atoms' positions and velocities via a force field. NeuralIL can be used to generate these forces in each step of the velocity Verlet algorithm, replacing the more traditional methods. For NeuralIL, the Jax implementation with the VJP to extract forces is central to the efficiency of the model. Due to the code's Jax implementation, the model can also be used to run Jax-MD [48], a software MD-subpackage built on Jax, allowing the model to run on GPUs. All MD simulations completed with NeuralIL will hereafter be referred to as Jax-MD.

As mentioned in this section, MD simulations are determined by which ensemble is chosen to simulate in, holding some thermodynamic properties constant in the environment. In the *NPT* ensemble, it is possible to let the unit cell values fluctuate, varying the cell dimensions while the simulation is running. A non-isotropic extension of this methodology was integrated into the code of Jax-MD by Sebastian Bichelmaier and his team as a part of his Ph.D. work [21]. The premise of flexible cell simulations should provide a more favorable simulation environment for solids, as the cell shape is crucial for predicting their properties [21].

2.6 Phase transition of DMMgF

A study by Szafrański et.al. [18] shows details about the phase transition of DMMgF. In the article, they show how DMMgF has a characteristic phase transition occurring experimentally at temperatures between 262K and 264K. The phase transition comes with a significant structural order-disorder difference that can be seen in the systems dimethylamine ($(CH_3N)_2 - NH_2$) molecules. The low-temperature phase is shown to have significant order in the molecules, while the high-temperature phase shows a disorder. As a consequence, an order parameter can be defined to check which phase a configuration is in. We define this order parameters as a sum of middle vectors pointing from the nitrogen atom to the connecting point between each methyl molecule. Figure 2.25 shows an example of a dimethylamine molecule with its corresponding middle vector

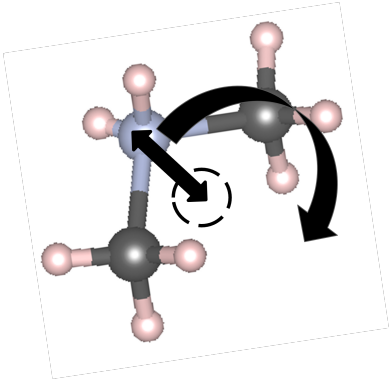


Figure 2.25: A middle vector for a CH_3NCH_3 molecule in the perovskite system, illustrated by Kristian Berland. The connecting point between the two methyl molecules is shown by the dashed sphere. The white, black, and blue spheres denote the hydrogen, carbon, and nitrogen atoms, respectively.

The rotated arrow in figure 2.25 illustrates the re-orientation that certain dimethylamine molecules would experience under an order-disorder phase transition. Equation (2.29) shows the formula for the order-parameters calculated for each dimethylamine molecule within a DMMgF configuration:

$$\text{Order-parameter} = \frac{\sum_n ||\text{Middle vectors}||_2}{m}. \tag{2.29}$$

Here, the sum is normalised by m, signifying the number of dimethylamine molecules in the configurations. A drop in order-parameter values towards a value of 0 would indicate that the vectors had summed to 0, illustrating a re-orientation of the molecules so that they point in opposite directions, signifying the disordered structure.

Chapter 3

Method

This chapter will first introduce the general goals and problems of the thesis. Then, the software used for the generation of data sets and general analysis of the DMMgF system will be presented. Details about the available training data will then be presented before details on how this data was used in the training of NeuralIL will be explained in detail. Finally, how the Jax-MD simulations were performed will be presented.

3.1 Problem description and goals of thesis

The goals of this master's thesis were briefly mentioned in the introduction. Since relevant theory has now been presented, the goals of the study can be explained a bit more thoroughly. In general, the study seeks to find out more about NeruralIL's predictive performance on DMMgF by predicting interatomic forces and comparing them to DFT calculations. Additionally, we aim to evaluate whether the force predictions obtained from NeuralIL be used to simulate stable MD. We refer to the stability of MD simulation in terms of the fluctuations in the total volume of the configurations during the simulation time. While no specific fluctuation range has been deemed acceptable for defining stability, fluctuations that cause non-physical atomic movements, and possible simulation crashes are considered non-optimal. Using NeuralIL, we investigate possible combinations of training data and parameters to see how this affects the stability of MD simulations.

The final goal of the thesis is to answer whether the NeuralIL-based MD simulations can reproduce an experimentally determined phase transition of DMMgF from the low to high-temperature phase. To determine this, we seek to reproduce the result shown by Szafrański et.al. in their study of the phase transition of the DMMgF system. Figure 3.1 shows their experimental data, where a volume drop between the two phases of the system occurs at temperatures between 262K and 264K.

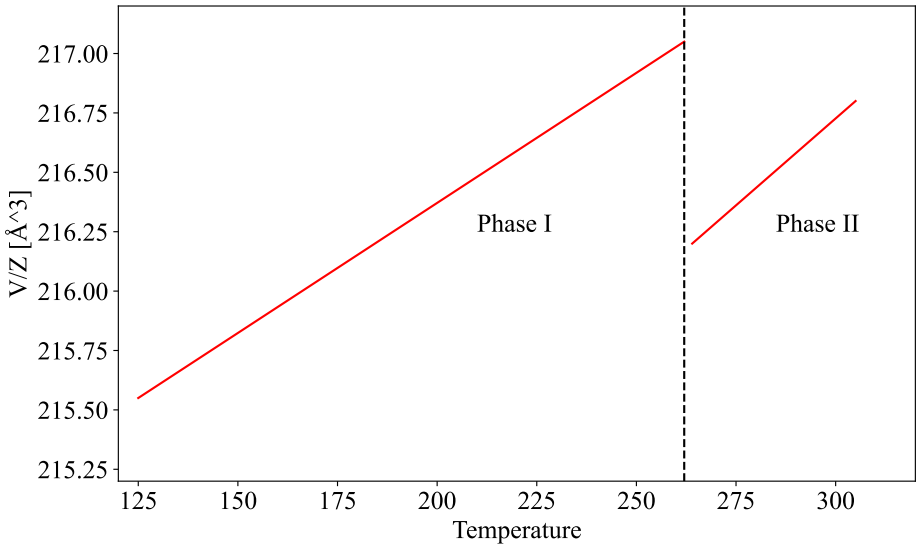


Figure 3.1: Experimental volume evolution of the DMMgF system for a temperature range of 120 to 320 K. The volume has a significant drop between temperatures of 262-264K, following a phase transition from a ordered to disordered phase. The figure is adapted to show linear fits to the experimental data in Figure 1 (b) from the study of Szafrański et.al. [18].

Using NeuralIL, the goal is to explore whether a stable model can reproduce the result shown in figure 3.1 with MD simulations in this temperature region. If such a result could be shown, it could indicate whether NeuralIL is capable of capturing the underlying physics of DMMgF in agreement with experimental data.

3.2 Software

3.2.1 Sigma2 clusters

Training of NeuralIL models and simulations of NeuralIL-based MD were completed on resources provided by Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway. Python scripts used were submitted as batch jobs to available GPU nodes on the computers through the slurm workload manager. For details regarding sigma2, see their home page[1].

3.2.2 VASP

The Vienna Ab initio Simulation Package (VASP) [49] was used to run molecular dynamics for the generation of data sets used in the training of NeuralIL. For more details on the data sets, see section 3.3. Be referred to VASP's wiki page[2] for details regarding calculations of MD using the software.

[1] https://www.sigma2.no/
[2] https://www.vasp.at/

3.2.3 ASE

The Atomic Simulation Environment (ASE) [50] is a Python package with a set of tools for working with atomic simulations. It was routinely used to extract individual atoms and key properties from the different configurations contained in the MD trajectories. As with VASP, details can be found in their documentation[3].

3.2.4 VESTA

The software Visualisation for Electronic and Structural analysis (VESTA) [51] was used to inspect and visualize configurations and individual dimethylamine molecules attained from the MD trajectories. Details about the VESTA software can be found in their documentation.[4]

3.2.5 Molcrys

The recently published python package molcrys[5] was routinely used to calculate the order-parameters introduced in section 2.6. The package is developed by the material theory and informatics team at NMBU.

3.3 Training data

All data sets used for the training of NeuralIL were created by Rasmus André Tranås using the VASP software. The data sets were created by running several DFT-based MD simulations of DMMgF. In total, six simulations were completed using NVT ensembles. The first five simulations had a temperature of 300K, using DFT-relaxed lattice constants scaled by a factor varying between 0.98 and 1.02. The final simulation was conducted for a higher temperature of 600K and was completed with a lattice constant scaled by a factor of 1.0. The ground truth forces used in training were routinely provided by DFT calculations for each configuration. The trajectories were compiled into *.json* files, resulting in six different data sets with either different volumes or different temperatures. Table 3.1 shows an overview of the different data sets used in this thesis.

Table 3.1: Different data sets used in training of NeuralIL.

Data set	Lattice constants scaling factor	Temperature [K]	Configurations
1	0.98	300	2620
2	0.99	300	2472
3	1.0	300	2292
4	1.01	300	2275
5	1.02	300	2176
6	1.0	600	2275

[3]https://wiki.fysik.dtu.dk/ase/
[4]http://www.jp-minerals.org/vesta/en/
[5]https://gitlab.com/m7582/molcrys/

3.4 Machine learning framework

In general, creating a robust machine learning model for any given problem requires detailed thought and exploration of which data sets and parameters to train with. Figure 3.2 shows how a general machine learning framework is adapted to this study using NeuralIL:

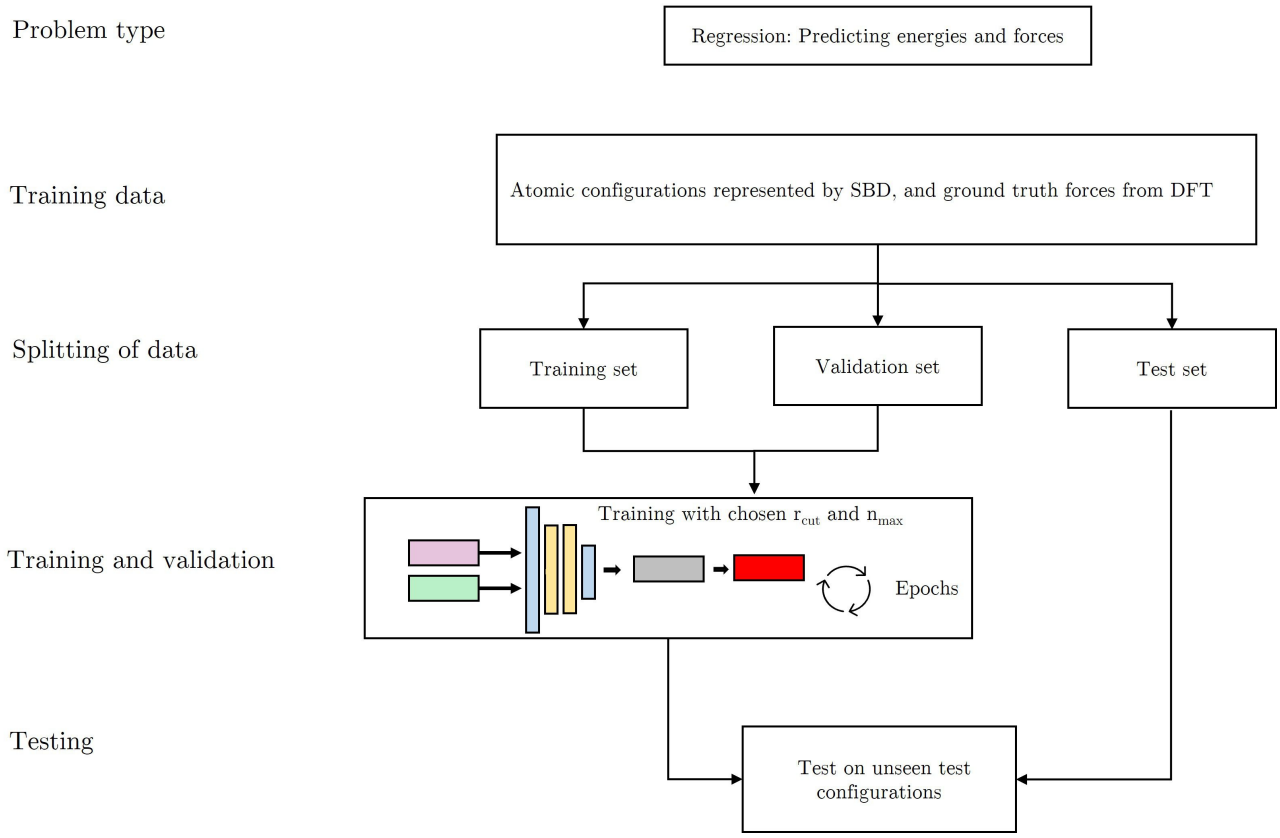


Figure 3.2: An illustration of the general machine learning framework used in this study.

Figure 3.2 leaves out details about the different steps of the framework used in this study. We will now go into detail about which combinations of data sets from 3.1 were chosen to train different models, and which parameter combinations were chosen.

3.4.1 Hyperparameters

Some hyperparameters of NeuralIL were unchanged in this study, regardless of which training framework was used in the analysis. Table 3.2 shows these hyperparameters:

Table 3.2: Hyperparameters remaining unchanged in the training of NeuralIL

Loss function force parameter	0.20 eV/Å
Loss function energy parameter	0.020 eV/Å
Core widths	128-64-32-16-16
Mini batch size	4
Validation batch size	32

Table 3.2 shows the loss function parameters, which directly affect the influence of the atomic

forces and energies on the loss function shown in equation (2.26). A larger value of the force parameter is chosen so that the forces will have a larger influence on model training. The validation batch size is chosen so the model validates on a batch of samples from the validation set, rather than the whole data all at once. Similarly to mini-batches explained in section 2.4.3, this has memory advantages in training but can result in slower training.

The only hyperparameter that was changed for the training of different models was the number of epochs. Initially, the number of epochs was set to 50 for all models, but some models were re-trained with a different amount as explained in section 3.4.5.

3.4.2 NeuralIL training

To ensure that we were working with a robust version of NeuralIL when simulating JAX-MD, several different models were trained, validated, and tested. The analysis was performed with the goal of finding a NeuralIL model that could show stability when used in the simulation of JAX-MD, avoiding simulation crashes and un-physical results. The different models trained followed separate training methods; *a simple training* and *a mixed training*. These two ways of training only differ with regards to which data sets from table 3.1 are used in the training of the NeuralIL models, and not other specifics like the model parameters.

Simple training

Figure 3.3 shows an illustration of the simple training method. All NeuralIL models following the simple training method were trained on configurations only extracted from data set 3 from table 3.1. All configurations contained in the single data set were used in training. The data was then split into training, validation, and test sets. 20 randomly drawn configurations were withheld as a test set from data set 3, with the remaining data being split into an 80% training and 20% validation split. The model was then trained and validated on the training and validation split, with chosen values for r_{cut} and n_{max} as described in section 3.4.3, alongside the constant choices of hyperparameters shown in table 3.2. To get a larger test region for the models, test sets containing 20 randomly drawn configurations were extracted from the remaining data sets in table 3.1, resulting in six different test sets for the model to be tested on. The trained model was finally tested on all six test sets.

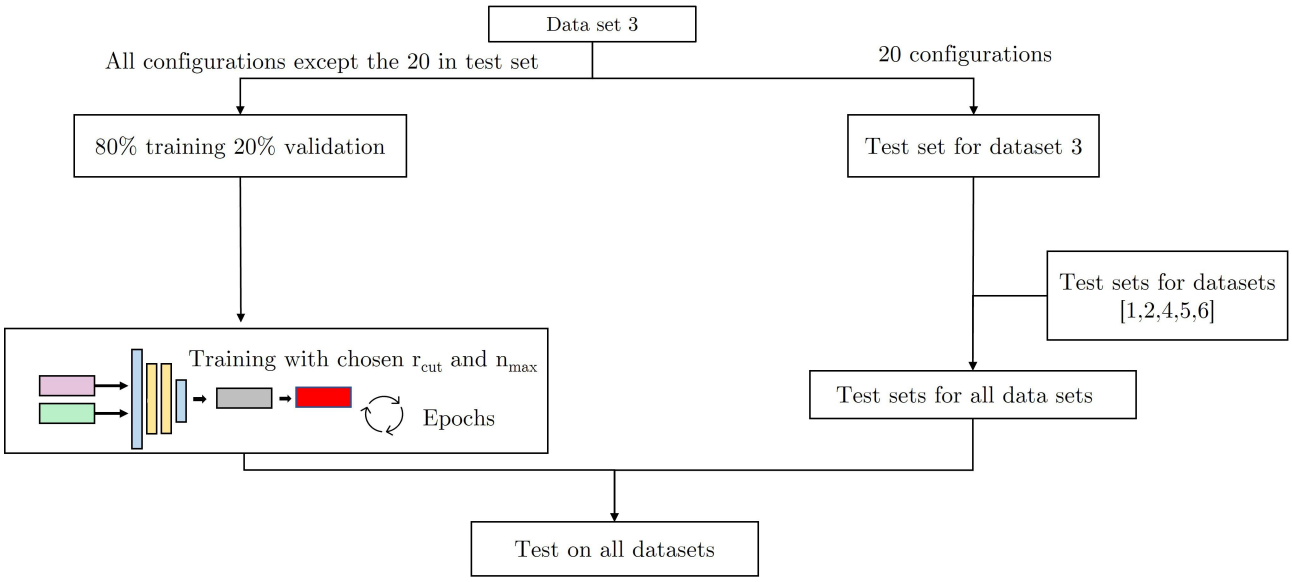


Figure 3.3: Illustration of the simple training method. All data sets are extracted from the data sets in table 3.1. The model splits data set 3 into a training, validation, and test set. A NeuralIL model is trained on the training set, and validated on the validation set before it is tested on all six external test sets extracted from the data sets in table 3.1.

Mixed training

Figure 3.4 shows an illustration of the mixed training method. In contrast to the simple training, all models following the mixed training used a subset of 400 randomly drawn configurations from each of the data sets in table 3.1 to train a NeuralIL model. The 400 configuration subsets totaled a set of 2400 configurations, which were split into 80% training and 20% validation split. The model was then trained and validated on this training and validation split, with chosen values for r_{cut} and n_{max} as described in section 3.4.3, alongside the constant choices of hyperparameters shown in table 3.2. To extract test sets that were ensured to be different than the ones used in training, the index labels of the used configurations were saved to a Python dictionary. Under testing, 20 test configurations were randomly drawn from the data sets in table 3.1, and ensured to be different from the ones used in training by indexing the labels in that same dictionary.

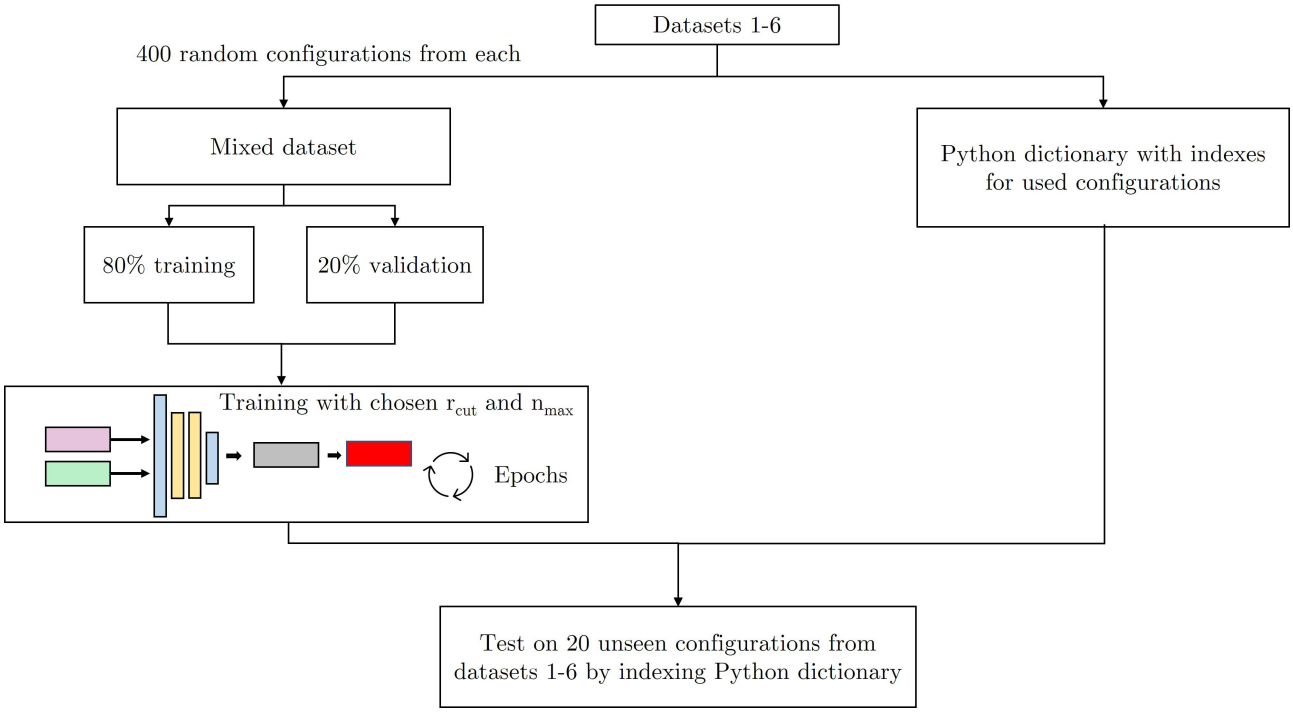


Figure 3.4: Illustration of the mixed model training method. All data sets are extracted from the data sets in table 3.1. 400 random configurations from data set 1-6 into a mixed data set containing 2400 configurations. This data set is split into training and validation sets that are used in the training of a NeuralIL model. Test sets are extracted based on indexing labels that are saved to a Python dictionary.

3.4.3 Parameter optimization and model combinations

Ensuring that we found the most optimized NeuralIL model in the context of the two training methods introduced in 3.4.2, we explored a range of parameter combinations for both of them. The parameter combinations explored were the cut-off radius r_{cut} and the n_{max} parameters. Several combinations of these parameters were explored, varying r_{cut} between values of 3.5 Å and 6 Å, and n_{max} between 4 and 6. These parameter combinations were tested for an equal amount of models following the simple training method and the mixed training method. This resulted in a total of 14 different NeuralIL models, with both different training data and parameters. Table 3.3 shows these 14 models. We denote the models with names *simple* or *mixed* based on which training method in section 3.4.2 was used.

Table 3.3: Different models used in the parameter optimization. The parameters containing equal colors signify equal values.

Model name	r_{cut}[Å]	n_{max}
Simple model 1	3.5	4
Simple model 2	5	4
Simple model 3	5	5
Simple model 4	5	6
Simple model 5	6	4
Simple model 6	6	5
Simple model 7	6	6
Mixed model 1	3.5	4
Mixed model 2	5	4
Mixed model 3	5	5
Mixed model 4	5	6
Mixed model 5	6	4
Mixed model 6	6	5
Mixed model 7	6	6

Choosing to focus on the parameters r_{max} and n_{max}, we have excluded many of the model hyperparameters that could be explored in an extended hyperparameter optimization. Among the ones excluded are the loss function, activation functions, and layer widths.

3.4.4 Selection of models for simulation

Simulating Jax-MD for each of the 14 models in table 3.3 would result in a quite extensive analysis. Therefore, only four models were used. Of the four models, one model was simple, and three were mixed. The models were selected by which had the smallest *root mean square error* (RMSE) and *mean absolute error* (MAE) between the ground truth and predicted forces. Small mean error values of these two error metrics over all the six test sets extracted from each data set in 3.1 were used as a final performance criterion for selection. Small errors would mean that the models predict forces closer to the ground truth forces calculated by DFT, which was deemed as preferred when performing MD simulations.

Parity plots showing the predicted forces against the DFT ground truth force calculations were also inspected to ensure the models had reasonable predictions on all test sets, with no distinct outlier predictions.

3.4.5 Re-training of models

After observation of non-optimal volume fluctuations, the models selected for Jax-MD were trained again with a reduced amount of epochs, determined by where the errors on the training in the model had converged. This technique is usually referred to as *Early stopping* [52], from where the learning is stopped based on the errors having converged over the validation set. Further training beyond this point would make the generalized performance worse [52], by increasing the chance of a model overfitting. In this work, we choose our *convergence point* as the number of epochs where the model only sees a slight decrease in MAE over the validation set. The chosen convergence points are only based on observation of the graphs, whereas more

optimal points may have been found by using more complex methods like specific stopping criteria [52]. See section 4.2.3 for more details on the models that were re-trained, and the convergence points chosen for each of them.

3.5 Jax-MD

All Jax-MD simulations were performed in an NPT ensemble with flexible cells, following the implementation of Sebastian Bichelmaier [21]. The pressure in the simulations was fixed at 1 bar, with a constant temperature defined by a set value. All simulations were completed with 150 thousand timesteps of 0.25 fs, totaling a time of 37.5 ps for every simulation. The simulations were conducted with a barostat constant, and coupling constant of $\tau_{\mathrm{p}} = 1500$ fs and $\tau_{\mathrm{t}} = 500$ fs, respectively. The constants were set with recommendations from Bichelmaier. See [21] for a detailed explanation of these parameters in the context of flexible cell simulations. The initial structure of the Jax-MD was loaded from a VASP *CONTCAR* file used in the generation of dataset 3. The CONTCAR file defined a supercell containing 384 atoms in the low-temperature phase of the DMMgF system. The configurations resulting from Jax-MD simulations were written to ase trajectory files for every 25th simulation step.

3.6 Exploration of phase transition temperatures

The temperature region explored for the DMMgF system ranged from 250K to 335K, with intervals of 5K between each simulated temperature. An extra simulation at 262.5K was completed to attain more detail close to the experimental phase transition temperature laying between 262K and 264K.

The first 400 written configurations (10 thousand MD steps) were excluded from the mean volume calculations of the resulting MD trajectories. This was done to account for possible thermalization in the cell. The mean volumes were plotted per formula unit, against the different temperatures simulated, where a drop in agreement with figure 3.1 would be expected had the phase transition occurred. The order-parameters defined by equation (2.29) were also inspected for certain trajectories to see if the phase transition had occurred.

Chapter 4

Results

This chapter will provide the results of the study, divided into four sections. Firstly, the force predictions of the model will be presented by tables and parity plots. Secondly, the results from JAX-MD simulations will be presented for selected models. Thirdly, a section presenting the result of the exploration of the temperatures close to the experimental phase transitions of DMMgF will be shown. Finally, results from an additional simulation started from the high-temperature of DMMgF will be shown.

4.1 Force predictions

This section will provide the results of the force predictions made by the 14 different NeuralIL models presented in 3.3. Only a select amount of parity plots will be shown, see appendix A for all plots.

4.1.1 Mean errors of predictions

Table 4.1 shows the mean predictions of all 14 models on all the respective test sets extracted from the data sets in table 3.1.

Table 4.1: Table of force prediction mean errors for all model combinations.

Model	r_cut [Å]	n_max	RMSE [eV/Å]	MAE [eV/Å]
Simple model 1	3.5	4	0.1298	0.0964
Simple model 2	5	4	0.1541	0.1143
Simple model 3	5	5	1.2237	0.2468
Simple model 4	5	6	0.1706	0.1220
Simple model 5	6	4	0.1754	0.1305
Simple model 6	6	5	0.2032	0.1420
Simple model 7	6	6	0.1840	0.1389
Mixed model 1	3.5	4	0.0693	0.0516
Mixed model 2	5	4	0.0583	0.0445
Mixed model 3	5	5	0.0597	0.0455
Mixed model 4	5	6	0.0532	0.0406
Mixed model 5	6	4	0.0795	0.0609
Mixed model 6	6	5	0.1158	0.0859
Mixed model 7	6	6	0.0589	0.0450

Table 4.1 four distinct colored rows depict the best simple model, and the three best-mixed models in order, colored in green, blue, yellow, and orange, respectively. Overall the mean errors shown in the table are generally lower for all the mixed models compared to the simple models. This comes as no surprise, given that the NeuralIL models following the mixed training were trained on configurations from all the data sets, compared to the simple models only being trained on data set 3. Since the mixed models have seen a number of configurations from the same sets they are tested on in training, the models can be said to interpolate well. In contrast, the simple models have generally worse performance, with significantly higher error values. The parity plots give an insight into why the mean errors are generally high for the simple models.

4.1.2 Parity plots

Figure 4.1 show parity plots of the force predictions done by simple model 1, colored in green in 4.1.

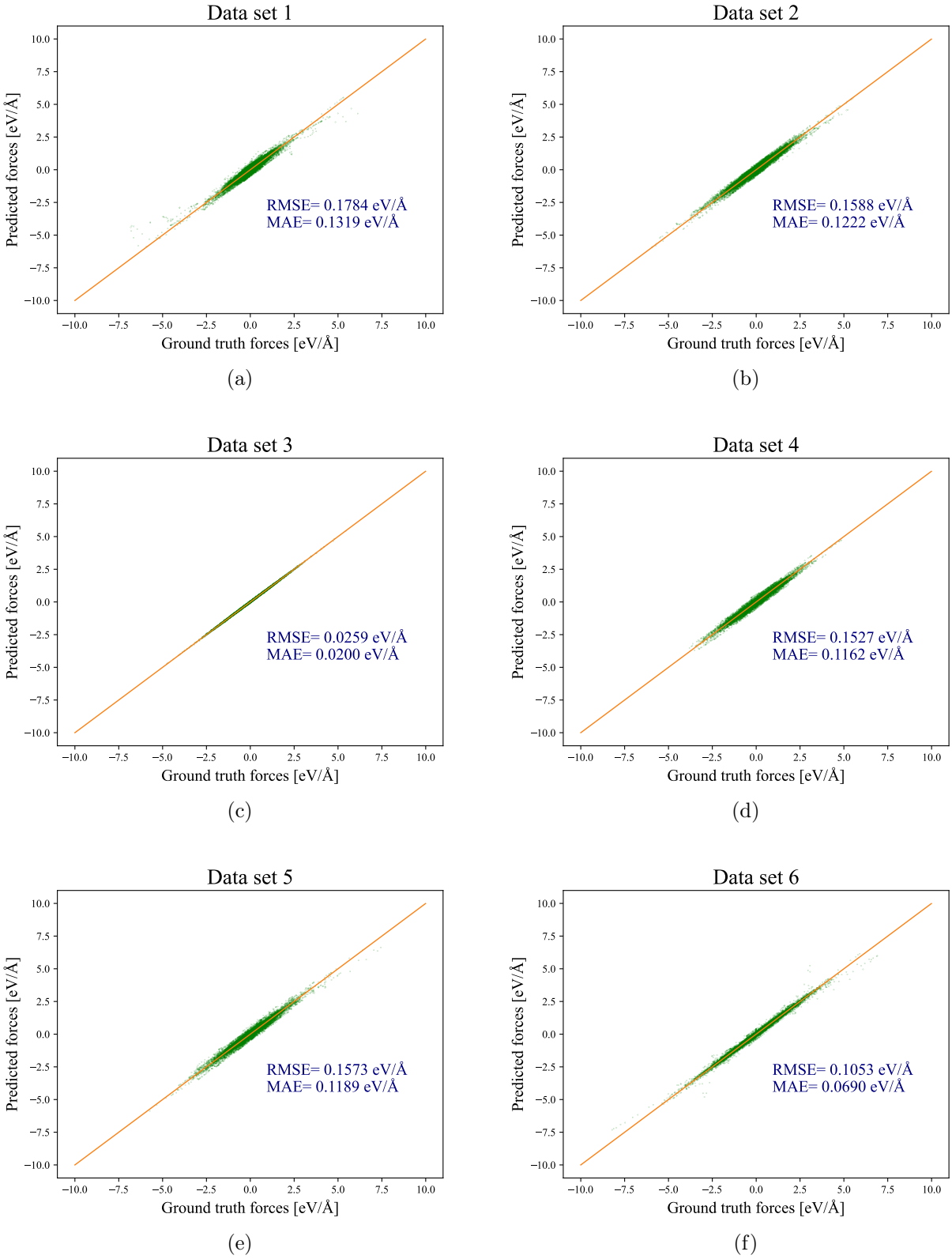


Figure 4.1: Parity plots of predicted forces for simple model number 1 against ground truth forces from DFT. The title of the plots denotes which test set is predicted on i.e: (a): Predicted forces on a test set of 20 configurations extracted from data set 1. The predicted forces are plotted as scatter points in green, where a prediction having zero error would lay on the orange line.

The parity plots in figure 4.1 shows all six of simple model 1's predictions on all test sets extracted from the data sets in table 3.1. As can be seen from plot (c), the model interpolates well as the prediction errors associated with the test set from data set 3 is remarkably low with a value of MAE = 0.020 eV/Å. However, as the model was only trained on configurations from data set 3, it extrapolates badly, as seen from the other plots in figure 4.1 containing higher error values.

Figure 4.2 shows the force predictions on each test set by mixed model 4, which had the smallest mean errors, colored in blue in table 4.1.

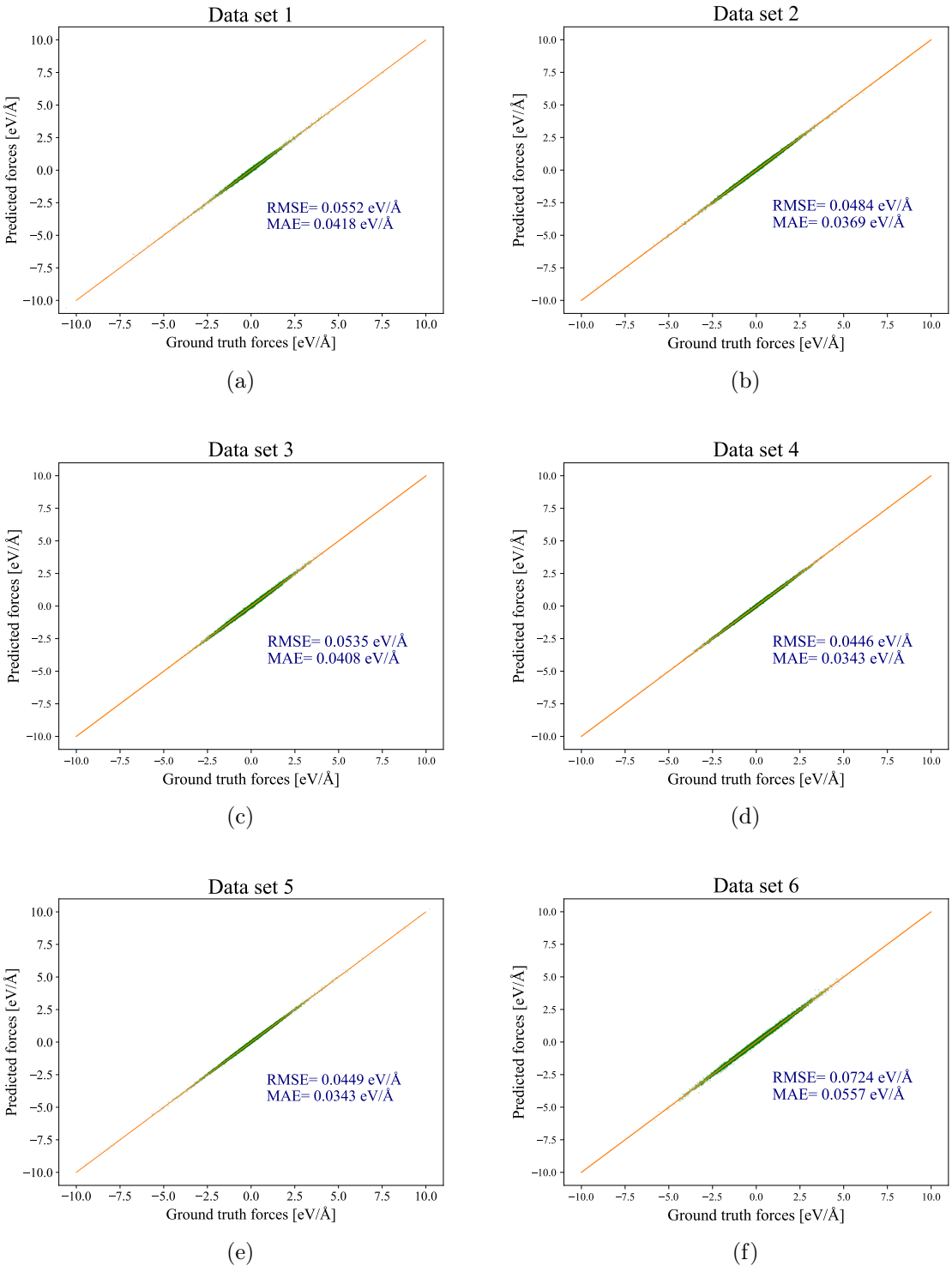


Figure 4.2: Parity plots of predicted forces for mixed model 4 against ground truth forces from DFT. The title of the plots denotes which test set is predicted on, i.e: (a): Predicted forces on a test set of 20 configurations extracted from data set 1. The predicted forces are plotted as scatter points in green, where a prediction having zero error would lay on the orange line.

The parity plots show an overall accurate performance for mixed model 4 over all the test sets, with a slightly higher error for the high-temperature data set in parity plot (f). The over all better performance is a direct result of including configurations from all six data sets in

training, as explained in section 3.4.2. However, the parity plots give no indication of model extrapolation, as the test sets do not cover areas that the model has not seen in training. Still, the model shows remarkably good interpolation performance with error values as low as MAE = 0.0343 eV/Å, as seen in plot (d) and (e).

4.2 Volume fluctuations from Jax-MD

4.2.1 Initial models

The four models highlighted with colored rows in table 4.1 were selected to simulate Jax-MD based on the performance criteria of lowest errors. Since, the three mixed models showed much lower errors in the force predictions, three of these models were selected, as opposed to the one simple model. Figure 4.3 shows the Jax-MD volume fluctuations plotted per formula unit for these four models for the first 40 thousand of 150 thousand simulation steps.

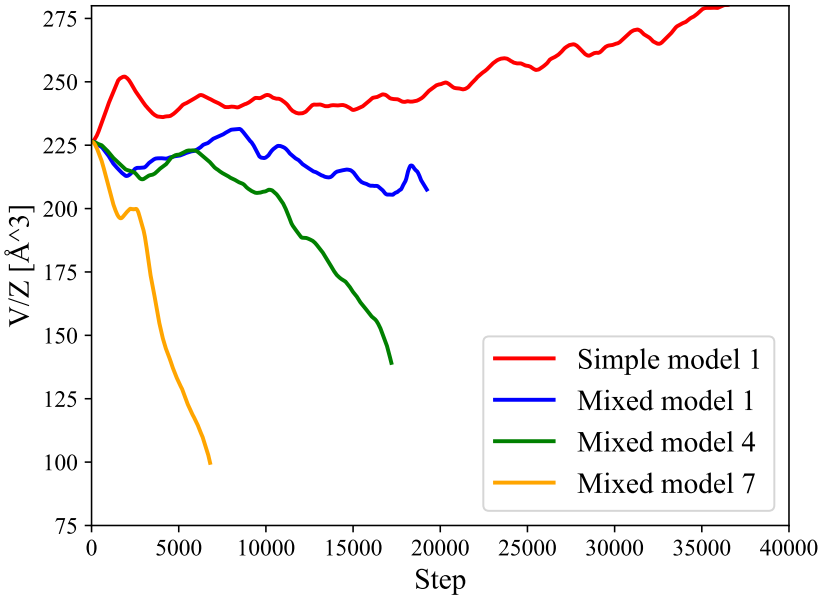
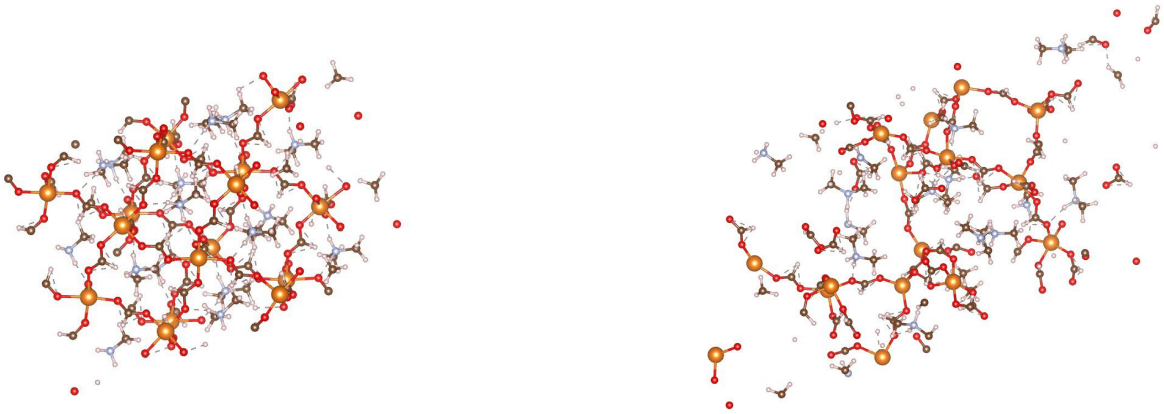


Figure 4.3: Volume fluctuations of the four selected models plotted for the first 40 thousand Jax-MD steps. All models show non-physical volume fluctuations, resulting in simulation crashes.

All four trajectories show non-physical fluctuations, either with a gradual decrease or increase of volume until the eventual simulation crash.

4.2.2 Simulation crashes

Visualizing the atomic configurations gives an insight into what happens at the atomic level when the volumes either increase or decrease towards un-physical values in figure 4.3. Figure 4.4 shows the initial atomic configuration, and the last configuration attained simple model 1's trajectory in figure 4.3.



(a) (b)

Figure 4.4: (a) Initial configuration in the simulation. (b) Last configuration after 100 thousand MD steps using simple 1. The atoms move away from each other in an un-physical manner. Mg atoms are colored orange, nitrogen blue, carbon brown, and hydrogen white.

Figure 4.4 shows how the atoms move away from each other in the configuration, breaking the atomic structure and causing the simulation to crash. The explosion can be concretely shown by summing all the pairwise distances between each of the carbon atoms in every configuration along the trajectory. Figure 4.5 shows the evolution of this sum along the trajectory, up to 100 thousand MD steps at which the simulation was terminated.

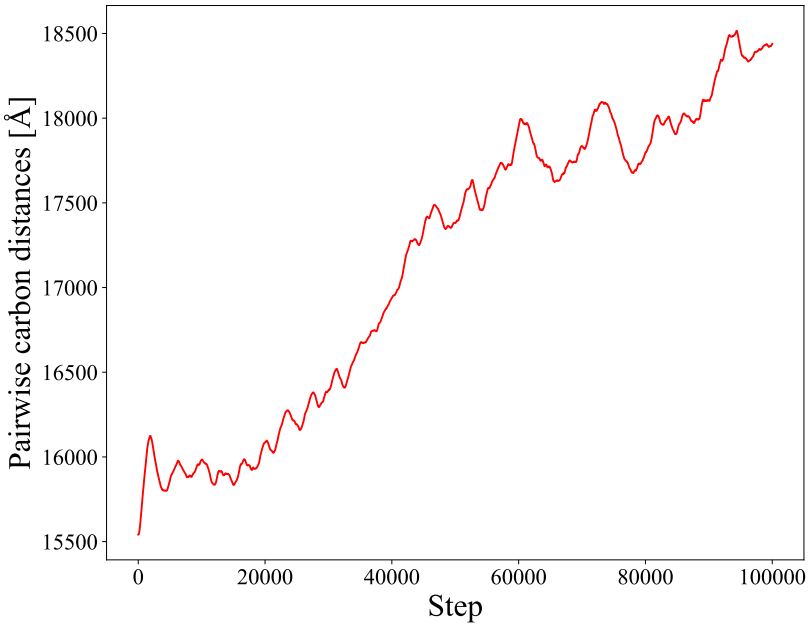


Figure 4.5: Sum of pairwise distances between the 80 carbon atoms in each configuration along the JAX-MD trajectory using simple model 1. The sum increases over the simulation time.

As can be seen, by the sum, the distance increases steadily for the whole simulation time,

coinciding with the explosion shown in figure 4.4.

In the case of simple model 1, the explosion can be seen with a correspondingly high increase in the volume over the trajectory as shown by the red line in figure 4.3. In other simulations, the opposite behavior can be observed, as exemplified by the simulation of mixed model 7. In this simulation, the volume decreases rapidly before the simulation eventually crashes, shown by the yellow graph in figure 4.3. Figure 4.6 shows the first and last configuration attained in the trajectory of mixed model 7 in figure 4.3.

(a) (b)

Figure 4.6: (a) Initial configuration in the simulation. (b) Last snapshot of configuration before simulation crash. The atoms have moved closer to each other than in the initial configuration. Mg atoms are colored orange, nitrogen blue, carbon brown, and hydrogen white.

In the snapshots, it is clear that the atoms' movements cause the cell to implode, corresponding with the decrease in volume.

Similar to figure 4.5, the sum of pairwise carbon distances in mixed model 7's trajectory is shown in figure 4.7.

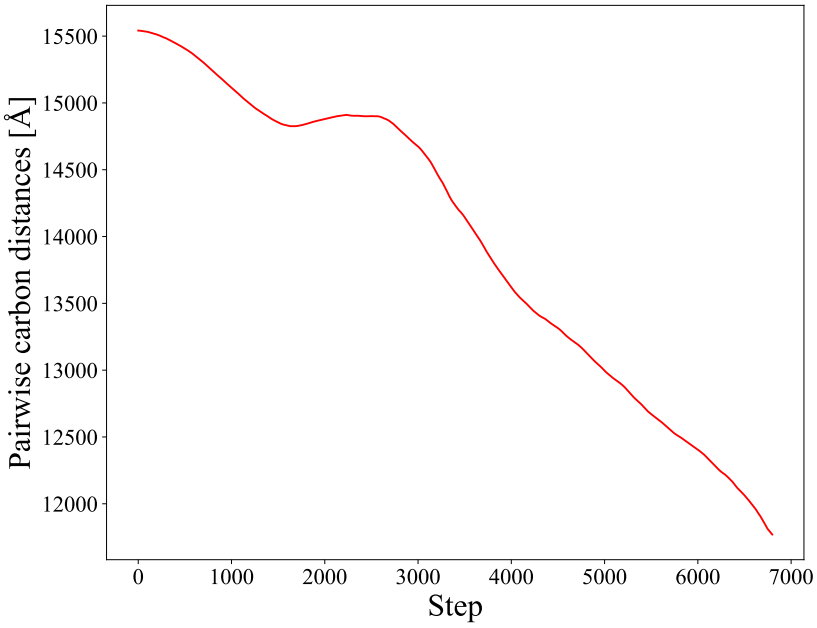


Figure 4.7: Sum of pairwise distances between the 80 carbon atoms in each configuration along the JAX-MD trajectory using mixed model 7. The sum decreases over the simulation time.

The pairwise distances decrease in agreement with the implosions we see in the snapshots of figure 4.6. As a consequence of the crashing, we hypothesized the models to be overfit on the training data. The models were thus re-trained according to an early stopping method, as explained in section 3.4.5.

4.2.3 Validation metrics for re-training

The four models initially used for JAX-MD simulations were trained again, with a smaller amount of epochs to avoid the potential overfit, as explained in section 3.4.5. Figure 4.8 shows the MAE values of all the initial models over the validation set. Here, the number of epochs for the initially trained models is shown by red vertical lines, whereas the convergence points chosen as the number of epochs to re-train the models with are shown by green verticle lines.

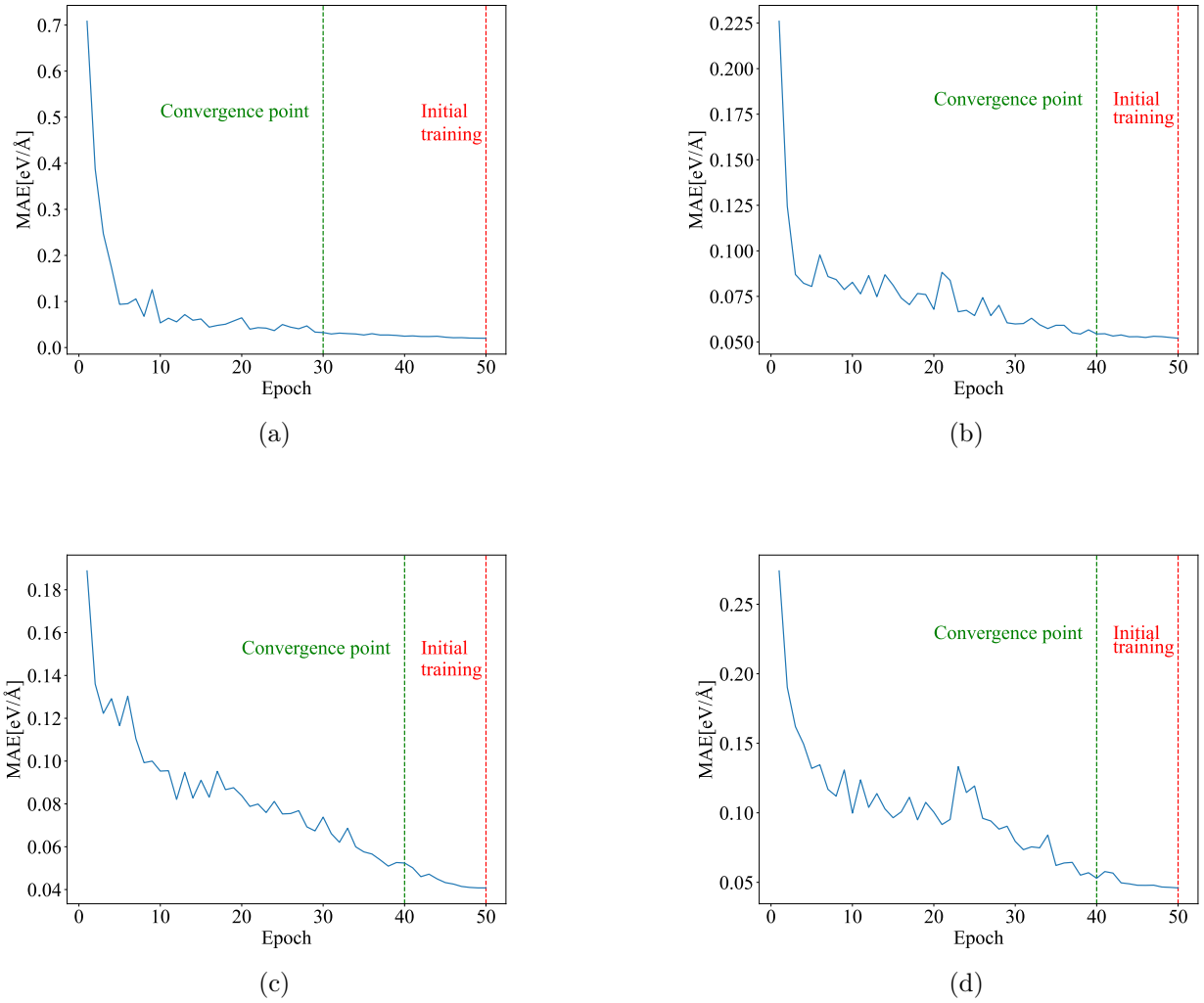


Figure 4.8: Validation metrics of the four initially selected models that simulated JAX-MD. Vertical lines are drawn in red and green denoting the number of epochs for the initial model training, and the convergence points chosen as amount of epochs for re-training. (a) Simple model 1. (b) Mixed model 1. (c) Mixed model 4. (d) Mixed model 7.

Plots (b)-(d) in figure 4.8 shows that model convergence in training is approximately reached for the mixed models at 40 epochs, where only small decreases of the *MAE* can be seen for the next 10 epochs. For the simple model in plot (a), an approximate convergence point is reached at 30 epochs. These convergence points where chosen as the number of epochs to train the models again, and perform new JAX-MD simulations.

4.2.4 Re-trained models

Figure 4.9 shows the JAX-MD volume fluctuations for the four re-trained models:

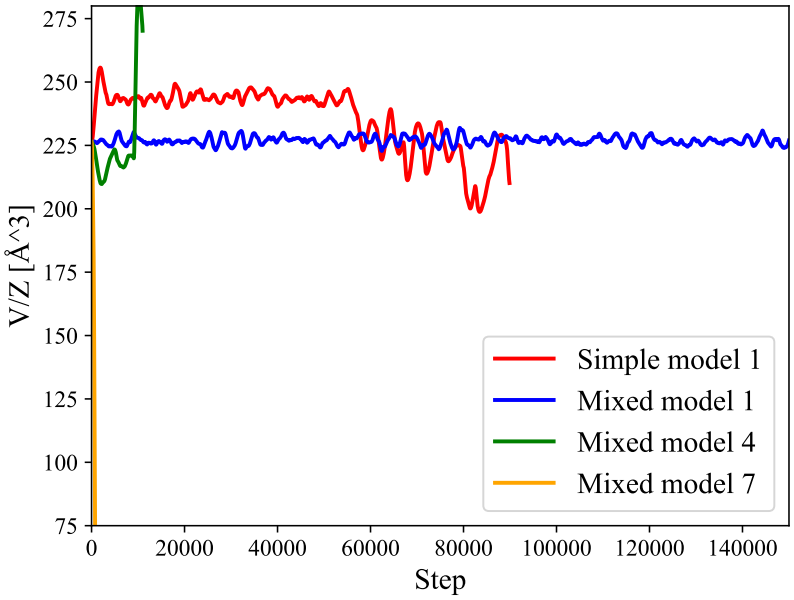


Figure 4.9: Volume fluctuations of the four selected models. Mixed model 1 shows stability in the simulations, while the other models show non-stability, resulting in simulation crashes.

The plot shows how the re-trained mixed model 1 gives stable volume fluctuations over the simulation time, while the other models result in simulation crashes. Of the three models showing crashing in the simulations, mixed model 7 and 4 do so at earlier times than simple model 1, although these models have smaller errors in the force predictions. Table 4.2 shows the mean error values of the re-trained models.

Table 4.2: Table of mean errors of force predictions over all data sets for the four re-trained models.

Model	**r_cut[Å]**	**n_max**	**RMSE[eV/Å]**	**MAE[eV/Å]**
Simple model 1	3.5	4	0.1381	0.1028
Mixed model 1	3.5	4	0.0819	0.0604
Mixed model 4	5	6	0.0689	0.0486
Mixed model 7	6	6	0.0645	0.0502

The re-trained models have worse performance over the six test sets, as shown by the larger error values compared to the errors of the initial 50 epoch models shown in table 4.3.

Table 4.3: Table of mean errors of force predictions over all data sets for the four models that were later re-trained.

Model	**r_cut[Å]**	**n_max**	**RMSE[eV/Å]**	**MAE[eV/Å]**
Simple model 1	3.5	4	0.1298	0.0964
Mixed model 1	3.5	4	0.0693	0.0516
Mixed model 4	5	6	0.0532	0.0406
Mixed model 7	6	6	0.0589	0.0450

4.3 Exploration of phase transition temperatures

As explained in section 3.6, temperatures between 250K and 335K were simulated to look for the phase transition of DMMgF. All simulations were performed with re-trained mixed model 1, which showed stability in figure 4.9. Figure 4.10 shows the mean volume calculations for all the temperatures simulated in the region.

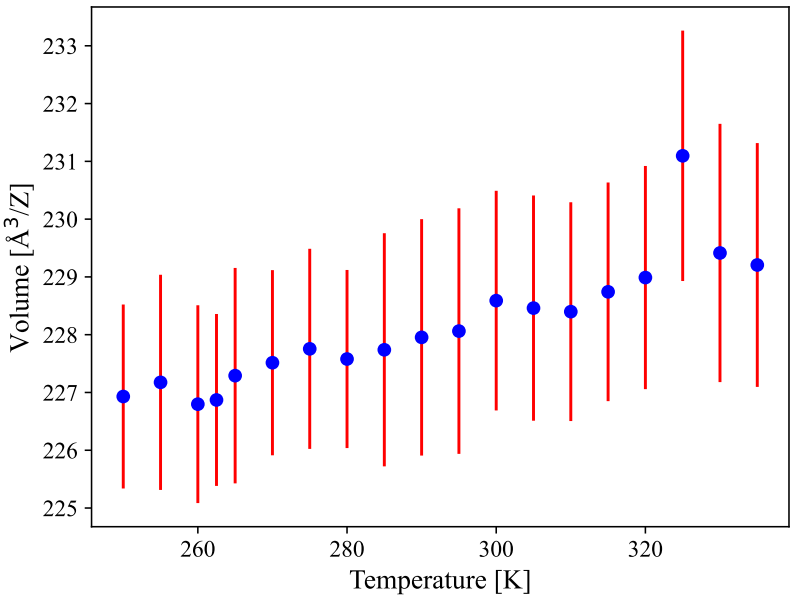


Figure 4.10: Plot of mean volumes vs temperatures for different JAX-MD trajectories. The mean volumes are shown in blue with corresponding standard deviations in red.

Figure 4.10 shows no large drop in volume over the experimental phase transition temperatures at 262-264K. Rather, a gradual increase in the mean volumes can be seen for increasing temperatures. An outlier point can be seen at 325K, with significantly larger mean volumes than the other simulated temperatures.

Figure 4.11 shows the order parameters calculated for the outlier trajectory at 325K.

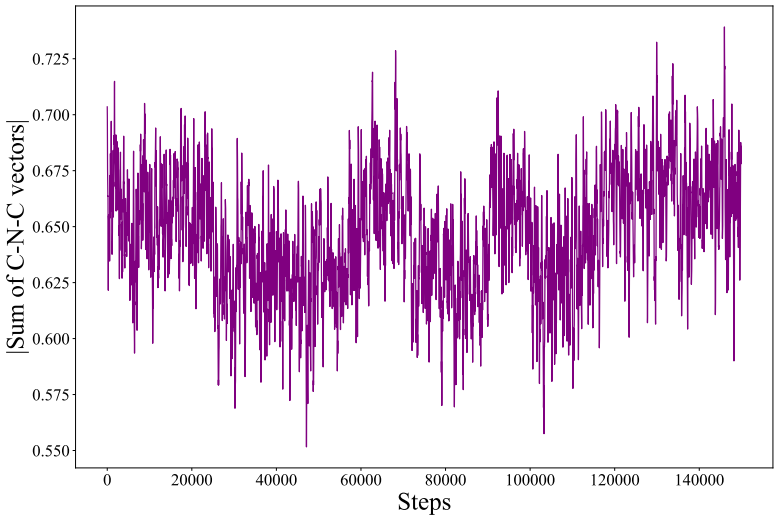


Figure 4.11: Plot of order parameters for a JAX-MD simulation at 325K. The value of the order parameters fluctuates between the different simulation intervals, before returning to the orderer value of 0.66.

.

Figure 4.11 shows small drops in the order parameters during the simulation, before it moves back to the starting values centered around 0.66. The small drops in the parameters indicate some re-orientation towards a more disordered structure. However, the re-orientations are associated with a volume increase, rather than the expected decrease had a full phase transition occurred in the cell. Furthermore, the drop in order parameter value is not significantly large, only changing from approximately 0.66 to 0.625 in the plot. Possibly, only a few molecules in the cell have experienced some structural changes, before they quickly rotate back to an ordered structure.

In contrast to this, the order-parameters of the configurations attained from trajectories at simulation temperatures close to the experimental phase transition range of 262-264K show no changes toward a disorder. Figure 4.12 shows the order parameters of the MD-trajectory simulated at 265K.

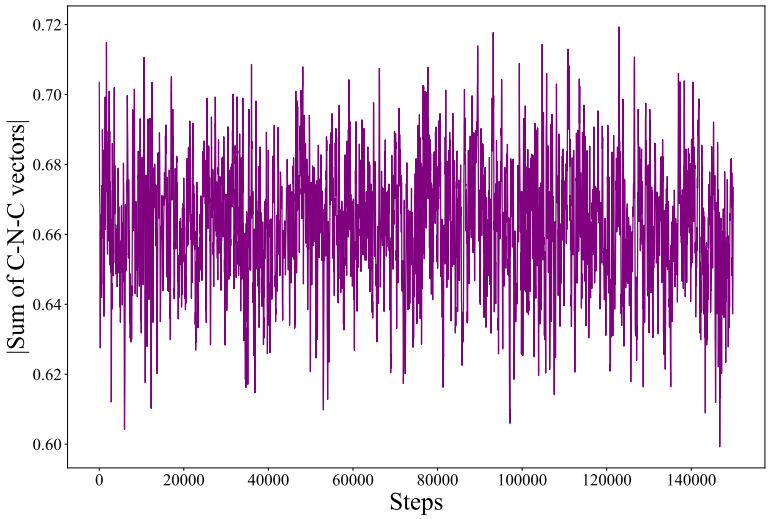


Figure 4.12: Plot of order parameters for a JAX-MD simulation at 265K. The value of the order parameters is centered around a value of 0.66, with small deviations.

The figure clearly shows stability in the order-parameters, with small fluctuations around a value of 0.66, indicating that no structural changes in the dimethylamine molecules gave occurred during the simulation time.

4.3.1 Simulation of high-temperature disordered-phase

Up til now, all simulations were initialized from an ordered arrangement of DMMgF, but what happens if we initialize it from the disordered phase? To explore this, we initialized a simulation with a 144-atom supercell arranged in the disordered phase of the material. Similar to earlier, the simulation was performed with the stable re-trained mixed model 1. Figure 4.13 shows the volume fluctuations of this simulation.

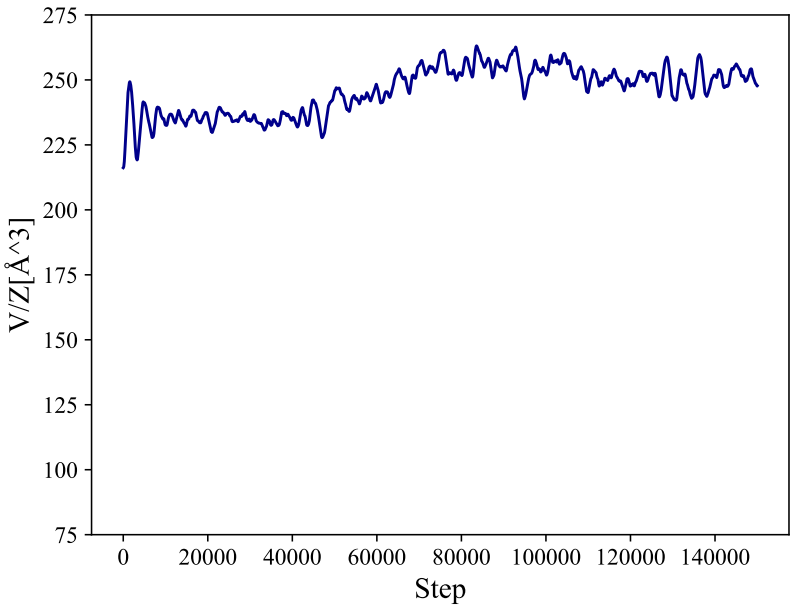


Figure 4.13: Volume fluctuation plotted against simulation step for the simulation performed with a disordered starting simulation structure at 250K. The volume increases over the simulation time frame, but stays reasonably stable.

Not only is the volume evolution stable, but it increases significantly over the simulation time. While this result is peculiar, it might indicate that the disordered structure might be associated with an increase in volume, rather than a decrease when using the flexible cell NPT simulations. This also coincides with the outlier simulation of 325K seeing disordered structural changes in figure 4.11, and a corresponding volume increase in figure 4.10. Figure 4.14 shows the evolution of the order parameters calculated from the trajectory of the simulation started in the disordered phase of DMMgF.

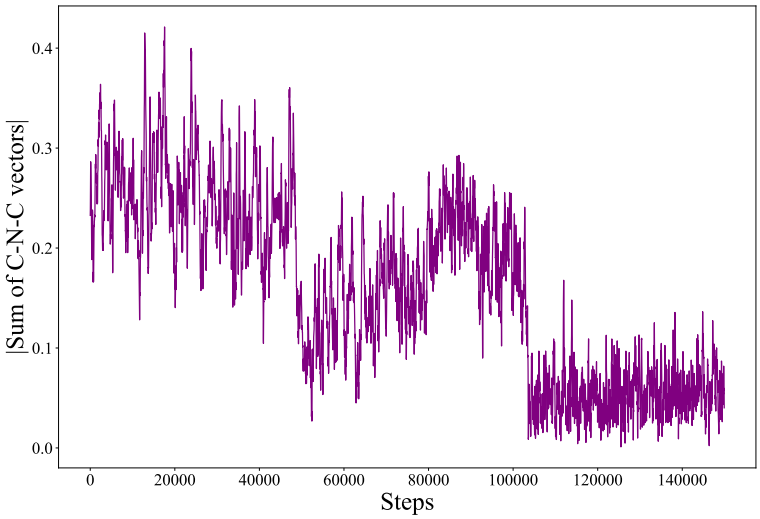


Figure 4.14: Plot of order-parameters for a JAX-MD simulation started in the high-temperature disordered phase of DMMgF. The value of the order-parameter drops over the simulation time.

The order-parameters drop from values of approximately 0.26 to 0.05, indicating that the structure has become even more disordered than the starting simulation structure was. When inspecting the six dimethylamine molecules contained in the configurations, it's clear that some order was still present in the starting simulation structure. Figure 4.15 shows the dimethylamine molecules and their respective middle vectors in the starting simulation structure:

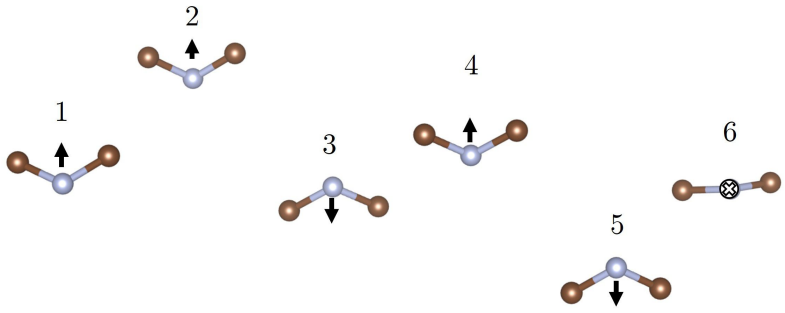


Figure 4.15: The six Dimethylamine molecules in the initial configuration of the DMMgF supercell containing 144 atoms. All hydrogen atoms are emitted for visualization purposes. The middle vector of molecule 6 points into the paper plane, shown by a circle with a cross inside.

As can be seen, pair-wise middle vectors for molecules 2 and 3, and 4 and 5 approximately sum to 0. However, molecules 1 and 6 have vectors that do not sum to 0, meaning a certain order is still present in the configuration. Figure 4.16 shows the last configuration attained in the trajectory, where the order parameters in figure 4.14 has dropped to values close to 0.

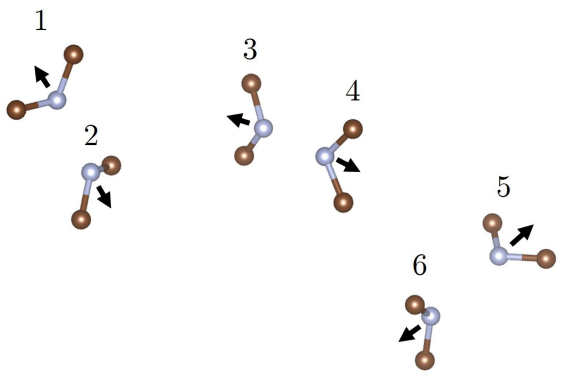


Figure 4.16: The six dimethylamine molecules extracted from the last configuration from the simulation using a disordered starting configuration. All hydrogen atoms are emitted for visualization purposes. Molecule labels 1 to 6 do not denote the same molecules as in figure 4.15, and are only for visualization purposes.

Here, the molecules have re-oriented themselves so that each middle vector approximately sums to values around 0, which coincides with the order parameters in the last simulations steps as shown in figure 4.14. Specifically, molecules 1 and 2, 3 and 4, and molecules 5 and 6 have vectors that point in opposite directions in space, summing their middle vectors to values close to 0. The re-orientation illustrates how NeuralIL used for MD can result in reasonably logical structural changes in DMMgF when the simulation is initialized in the high-temperature disordered phase.

Chapter 5

Discussion

This chapter will discuss the results, starting with the force predictions made by NeuralIL, and the different Jax-MD simulations completed. For the force predictions, most of the focus will be on the results, while for Jax-MD the methods will also be discussed. Then, the discussion will be brought over to the results achieved as part of the search for the phase transition. Finally, general improvements in the framework of NerualIL-based MD will be discussed.

5.1 Force predictions using NeuralIL

The force predictions made by NeuralIL show that the model generally has a robust predictive performance on the DMMgF system. MAE values as low as 0.020 ev/Å was shown in plot (c) in figure 3, which is lower than the mean absolute difference of 0.59 eV/Å in that same test set. The errors can be pushed remarkably small, meaning the predictions come close to the ground truth forces calculated by DFT. While the predictive performance is strong, this can come with some caveats when the model wants to extrapolate to unseen test regions. In this thesis, models following the mixed training method were trained on six data sets, where all test sets were extracted from the same sets as the training sets, see section 3.4.2. This way, all the testing done with a trained mixed model only gave an indication of the model's strong interpolative performance, and not how well it will extrapolate to a region of configurations not seen in training. Figure 5.1 shows an illustration of the extrapolation problem, with data sets and an arbitrary extrapolation region of configurations in configuration space.

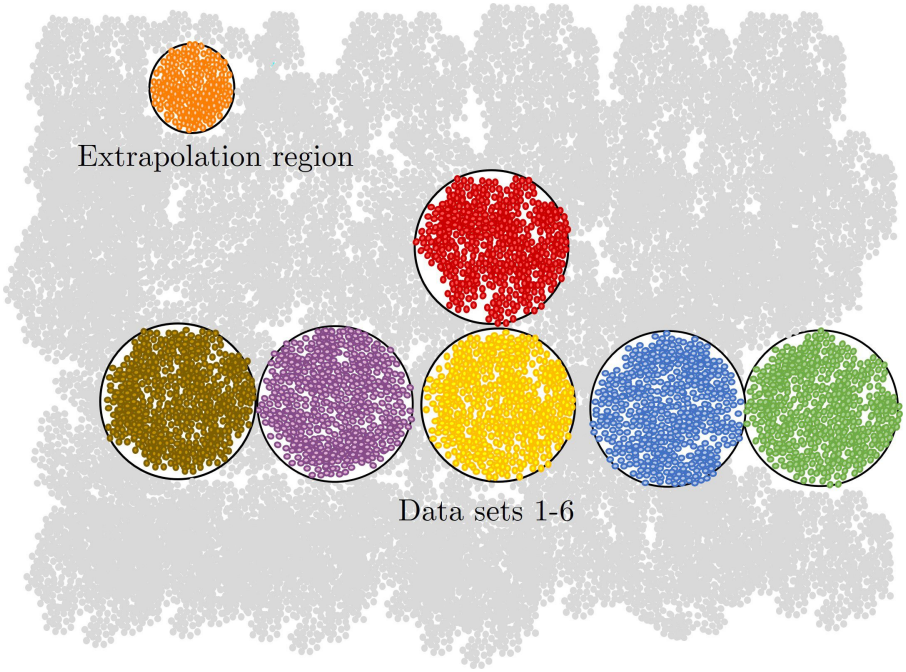


Figure 5.1: A two-dimensional illustration of the configuration space of an MD simulation. Scatter points denote distinct configurations for the DMMgF system that can be attained in an arbitrary MD simulation. The six data sets used in the mixed training method in this thesis are illustrated as circles of scatter points, where all configurations within the circle are contained in the respective data sets. An extrapolation region outside the data sets is shown in orange.

Since the test sets extracted all come from the same regions as the data sets shown in figure 5.1, no indication of the model's performance in such an arbitrary extrapolation region for the DMMgF system can be made from just the force predictions alone. This fact is even more evident when looking at the different results attained from the JAX-MD simulations.

5.2 NeuralIL used for Jax-MD

A series of NeuralIL models have been used to run flexible cell NPT JAX-MD simulations. Inspecting the results, it's clear that the selected models having the smallest mean errors resulted in completely non-physical results when used to simulate JAX-MD. In fact, only one model showed stable behavior, after a re-training procedure with early stopping was completed, as shown by 4.9. The cause of these crashes is likely that the flexible functional forms of an MLFF can make them uncertain when used in MD simulations. For NeuralIL, this is no exception, as subtle errors in the force predictions can become destructive as they accumulate along a calculated MD-trajectory [32]. Simulations can be kept stable if these errors are kept sufficiently small, which is not achieved easily without a robust way to train the NeuralIL models. Several weaknesses in the methodology can be pointed out as reasons for the lack of stability seen in our JAX-MD simulations. Among these are the data sets used in the training of the models.

5.2.1 Training data and model stability

Firstly, it's likely that the training data is quite sparse with regard to simulating Jax-MD for DMMgF. Of the six data sets shown in table 3.1, only one high-temperature data set of 600K was included to account for high-temperature configurations with atoms having small interatomic distances. The other five data sets are only varied in terms of volume by varying the scaling of the DFT-relaxed lattice constants over a small range. Likely, these data sets are not representative of the configuration space illustrated in figure 5.1, causing the model to have high errors in regions not represented by the training data. As discussed in section 5.1, the models in this thesis do not give any indication of extrapolation beyond the training data sets. In the case of simulation in Jax-MD this becomes a problem, as the force predictions we have attained do not give any indication if the models will show stability in the simulations or not. In fact, comparing the error values of the initial and re-trained models in tables 4.3 and 4.2, its clear that mixed model 1 achieved stability when having higher mean errors over the test sets. In other words, it's challenging to assert whether a model will show stability in Jax-MD before any simulation is performed. The selected models we chose from the mixed training to simulate Jax-MD in this study likely suffered from this fact, where the low errors in testing did not correspond to low errors being retained throughout the configurations of the MD simulations, causing eventual crashes.

5.2.2 Overfitting

Another likely reason for the crashing seen in both 4.9 is a possible overfit of the models on the training data. For the model following the simple training method, an overfit was highly likely given that all atomic configurations trained on are extracted from the same data set. Due to this reason, the simple model crashes in both the Jax-MD simulations in figure 4.9, and after re-training in figure 4.9. For the mixed models, an overfit was also likely as seen from mixed model 1 becoming stable after the re-training, shown by figure 4.9.

Reasons for the overfitting have to be seen in both the context of the full training methodology and NeruallL's strong ability to learn from the features contained in the training data. While the model has an incredible amount of potential for interpolation, this also means that the model is more likely to overfit when training architecture is pushed in terms of epochs, layer widths, and mini-batches. Extra care has to be taken when choosing these model parameters in combination with the training data. In our case, we might have included too many configurations in training with little diversity, leading to a high chance of collinearity within the features of the training data, while also combining this with the choice of too many epochs, wide network layers, and small mini-batch sizes. This likely made the models overshoot the bias-variance trade-off, making their extrapolative performance bad. Although an early stopping made us achieve a stable model in the end, earlier incorporation of the technique should have been made to reduce the chance of the initial models being overfitted.

5.2.3 Cut-off radius

In addition to the effects of the training data and possible overfitting on them, the cut-off radius r_{cut} can also have a direct effect on NeuralIL's performance in Jax-MD. As shown by both the initial Jax-MD simulations in figure 4.3, and after model re-training in figure 4.9, the models containing higher r_{cut} values cause simulation crashes, which occurs gradually faster for increasing values of the cut-off. Possible reasons might be that the inclusion of more neighbors

might raise the chance of noisy features being included in the SBDs. Exactly what these noisy features are in the context is hard to determine and may, unfortunately, be a consequence of the loss of the underlying physics when using the general regression functional of MLFFs.

5.3 Exploration of phase transition temperatures

Although earlier studies like the study by Peitao Liu et al. of zirconia [53], have been able to show the phase transition of a solid using MLFF-based MD, this study does not provide a clear observation of the phase transition of DMMgF. While no clear phase transition was shown, the mean volume proves to steadily rise over the temperature regime explored, which is expected to happen following the general curves of figure 3.1. The outlier point at 325K shows that the model at some temperatures rotates certain molecules slightly, indicating that the model can possibly predict structural changes in the system. Furthermore, the additional simulation which was conducted from an initial simulation structure in the high-temperature phase clearly showed that NeuralIL-based MD could predict reasonable re-orientations of the dimethylamine molecules in agreement with what was expected in the high-temperature disordered phase.

Likely, the full phase transition couldn't be reproduced due to simplifications made with regard to the training methodology. While weaknesses in the training methodology already have been discussed, some more comments about the training data should be made in the context of DMMgF's phases. Inspecting the order-parameters of the six data sets used in training of the mixed models, it's clear that most configurations are in the low-temperature phase of the DMMgF system. Again, this makes it unclear if the model is able to extrapolate toward a new region in configuration space not included in the training data, in this case being the other phase we wanted it to transition to.

Among other reasons for the phase transition not showing might be the number of atoms contained in the supercell of the simulation. As seen in a study by Wu et.al. of hafnia [54], the amount of atoms in the supercell directly affects the probability distribution of atomic displacements in the cell for a given temperature range. Exploring variations of larger supercells, longer simulation times, and larger temperature regions could make the probability of observing a phase transition higher due to nucleation growth.

Finally, the methodology using flexible cell simulation is relatively new and might induce unknown factors into the simulation compared to methods with different ensembles. This is evidenced further by the volume increase for the simulation starting in the disordered high-temperature phase, while a decrease coinciding with figure 3.1 was expected for the structural changes observed. It's therefore quite possible that searching for the phase transition with an NPT ensemble without flexible cells could possibly show the phase transition clearly.

5.4 General improvements on the NeuralIL framework

As shown by the results in this master's thesis, it's highly likely that an initially trained NeuralIL model might show results of non-physical results when used for MD simulations. For any future studies using NeuralIL to simulate MD, there are suggestions for improvements on the general framework used in this study.

5.4.1 Training data

Firstly, significant thought should be given to the training data included in the NeuralIL models used for simulations. Since the model's interpolative capacity is so strong, diverse training data is needed to ensure that the model can extrapolate considerably well in MD simulations. While it can be easy to state that good data sets are needed for a model to be robust, exactly what data sets are necessary for a model to achieve the desired performance in MD can be challenging to assert for any given problem. Thankfully, efforts to tackle this have been explored in the literature recently. The newest NeuralIL article [32] has come up with a framework, where high-information configurations are selected and reintroduced into NeuralIL's training based on uncertainty estimations. The method uses deep ensembles [55] of NeuralIL models trained on a small number of configurations to perform several NeuralIL-based MD simulations. Several of the resulting trajectories that contain configurations with high values of the uncertainty estimators have those configurations selected and re-introduced into model training [32]. The models that were re-trained with high-error configurations show stability in new MD simulations performed, where simulation crashes and un-physical results were previously observed [32]. Such a framework focuses on the gradual introduction of highly valuable training configurations to an initially trained model, achieving robust performance and efficient training. Furthermore, the chance of overfitting may also be reduced by restricting the number of configurations, while also increasing their diversity within the material's configurational space. While this framework may ensure stability in the models, the inclusion of specific data sets might also be a good idea in certain problems where complex material behavior is the wanted result. For example, the inclusion of disordered structures in an already stable NeuralIL model may have given it an advantage with regard to predicting the phase transition of DMMgF in our study.

5.4.2 Overfitting

Along with better frameworks for including representative and diverse training data, real care has to be made with regard to model overfitting. While the deep ensemble methodology might reduce the chance of potential overfitting, techniques like early stopping might also be used in parallel to minimize potential overfitting possibilities. Furthermore, the introduction of L1 or L2 regularization and dropout layers [26, p.536] in the ResNet architecture of NeuralIL may also be considered for the same reasons.

Additionally, feature selection techniques might be explored to ensure only the necessary features attained from the SBDs are included. While the SBDs are optimally complete, in cases where training configurations may be significantly similar, feature selection tools might be useful in data pre-processing before training the NeuralIL models. Moreover, feature selection might also be a valid approach for models trained with higher values of the cut-off radius, given

that noisy features may become apparent.

5.4.3 Parameter choices

Based on our study, initial parameter choices of $r_{cut} = 3.5$ Å and $n_{max} = 4$ might be reasonable choices when NeuralIL is used for MD simulations of DMMgF. While models with other parameters achieved higher predictive performance, our only stable model was achieved with these parameters, validating these parameter choices for further use of NeuralIL-based MD on DMMgF. However, values of $r = 5$ Å has seen stable performance for other materials, for example in the studies of hafnia by Bichelmaier [21], suggesting that higher values of the cut-off radius should not be ruled out without further exploration.

Chapter 6

Conclusion

6.1 Conclusion

In this master's thesis, a machine learning force field called NeuralIL has been used to predict atomic forces of a hybrid organic-inorganic perovskite system (DMMgF). The model predictions have been used to access its applicability in molecular dynamics simulations for the material, where a flexible cell environment of Jax-MD was used. A model showing stability in Jax-MD simulations was selected for an extended analysis, where it was assessed if NeuralIL-based MD simulations could reproduce the phase transition of DMMgF in accordance with experimental data.

NeuralIL's force predictions show how the model is capable of reproducing atomic forces for the system comparable to ab initio calculations made by DFT. Force predictions with MAE as low as 0.020 eV/Å were seen in testing, emphasizing how the model is highly-capable of predicting the interatomic forces accurately for solids.

Results from the MD simulations show that NeuralIL has the capability to perform stable MD simulations comparable with traditional methods. Despite achieving stable simulations, the work also illustrated the challenges associated with MLFF-based MD simulations, where many simulations crashed due to the accumulation of predictive errors. Improving the framework to avoid such crashes in future simulations is heavily dependent on the inclusion of more representative data sets, and reducing the chances of NeuralIL overfitting on them.

The phase transition of DMMgF was not shown in accordance with experimental data. Although a definitive phase transition was not observed, certain simulations showed positive indications of molecular rotations concurrent with what was expected in the high-temperature phase of the system. Further work could explore if these indications are just random simulation artifacts, or if alternative simulation techniques and more robust training methodologies potentially could capture the phase transition accurately.

6.2 Further work

There are many suggestions for further work that can be made based on this study. Perhaps most important are the inclusions of more representative data sets for the model to train on. More varied data sets would likely reduce the chance of the model overfitting, and make it extrapolate better to the unseen configurations that are seen in MD simulations.

Ways to include such training data sets without performing expensive DFT-based MD calculations have already been explored in the literature. A study by Carrete et.al. [32], shows how configurations with high uncertainties can be included in the training through deep-ensemble methods, allowing the model to improve its stability in MD simulations. This kind of active learning approach could serve as an efficient method to ensure the stability of NeuralIL models used for MD simulations and should therefore be explored in future studies.

Additionally, regularization and feature selection methods could be introduced into NeuralIL as methods to reduce the chances of overfitting.

Finally, there are other improvements that could be made for the analysis to possibly show the phase transition more accurately. NPT ensembles without the flexible cell implementation could be used to see if the reason for the phase transition not showing was due to the novelty of the ensemble methods used. Longer simulations and larger supercells could also be explored, as they possibly make an observation of the phase transition more likely due to nucleation growth.

Bibliography

[1] Peter J Wellmann. *The search for new materials and the role of novel processing routes*. 2021. URL: https://link.springer.com/article/10.1007/s43939-021-00014-y.

[2] Fraunhofer Institute for Solar Energy Systems ISE. *Photovoltaics Report*. Tech. Rep. Fraunhofer ISE, 2023. URL: https://www.ise.fraunhofer.de/en/publications/studies/photovoltaics-report.html.

[3] Guus J. M. Velders, A. R. Ravishankara, Melanie K. Miller, Mario J. Molina, Joseph Alcamo, John S. Daniel, et al. "Preserving Montreal Protocol Climate Benefits by Limiting HFCs". In: *Science* 335.6071 (2012), pp. 922–923. DOI: 10.1126/science.1216414. eprint: https://www.science.org/doi/pdf/10.1126/science.1216414. URL: https://www.science.org/doi/abs/10.1126/science.1216414.

[4] David Boldrin. "Fantastic barocalorics and where to find them". In: *Applied Physics Letters* 118 (Apr. 2021), p. 170502. DOI: 10.1063/5.0046416.

[5] Hannah Ritchie, Max Roser, and Pablo Rosado. "Energy". In: *Our World in Data* (2022). URL: https://ourworldindata.org/energy.

[6] H.-O. Pörtner, D.C. Roberts, M. Tignor, E.S. Poloczanska, K. Mintenbeck, A. Alegría, et al., eds. *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge, UK and New York, NY, USA: Cambridge University Press, 2022, p. 3056. DOI: 10.1017/9781009325844.

[7] Mark Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. 1st. New York, NY: Oxford University Press, 2010.

[8] "Accuracy and Methods beyond "Standard" Calculations". In: *Density Functional Theory*. John Wiley Sons, Ltd, 2009. Chap. 10, pp. 209–233. ISBN: 9780470447710. DOI: https://doi.org/10.1002/9780470447710.ch10.

[9] Radu Iftimie, Peter Minary, and Mark E. Tuckerman. "Ab initio molecular dynamics: Concepts, recent developments, and future trends". In: *Proceedings of the National Academy of Sciences* 102.19 (2005), pp. 6654–6659. DOI: 10.1073/pnas.0500193102.

[10] Thomas Kühne. "Second generation Car–Parrinello molecular dynamics". In: *Wiley interdisciplinary reviews: Computational Molecular Science.* 4 (July 2014), p. 391. DOI: 10.1002/wcms.1176.

[11] Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, et al. "Machine Learning Force Fields". In: *Chemical Reviews* 121.16 (2021). PMID: 33705118, pp. 10142–10186. DOI: 10.1021/acs.chemrev.0c01111.

[12] Albert P. Bartó k, Mike C. Payne, Risi Kondor, and Gábor Csányi. "Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons". In: *Physical Review Letters* 104.13 (Apr. 2010). DOI: 10.1103/physrevlett.104.136403.

[13] Aldo Glielmo, Claudio Zeni, and Alessandro De Vita. "Efficient nonparametric n-body force fields from machine learning". In: *Physical Review B* 97.18 (May 2018). DOI: 10.1103/physrevb.97.184307.

[14] Yu Xie, Jonathan Vandermause, Lixin Sun, Andrea Cepellotti, and Boris Kozinsky. "Bayesian force fields from active learning for simulation of inter-dimensional transformation of stanene". In: *npj Computational Materials* 7 (Mar. 2021), p. 40. DOI: 10.1038/s41524-021-00510-y.

[15] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan Ee. "Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics". In: *Physical Review Letters* 120 (July 2017). DOI: 10.1103/PhysRevLett.120.143001.

[16] Hadrián Montes-Campos, Jesús Carrete, Sebastian Bichelmaier, Luis M. Varela, and Georg K. H. Madsen. "A Differentiable Neural-Network Force Field for Ionic Liquids". In: *Journal of Chemical Information and Modeling* 62.1 (2022). PMID: 34941253, pp. 88–101. DOI: 10.1021/acs.jcim.1c01380.

[17] Kwang-Hwi Cho, Kyoung Tai No, and Harold A. Scheraga. "A polarizable force field for water using an artificial neural network". In: *Journal of Molecular Structure* 641.1 (2002), pp. 77–91. ISSN: 0022-2860. DOI: https://doi.org/10.1016/S0022-2860(02)00299-5.

[18] Marek Szafrański, Wen-Juan Wei, Zhe-Ming Wang, Wei Li, and Andrzej Katrusiak. "Research Update: Tricritical point and large caloric effect in a hybrid organic-inorganic perovskite". In: *APL Materials* 6.10 (2018), p. 100701. DOI: 10.1063/1.5049116.

[19] Ralf Schneider, Amit Raj Sharma, and Abha Rai. "Introduction to Molecular Dynamics". In: *Computational Many-Particle Physics*. Ed. by H. Fehske, R. Schneider, and A. Weiße. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–40. ISBN: 978-3-540-74686-7. DOI: 10.1007/978-3-540-74686-7_1. URL: https://doi.org/10.1007/978-3-540-74686-7_1.

[20] JE Lennard-Jones. "On the Determination of Molecular Fields. II. From the Equation of State of a Gas". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106.738 (1924), pp. 463–477.

[21] Sebastian Bichelmaier. "Ab-initio modelling of material properties using elements of artificial intelligence". Dissertation. repositUm: Technische Universität Wien, 2023. DOI: 10.34726/hss.2023.90200.

[22] Lauri Himanen, Marc O.J. Jäger, Eiaki V. Morooka, Filippo Federici Canova, Yashasvi S. Ranawat, David Z. Gao, et al. "DScribe: Library of descriptors for machine learning in materials science". In: *Computer Physics Communications* 247 (2020), p. 106949. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2019.106949.

[23] Albert P. Bartók, Risi Kondor, and Gábor Csányi. "On representing chemical environments". In: *Phys. Rev. B* 87 (18 May 2013), p. 184115. DOI: 10.1103/PhysRevB.87.184115.

[24] Marcin Novotni and Reinhard Klein. "3D Zernike Descriptors for Content Based Shape Retrieval". In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM '03. Seattle, Washington, USA: Association for Computing Machinery, 2003, pp. 216–225. ISBN: 1581137060. DOI: 10.1145/781606.781639.

[25] Emir Kocer, Jeremy K. Mason, and Hakan Erturk. "Continuous and optimally complete description of chemical environments using Spherical Bessel descriptors". In: *AIP Advances* 10.1 (2020). DOI: 10.1063/1.5111045.

[26] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning, 3rd Ed.* Birmingham, UK: Packt Publishing, 2019, p. 748. ISBN: 978-1789955750.

[27] Alekseĭ Grigorevich Ivakhnenko and Valentin Grigorevich Lapa. "CYBERNETIC PREDICTING DEVICES". In: 1966.

[28] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

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

[30] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[31] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[32] Jesús Carrete, Hadrián Montes-Campos, Ralf Wanzenböck, Esther Heid, and Georg K. H. Madsen. "Deep ensembles vs committees for uncertainty estimation in neural-network force fields: Comparison and application to active learning". In: *The Journal of Chemical Physics* 158.20 (May 2023). ISSN: 0021-9606. DOI: 10.1063/5.014690.

[33] Dongwei Chen, Fei Hu, Guokui Nian, and Tiantian Yang. "Deep Residual Learning for Nonlinear Regression". In: *Entropy* 22.2 (2020). ISSN: 1099-4300. DOI: 10.3390/e22020193.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. DOI: 10.1109/CVPR.2016.90.

[35] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3.

[36] Luke Metz, James Harrison, C. Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, et al. *VeLO: Training Versatile Learned Optimizers by Scaling Up*. 2022. arXiv: 2211.09760 [cs.LG].

[37] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[38] Geoffrey E. Hinton. *Neural Networks for Machine Learning: Lecture 6a, Overview of mini-batch gradient descent*. Retrieved from https://www.youtube.com/watch?v=O3sxAc4hxZU. 2012.

[39] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2014. arXiv: 1308.0850 [cs.NE].

[40] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[41] Y. Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 5 (Feb. 1994), pp. 157–66. DOI: 10.1109/72.279181.

[42] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[43] *Flax: A neural network library and ecosystem for JAX designed for flexibility*. URL: https://github.com/google/flax.

[44] Igor Babuschkin et al. 2020. URL: https://github.com/deepmind.

[45] Matteo Hessel David Budden. "Using JAX to accelerate our research". In: (2020). URL: https://www.deepmind.com/blog/using-jax-to-accelerate-our-research.

[46] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. *Automatic differentiation in machine learning: a survey*. 2018. arXiv: 1502.05767 [cs.SC].

[47] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[48] Samuel S. Schoenholz and Ekin D. Cubuk. *JAX, M.D.: A Framework for Differentiable Physics*. 2020. arXiv: 1912.04232 [physics.comp-ph].

[49] G. Kresse and J. Hafner. "Ab initio molecular dynamics for liquid metals". In: *Phys. Rev. B* 47 (1 Jan. 1993), pp. 558–561. DOI: 10.1103/PhysRevB.47.558.

[50] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, and Marcin Dułak et al. "The atomic simulation environment—a Python library for working with atoms". In: *Journal of Physics: Condensed Matter* 29.27 (2017). Url: http://stacks.iop.org/0953-8984/29/i=27/a=273002, p. 273002.

[51] K. Momma and F. Izumi. "VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data". In: *J. Appl. Crystallogr.* 44 (2011). Url: http://stacks.iop.org/0953-8984/29/i=27/a=273002, pp. 1272–1276.

[52] Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_5. URL: https://doi.org/10.1007/978-3-642-35289-8_5.

[53] Peitao Liu, Carla Verdi, Ferenc Karsai, and Georg Kresse. "Phase transitions of zirconia: Machine-learned force fields beyond density functional theory". In: *Phys. Rev. B* 105 (6 Feb. 2022), p. L060102. DOI: 10.1103/PhysRevB.105.L060102.

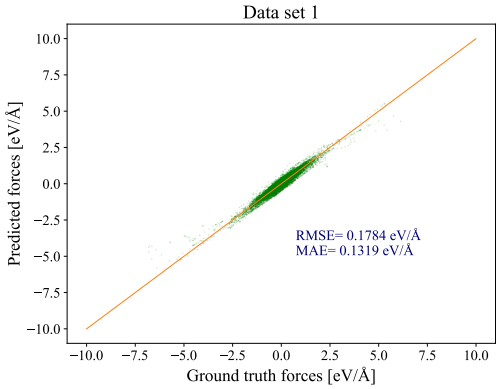
[54] Jing Wu, Yuzhi Zhang, Linfeng Zhang, and Shi Liu. "Deep learning of accurate force field of ferroelectric HfO_2". In: *Phys. Rev. B* 103 (2 Jan. 2021), p. 024108. DOI: 10.1103/PhysRevB.103.024108.

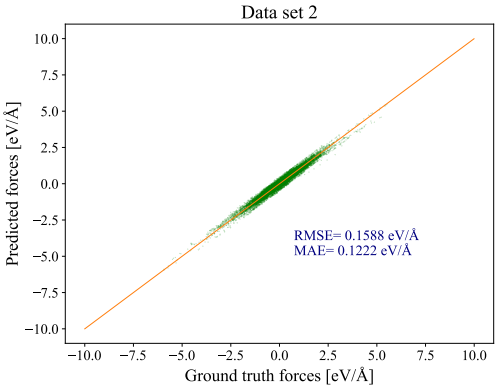
[55] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. 2017. arXiv: 1612.01474 [stat.ML].

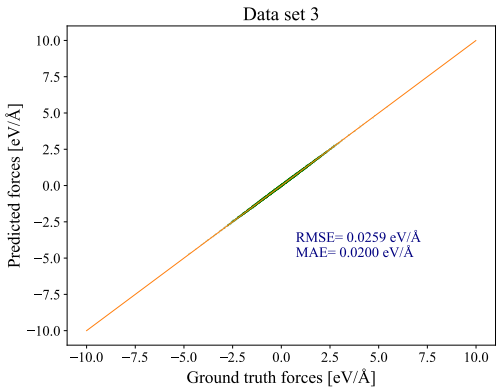
Appendix A

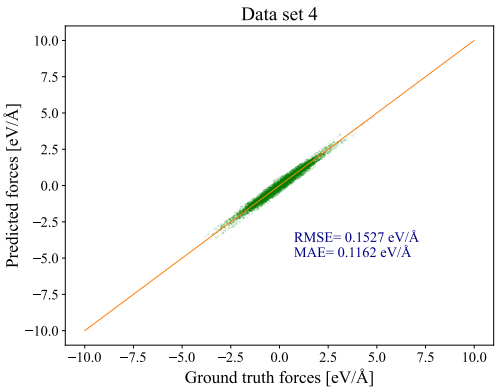
Appendix A lists all parity plots of all predictions over all six test sets from table 3.1 made by all initial and re-trained models. Each page has predictions made by one model, where the figure title names the respective model. All parity plots are denoted with titles signifying which data set of table 3.1 the test sets are extracted from.

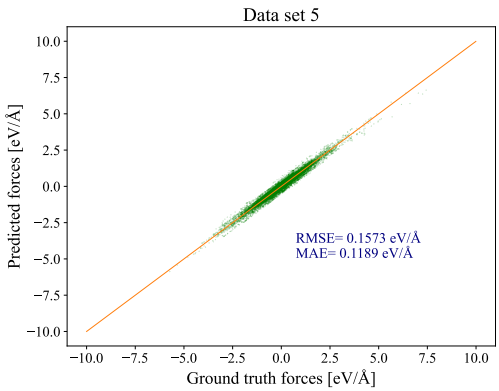
Simple model 1

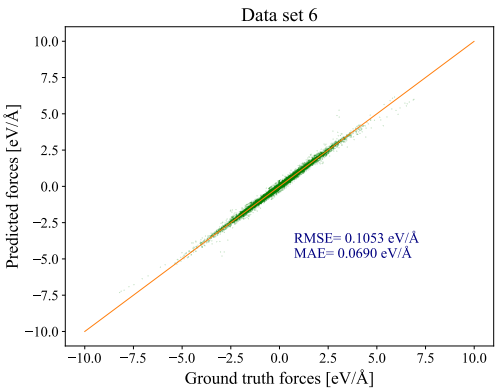




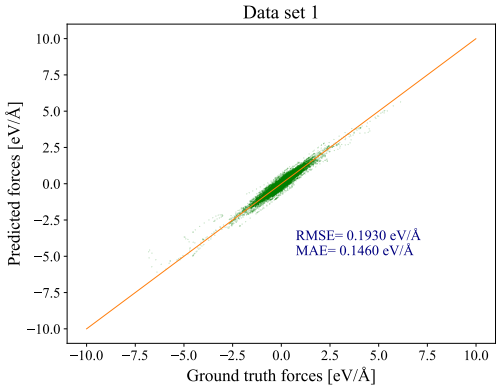


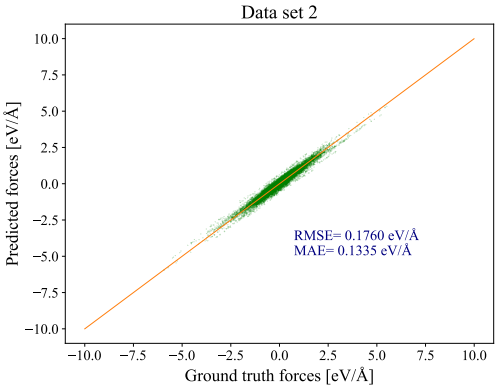


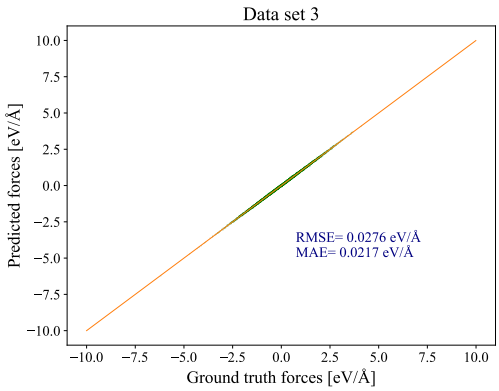


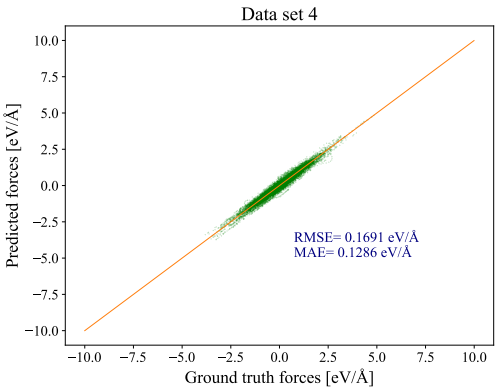


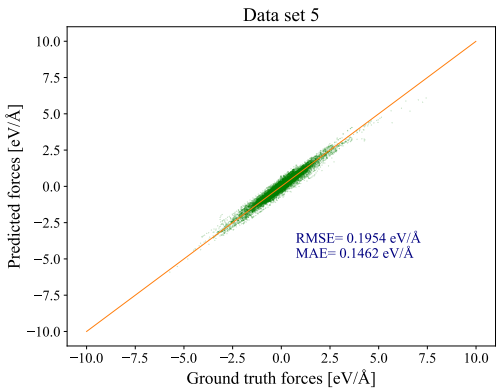
Simple model 2

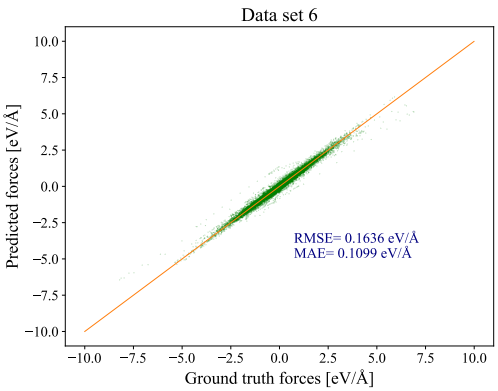




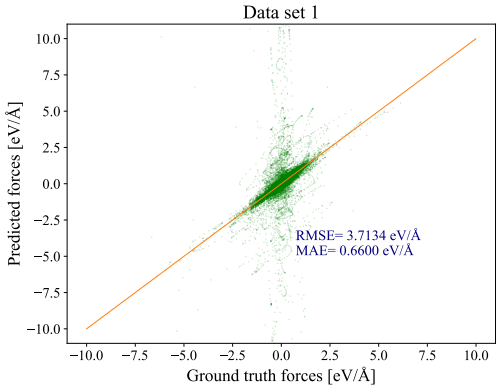


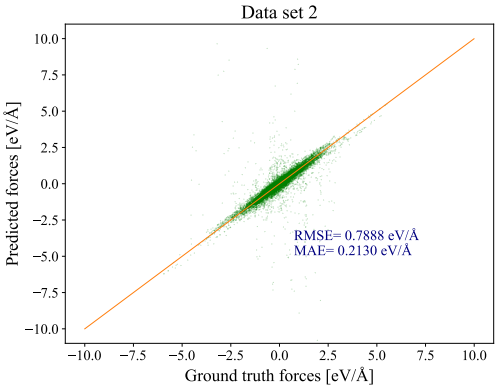


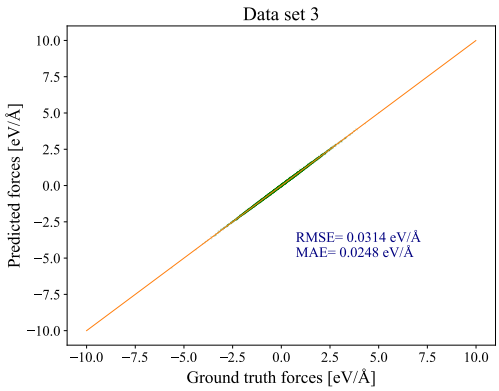


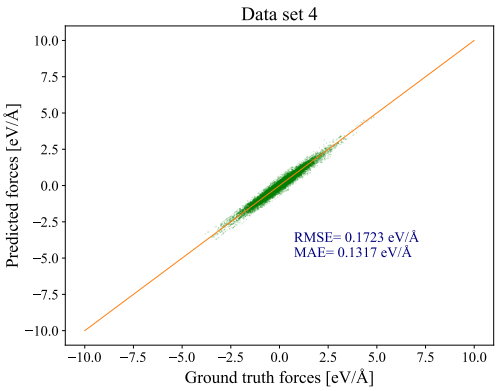


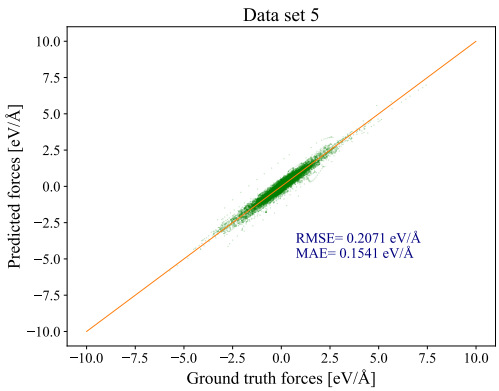
Simple model 3

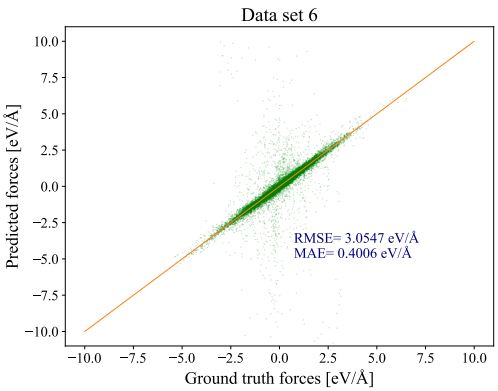




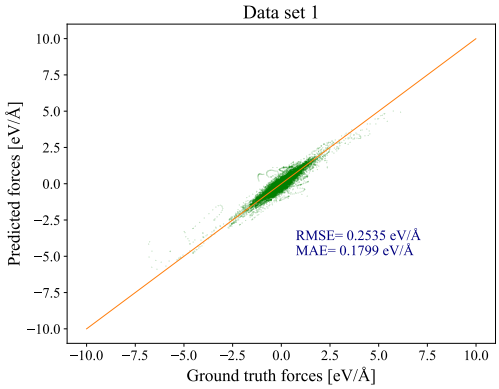


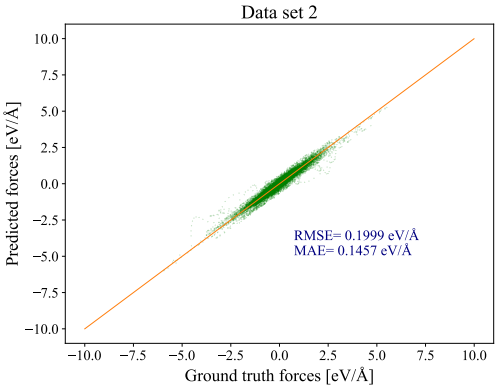


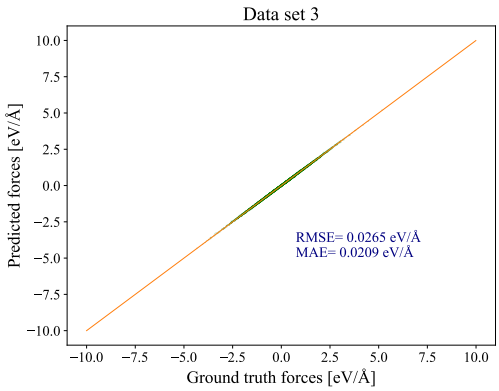


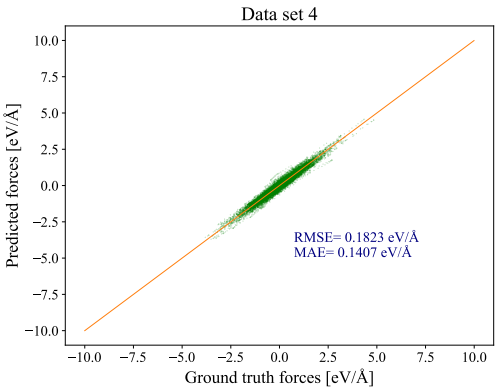


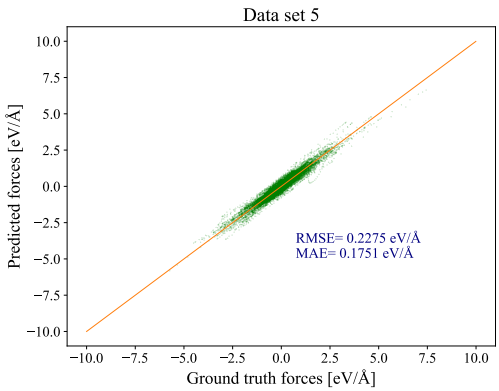
Simple model 4

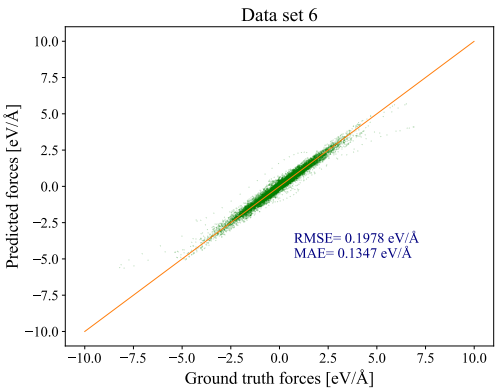




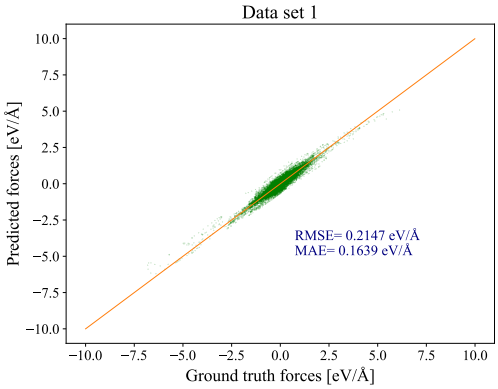


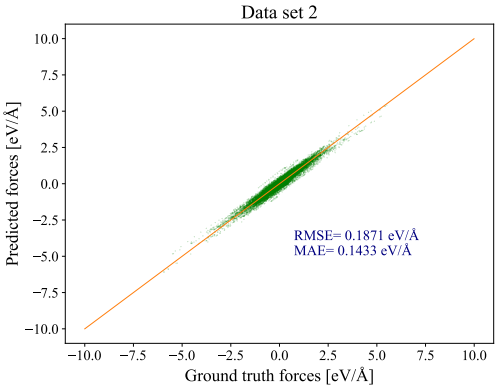


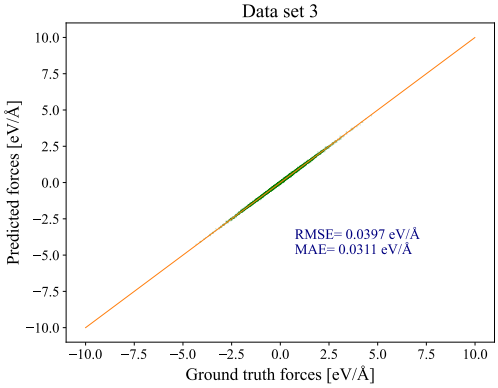


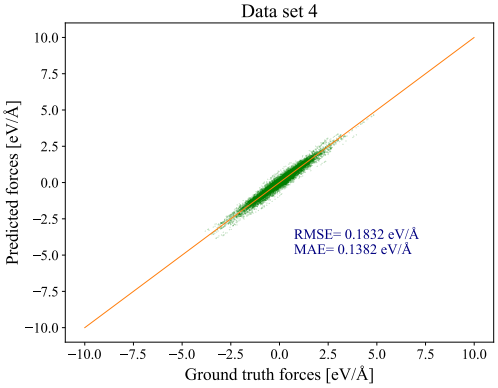


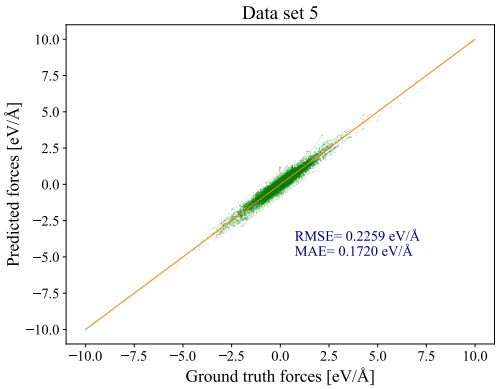
Simple model 5

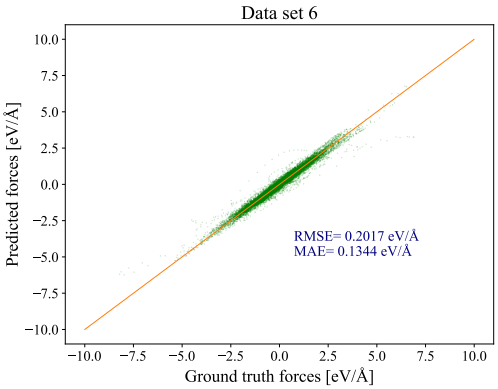




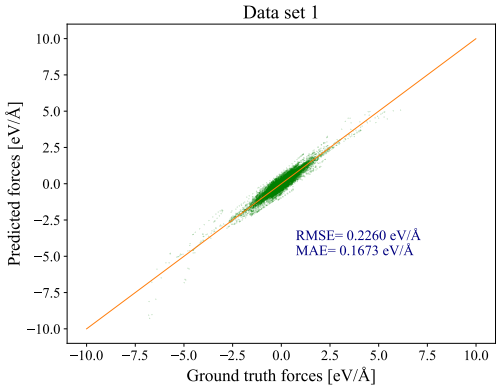


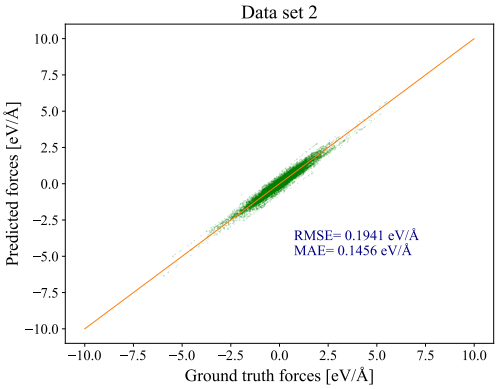


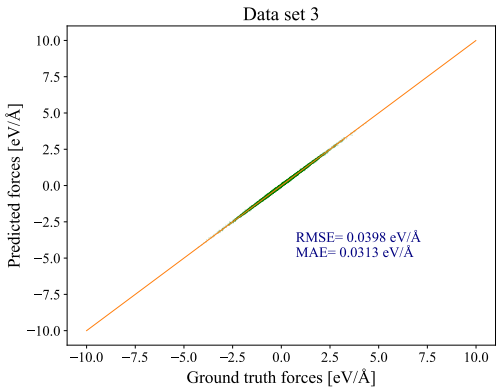


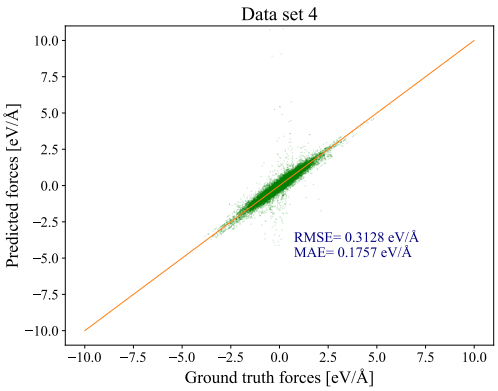


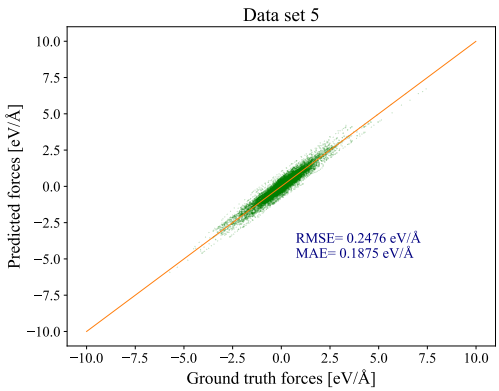
Simple model 6

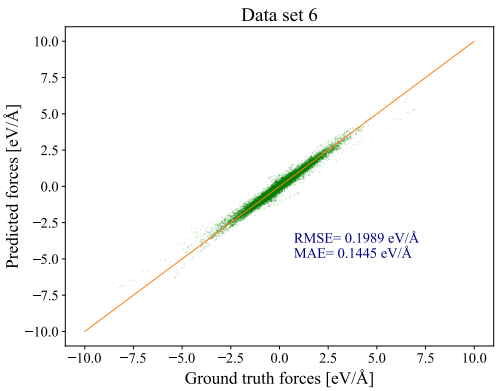




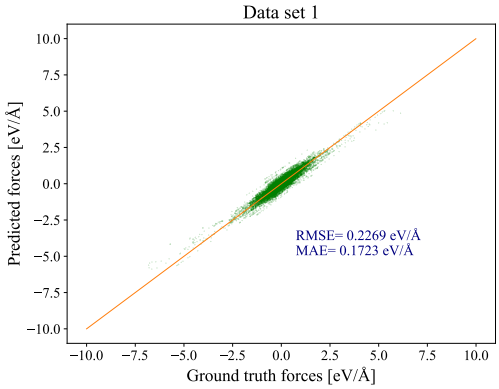


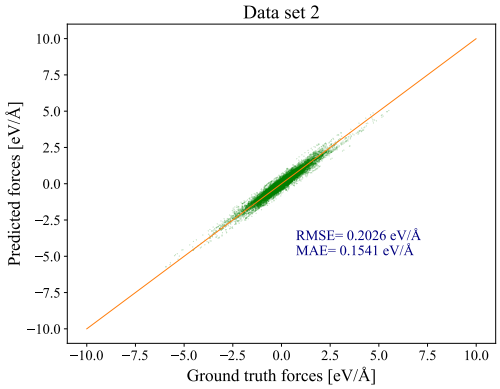


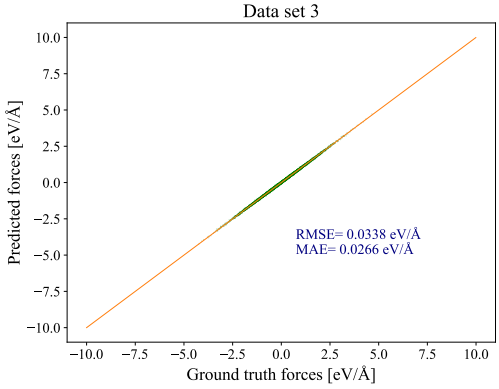


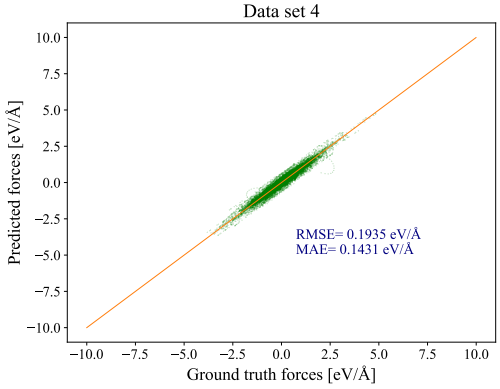


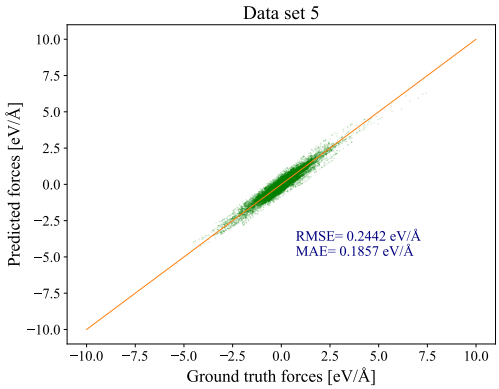
Simple model 7

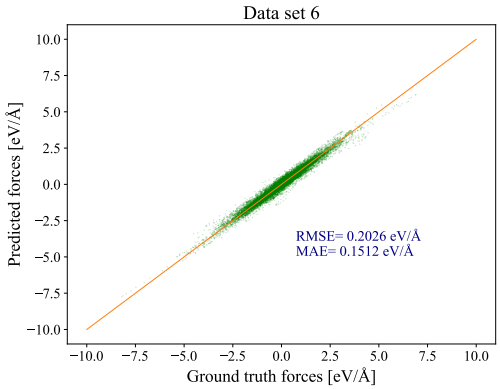




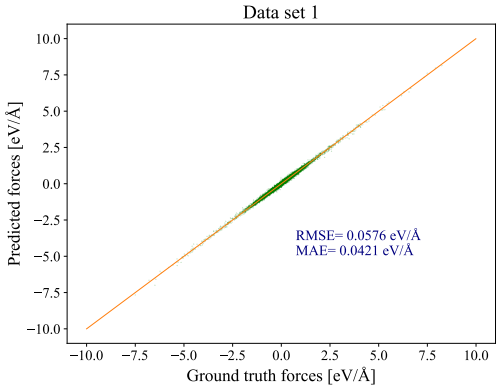


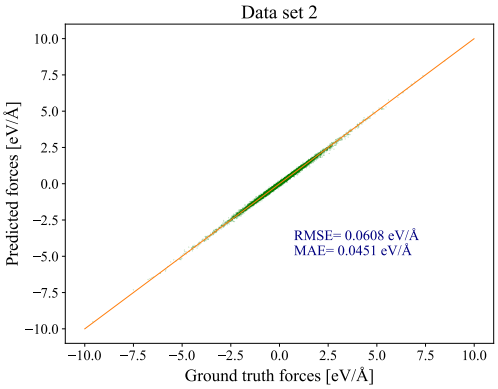


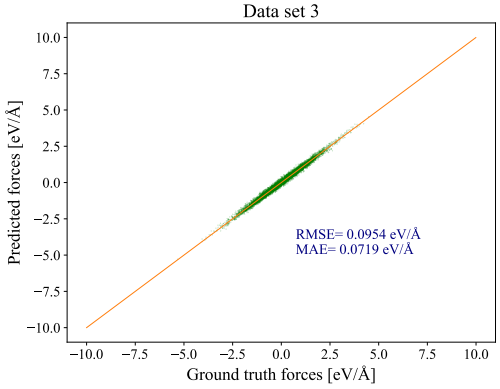


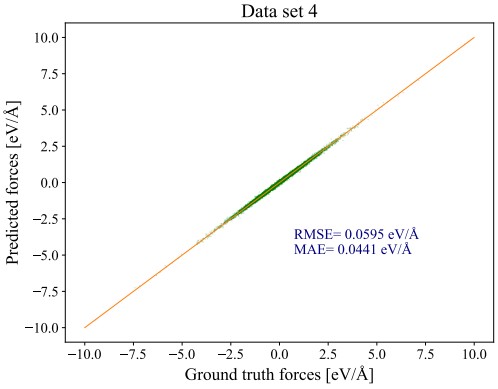


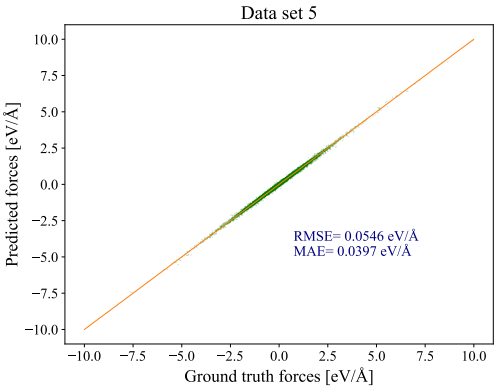
Mixed model 1

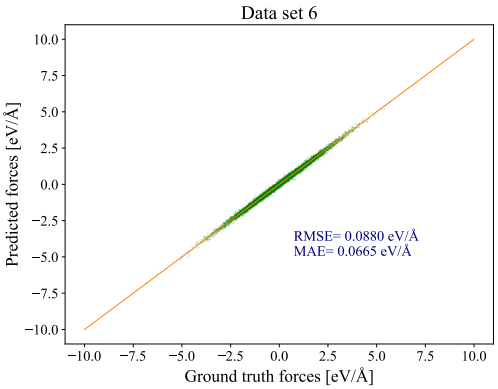




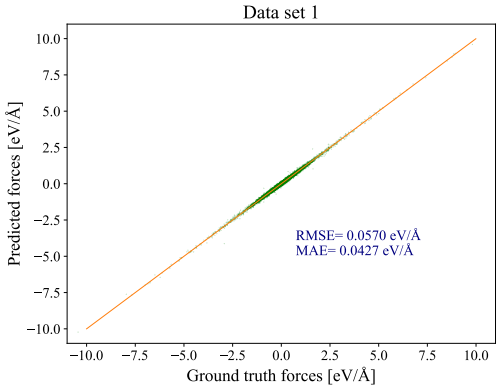


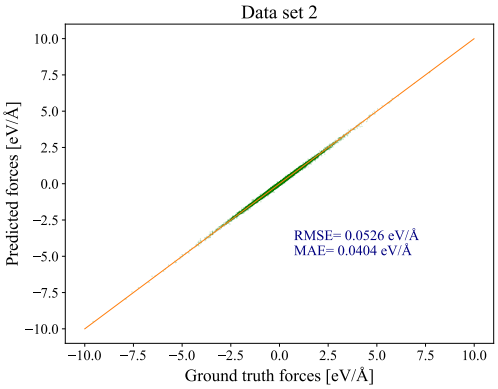


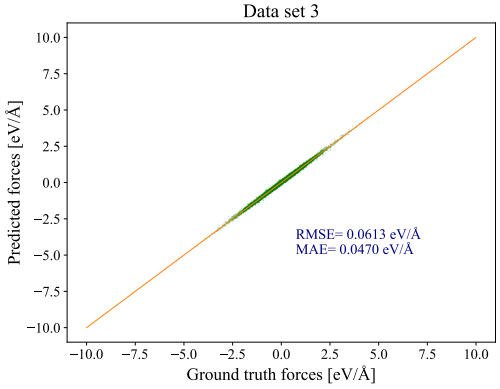


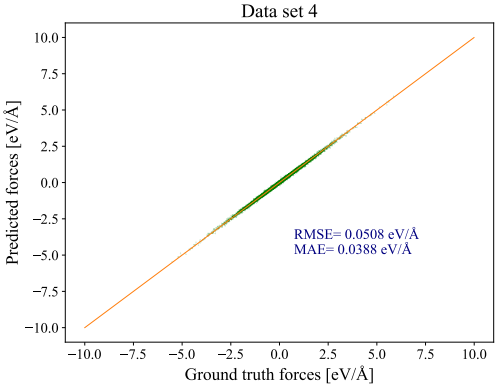


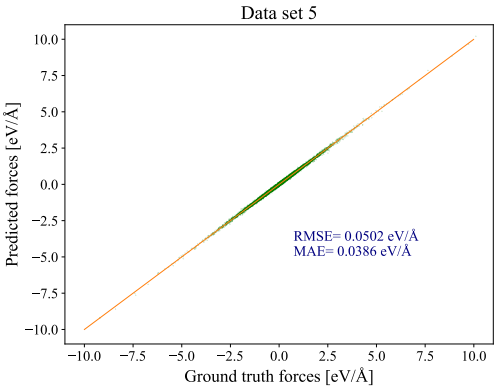
Mixed model 2

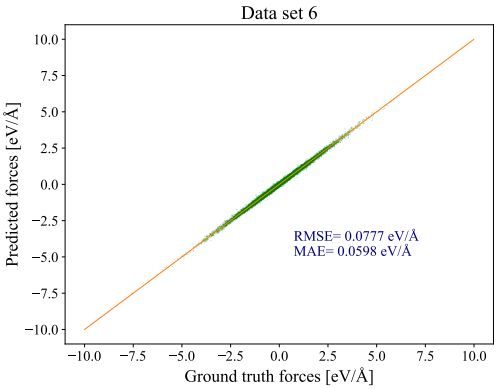




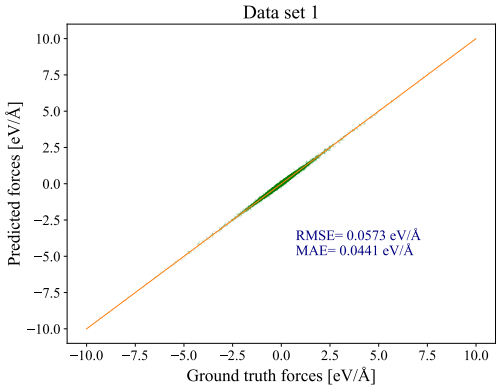


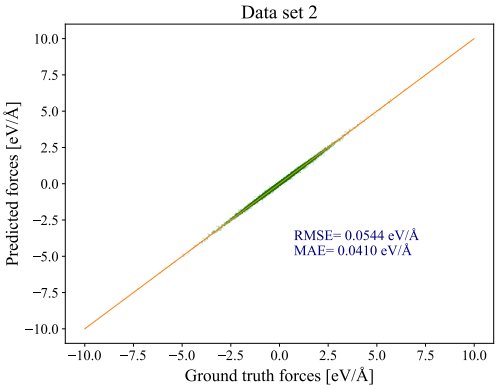


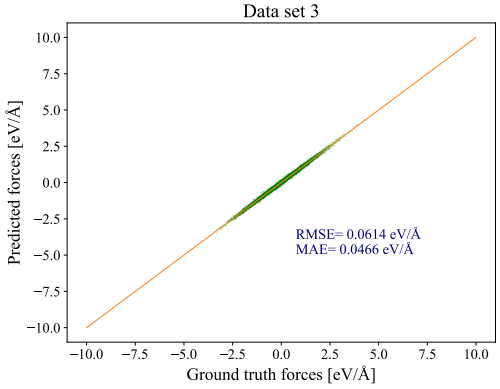


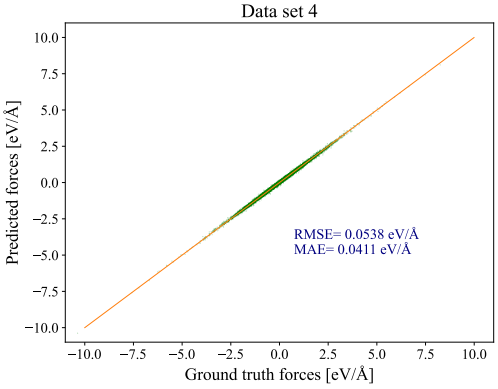


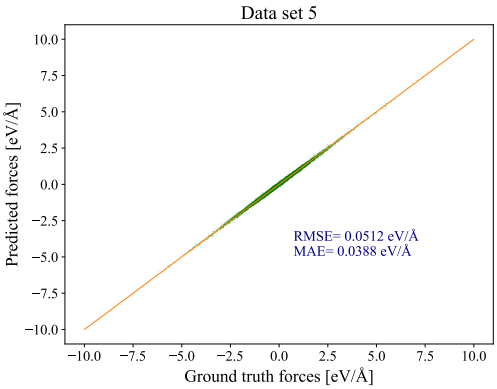
Mixed model 3

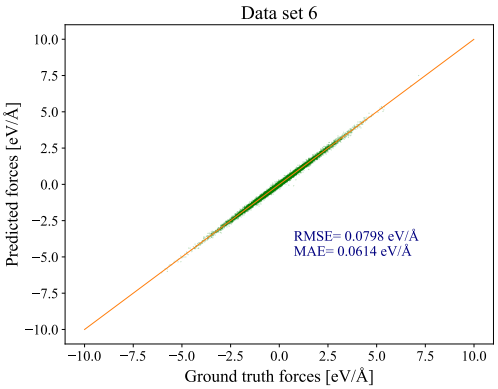




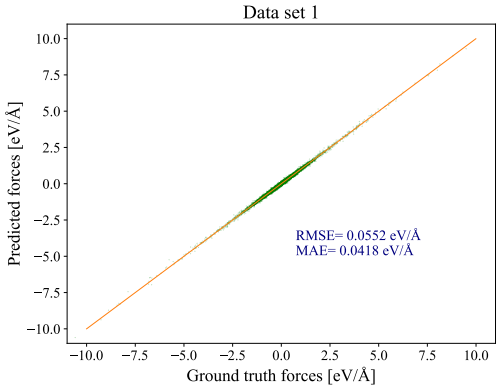


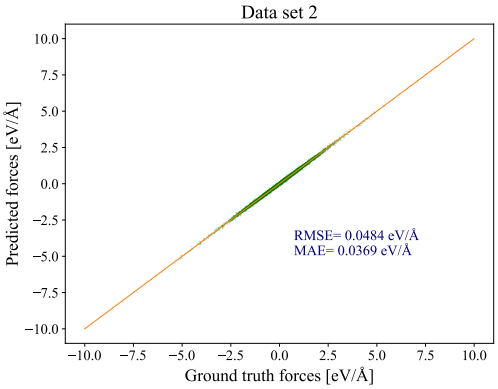


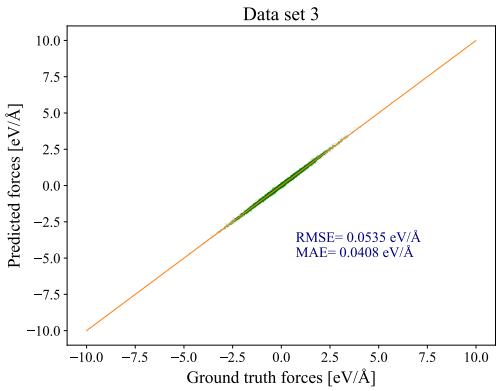


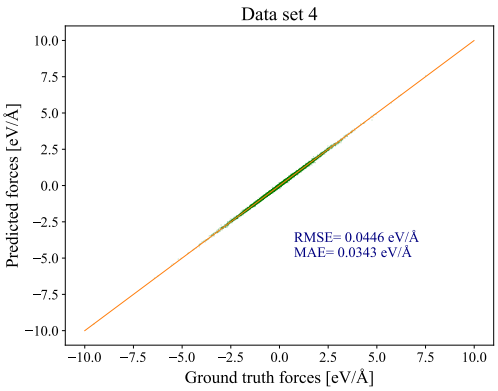


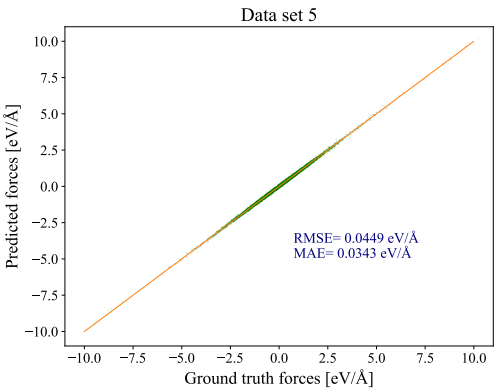
Mixed model 4

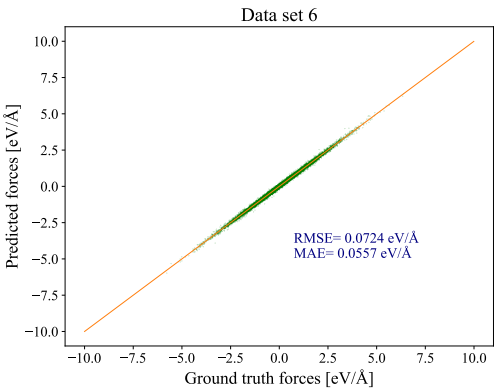




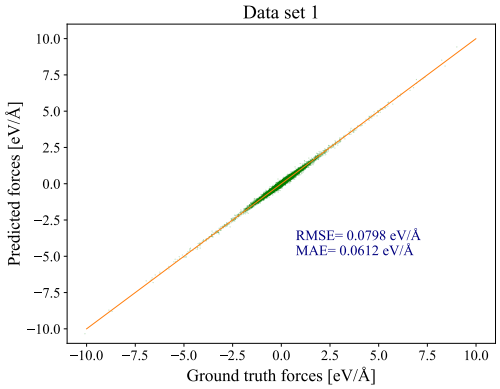


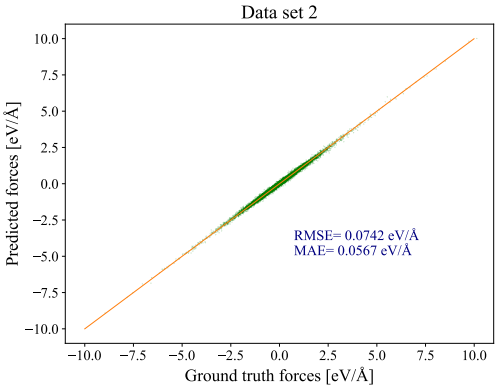


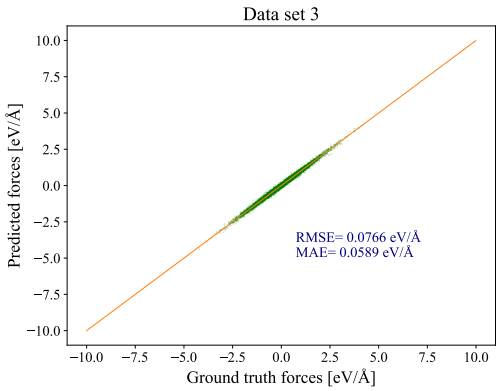


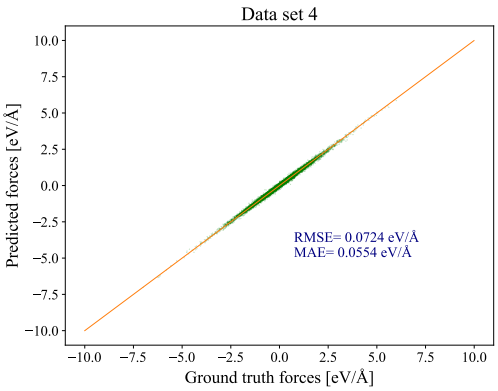


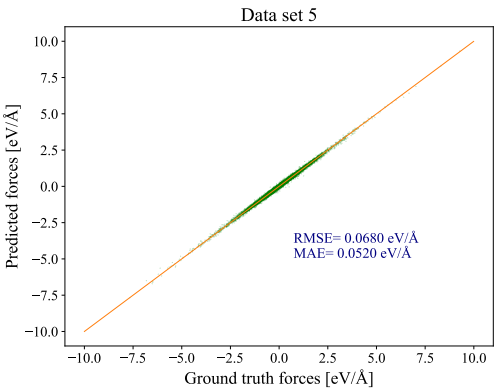
Mixed model 5

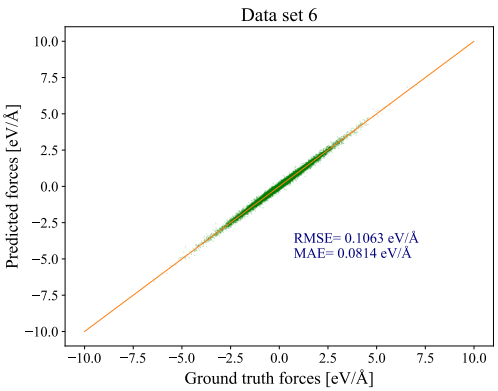




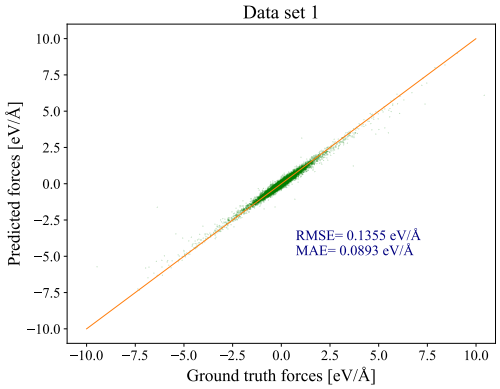


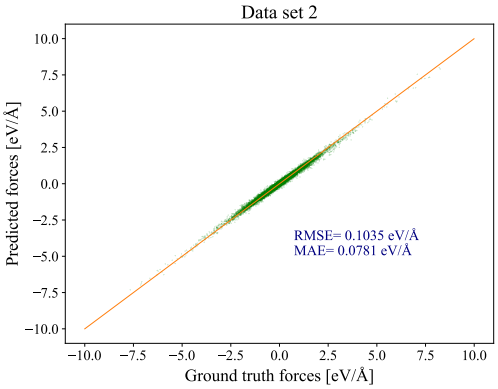


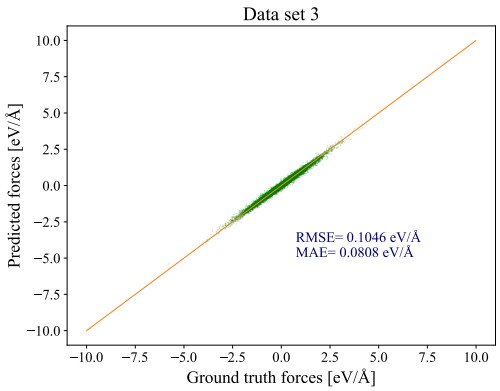


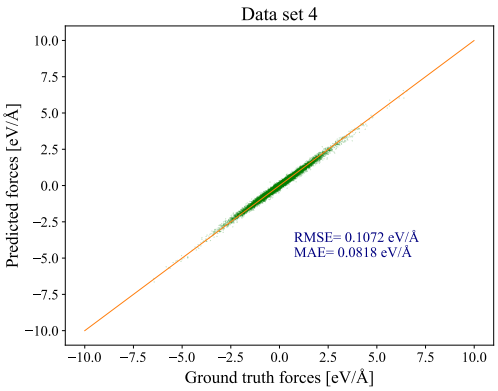


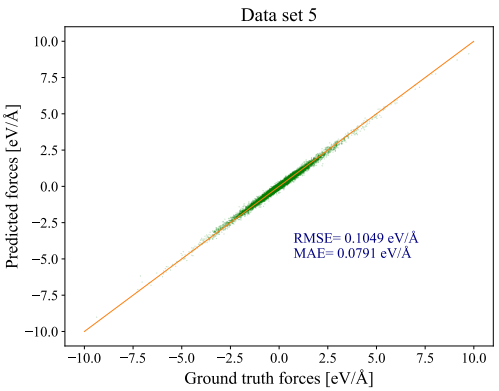
Mixed model 6

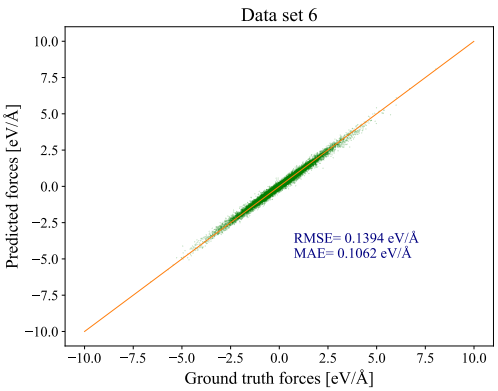




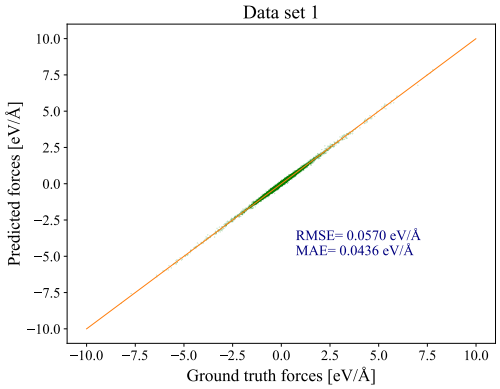


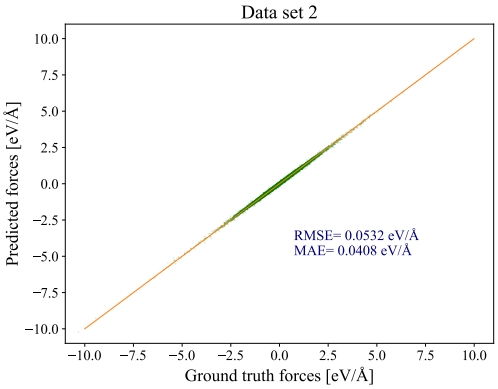


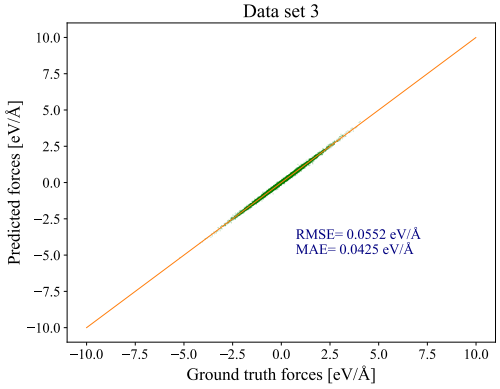


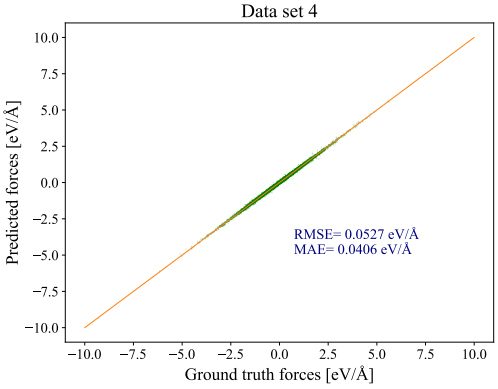


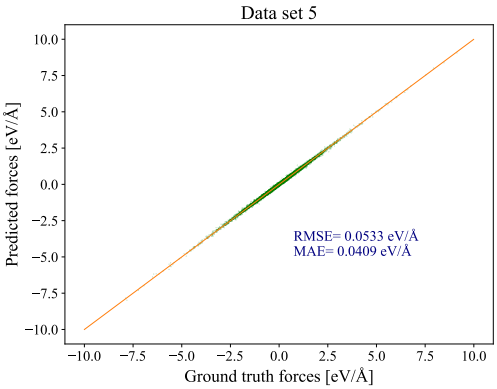
Mixed model 7

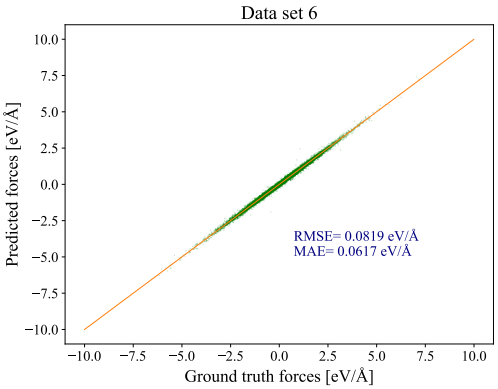




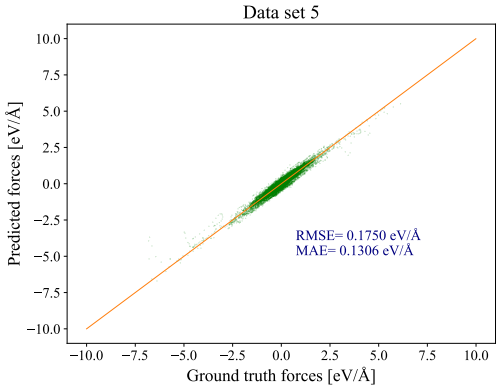


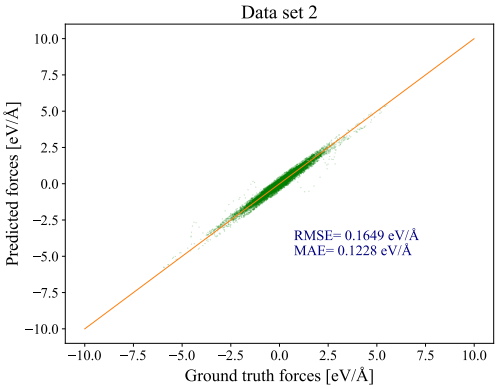


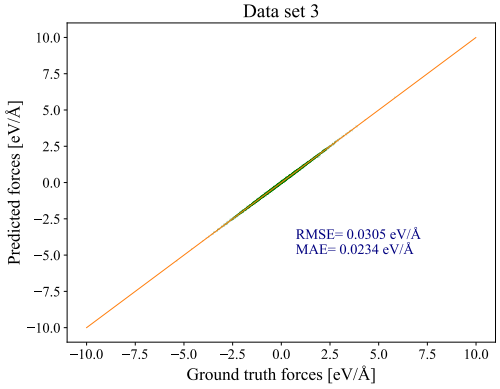


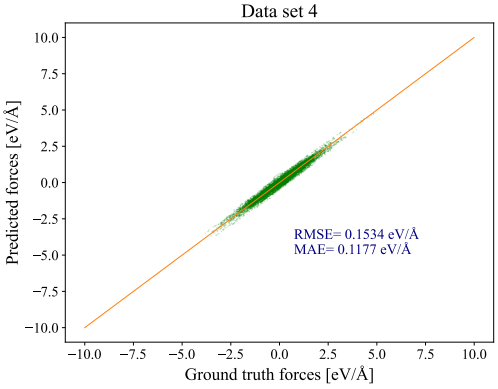


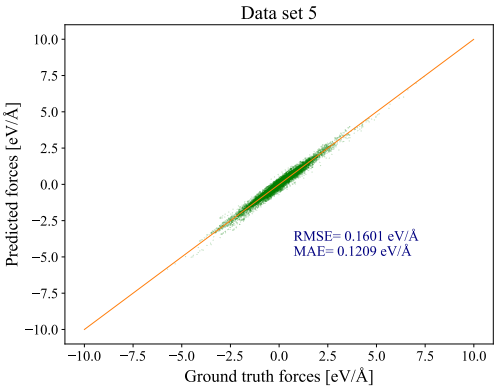
Re-trained simple model 1

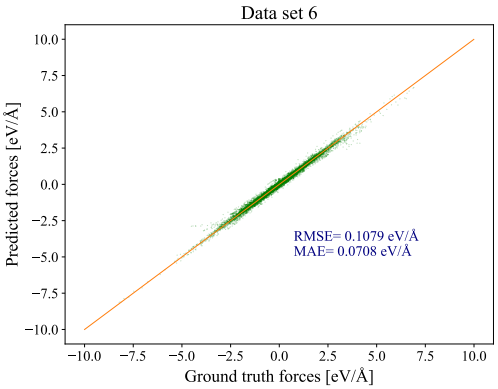




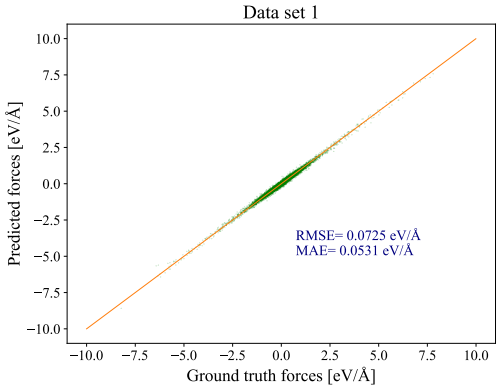


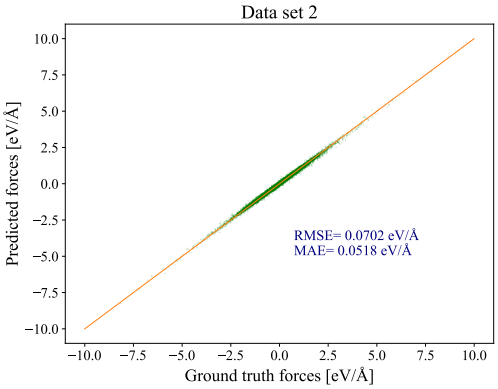


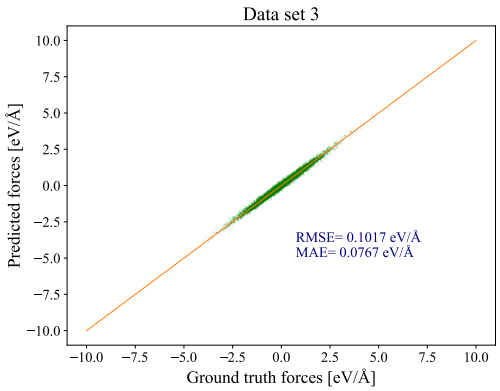


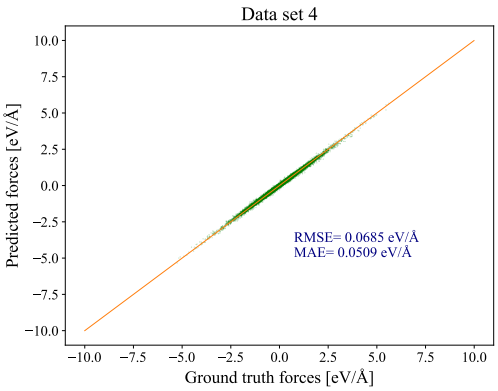


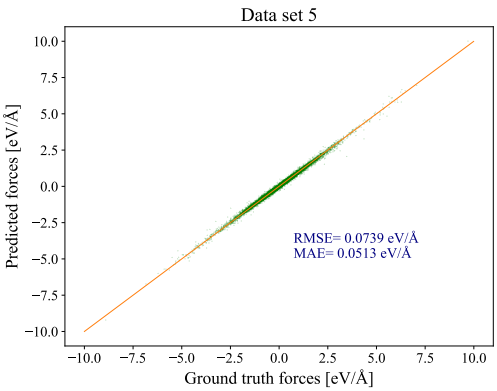
Re-trained mixed model 1

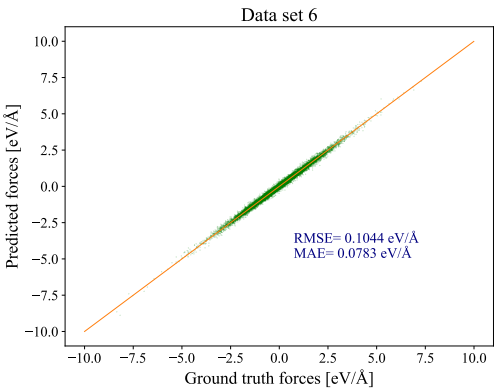




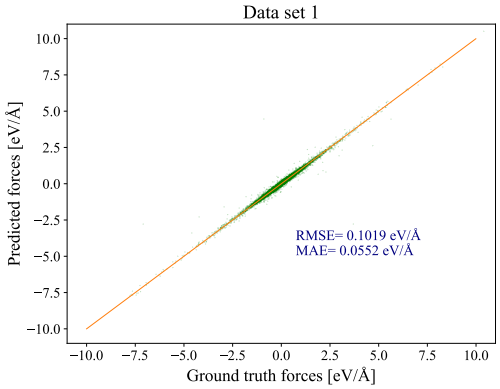


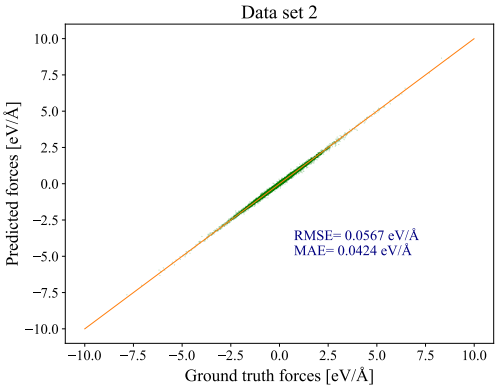


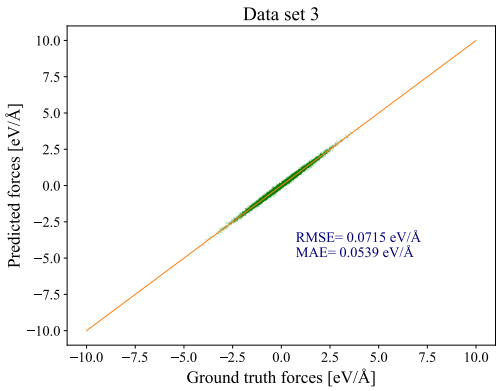


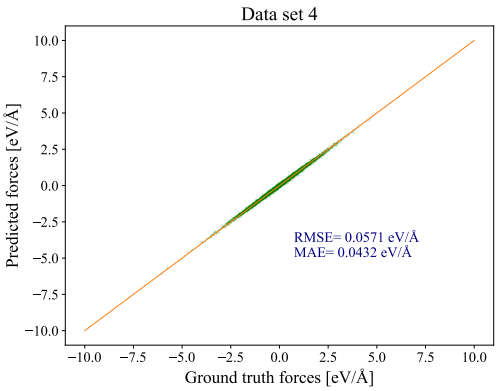


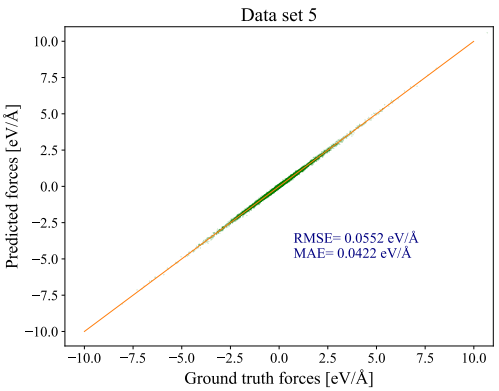
Re-trained mixed model 4

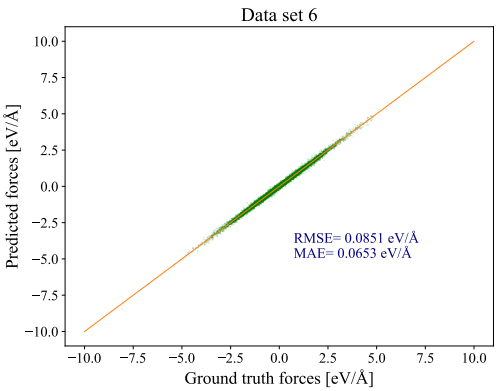




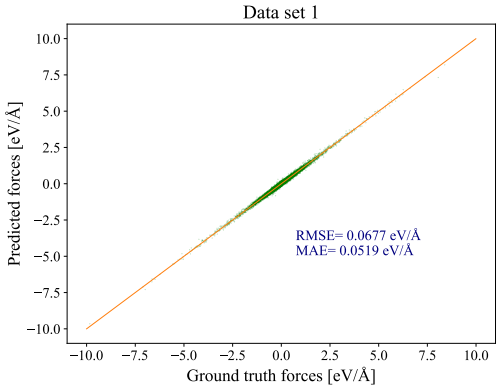


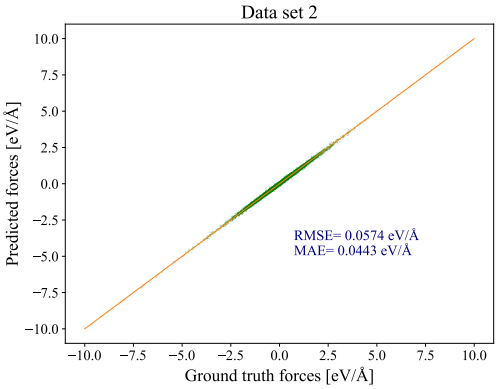


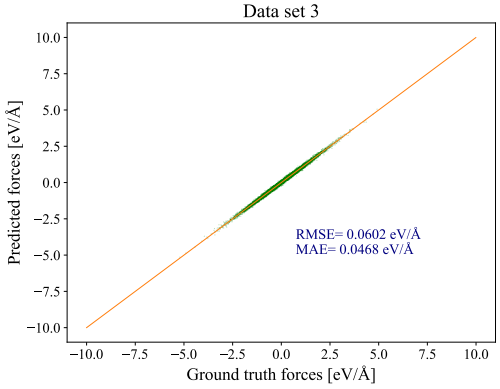


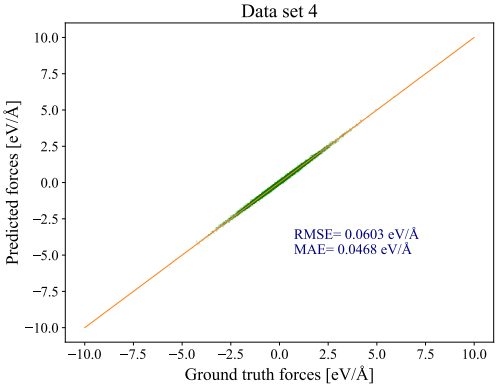


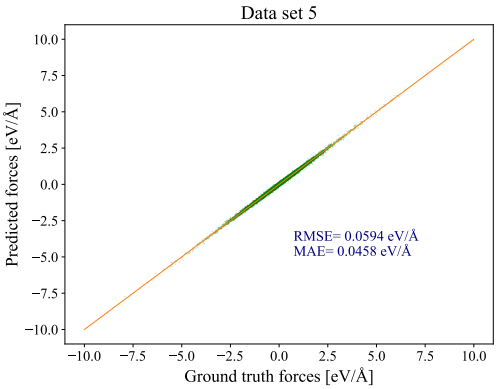
Re-trained mixed model 7

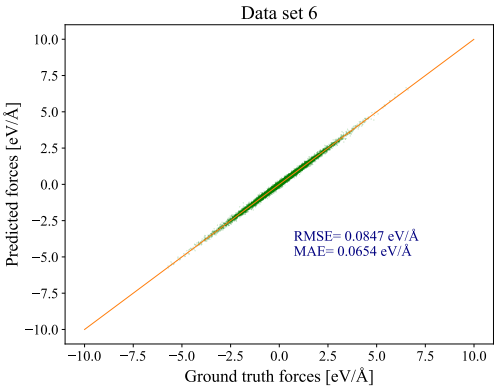




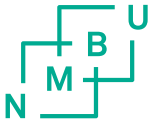








This page is intentionally left blank.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway