ETH zürich

Master Thesis

Two-Dimensional Machine Learning Force Fields: Universal Models and Alloy Applications

Fabian Jaeger

ETH Zurich
Institute of Materials Theory

Supervised by

Prof. Dr. Nicola Spaldin
Dr. Jonathan Schmidt

October, 2023

Acknowledgments

I would like to extend my sincerest gratitude to Prof. Nicola Spaldin for providing me with the incredible opportunity to work in her research group.

I am also immensely grateful to my supervisor Dr. Jonathan Schmidt for his invaluable mentorship and the countless lessons I have learned under his guidance. It has been nothing but enjoyable to work with him.

Lastly, to my family, who made everything possible.

Contents

Abstract

This thesis explores the potential and advantages of machine-learned force fields in various applications pertaining to material physics. Through research on two model architectures MACE and M3GNet with various training strategies, it has been found that these computational techniques can be used to complement or in certain cases challenge traditional computational techniques such as Density Functional Theory (DFT) and Cluster Expansion (CE). Notably, the M3GNET trained universal force field on an 85,279 large two-dimensional compound dataset has shown a two-fold improvement to previous work by Wang et al. [40] with an energy MAE of 77 $\mathrm{meV/atom}$ after the relaxation. The result introduces the possibility for greater accuracy in high-throughput screenings of novel two-dimensional prototype materials and the creation of an enhanced database of stable two-dimensional compounds.

Additionally, the applicability of these universal force fields to an alloy specific force field using transfer learning was investigated for transition-metal disulfide systems $IrRuS_2$ $MoTaS_2$, $MoWS_2$, $NbMoS_2$, $TiFeS_2$, $TiNbS_2$, $TiTaS_2$, $TiVS_2$ and $ZrTaS_2$, showing promising improvements in performance in certain cases. Furthermore, for a subset of these alloys, the material specific force fields were then employed to predict the formation energy and compared to results obtained from Cluster Expansion. Given that our machine learning predictions yield improved performance across the board but don't suffer from the inherent limitations of on-lattice models such as Cluster Expansion, we foresee our results paving new avenues in computational modelling of alloys.

Chapter 1

Introduction

In 1929 British physicist Dirac notably stated that:

> *The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much to complicated to be soluble. It there fore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation*

Almost a century later, Dirac's statement remains fundamentally true. Despite our thorough understanding of the Schrödinger equation – the governing equation of quantum mechanics that uniquely describes the dynamics of any quantum system – solving such an equation remains a formidable challenge. Indeed, analytical solutions are only possible for a minuscule subset of very simple systems, namely Hydrogen. This is due to the exponential complexity that arises in modelling even slightly more involved real-world system.

In large part, the history of quantum mechanics since its inception has thus been the pursuit of meaningful approximations and efficient numerical methods. Techniques such as Hartree-Fock theory, density functional theory (DFT) and perturbative approaches such as Moeller-Plesset and an array of other quantum chemistry frameworks have emerged from this endeavour. Despite these being approximations, they have enabled the prediction and understanding of a host of chemical and physical phenomena, offering increasing insights into the quantum world.

In recent decades, the staple of computational chemistry has been DFT which has proven extremely successful in predicting and understanding material properties. DFT methods however have several limitations, for example in incorporating disorder and describing properties at finite temperatures. Moreover, depending on the choice of exchange-correlation functional DFT predictions van vary significantly. This means that for certain classes of materials, the validation of material properties continues to rely heavily on

experimental verification, a processes that can be equally time-consuming and costly.

Another main limitation of DFT is its poor scalability and high cost, which limits its ability to model large and complex crystalline systems, such as alloys. To overcome this limitations the cluster expansion (CE) technique in conjunction with DFT has been a staple in the last few decades for modelling complex alloys and highly-disordered systems. However, the applicability and precision of Cluster Expansion heavily relies on the underlying selection of clusters and still requires some understanding of the systems behaviour which severely hinders its generalisation capability to other systems. The primary limitation of Cluster Expansion is its inherent assumption that the atomic configurations or clusters remain static and don't significantly relax or change position, which can compromise its accuracy when a material undergoes physical or chemical changes causing atoms to move and minimize the system's energy.

In recent years, motivated by successes of machine learning model in domains such as vision recognition, language modelling and more recently molecular modelling, machine learning force fields have become more and more prevalent with the hope of complementing or someday all together replacing DFT and Cluster Expansion. Force fields are mathematical functions that describe the potential energy and forces experienced by atoms in a molecular or crystalline system. Classical force fields, rely on empirical parameters informed by experimental data or quantum mechanical calculations. However, these approaches often suffer from limitations in accuracy and transferability. Unlike these traditional methods, Machine Learning Force Fields (MLFF) employ a data-driven approach, utilising a vast training dataset created through Density Functional Theory (DFT) calculations for various atomic arrangements. This includes information such as atomic positions $\{\mathbf{r}_i^{\text{DFT}}\}$, energies $\{E_i^{\text{DFT}}\}$, forces $\{\mathbf{F}_i^{\text{DFT}}\}$, stresses $\{\boldsymbol{\sigma}_i^{\text{DFT}}\}$, and if applicable other material properties.

These machine learning methods can come in many flavours notably but not exclusively, Kernel-based methods [29] and neural networks (NN)[1]. Kernel-based methods rely on specific kernels to define the input-output mapping, which impose severe restrictions (strong inductive biases) on its capability to generalise to any material. Such restrictions have the benefit of requiring less data compare to neural networks to obtain reasonable predictions, provided the capacity of kernel method can capture the complexity of the problem. For instance, kernel-based machine learning force fields such as Gaussian Approximation Potential (GAP) have been applied to a range of alloy compositions, such as the binary system AgPb, and have shown competitive accuracy when compared to Cluster Expansion [26, 23].

While kernel-based methods, such as Gaussian processes, are viable options, the growing availability of DFT datasets and growing computational capability make neural networks a more promising choice for force fields. A key advantage of NNs is their ability to accommodate any functional dependency providing them with the flexibility to model complex and simple systems alike. It is beneficial to then embed specific inherent biases

[1] All these regression models establish a functional relationship between the position of atoms in a systems and the potential energy surface (PES)

from the data into the model, a strategy which has driven the success of convolutional neural networks (CNNs) in image recognition. CNNs specifically encode translational invariance in their layers without needing to learn this symmetry from data.

When modelling molecules or crystals, we deal with data represented as structures that naturally exist in three-dimensional space. Therefore it is crucial for neural networks to respect the inherent symmetries of this space, represented by the $SE(3)$ group, encompassing translations, rotations and reflections.

Such $SE(3)$-symmetry-aware force fields have been constructed using graph neural networks and have delivered state-of-the art accuracy for material property predictions - notable examples include Nequip [32], Mace [44] and M3GNet [18]. Owing to their substantial improvements in accuracy over traditional techniques like classical neural networks or kernel methods, graph neural networks have emerged as a promising tools for constructing universal machine learning force fields (UFF)[2]. These models are 'universal' in their ability to be applied to a wide range of materials, extending beyond the specific atomic arrangements of types they were initially trained on. Once these universal force fields are trained, the fast inference time enables the prediction of any material property at speeds of classical methods, facilitating high-throughput searches across vast material databases, a feat previously unattainable using only DFT. In the past year, for instance there has been significant work on the creation of a three-dimensional universal machine learning force field using the M3GNet architecture [33].

Historically, two-dimensional UFF research has been less prevalent than its three-dimensional counterpart, largely due to challenges in acquiring suitably sized two-dimensional datasets. In this thesis, we aim to replicate the success of three-dimensional UFFs in creating a universal force field for two-dimensional structures by capitalizing on a newly curated database based on the work of Wang. et. al. [40]. We utilize different models, experiment with various training methods by adjusting model and data hyperparameters and assess the performance of each model. This study aims to provide a set of recommendations for the scientific community on the most-effective practices.

We explore whether there are potential advantages to applying transfer learning from these universal machine learning force fields to material-specific force fields. This analysis is conducted using a two-dimensional transitional metal disulfide alloy dataset. Furthermore given the widespread use of Cluster Expansion in predicting alloy formation energies, we aim to determine whether these universal force fields can be directly trained to predict these energies and at which accuracy. This allows us to ascertain whether or not these models can serve as an alternative to Cluster Expansion.

The remainder of this thesis is organised as follows. We start by giving a brief overview of Density Functional Theory. A quick introduction to Cluster Expansion and its role in alloy modelling will be presented alongside. This will be followed by an introduction into deep learning where we discuss the most basic neural network architecture, the multilayer perceptron, how neural networks learn and briefly talk about transfer learning. In

[2] In literature also sometimes referred to as universal inter-atomic potentials.

the subsequent chapter we will give an overview of geometric graph neural networks. We discuss the role of symmetry and how different architectures embed these into their models. Lastly we state and discuss our results, give a conclusion and an outlook of possible future avenues of research.

Chapter 2

Theory and Computational Methods

2.1 Schrödinger equation

The energy of a many-body system with N electrons and M nuclei is determined by the Schrödinger equation:

$$\hat{H}_{\text{tot}}\Psi_i(\mathbf{r}_1,\dots\mathbf{r}_N;\mathbf{R}_1,\dots\mathbf{R}_M) = E_i^{\text{tot}}\Psi_i(\mathbf{r}_1,\dots\mathbf{r}_N;\mathbf{R}_1,\dots\mathbf{R}_M) \tag{2.1}$$

Here $\mathbf{r}_i$ and $\mathbf{R}_i$ denote the electronic and nucleus positions respectively. The Hamiltonian that describes such a system is given by:

$$\begin{aligned}\hat{H}_{\text{tot}} &= \hat{T}_{\text{e}} + \hat{T}_{\text{n}} + \hat{V}_{\text{en}} + \hat{V}_{\text{ee}} + \hat{V}_{\text{nn}} \\ &= -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \frac{1}{2}\sum_{A=1}^{M}\frac{1}{M_A}\nabla_A^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i-\mathbf{R}_A|} \\ &\quad + \sum_{i=1}^{N}\sum_{j<i}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} + \sum_{A=1}^{M}\sum_{B<A}^{M}\frac{Z_A Z_B}{|\mathbf{R}_A-\mathbf{R}_B|}\end{aligned} \tag{2.2}$$

where $\hat{T}_{\text{e}}$, $\hat{T}_{\text{n}}$, $\hat{V}_{\text{en}}$,$\hat{V}_{\text{ee}}$ and $\hat{V}_{\text{nn}}$ represents the electron kinetic energy, nuclei kinetic energy, electron-nuclei potential, electron-electron potential and nuclei-nuclei potential respectively and i, j iterate through the N electrons and A, B through the M nuclei.

The commonly used Born-Oppenheimer approximation makes the assumption, that due to their greater mass, nuclei move more slowly compared to electrons. This allows for a decoupling of the electronic and nuclear degrees of freedom in which the kinetic term of the nuclei can be neglected. The nuclear problem is simplified to be

$$\hat{V}_{nn}\Psi = E_{nn}\Psi \tag{2.3}$$

while the electronic problem $\hat{H}\Psi = E\Psi$ is described by the Hamiltonian

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i-\mathbf{R}_A|} + \sum_{i=1}^{N}\sum_{j<i}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} = \hat{T} + \hat{V}_{en} + \hat{V}_{ee} \tag{2.4}$$

with total energy $E_{\text{tot}} = E + E_{nn}$.

2.2 Density Functional Theory

As the name density functional theory suggests, the theory is formulated using an energy functional of the electron density. The Hohenberg-Kohn theorems provide us with the knowledge that there is a unique mapping between ground state electronic densities and external potentials and that the former minimize the energy of an unknown universal energy functional [1].

Analogous to the Hartree-Fock equations, the Kohn-Sham formalism maps the fully interacting problem to a simplified non-interacting single electron problem but with a modified energy functional $E_{\text{int}}[\rho]$. These electrons move within an effective "Kohn-Sham" single particle potential $v_{\text{eff}}(\mathbf{r})$ [20].

The central quantity of interest in the Kohn-Sham formalism is the electron density formed by the N occupied single-particle orbitals:

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} |\varphi_i(\mathbf{r})|^2 \tag{2.5}$$

The single-particle Kohn-Sham orbitals $\varphi_i(\mathbf{r})$ are obtained from the Schrödinger equation:

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + v_{\text{eff}}\right)\varphi_i(\mathbf{r}) = \epsilon_i\varphi_i(\mathbf{r}) \tag{2.6}$$

where the effective potential v_{eff} is given by

$$v_{\text{eff}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + \int \mathrm{d}^3r' \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} + \frac{\delta E_{xc}[\rho(\mathbf{r})]}{\delta\rho(\mathbf{r})} \tag{2.7}$$

The total energy is then obtained from the following energy functional:

$$E_{\text{int}}[\rho(\mathbf{r})] = T_S[\rho] + E_H[\rho(\mathbf{r})] + E_{\text{ex}}[\rho(\mathbf{r}] \tag{2.8}$$

$$= \sum_i \epsilon_i - \frac{1}{2}\int \mathrm{d}^3r' \int \mathrm{d}^3r \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} - \int \mathrm{d}^3r\rho(\mathbf{r})\frac{\delta E_{\text{xc}}(\rho(\mathbf{r}))}{\delta\rho(\mathbf{r})} + E_{xc}[\rho(\mathbf{r})] \tag{2.9}$$

From the effective potential (2.7) it is evident we need to make some approximation for the exchange energy $E_{xc}[\rho(\mathbf{r})]$.

The general flow of the self-consistent field calculation within DFT is show in Figure (2.1). The aim of the self-consistent field calculation is to obtain a solution of the energy-density functional for which the difference in energy between two consecutive steps satisfies the convergence hyperparameter defined by the user.

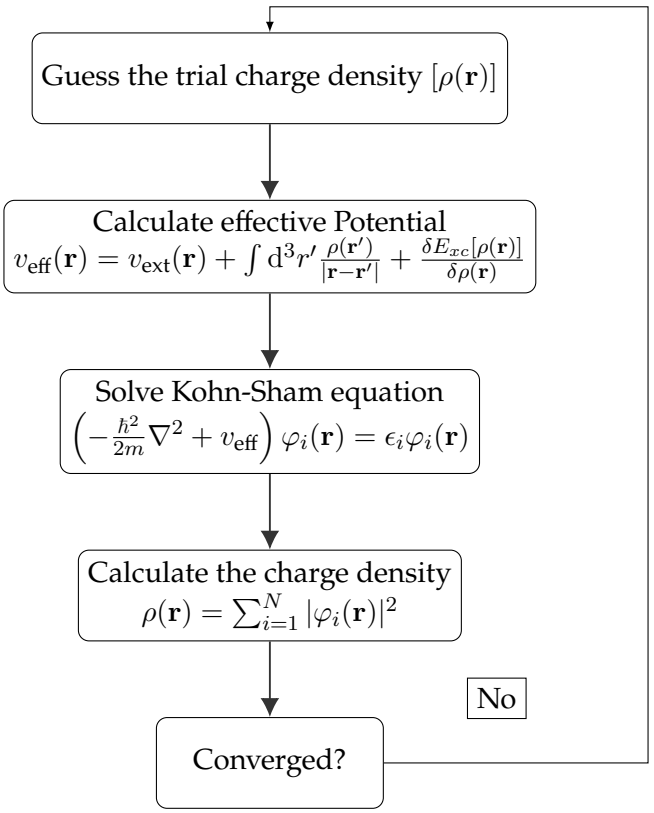


Figure 2.1: Flowchart for the self-consistent calculations in DFT. Once the trial charge density converges in energy we have found the ground state energy which we can then use to calculate the electronic properties of the system.

2.2.1 PBE Exchange-Correlation functional

The Perdew-Burke-Ernzerhof (PBE) [5] is one of the exchange correlation functional one can employ in equation (2.9). It is a generalized gradient approximation (GGA) that improves upon the widely used Local Density Approximation (LDA)

$$E_{xc}^{\text{LDA}}[\rho(\mathbf{r})] = \int \rho(\mathbf{r}) \epsilon_{xc}(\rho(\mathbf{r})) \mathrm{d}^3 r \tag{2.10}$$

where ϵ_{xc} is the exchange correlation energy per particle obtain for a Jellium model of a uniform electron gas with density $\rho(\mathbf{r})$, by accounting also for gradients of the electron density, i.e

$$E_{xc}^{\text{GGA}}[\rho(\mathbf{r})] = \int \rho(\mathbf{r}) \epsilon_{xc}(\rho(\mathbf{r}), \nabla\rho(\mathbf{r})) \mathrm{d}^3 r \tag{2.11}$$

The popularity of the use of PBE functionals in computational chemistry and materials science can be traced back to its substantial improvement of its predictions in addition to not leading to a significant increase in computational overhead when compared to standard LDA methods.

More complicated forms such as hybrid functional that expand on GGA methods by additionally including exact exchange energies from Hartree-Fock calculations also exist, but will not be discussed further in detail.

2.2.2 Obtaining forces and stresses in DFT

Many important material properties cannot directly be derived from the energy obtained from DFT calculations but will depend on derivatives of this energy. Of importance are mainly forces and stresses acting on the system, which allows us to optimise structural parameters and derive vibrational properties such as phonons. Classically the force can directly be obtained from the potential energy via the gradient with respect to the atomic coordinates:

$$\mathbf{F} = -\nabla_{\mathbf{R}} U(\mathbf{R}) \tag{2.12}$$

In quantum mechanics this connection between forces and energies can be established through Perturbation theory in the form of the Hellman-Feynman theorem which can be derived from the reasonable assumption that in quantum mechanics we have

$$\mathbf{F} = -\nabla_{\mathbf{R}} \langle E \rangle \quad \text{with} \quad E = \langle \Psi | \hat{H} | \Psi \rangle \tag{2.13}$$

In this case we simply have to pay attention to the fact that the wavefunctions are implicit functions of the atomic coordinates too, requiring the use of the chain rule when differentiating. It is useful abstracting the differentiating to any quantity that implicitly appears in the Hamiltonian and in the wavefunction through the variable λ. As such we get

$$\begin{aligned}
\frac{\partial E}{\partial \lambda} &= \frac{\partial}{\partial \lambda} \langle \Psi | \hat{H} | \Psi \rangle \\
&= \left\langle \Psi \left| \frac{\partial \hat{H}}{\partial \lambda} \right| \Psi \right\rangle + \left\langle \frac{\partial \Psi}{\partial \lambda} \mid \hat{H} | \Psi \right\rangle + \left\langle \Psi | \hat{H} | \frac{\partial \Psi}{\partial \lambda} \right\rangle \\
&= \left\langle \Psi \left| \frac{\partial \hat{H}}{\partial \lambda} \right| \Psi \right\rangle + E \left\langle \frac{\partial \Psi}{\partial \lambda} \mid \Psi \right\rangle + E \left\langle \Psi \mid \frac{\partial \Psi}{\partial \lambda} \right\rangle \\
&= \left\langle \Psi \left| \frac{\partial \hat{H}}{\partial \lambda} \right| \Psi \right\rangle + E \frac{\partial}{\partial \lambda} \langle \Psi \mid \Psi \rangle
\end{aligned} \tag{2.14}$$

The last term vanishes since we get a normalized ground state wave function in DFT $\langle \Psi | \Psi \rangle = 1$, leading to the so-called Hellman-Feynman theorem:

$$\frac{\partial E}{\partial \lambda} = \langle \Psi | \frac{\partial \hat{H}}{\partial \lambda} | \Psi \rangle \tag{2.15}$$

Setting $\lambda = \mathbf{R}_i$ to be the cartesian vector on atom i, one can show from the many-body Schrödinger equation within the Born-Oppenheimer approximation (using the charge density representation of the total energy as in DFT) applying the Hellman-Feynman Theorem yields:

$$\mathbf{F}_A = -\frac{\partial E}{\partial \mathbf{R}_A} = \langle \Psi | \frac{\partial \hat{H}}{\partial \lambda} | \Psi \rangle = -\frac{\partial E_{\text{nn}}}{\partial \mathbf{R}_A} - \int \rho(\mathbf{r}) \frac{\partial V_{\text{en}}}{\partial \mathbf{R}_A} \mathrm{d}\mathbf{r} \tag{2.16}$$

where V_{en} are the nuclei-electron interaction potential and E_{nn} the nuclei-nuclei contribution to the total energy $E_{\text{tot}} = E + E_{\text{nn}}$. The force depends thus only on the charge density as the other terms are fixed for a given set of atoms (fixed atomic charge Z).

The Cauchy stress tensor σ is defined as the negative gradient of the energy per unit volume $\Omega = \mathbf{a} \cdot \mathbf{b} \times \mathbf{c}$ (referred to as strain)

$$\sigma_{\alpha\beta} = \frac{1}{\Omega} \frac{\partial E}{\partial \epsilon_{\alpha\beta}} \tag{2.17}$$

In contrast to the force which is a microscopic quantity as one formulates it for each of the atoms, the stress is a macroscopic quantity that in a three dimensional system takes the form of a 3×3 matrix.

It can be shown that the stress can be calculated through

$$\sigma_{\alpha\beta} = -\frac{1}{\Omega} \sum_i \left\langle \Psi \left| \frac{1}{2\mu_i} \nabla_{i\alpha} \nabla_{i\beta} - \frac{1}{2} \sum_{j \neq i} \frac{(\mathbf{q}_j - \mathbf{q}_i)_\alpha (\mathbf{q}_j - \mathbf{q}_i)_\beta}{q_{ij}} \left(\frac{\mathrm{d}}{\mathrm{d}q_{ij}} \right) \hat{V} \right| \Psi \right\rangle \tag{2.18}$$

where i, j run over all electron and nuclei and $\mathbf{q}_i = \{\mathbf{r}_j, \mathbf{R}_k\}$ representing all spatial coordinates of the nuclei and electrons.

2.3 Formation Energy

The formation energy represents the change in energy when a compound forms from its constituent elements in their most stable states. In general the formation energy is computed as follows:

$$E_F = E^{\text{total}} - \sum_i \lambda_i E_i^{\text{element}} \tag{2.19}$$

where λ_i is the multiplicity of the element in the formula unit of the compound. However since we are going to be concerned with the formation energy of ternary alloys (AB) where A and B are binary structures, this formula can conventionally be rewritten as

$$E_F = E(AB) - [xE(A) + (1-x)E(B)] \tag{2.20}$$

where E_F is the formation energy of the ternary alloy, $E(AB)$ the total energy of the alloy, $E(A)$ and $E(B)$ the energy of the most stable binary phases A and B respectively and x is the percentage of A in the alloy. Under standard conditions, a negative formation energy implies alloy stability and spontaneous formation, as the energy of the combined state is lower than the individual energies of the elements. Phrased differently, negative formation energies represent exothermic compounds in which the energy released during the formation of the compound, in the process of making new bonds, is greater than the energy required to break the existing bonds of the reactant elements.

2.4 Cluster Expansion and Alloy Modelling

The modelling of alloys with DFT poses many inherent challenges. One particular challenge is the exponential increase of possible atomic configuration in multicomponent systems with respect to the number of constituents and system size. For accurate thermodynamical predictions of alloy properties, detailed DFT calculations are required for all possible configurations, which is a very resource-intensive task.

Successful approaches that have attempted to tackle this problem areand off-lattice models such as gaussian approximation potentials (GAP), atomic Cluster Expansion (ACE) and machine learning force fields

Efficient strategies that have aimed to address this issues include Cluster Expansion (referred to as a on-lattice models) and off-lattice models such as Gaussian approximation potentials (GAP) and Atomic Cluster Expansion (ACE). This section will focus specifically on the conventional on-lattice Cluster Expansion approach, while ACE will be discussed later in section (3.7.1) [19, 23, 24].

In alloy systems, atoms can arrange themselves in various configurations, forming distinct clusters with specific atomic arrangements. Cluster Expansion uses these atomic clusters as the building blocks to represent the alloys energy by assuming that the total energy of the alloy is given by a sum of contributions from different clusters $\{\alpha, \beta \dots\}$ and their respective inter-cluster interactions (often also referred to as effective cluster interaction or ECI) denoted by J_α [2].

The methodical treatment of a multi-component substitutional system begins by mapping the original system to a lattice model. The underlying assumption here is that there is a unique mapping for any particular occupation, denoted as $\boldsymbol{\sigma}$[1] to an ideal lattice L, where the deviation of the atomic sites from the ideal lattice sites would be small. The configuration $\boldsymbol{\sigma}$ is defined by specifying the occupation of any lattice site p by a site variable σ_p.

In the case of a generic binary substitutional alloy $A_{1-x}B_x$ these 'occupation' number take the values +1 and -1 if the site is occupied by A or B respectively. [2]

For a multi-component system with N sites that are symmetrically equivalent and can be occupied by M different atomic species, a configuration state is described by an N-dimensional vector of site variables $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots \sigma_N)$. Assuming that the site p can be occupied by one of M species, the site variable σ_p takes one of $(\pm M/2, \dots, \pm 1)$ when M is even and $(\pm(M-1)/, \dots, \pm 1, 0)$ when M is odd.

Let $\mathbf{V}$ denote the multiplicity weighted ECI (effective cluster interaction)

$$V_\alpha = m_\alpha J_\alpha \tag{2.21}$$

[1] not to be confused with the stress σ

[2] It should be noted that the system can contain other components, e.g C in $A_{1-x}B_xC$ which are not explicitly indicated by the occupation number.

where $\mathbf{V} = [V_0, V_1, \ldots, V_{N_c-1}]^{\mathrm{T}}$ denotes the column vector of all ECIs and N_c the number of clusters considered. The Cluster Expansion then parametrises the energy (per atom) of the alloy as a polynomial in the occupation variables obtained through some choice of orthonormal basis [3]

$$E(\boldsymbol{\sigma}) = \sum_{\alpha} m_\alpha J_\alpha \langle \phi_\alpha(\boldsymbol{\sigma}) \rangle = \phi(\sigma)\mathbf{V} \tag{2.22}$$

where $\phi_\alpha(\boldsymbol{\sigma}) = [\phi_0, \phi_1, \ldots, \phi_{N_c-1}]$ is the cluster correlation function and is the average of the product of "spin" variables (or occupation number) σ_i defined for each site i in the cluster α:

$$\phi_\alpha(\boldsymbol{\sigma}) = \frac{1}{N_\alpha} \sum_{\beta \subset \alpha} \left(\prod_{i \in \beta} \sigma_i \right) \tag{2.23}$$

where the sum is over all the subclusters β contained within cluster α and N_α represents the number of subclusters in cluster α. This is a tensor product similar to the ACE and MACE construction as we will later see.

For a multi-component system however the cluster correlation function in equation (2.23) needs to additionally incorporate orthogonality with respect to the different atomic species M. One common choice are Chebyshev Polynomials:

$$\Theta_{b,M}(\sigma_p) = \sum_{m=0}^{M-1} c_m^{(b)} \sigma_p^{(b)} \quad (b = 0, 1, \ldots M-1) \tag{2.24}$$

where the coefficients $c_m^{(i)}$ are determined from the orthogonality condition:

$$\left\langle \Theta_{b,M} \middle| \Theta_{b',M} \right\rangle = \frac{1}{M} \sum_{\sigma_p} \Theta_{b,M}(\sigma_p)\Theta_{b',M}(\sigma_p) = \delta_{b,b'} \tag{2.25}$$

The tensor product of the site basis functions then spans a basis on the collective space of N sites:

$$\phi(\boldsymbol{\sigma}) = \prod_{p=1}^{N} \Theta_{b_p,M}(\sigma_p) \tag{2.26}$$

which can be used as cluster correlation function in (2.22) to estimate the energy for multi-component alloys.

When considering all clusters α that are included in the summation in (2.22), it follows that the Cluster Expansion is capable of representing any function $E(\sigma)$ of a configuration σ through suitable choices of J_α. However, the real merit of the Cluster Expansion lies in its practical application. Even a relatively small number of terms in the expansion can effectively capture the most important contributions to the total energy.

[3] Note the similarities of the energy with the Ising Hamiltonian

2.4.1 ECI Selection

As a last step, the coupling coefficients (ECIs) J_α have to be determined. The most straightforward method involves minimising the difference between the total energies of a relatively small number of crystal structures with different ionic configurations obtained from DFT and comparing those to the predictions of the total energy from the truncated Cluster Expansion. This approach is referred to as structure inversion method and was very prevalent in the early days of Cluster Expansion.

As the Cluster Expansion method matured and the number of terms in the expansion (and data points changed from a handful to thousands) the regression technique embraced big-data techniques such as simulated annealing, genetic algorithms and also Bayesian methods [23].

Structure inversion

In structure inversion we use the total energy of small number N_s of structures obtained from DFT as the training set and determine $\mathbf{V}$ and in turn J_α through standard least-squares (LS) regression technique. The loss function to be minimised is chosen to be [41]

$$\mathcal{L}_{\mathrm{LS}}(\mathbf{V}) = \sum_{i=1}^{N_s} \left[E_i - \sum_{a=0}^{N_c-1} V_a \phi_a \left(\boldsymbol{\sigma}_i \right) \right]^2 \tag{2.27}$$

where $E_i = E(\sigma_i)$ is the DFT energy of configuration σ_i. A well known result of least-squares regression is that the minima of this loss function with respect to $\mathbf{V}$ possesses a closed-form solution of

$$\hat{\mathbf{V}}_{\mathrm{LS}} = \left[\mathbf{X}^{\mathrm{T}} \mathbf{X} \right]^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{E} \tag{2.28}$$

In this setup, $\mathbf{X}$ represents an $N_s \times N_c$ matrix where $X_{i\alpha} = \phi_\alpha(\boldsymbol{\sigma} i)$. The vector $\mathbf{E}$ is a column vector comprised of N_s directly computed energies, denoted by $\mathbf{E} = [E1, E_2, \ldots, E_{N_s}]^{\mathrm{T}}$. To avoid overfitting it's typically necessary for the number of training data points (N_s) to significantly outnumber the number of clusters (N_c) if using standard least squares regression.

One can characterise the fitting error by the following root mean squared error (RMSE) of fitted energies

$$\Delta_{\mathrm{fit}} = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \left[E_i - \sum_{a=0}^{N_c-1} \hat{V}_a \phi_a \left(\sigma_i \right) \right]^2} \tag{2.29}$$

This equation evaluates the square root of the average squared differences between the observed energies and the predicted values based on the fitted model.

2.5 Machine and Deep Learning

Machine learning approaches can be divided into three main paradigms, supervised, unsupervised and reinforcement learning. In this thesis we will mostly be concerned

with supervised learning where given a dataset $\mathcal{D}$ consisting of tuples of feature vectors $\mathbf{x}$ and labels y_i $\{(\mathbf{x}_i, y_i\})$ serving as input to the machine learning algorithm, we try to predict the output labels $\hat{y}_i$ given the input feature vector $\mathbf{x}_i$. Depending on whether the labels domain $\hat{y}_i$ is on a finite set of classes or the space of real numbers we are concerned with a classification or regression task respectively.

In recent times, deep learning, a technique grounded on using neural networks and a sub field of machine learning has been behind the success of many of the breakthroughs in AI such ChatGPT and AlphaFold. This success is driven by the capacity of neural networks to greatly benefit from ever increasing quantities of data. Additionally the increase in computational abilities of modern machines has enabled deeper and deeper networks[4]. Although neural networks have outgrown the capabilities of more traditional machine learning algorithms such as support-vector machines and kernel-based models, to name a few, they have major pitfalls such as interpretatbility and uncertainty estimation which still limits their broad adoption and applicability in the physical and medical sciences.

As alluded in the introduction, neural networks fundamental strength however lies in their ability to capture any functional dependence between input and output, allowing them to be used on any type of data. More precisely the universal approximation theorem states that a feed-forward neural network with a single hidden layer can approximate any continuous function given enough hidden neurons and appropriate activation functions. The downside of this is that since the parameter space to explore is larger, complex input-output relations can require more data than more traditional approaches to get similar outcomes for certain relatively simpler problems.

Apart from parameters that are learned during training, machine learning models also depend on hyperparameters. These are manually determined by the operator or through an automated procedure and can significantly impact the performance of a model, which can be be viewed as another disadvantage of machine learning, given that an extensive hyperparameter search - exploring numerous hyperparameter combinations - is often required to identify the most optimal model.

The most notable hyperparameters include the learning rate α in gradient descent (elaborated on in section (2.5.2)) and the batch size. The batch size determines the number of data points used for training in one cycle before the model parameters are updated.

2.5.1 Multi-Layer Perceptron

The perceptron is the simplest supervised learning neural network with a single input and output layer representing our predictor quantity. Assuming the training data has d inputs and a single output y, the perceptron prediction is given by

$$\hat{y} = \text{sign}\{\mathbf{W}\mathbf{x} + b\} = \text{sign}\{\sum_{j=1}^{d} w_j x_j + b\} \tag{2.30}$$

[4]deeper referring to the number of layers used in a neural network.

where the input layer represents the information from d-feature vectors $\mathbf{x} = [x_1, \dots x_d]$ in its d nodes, $\mathbf{W} = [w_1, \dots w_d]^T$ representing all the edges between the input nodes to the single output node and $b \in \mathbb{R}$ corresponding to the bias. Subsequently the sign function, serving as a so-called activation function Φ, maps the aggregated value $\sum_{i=1}^{d} w_i x_i + b$ to the values $+1$ or -1.

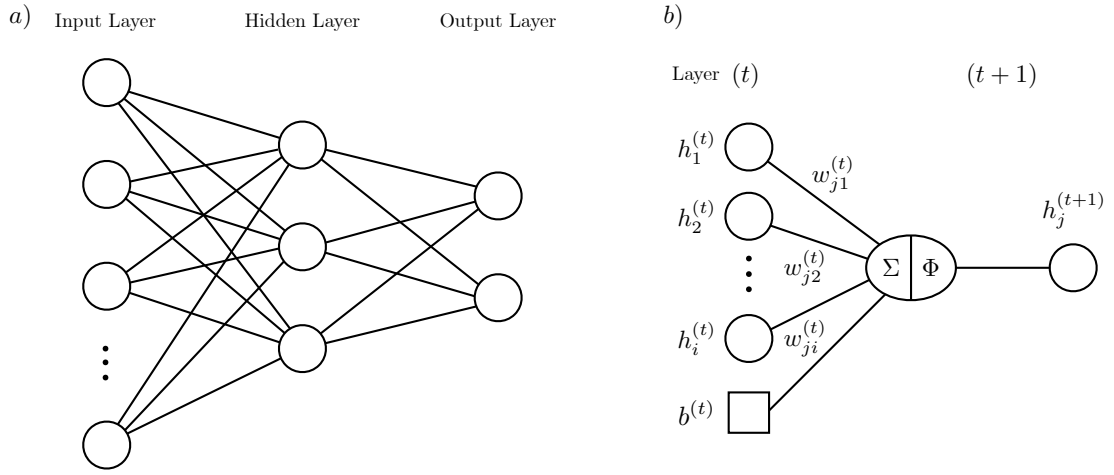


Figure 2.2: a) Schematic representation of an MLP b) Pictorial representation of the steps involved in node output computations.

Although the original perceptron was an algorithm for supervised learning for binary classifiers, its core principles can be extended to include other types of learning and capture more complex functions by relaxing the condition on the activation function to include non-linear function and introduce so-called hidden-layers as well as a variable output layer dimension. Such types of neural networks go under the name of *Multi-layer Perceptron (MLP)* and are often interchangeably used with feed-forward neural networks. In such networks each of the neurons in the previous layer feed into each neuron in the successive layer, for all the layers going from input to output. The perceptron can thus really be seen as the fundamental unit of computation or neuron in a neural network from which most modern architectures are constructed.

Formally if an MLP contains $p_1, \dots p_k$ units in each of its k-hidden layers then the vector representations of these hidden states denoted by $\mathbf{h}_1, \dots \mathbf{h}_k$ have dimensionalities $p_1, \dots p_k$. The weights of the connections between the input layer and the first hidden layer are then contained in the matrix $\mathbf{W}^1$ with size $d \times p_1$ where d is the input dimension of the features. The weights between the t-th hidden layer and the $(t+1)$-th hidden layer are denoted by a $p_t \times p_{t+1}$ matrix $\mathbf{W}^{(t)}$[5]. The weight matrix connecting the final layer is then given by $o \times p_k$ where o represent the number of nodes in the output layer and p_k the number of hidden units in the last hidden layer. As such an MLP follows the following

[5]So the number of units in the $(t+1)$-th layer defines the number of rows, whereas the number of units in the t-th layer defines the number of columns

recursive procedure:

$$\begin{aligned} \mathbf{h}^{(1)} &= \Phi(\mathbf{W}^{(1)}\mathbf{x} + b^{(1)}) && \text{[Input to Hidden Layer]} \\ \mathbf{h}^{(t+1)} &= \Phi(\mathbf{W}^{(t+1)}\mathbf{h}^{(t)} + b^{(t)}) \quad \forall t \in \{1, \dots, k-1\} && \text{[Hidden to Hidden Layer]} \\ \mathbf{o} &= \Phi(\mathbf{W}^{(k+1)}\mathbf{h}^{(k)}) && \text{[Hidden to Output Layer]} \end{aligned} \tag{2.31}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input feature vector and Φ is the activation function.

The individual neuron contributions (different components of the hidden state vector $\mathbf{h}^{(t)}$ for the $(t+1)$-th hidden layer taking inputs from the t-th hidden layer $\mathbf{h}^{(t)}$) are then explicitly given by

$$h_j^{(t+1)} = \Phi\left(\sum_{i=1}^{p_t} w_{ji}^{(t+1)} h_i^{(t)} + b^{(t)}\right) \tag{2.32}$$

where $w_{ji}^{(t+1)}$ is the component of the weight matrix $\mathbf{W}^{(t+1)}$ and $h_i^{(t)}$ represent the hidden components of the t-th hidden layer that serve as input to the $(t+1)$-th layer.

Once the architecture has been decided on (i.e how many layers, what kind of activation functions), we find that the network outputs $\hat{y}_n$ depend soley on the input $\mathbf{x}$ and the parameters $\boldsymbol{\theta} = \{\mathbf{W}^{(t)}, b^{(t)}\}, t \in 1, \dots k$ with $\boldsymbol{\theta} \in \Theta$. Formally we can thus represent the predictions $\hat{y}_n$ of the Neural Network as the following functional dependence:

$$\hat{y}_n = f(\mathbf{x}_n; \boldsymbol{\theta}) \tag{2.33}$$

2.5.2 Training and Evaluation

The goal of the training process to learn the functional representation $\hat{y}_n$ in equation (2.33), given a dataset $\mathcal{D}$ by modifying the weights and biases. To quantity the validity of the trained functional dependence, a scalar-valued *loss-function* $\mathcal{L} : \Theta \to \mathbb{R}$ (or cost-function) is introduced[6].

In regression problems (where the estimator $\hat{y}$ is real-valued), common loss functions include L_1 and L_2 losses:

- Mean squared error (MSE)

$$\text{MSE}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2 \tag{2.35}$$

[6]For example in classification tasks, loss functions usually take the forms of some type of categorical cross entropy or Kullback-Leibler Divergence[7] (in analogy to entropy in information theory):

$$\text{KL} = -\sum_{n=1}^{N}(y_n \log(\hat{y}_n) + (1 - y_n)\log(1 - \hat{y}_n)) \tag{2.34}$$

- Root-mean squared error (RMSE)

$$\text{RMSE}(\boldsymbol{\theta}) = \sqrt{\text{MSE}(\boldsymbol{\theta})} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n;\boldsymbol{\theta}))^2} \tag{2.36}$$

- and Mean absolute error (MAE)

$$\text{MAE}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}|y_n - f(\mathbf{x}_n;\boldsymbol{\theta})| \tag{2.37}$$

We note that the MSE and RMSE penalize large residuals ($\hat{y} - y$) more compared to MAE resulting in more outlier-sensitive training. Evidently the choice of loss function can strongly affect the quality and efficiency of training.

Optimisation algorithms: Gradient descent

Once a loss function $\mathcal{L}$ is defined, the goal of training is then to find the minima of this loss function with respect to the implicit parameters $\boldsymbol{\theta}$. These parameters represent the optimal set

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\Theta}{\text{argmin}}\ \mathcal{L}(\boldsymbol{\theta}) \tag{2.38}$$

The most basic differentiable optimisation algorithm is gradient descent. This iterative-first order method aims to locate the minima of a function, considering only gradient data and neglecting curvature information.

Initially, a random starting point for the network parameters $\boldsymbol{\theta}_0$ is selected. Subsequently, through the process of gradient descent, the parameters are iteratively updated by taking steps equivalent to the negative of the gradient (or estimation of the gradient) of all training samples:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla_\theta \mathcal{L}(\boldsymbol{\theta})\ |_{\boldsymbol{\theta}_t} \tag{2.39}$$

where α_t is the step-size or learning rate. Such a gradient step in which the entire dataset is used to update the parameters $\boldsymbol{\theta}$ is referred to as *epoch*. If the learning rate α is too small, we run the risk of getting stuck in a local minimum whereas if α is too large, we may be unable to locate a minimum. Such t-iterations (epochs) are performed until the optimisation method reaches a stationary point where the gradient $\mathbf{g}_t = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\ |_{\boldsymbol{\theta}_t}$ vanishes. This set of parameters corresponds to the optimal set of parameters $\boldsymbol{\theta}^*$. [8]

Popular extensions of gradient descent that are commonly employed in practice are stochastic gradient descent and its variant ADAM. These methods prove advantageous, largely due to the fact that stochastic gradient methods update the parameters $\boldsymbol{\theta}$ after each batch rather than after the completion of the full training set, thereby speeding up the training processes significantly.

[8] Concretely for the case of an MLP, gradient descent thus updates all the weights matrices $\mathbf{W}$ and the bias vector $\mathbf{b}$ in each epoch.

Updating the weights and biases[9] and computing the gradients in the optimization algorithms over all the parameters $\boldsymbol{\theta}$ that appear in (2.39) is done using the back-propagation algorithm. In modern machine learning framework such as PyTorch [21] and Tensorflow [9] the backpropagation algorithm is efficiently implemented through reverse-mode automatic differentiation.

2.5.3 Training, Testing and Validation

In practice not the whole data set $\mathcal{D}$ is used for training but a small subset of validation and testing samples are retained for evaluation and thus referred to as holdout sets. This is crucial since we want to evaluate the performance of the model on unseen data to ensure it generalizes well. These are typically defined in percentages of the whole dataset with common splits including (80,10,10) and (70,15,15) for training, validation and testing respectively.

Typically, after one epoch of optimizing the model parameters $\boldsymbol{\theta}$ using the training dataset, the model is evaluated on the validation dataset to obtain an intermediate result to how well the model generalizes. The validation loss can be used to probe whether under- or overfitting is present and inform necessary changes to hyperparameters and neural network architecture or the addition of regularization techniques to improve generalization.

After the model has been optimized on the validation and training set, the test set serves as the final evaluation of the trained model as an unbiased measure to how well the model predicts unseen data. The evaluation of the model on this dataset should not inform any further architecture or hyperparameter changes since that could incur an overfitting to the test set, which defeats the purpose of it as an unbiased estimate in the first place.

2.5.4 Generalisation and Bias-Variance tradeoff

Evidently from the last section, the goal of any optimisation algorithm is not the loss function minimisation but rather the generalisation capability of a model. To better understand what yields good generalization capabilities it is beneficial to consider machine learning in a probabilistic framework in which our models are described as a distribution of possible functions. Such a probabilistic perspective can be enlightening especially when we wish to additionally treat uncertainties for our predictors and formally define notions of generalization, underfitting, overfitting and capacity.

One such fundamental concept that quantifies the generalization capability of a model is the bias-variance tradeoff[10]. If we chose the mean-squared error to evaluate our model it can be shown that it is dictated by the bias and variance of the model as follows:

$$\mathrm{MSE}(\boldsymbol{\theta}) = \mathbb{E}[(\hat{y} - y)^2] = \mathrm{Var}[\hat{y}] + \mathrm{Bias}^2[\hat{y}] \tag{2.40}$$

[9] The error from the forward pass needs to propagate back through the network to determine which nodes contribute to the overall loss and inform how to update them to obtain a smaller overall, loss.

[10] Although the explicit dependence is on the predictor we we will denote the functional dependence on the implicit parameter $\boldsymbol{\theta}$ of the predictor (recall $\hat{y} = f(\mathbf{x}, \boldsymbol{\theta})$).

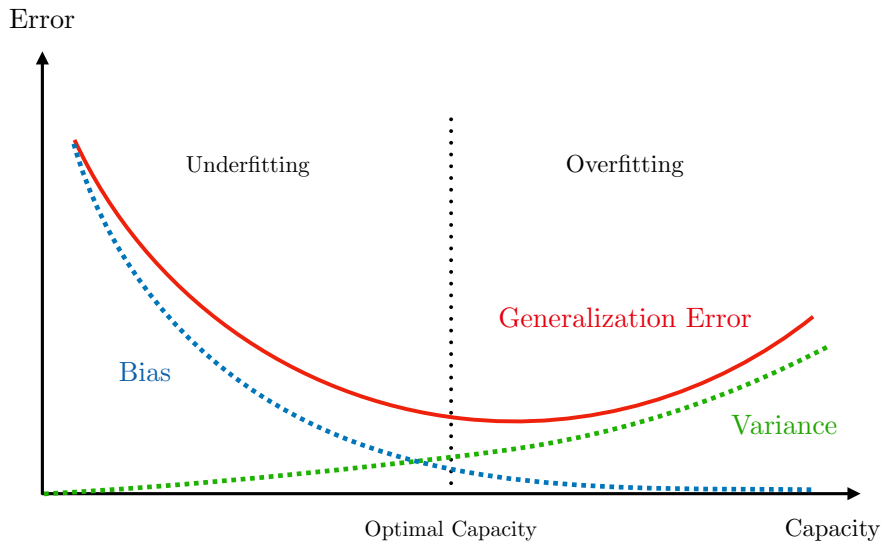


Figure 2.3: Visualisation of the Bias-Variance tradeoff. The y-axis can be interpreted as an Error.

The bias of the estimator $\hat{y}$:

$$\text{Bias}[\hat{y}] = \mathbb{E}[\hat{y}] - y = \mathbb{E}[f(\mathbf{x}; \boldsymbol{\theta})] - y \tag{2.41}$$

measures the expected deviation from the true value of the function or parameter[11]. High bias leads to a model that is too simple (small capacity) to fit the data properly, resulting in underfitting.

The variance

$$\text{Var}[\hat{y}] = \mathbb{E}[\hat{y}]^2 - \mathbb{E}[(\hat{y})^2] = \mathbb{E}[f(\mathbf{x}; \boldsymbol{\theta})]^2 - \mathbb{E}[(f(\mathbf{x}; \boldsymbol{\theta}))^2] \tag{2.42}$$

provides a measure of the deviation from the expected value that any particular sampling of the data is likely to cause. High model variance indicates that the model is too complex and learns the random fluctuations (or aleatoric error) in the training data, resulting in overfitting. Such models perform very well on the training data but are unable to generalise well to unseen data.

From equation (2.40) we see that the goal in machine learning models is to strike the right balance between bias and variance since reducing one type of error (bias or variance) usually leads to the increase of the other, as can be seen in Figure (2.3). The bias-variance tradeoff indicates that the optimal model complexity with the smallest generalization error is obtained by balancing model simplicity (large bias) and complexity (high variance).

[11] An unbiased estimator is one for which $\text{Bias}(\hat{\boldsymbol{\theta}}) = 0$ for which we get $\mathbb{E}[\hat{y}] = y$.

2.5.5 Transfer Learning

Instead of randomly initializing the parameters $\boldsymbol{\theta}_0$ of a model, transfer learning leverages pre-trained models with pre-determined parameters $\boldsymbol{\theta}_0$ as a starting point for training on a different but similar task. The motivation for transfer learning is that pre-trained models on a large dataset, have already learned how to extract relevant features and general patterns of the data and instead of discarding this knowledge it can be employed on a different problem to improve the accuracy. In a neural network, this can mean that most of the layers are 'frozen' and only the last few layers of the Neural Network are modulated to learn the intricacies of the current problem.

Compared to normal training, transfer learning thus requires much less data since we are essentially learning a much smaller neural network this usually results in similar or improved performance to normal training with a much faster convergence rate.

Chapter 3

Geometric GNNs and Representation Learning

Graph Neural Networks (GNNs) represent a new class of machine learning models designed to exploit relationships present in data that can naturally be modelled using graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that are comprised of a set of objects $\mathcal{V}$ (nodes) and their pairwise relationships $e_{ij} \in \mathcal{E}$ (edges). Such GNNs have been successfully applied to social systems and networks and as recommender systems.

Borrowing from the success of graph neural networks from other domains, materials representation learning using *geometric graph neural networks* [1]has emerged as a promising candidate for the creation of universal machine learned force fields (UFFs), able to predict forces (node-level predictions) of atoms and energies (graph-level prediction) for a large set of structures and chemistries.

Materials serving as inputs to such MLFFs are represented using geometric graphs. Geometric graphs $\mathcal{G} = (\mathbf{A}, \mathbf{S}, \vec{\mathbf{C}}, \vec{\mathbf{V}})$ in contrast to normal graphs embed each node in euclidean space in addition to modelling scalar quantities embedded in the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, describing the connectivity of each node in the graph and $\mathbf{S} \in \mathbb{R}^{n \times f}$ describing scalar features $\mathbf{s}_i \in \mathbf{R}^f$ such as the individual atom types Z_i of the i-t atom in the unit cell. Here n denotes the number of nodes and f the scalar feature dimension. In such geometric graphs in addition to these scalar values we also have geometric quantities such as velocity or forces, represented by $\vec{\mathbf{V}} \in \mathbb{R}^{n \times 3}$ and node coordinates $\vec{\mathbf{C}} = [\mathbf{r}_1, \dots \mathbf{r}_n] \in \mathbb{R}^{3 \times n}$ where n is the number of nodes.

In this chapter, we will elaborate on the current landscape of these universal machine learning force fields. We will describe the current most prevalent machine learning force field architectures, introduce message-passing neural network that serve as a foundation for many of these and discuss challenges, tradeoffs and avenues of improvement. A common theme among many architectures is the efficient embedding of symmetry.

[1]GNNs that embed symmetries are often referred to as geometric GNNs.

As such, a discussion of the required symmetries for a crystalline system and different implementations is conducted.

3.1 Machine learned Force Fields

Current implementations of MLFFs can broadly be classified as either descriptor-based or end-to-end [29].

Descriptor-based MLFF were the first types of models to be introduced that use fixed encoding rules of the local atomic environment are used as inputs to the neural network. Examples include the original Behler-Parrinello force field network [7, 8, 29].

End-to-End MLFF, which we use in this thesis, are a more recent development, where the embedding of the chemical charges, Cartesian coordinates and other node features from geometric graphs are learned by the model. Most such MLFF frameworks employ a *Message-Passing Scheme* on graphs formalized by Gilmer et al. [10], which we will introduce in section (3.3). Through this message passing between nodes (essentially an information exchange), chemical interactions can be modelled without the need of hard-coding them, thus allowing for greater flexibility and variety of interactions.

The training of MLFF on graphs follows the same concepts as traditional neural network training introduced in the last chapter. In a first step, the atomic energies $\hat{E}_i$ are predicted from the model using the atomic positions $\mathbf{r}_i$ and chemical elements Z_i. The total energy $\hat{E}$ is then obtained as the sum of the individual atom contributions $\hat{E}_i$

$$E = \sum_{i=1}^{N} \hat{E}_i \tag{3.1}$$

and the forces as negative gradient of that PES with respect to the atomic coordinates (this works because of conservative forces):

$$\mathbf{F}_i = -\boldsymbol{\nabla}_{\mathbf{r}_i} E\left(\left(\mathbf{r}_1, Z_1\right), \ldots, \left(\mathbf{r}_n, Z_n\right)\right) \tag{3.2}$$

To ensure energy conservation, most methods compute the per-atom force as the gradient of the predicted energy. The stress $\boldsymbol{\sigma}$ is computed using

$$\boldsymbol{\sigma} = \frac{1}{V} \frac{\partial E}{\partial \epsilon} \tag{3.3}$$

where V is the volume and ϵ the strain.

To assess the accuracy of the model we formalize the notion of a loss function represented as a linear combination of energy, forces and stress losses for MLFFs:

$$\mathcal{L} = \lambda_E \mathcal{L}_E(\hat{E}, E_{\text{DFT}}) + \lambda_F \mathcal{L}_F \left(\hat{\mathbf{F}}, \mathbf{F}_{\text{DFT}}\right) + \lambda_\sigma \mathcal{L}_\sigma(\hat{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) \tag{3.4}$$

where $\lambda_E, \lambda_F, \lambda_\sigma$ are the weights for energies, forces and stress, the quantities with hats correspond to the predicted values and the DFT values as ground truths. $\mathcal{L}_E, \mathcal{L}_F, \mathcal{L}_\sigma$ represent the mean absolute error, as introduced in the last chapter.

Much like the broader field of machine learning, achieving a suitable tradeoff between theoretical expressiveness [45] of representations and efficiency is paramount. Given the inherent difficultly in modelling interactions of complex structures, however, compounded by scalability issues that commonly arise in Graph Neural Networks, the creation of UFF presents itself as particularly challenging.

3.1.1 Symmetry and equivariance

The third challenge is the successful and efficient incorporation of symmetry into our models. Concretely, we want the predictions of the model not to depend on the arbitrary choice of reference frame in which we describe our structure, which requires our models to satisfy:

- *Permutation invariance*: We denote the permutation operation by P_σ then for any geometric graph node we require the following for permutation invariance:

$$P_\sigma \mathcal{G} = (P_\sigma \mathbf{A} P_\sigma^T, P_\sigma \mathbf{S}, P_\sigma \vec{\mathbf{V}}, P_\sigma \vec{\mathbf{C}}) = (\mathbf{A}, \mathbf{S}, \vec{\mathbf{V}}, \vec{\mathbf{C}}) \tag{3.5}$$

- *SE(3) equivariance*: Consists of translation and rotations (we may also consider $SE(3)$ which additionally includes reflections). Any scalar/vector or tensor predictor of our model $\boldsymbol{f}$ should be invariant/equivariant to any group operation in $E(3)$[2] respectively. Rotations $\mathcal{R}$ and translations $\mathbf{t}$ should act on the geometric graph nodes $(\mathbf{s}_i, \vec{\mathbf{v}}_i, \vec{\mathbf{r}}_i)$ as follows

$$(\mathbf{s}_i, \mathcal{R}\vec{\mathbf{v}}_i, \mathcal{R}(\vec{\mathbf{r}}_i + \mathbf{t})) \quad \forall i \in \mathcal{V} \tag{3.6}$$

  Meaning rotations and reflections should only act on vector $\vec{\mathbf{v}}_i \in \vec{\mathbf{V}}$ and coordinates $\vec{\mathbf{r}}_i \in \vec{\mathbf{C}}$ but leave the scalar features $\mathbf{s}_i \in \mathbf{S}$ invariant and translations only act on the coordinates $\vec{\mathbf{r}}_i \in \mathbf{C}$.

Formally a function $f : X \to Y$ is equivariant with respect to the group G that acts on X and Y if:

$$D_Y[g] f(x) = f(D_X[g]x), \quad \forall g \in G, \forall x \in X \tag{3.7}$$

where $D_X[g]$ and $D_Y[g]$ are the representations of the group element g in the vector spaces X and Y, respectively. Invariance is a type of equivariance where $D_Y[g] = \mathbf{1}$ for all elements of the group $g \in G$.

For example, the energy prediction, corresponding to a scalar value, of our model for an arbitrary 3D molecular or crystal structure should not change (remain invariant) if the molecule or crystal is translated or rotated in 3D space (equivalent to a translation or rotation of the underlying reference frame). The energy-predictions should thus be $SE(3)$-*invariant*. The per-atom-force predictions (vectors) however need to exhibit $SO(3)$-*equivariance* and translational invariance. This is because we want the force vectors to adjust their orientation in response to any rotation applied to the input structure accordingly. This has the convenient consequence that unlike invariant models that rely on the

[2] $SE(3)$ stands for special Euclidean group in 3D and is the semidirect product of $SO(3)$ with a translation group.

derivative of the energies to obtain the forces, equivariant MLFFs can directly predict the forces [51].

Implementing translation invariance in our model is straightforward due to the fact that most physical quantities of interest rely on the relative distances between entities rather than their absolute positions. As such, if we simply featurize distances and not absolute position, our predictor $\boldsymbol{f}$ will be translational invariant.

Incorporating the $SO(3)$ rotation invariance/equivariance, mentioned above for energies and forces, jointly in a GNN model can be realized if under any element in the rotation group $g \in SO(3)$, represented in 3D space by $\mathcal{R}(g)$, and vector $\mathbf{r} \in \mathbb{R}^3$, the predictors $\boldsymbol{f}$ of our model satisfies (acting on the coordinate channel of the geometric graph):

$$\boldsymbol{f}(\mathcal{R}(g)\mathbf{r}) = D^{\ell}(g)\boldsymbol{f}(\mathbf{r}) \tag{3.8}$$

Here we exploit the fact that the irreducible representations of $SO(3)$ are indexed by different 'rotation orders' ℓ (or angular frequency in analogy to quantum mechanics) that decouple. Each of these different irreducible representations have dimensions $(2\ell+1)$ for $\ell \in \mathbb{N}$ and are unitary. In real space, these irreducible representations take the form of Wigner D-matrices D^{ℓ}, which map elements of $SO(3)$ to $(2\ell+1) \times (2\ell+1)$-dimensional matrices. The predictor function $\boldsymbol{f}$ in such a representation is thus indexed by the rotation order ℓ and we have $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^{(2\ell+1)}$.

The rotation orders $\ell = 0, 1, 2$ correspond to scalars, vectors in 3-space and symmetric traceless matrices respectively. In the case of scalars these Wigner-D matrices are given by $D^{(0)}(g) = 1$ since scalars do not change under rotation essentially corresponding to the *invariance* property where the can recover the energy prediction, while for $\ell = 1$ corresponding to 3-vectors we get $D^{(1)}(g) = \mathcal{R}(g)$ where $\mathcal{R}$ represents the rotation matrix and the predictor function can represent force vectors.

One common way to encode the geometric information contained in $\mathbf{C}$ into $SE(3)$-equivariant features is through a spherical harmonics representation. Spherical harmonics map an input $3D$ vector $\mathbf{r}$ from the coordinate matrix $\mathbf{C}$ to a $(2\ell+1)$ dimensional vector and are equivariant to order ℓ rotations $\mathcal{R}(g)$. That is for all $g \in SO(3)$ we have:

$$Y_m^{\ell}(\mathcal{R}(g)\mathbf{r}) = \sum_{m'} D^{\ell}_{mm'}(g) Y^{\ell}_{m'}(\mathbf{r}) \tag{3.9}$$

The $2\ell+1$ spherical harmonics indexed by m represent the irreducible representations of $SO(3)$ for the closed subspace of ℓ-ordered rotations. They are indexed by the 'magnetic number' m. As such this can also be written in vector notation where for an order-ℓ rotation we obtain a spherical harmonics vector of dimension $(2\ell+1)$. Here $D^{\ell}(g)$ represents the Wigner D matrix as above.

One way to enforce symmetry in our predictions would be through symmetry-aware data augmentation. However, since translation and rotation are continuous symmetry operations in theory an infinite number of data points have to be generated to capture all possible symmetry operations in the data. Furthermore, there are no guarantees the

final model actually enforces true equivariance or simply creates approximate equivariant prediction.

A much more data-efficient approach would be incorporating symmetry constraints right in our model (or more precisely, each layer of the model), which serve as very strong inductive biases that can help narrow the parameter space the model needs to search (reducing the degree of freedoms of learning). Such symmetry-aware models have had significant success in recent years and come in many different flavors. These will be elaborated on in the next section.

3.2 Outline of existing molecular MLFFs

Existing 3D GNN layers used for MLFFs can broadly be categorized by their *tensor order* ℓ and *body order* v and whether the intermediate node features are invariant or equivariant. Tensor order refers to the discussion in the previous section about rotation orders for the featurization of the geometric information, such as positions. Networks with $\ell = 0$ are invariant 3D GNNs that only contain scalar features, such as SCHNet (2017) [14] and DimeNet (2020) [34]. This mean concretely that only invariant geometric features such as distances and angles are encoded and featurized for our nodes and edges and so all internal features over the entire network are going to be rotation and translational invariant (i.e not change when the input structure is translated or rotated) and transform according to $D^{\ell=0}(g) = \mathbf{1}$ in equation (3.8) [36, 45].

Equivariant 3D GNNs not only capture these scalar-invariant features in their nodes and edges but can also accommodate vector type features with $\ell = 1$, such as PaiNN (2021) [28] and even higher-order features with $\ell \geq 1$, for example Tensor Field Network (2018) [15], Nequip (2021) [32], Mace (2022) [30]. These vector and tensor features undergo non-trivial, equivariant transformations when the input structure is rotated or translated. Most of these models are based on the e3nn (Euclidean Neural Networks) framework [35].

While Tensor Field Networks, Nequip and Mace rely on tensor contractions involving Clebsch-Gordan coefficient to model equivariant interactions, PaiNN [43] takes a different approach by directly modelling equivariant interactions in Cartesian space and therefore avoiding the need for tensor contractions.

All the aforementioned GNN models rely on constructing message-passing involving the interactions of at most three local neighbors. Higher-order models such as Mace take inspiration from atomic cluster expansion and the decomposition of the potential energy surface (PES) as a linear combination of $\nu + 1$-body-ordered function (with ν representing the correlation order). Each body order indicates how many nodes are involved in the message construction. The relation between the two is given by $d = \nu + 1$ with the additional $+1$ originating from the counting the central atom i.):

$$E_i = f_1(\sigma_i) + \sum_{j}^{N} f_2(\sigma_i, \sigma_j) + \sum_{j_1,j_2} f_3(\sigma_i, \sigma_{j_1}, \sigma_{j_2}) + \cdots + \sum_{j_1,\ldots j_v} f_v(\sigma_i, \sigma_{j_1}, \ldots, \sigma_{j_v}) \quad (3.10)$$

Here i signifies the index of the central atom, j corresponds to the index of the neighbor node from i, i.e $j \in \mathcal{N}(i)$ and $\sigma_i = (\mathbf{r}_i, Z_i)$ denotes the atom state. A body-ordered expansion of the PES is beneficial when the series can be truncated at some finite body order d. This is feasible when higher-order terms are sufficiently insignificant to not substantially influence the overall PES and there is empirical evidence that for many systems this is the case [19].

3.2.1 Periodic information and long-range interactions

The decomposition of the total energy into atomic energies used in most MLFFs, as shown in the equation (3.1), is an approximation of the PES. The energy E_i for each atom is computed through some type of message-passing scheme that only takes into account information from other nodes within a small cutoff radius r_c around the central atom i. However, this localized approach fails to account for long-range effects. This issue is particularly problematic for crystalline MLFFs. Although the aforementioned molecular MLFFs frameworks are compatible with crystalline data, the unique structure of crystals with a periodically repeating unit cell offers a challenge. Therefore, in addition to $E(3)$-equivariance and permutation invariance we also need crystalline MLFFs to consider periodic information[3] and long-range effects to construct a geometrically-complete model.

In other words, effective crystalline representational learning has to additionally incorporate the transmission of periodic information (such as lattice lengths) in addition to the local geometric and atom state features. Periodic invariant multi-edge graphs that capture this periodic information are for example Matformer [42], PotNet [49] and Ewald-MP [46] which include additional features during the message-passing scheme to address the limitation of lacking periodic information. For example, Ewald-MP captures periodic information by a long-range signal (large frequency cutoff) in Fourier space while maintaining the short-range signal of the local environment due to a small cutoff radius for the message-passing scheme. A comparison of many of the frameworks can be seen in Table (3.1).

3.3 Message-Passing Neural Networks

Message-passing Neural Networks (MPNN) [11] represent a general class of permutation-equivariant Graph Neural Networks where graph node representations, represented using a T-Layer MLP, are updated by aggregating messages received from their neighbours. A visual representation of the MPNN framework can be seen in Figure (3.1).

Concretely, the message passing operation starts off by constructing $\mathbf{m}_i^{(t)}$ for each node $i \in \mathcal{V}$ in layer t by aggregating message functions $\psi^{(t)}$ from all neighboring nodes $j \in \mathcal{V}$

[3] Crystalline materials are therefore often represented as periodic graphs.

Models	Periodic Invariant	Symmetry	Periodic Pattern	Body Order (layer)
SchNet (2017)	✓	E(3) invariant	✗	2
CGCNN (2018)	✓	E(3) invariant	✗	2
TFN (2018)	✓	E(3) equivariant	✗	2
MEGNET (2019)	✓	E(3) invariant	✗	2
ALIGNN (2021)	✓	E(3) invariant	✗	3
PaiNN (2021)	✓	E(3) equivariant	✗	3
Nequip (2021)	✓	E(3) equivariant	✗	2
M3GNet (2022)	✓	E(3) invariant	✗	3
MACE (2022)	✓	E(3) equivariant	✗	higher-order
Matformer (2022)	✓	E(3) invariant	✓	2
PotNet (2023)	✓	E(3) invariant	✓	2
Ewald-MP (2023)	✓	E(3) invariant	✓	2

Table 3.1: Overview of different Graph Neural Networks (crystal graph neural networks with periodicity information and conventional molecular GNNs) mentioned in this thesis [51]

in the vicinity $\mathcal{N}(i)$ of node i via a permutation-invariant local pooling operation $\bigoplus$ [4]:

$$\mathbf{m}_i^{(t)} = f_{\text{AGG}}\left[\mathbf{h}_i^{(t)}, \{\mathbf{h}_j^{(t)}, \mathbf{e}_{ij} \mid j \in \mathcal{N}(i)\}\right] = \bigoplus_{j \in \mathcal{N}(i) \mid (i,j) \in \mathcal{E}} \psi^{(t)}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}) \tag{3.11}$$

For each message-passing iteration, the hidden state of each node $i \in \mathcal{V}$ for the next layer $(t+1)$ is then updated according to

$$\mathbf{h}_i^{(t+1)} = f_{\text{UPD}}[\mathbf{h}_i^{(t)}, \mathbf{m}_i^{(t)}] = \phi^{(t)}\left(\mathbf{h}_i^{(t)}, \bigoplus_{j \in \mathcal{N}(i) \mid (i,j) \in \mathcal{E}} \psi^{(t)}\left(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}\right)\right) \tag{3.12}$$

Both the message $\psi^{(t)}$ and update function $\phi^{(t)}$ represent learnable differentiable functions in the form of neural networks such as MLPs.

After an iteration of T message construction steps, the information in the hidden states can be used for further downstream tasks such as node/graph classification or edge prediction through some readout step.

In the case of graph-level predictions yielding the predictor $\hat{y}_n$ (such as the energy of a structure) we can interpret the readout step as another learnable aggregation function R of the feature vectors $\mathbf{h}_i^{(T)}$ for the final layer T over the entire graph:

$$\hat{y} = R(\{\mathbf{h}_i^{(T)}, i \in \mathcal{V}\}) \tag{3.13}$$

This global-pooling layer has to also be permutation invariant (i.e include operations such as element-wise sums or mean) since permuting the ordering of graph nodes and edges should not alter the final prediction.

Since MPNNs construct their message functions from messages received from their immediate neighbours, a t-layer MPNN scheme is in principle able to represent a t-hop

[4] Can represent the element-wise sum or mean for example

framework (i.e is able to communicate with nodes that are t hops away). However, due to GNNs oversmoothing and oversquashing characteristics, this long-range capturing of information is often not observed in practice.

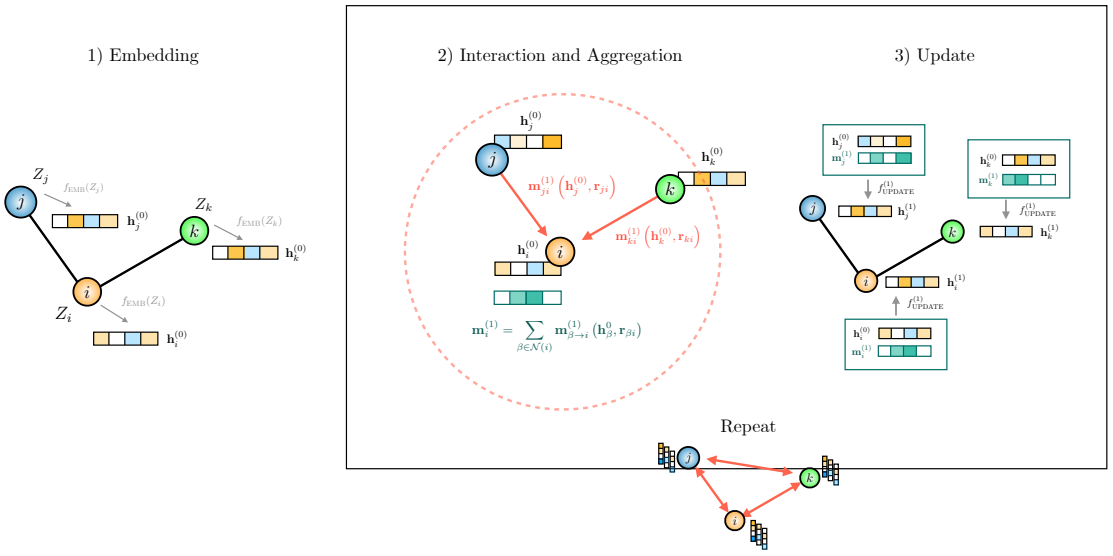


Figure 3.1: Illustration of the neural message-passing scheme [11]

Within the MPNN formalism, other common forms of machine learning layers such as convolutional and self-attention can be obtained.

For a convolutional layer, we use fixed weights for the edges a_{ij} between nodes i and j in the aggregation scheme. The node features of the neural network (or hidden state) $\mathbf{h}_i$ are updated using:

$$\mathbf{h}_i = \phi\left(\mathbf{h}_i, \bigoplus_{j \in \mathcal{N}_i} a_{ij}\psi(\mathbf{h}_j)\right) \tag{3.14}$$

We recover the convolutional GNNs from MPNNs (compare with eq. (3.12)) by restricting the messages ψ to be constructed only from the features of the local node (and not including neighbouring node features $\mathbf{h}_j$ of its connecting edge features $\mathbf{e}_{ij}$) with a fixed edge weight $\psi(\mathbf{h}_i, \mathbf{h}_j) = a_{ij}\psi(\mathbf{h}_j)$

In the self-attention layer, the interactions are not static but are obtained implicity from the interaction between node i and j through a learnable self-attention coefficient $e_{ij} = e(\mathbf{h}_i, \mathbf{h}_j)$ resulting in the hidden state:

$$\mathbf{h}_i = \phi\left(\mathbf{h}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{h}_i, \mathbf{h}_j)\psi(\mathbf{h}_j)\right) \tag{3.15}$$

The connection to MPNNs can be established via $\psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = a(\mathbf{h}_i, \mathbf{h}_j)\psi(\mathbf{h}_j)$

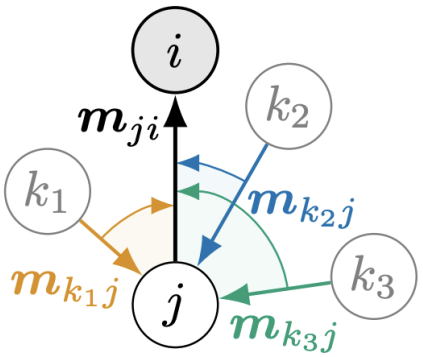


Figure 3.2: Directional Message Passing as introduced in DimeNet [34]. Note the aggregation away from the local site i, where all the node indices k are part of the neighbourhood $\mathcal{N}(i)$.

Evidently we have the following representational containment *convolution* $\subseteq$ *attention* $\subseteq$ *message-passing* since convolutional layers further correspond to an attentional layer with $a(\mathbf{h}_i, \mathbf{h}_j)$ being fixed.

3.4 Cartesian-invariant MLFFs ($\ell = 0$)

Invariant GNN layers aggregate scalar quantities from local neighbourhoods $\mathcal{N}(i)$ via scalarising the geometric information $\vec{\mathbf{V}}$:

$$\mathbf{s}_i^{(t+1)} = f_{\text{UPD}}^{(0)}\left(\mathbf{s}_i^{(t)}, f_{\text{AGG}}^{(0)}(\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \vec{\mathbf{v}}_j^{(t)}, \mathbf{e}_{ij})\right) \tag{3.16}$$

For e.g. SchNet[14] uses relative distances between the nodes $\|\mathbf{r}_{ij}\| = |\mathbf{r}_j - \mathbf{r}_i|$ as edge features $\mathbf{e}_{ij}$ to scalarise local geometric information

$$\mathbf{s}_i^{(t+1)} = f_{\text{UPD}}^{(0)}\left(\mathbf{s}_i^{(t)}, f_{\text{AGG}}^{(0)}(\mathbf{s}_j^{(t)}, \|\vec{\mathbf{r}}_{ij}\|)\right) = \mathbf{s}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} f_{\text{AGG}}^{(0)}(\mathbf{s}_j^{(t)}, \|\vec{\mathbf{r}}_{ij}\|) \tag{3.17}$$

Concretely in the aggregation and interaction step, the edge features embedded using a radial expansion are used to obtain a filter weight $W_{ij}^{(t)}$ determined by an MLP to inform the weight of each node scalar feature in the sum, i.e $\mathbf{s}_j \odot W_{ij}^{(t)}$.

This corresponds to a traditional message-passing scheme and as such we have body order 2 with complexity of $\mathcal{O}(nk)$ where n is the number of nodes and k the average 'connectivity' or degree[5].

[5]referring to the average number of bonds each atom in the structure forms

DimeNet [34] considers three-body interactions through angles $\mathbf{r}_{ij} \cdot \mathbf{r}_{ik}$ among triplets, as follows:

$$\mathbf{s}_i^{(t+1)} = \sum_{j \in \mathcal{N}(i)} f_{\text{UPD}}^{(0)} \left(\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}, \sum_{k \in \mathcal{N}(i) \setminus \{j\}} f_{\text{AGG}}^{(0)} \left(\mathbf{s}_j^{(t)}, \mathbf{s}_k^{(t)}, \|\vec{\mathbf{r}}_{ij}\|, \vec{\mathbf{r}_{ij}} \cdot \vec{\mathbf{r}}_{ik} \right) \right) \tag{3.18}$$

The updated scalar features are both $E(3)$ and translation invariant as the only geometric information used is the relative distances and angles, both of which remain unchanged under the action of $E(3)$ or translations.

3.4.1 M3GNet

In contrast to the other models we discuss, M3GNet [33] (Materials 3-body Graph Network) was built specifically for crystal structures[33]. It makes use of the multi-edge crystal graph representation (or crystal graph convolutional neural networks (CGCNN)) specifically designed for periodic crystal systems flattened as a line graph to encode the original edges to node features and angles between edges as edge features[17].

Similar to geometric graphs, a materials graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{C}, [\mathbf{M}, \mathbf{u}])$ includes node information $\mathbf{v}_i \in \mathcal{V}$, edge features $\mathbf{e}_{ij} \in \mathcal{E}$ and geometric information such as atomic positions $\mathbf{r}_i \in \vec{\mathbf{C}}$. As geometric features, we now have the optional global state information $\mathbf{u}$ and 3×3 lattice matrix $\mathbf{M}$[6].

Embedding

The graph structure is passed to a graph featurizer that embeds the pair atom distance $\mathbf{r}_{ij}$ up to a certain cut-off r_c to basis functions and the atomic number Z_i to element feature spaces. The element embedding converts the chemical elements into a node feature $\mathbf{v}_i^{(0)}$ of dimension 64. The bond distances are expanded using continuous and smooth basis functions which ensure that the first and second order derivatives vanish at the cut-off radius,

$$e_{\text{RBF},n}^{(0)}(\|\mathbf{r}\|) = \frac{1}{\sqrt{d_n}} \left[f_n(\mathbf{r}) + \sqrt{\frac{g_n}{d_{n-1}}} e_{n-1}(r) \right] \quad \text{with} \quad d_n = 1 - \frac{g_n}{d_{n-1}} \tag{3.19}$$

where

$$f_n(r) = (-1)^n \frac{\sqrt{2}\pi}{r_c^{3/2}} \frac{(n+1)(n+2)}{\sqrt{(n+1)^2 + (n+2)^2}} \left(\text{sinc}\left(r \frac{(n+1)\pi}{r_c} \right) + \text{sinc}\left(r \frac{(n+2)\pi}{r_c} \right) \right) \tag{3.20}$$

The initial bond $\mathbf{e}_{ij}^{(0)}$ is then a vector formed by these basis functions $e_{\text{RBF},n}^{(0)}$ with $n \in [0, N_{\text{RBF}}]$.

[6] These geometric coordinates are important to compute the forces and the lattice matrix as well as geometric coordinates are necessary for the computation of stresses.

Many-Body Computation

This module calculates the three-body interaction indices and their associated angles. There is essentially no change in node and edge features but simply a reformatting of the data. It uses the pairwise edge indices to create an index vector containing all three linked nodes with their associated angle. For example, if we have the pairwise edge connections (0,1), (0,2)), we would construct an index vector entry (0,1,2) which includes the associated three-body angle θ_{102}.

Main Block

This corresponds to the interaction block and consists of two steps, namely the many-body to bond module and standard graph convolutions.

The many-body to bond step computes the new bond information $\mathbf{e}_{ij}$ by considering the full-bonding environment $\mathcal{N}(i)$ of atom i with many-body angles θ_{ijk} and bond length $\mathbf{r}_{ij}$ and $\mathbf{r}_{ik}$. This is quite similar to the DimeNet framework, consisting of the following inputs for the aggregation message step:

$$\mathbf{m}_{ij}^{(t)} = \sum_{k \in \mathcal{N}(i) \setminus \{j\}} f_{\text{AGG}}^{(0)} \left(\mathbf{s}_j^{(t)}, \mathbf{s}_k^{(t)}, \|\vec{\mathbf{r}}_{ij}\|, \vec{\mathbf{r}_{ij}} \cdot \vec{\mathbf{r}}_{ik} \right) \tag{3.21}$$

The explicit construction for the many-body to bond step in M3GNet is given by:

$$\tilde{\mathbf{e}}_{ij}^{(t)} = \sum_{k \in \mathcal{N}(i)} J_l \left(z_{ln} \frac{r_{ik}}{r_c} \right) Y_l^0(\theta_{ijk}) \odot \sigma(\mathbf{W}_v \mathbf{v}_k + \mathbf{b}_v) f_c(\|\mathbf{r}_{ij}\|) f_c(\|\mathbf{r}_{ik}\|) \tag{3.22}$$

where J_l are the spherical Bessel functions with the roots at z_{ln}, Y_l^m is the spherical harmonics function, σ is the standard sigmoid activation function. The first part corresponds to a joint 2D basis for the interatomic distances and angles represented using the Fourier-Bessel basis: $\mathbf{a}_{\text{SBF}}^{(ki,ij)} \in \mathbb{R}^{N_{\text{SHBF}} \cdot N_{\text{SRBF}}}$ and is defined as

$$a_{\text{SBF}}^{(l,n)}(\mathbf{r}_{ik}, \theta) = \sqrt{\frac{2}{r_c^3 J_{l+1}^2(z_{ln})}} J_l \left(z_{ln} \frac{r_{ik}}{r_c} \right) Y_l^{m=0}(\theta) \tag{3.23}$$

The cutoff function in equation (3.22):

$$f_c(r) = 1 - 6 \left(\frac{r}{r_c} \right)^5 + 15 \left(\frac{r}{r_c} \right)^4 - 10 \left(\frac{r}{r_c} \right)^3 \tag{3.24}$$

ensures that the functions smoothly at the boundary, r_c since the Fourier-Bessel functions are not smoothly second-order differentiable there. We can see the above as the Neural Network learning to weigh each independent component l of the vector $\mathbf{e}_{ij}$ using the hidden state of the atom i and the relative distances through the componentwise multiplication.

Let g denote the non-linear activation function $g(x) = x\sigma(x)$, then the bond update for M3GNet is given by:

$$\mathbf{e}_{ij}^{(t+1)} = f_{\text{UPD}}(\mathbf{e}_{ij}^{(t)}, \tilde{\mathbf{e}}_{ij}^{(t)}) = \mathbf{e}_{ij}^{(t)} + g(\mathbf{W}_2\tilde{\mathbf{e}}_{ij}^{(t)} + \mathbf{b}_2) \odot \sigma(\mathbf{W}_1\tilde{\mathbf{e}}_{ij}^{(t)} + \mathbf{b}_1) \tag{3.25}$$

where all weights $\mathbf{W}$ and $\mathbf{b}$ appearing in (3.22) and (3.25) are learnable parameters in an MLP. Then using standard graph convolutions the internal state, bond information and optionally the global information are processed. The bond information is updated according to

$$\mathbf{e}_{ij}^{(t+1)} = \mathbf{e}_{ij}^{(t+1)} + \phi(\mathbf{v}_i^{(t)} \oplus \mathbf{v}_j^{(t)} \oplus \mathbf{e}_{ij}^{(t)} \oplus \mathbf{u}^{(t)})\mathbf{W}^{(0)}\mathbf{e}_{ij}^{(0)} \tag{3.26}$$

where $\phi(x)$ represents a gated MLP taking as input x the concatenated vectors consisting of node features $\mathbf{v}_i$ and $\mathbf{v}_j$, edge information $\mathbf{e}_{ij}$ calculated in (3.25) and the optional global state $\mathbf{u}$. A K-layer gated MLP can be represented as

$$\phi_K(x) = \left(\left(L_g^K \circ L_g^{K-1} \circ \dots L_g^1\right)(x)\right) \odot \left(\left(L_\sigma^K \circ L_g^{K-1} \circ \dots L_g^1\right)(x)\right) \tag{3.27}$$

where $L_g^k : x \mapsto \Phi\left(\mathbf{W}_k x + \mathbf{b}_k\right)$ is a single layer perceptron. The network on the left-hand side of (3.27) represents the normal network, where each output at the final layer K is element-wise $\odot$ multiplied by the node outputs from the gated MLP represented on the right-hand side of (3.27).

The internal node features are obtained by using the edge features in equation (3.26) and summing over $j \in \mathcal{N}(i)$ in analogy to the message aggregation step $\sum_j \mathbf{m}_{ji}^{(t+1)}$ for MPNNs:

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t+1)} + \sum_j \phi'(\mathbf{v}_i^{(t)} \oplus \mathbf{v}_j^{(t)} \oplus \mathbf{e}_{ij}^{(t)} \oplus \mathbf{u}^{(t)})\mathbf{W}'^{(0)}\mathbf{e}_{ij}^{(0)} \tag{3.28}$$

Out of the box, M3GNet initializes the gated MLPs in the graph convolution step to have two layers with 64 neurons in each layer.

After multiple such graph convolutions a final three-layer gated MLP $\phi_3(x)$ with neuron configuration [64,64,1] and no activation function[7] is used to predict the total energy.

3.5 Cartesian-equivariant MLFFs ($\ell = 1$)

For MLFFS with $\ell = 1$ we have to restrict updates and aggregations in the message passing scheme to certain vector operations to ensure equivariance. Such operations include scaling of vectors $\mathbf{s} \odot \mathbf{v}$ ($\odot$ represents element-wise multiplication), summation of vectors $\mathbf{v}_1 + \mathbf{v}_2$, linear transformation of vectors $W\mathbf{v}$, scalar products $\mathbf{v}_1 \cdot \mathbf{v}_2$, norms $\|\mathbf{v}\|$ and vector products $\mathbf{v}_1 \times \mathbf{v}_2$ [51] [28].

[7] Only in the last layer for the normal part of the gated MLP, the gated part still has an activation function

Cartesian-equivariant MLFFs propagate and update both scalar and vector message information with vector operations constrained to the example mentioned above:

$$\begin{aligned}
\mathbf{m}_i^{(t)} &= \sum_{j \in \mathcal{N}(i)} f_{\text{AGG}}^{(0)}(\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \vec{\mathbf{v}}_j^{(t)}, \mathbf{e}_{ij}) \\
\vec{\mathbf{m}}_i^{(t)} &= \sum_{j \in \mathcal{N}(i)} f_{\text{AGG}}^{(1)}(\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \vec{\mathbf{v}}_j^{(t)}, \mathbf{e}_{ij}) \\
\mathbf{s}_i^{(t+1)} &= \sum_{j \in \mathcal{N}(i)} f_{\text{UPD}}^{(0)}(\{s_i^{(t)}, \mathbf{v}_i^{(t)}\}, \{m_i^{(t)}, \mathbf{m}_i^{(t)}\}) \\
\vec{\mathbf{v}}_i^{(t+1)} &= \sum_{j \in \mathcal{N}(i)} f_{\text{UPD}}^{(1)}(\{s_i^{(t)}, \mathbf{v}_i^{(t)}\}, \{m_i^{(t)}, \mathbf{m}_i^{(t)}\})
\end{aligned} \tag{3.29}$$

For example, PaiNN [28] has interaction layers that aggregate scalar and vector features via a learned radial filter conditioned on the relative distance. It augments the invariant SchNet into equivariant flavor by projecting the interatomic distances via radial basis functions (corresponding to a linear transformation) and iteratively updating the vectors along with the scalar features:

$$\begin{aligned}
\mathbf{s}_i^{(t)} &= \mathbf{s}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} f_{\text{AGG}}(\mathbf{s}_j^{(t)}, \|\vec{\mathbf{r}}_{ij}\|) \\
\vec{\mathbf{v}}_i^{(t)} &= \vec{\mathbf{v}}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} f_2(\mathbf{s}_j^{(t)}, \|\vec{\mathbf{v}}_{ij}\|) \odot \vec{\mathbf{v}}_j^{(t)} + \sum_{j \in \mathcal{N}(i)} f_3(\mathbf{s}_j^{(t)}, \|\vec{\mathbf{r}}_{ij}\|) \odot \vec{\mathbf{r}}_{ij}
\end{aligned} \tag{3.30}$$

In the update step we apply a gated non-linearity on the vector features, which learns to scale their magnitude using their norm concatenated with the scalar features [16]. So the scalar and vector features are updated according to:

$$\begin{aligned}
\mathbf{s}_i^{(t+1)} &= \mathbf{m}_i^{(t)} + f_4(\mathbf{m}_i^{(t)}, \|\vec{\mathbf{m}}_i^{(t)}\|) \\
\vec{\mathbf{v}}_i^{(t+1)} &= \mathbf{m}_i^{(t)} + f_5(\mathbf{m}_i^{(t)}, \|\vec{\mathbf{m}}_i^{(t)})\| \odot \mathbf{m}_i^{(t)}
\end{aligned} \tag{3.31}$$

The vector features exhibit both $E(3)$ equivariance and translational invariance since the aggregation and update operations solely involve vector scaling, vector summation, linear transformations and scalar products, all of which are equivariance-preserving operations on vectors mentioned at the start of this section. On the other hand, the scalar features $\mathbf{s}_i$ remain invariant throughout the message-passing scheme, since we only rely on relative distance information as our geometric information.

3.6 Spherical tensor equivariant MLFFs $(\ell \geq 1)$

A general overview of such spherical equivariant MLFFs models can be seen in Figure (3.3). After an initial embedding of the scalar features of a graph such as the atomic numbers, multiple interaction blocks follow, each described by a tensor-MPNN layer consisting of

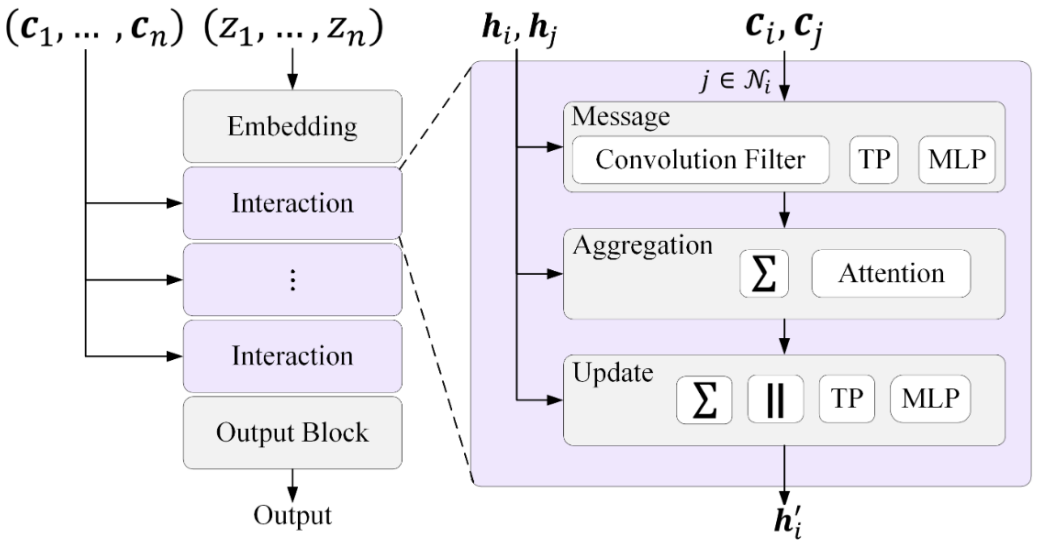


Figure 3.3: Architecture for spherical equivariant MLFFs. Figure taken from [51].

message construction, aggregation and update steps, which make use of tensor-products and Clebsch-gordan coefficients to retain equivariance [8]

3.6.1 Embedding block

The first block in most MLFFs is the chemical embedding block, in which the elements Z_i are one-hot encoded to vectors $\boldsymbol{\theta} : \{Z_i\} \rightarrow \mathbb{R}^a$ of lengths equal to the unique number of elements a in the system. The initial features $\mathbf{h}_i^{(0)}$ for each atom i are obtained by multiplying these one-hot embedding vectors $\boldsymbol{\theta}$ with a learnable weight matrix $\mathbf{W}$ of size $N_{\text{elements}} \times N_{\text{channels}}$ corresponding to a trainable self-interaction layer.

The node features after the initial embedding step can now, in addition to scalar $\mathbf{s}$ ($\ell = 0$) and vector features $\vec{\mathbf{v}}$ ($\ell = 1$), include higher-order feature vectors ($l_{\text{max}} \geq \ell \geq 2$). Many features for each rotation order can exist and are labelled by the channel index c ((i.e each element in the feature vector), the atom i and the representation index m. We will denote the features of node i by h_{icm}^{ℓ} or sometimes for brevity simply in vector notation $\mathbf{h}_i^{\ell}$.

[8] More precisely, to preserve equivariance across the entire message-passing scheme we will see that all filters and layers need to inhabit representations of $SO(3)$ that lead to outputs that are again equivariant. To ensure this, all operations will be restricted to tensor-products and linear operations.

3.6.2 Interaction block

Self-interaction

At the start of each message-construction and after the initial embedding among others (see Figure (3.4)), for NEQUIP and TFN linear operations on the node features $\mathbf{h}_i^\ell$ are performed. These are inspired by SCHNET and are also employed in TFN's and mix components of the feature vectors for each atom i in a learnable way through a weight matrix $\mathbf{W}$:

$$W(\mathbf{h}^\ell) = \sum_{c'} W_{cc'}^\ell \mathbf{h}_{ic'm}^\ell \tag{3.32}$$

To enforce equivariance, the weights are constant for every m for a given rotation order ℓ.

Message-construction

This tensor-based message passing scheme first computes a message $\mathbf{m}_i^{\ell_3} \in \mathbb{R}^{2\ell_3+1}$ through the aggregation over pairwise messages from neighboring nodes through a tensor-product operation $\text{TP}_{\ell_1,\ell_2}^{\ell_3}$:

$$\mathbf{m}_i^{(\ell_3,t)} = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{j\to i}^{(\ell_3,t)} = \sum_{j\in\mathcal{N}(i)} \text{TP}_{\ell_1,\ell_2}^{\ell_3}(\mathbf{r}_i - \mathbf{r}_j, \mathbf{h}_j^{(\ell_1,t)}) \tag{3.33}$$

where $\mathcal{N}(i)$ denotes the neighboring set of nodes j smaller than some cutoff radius hyperparamter r_c, i.e $\mathcal{N}(i) = \{j : \|\mathbf{r}_i - \mathbf{r}_j\|_2 \leq r_c\}$.

Both NEQUIP and TFN employ sum aggregation but of course alternatives exists. For example the more recent, EQUIFORMER (2023) [48] additionally weighs each message by attention.

Interaction

The TP operation in the message construction (3.33), crucially, makes use of the spherical harmonics (recall that these functions are $SO(3)$ equivariant, see equation (3.9)). Both NEQUIP and TENSOR-FIELD NETWORKS employ rotation-equivariant pointwise-convolution filters F

$$F_{cm}^{\ell_2,\ell_1}(\mathbf{r}) = R_c^{\ell_2,\ell_1}(r) Y_m^{\ell_2}(\hat{\mathbf{r}}) \tag{3.34}$$

composed of spherical harmonics $Y_m^{\ell_2}$ with rotation order ℓ_2 and magnetic number $m = -\ell_2, -\ell_2+1, \ldots, \ell_2$, that featurize the direction and a learnable radial function R, represented as an MLP:

$$R(r_{ij}) = \mathbf{W}_n \Phi(\ldots \Phi(\mathbf{W}_2 \Phi(\mathbf{W}_1 B(r_{ij})))) \tag{3.35}$$

that featurizes the relative distance r_{ij} between nodes[9]. Here Φ denotes the activation function, $\mathbf{W}$ the learnable weight matrices and $B(r_{ij})$ a basis embedding.

[9]Typically one uses three-layers in the MLP

For NEQUIP the non-linear activation functions for the radial MLP are SiLU activation functions

$$\text{SiLU}(x) = \frac{x}{1+e^{-x}} \tag{3.36}$$

and the initial radial embedding $B(r_{ij})$ is given by Radial Bessel functions

$$B(r_{ij}) = \frac{2}{r_c} \frac{\sin\left(\frac{n\pi}{r_c} r_{ij}\right)}{r_{ij}} f_{\text{env}}(r_{ij}, r_c) \tag{3.37}$$

The envelope polynomials f_{env} are the same as in DIMENET and satisfy $\lim_{r_{ij}\to 0} f_{\text{env}}(r_{ij}, r_{\text{cut}}) \to 0$ [34].

The index c in equation (3.34) refers to the channel index and denotes the multiple instances of each l-rotation order irreducible representations. What this concretely represents will become evident when we discuss the Clebsch-Gordan coefficients.

Because the radial functions only depend on the relative distance between nodes, the scalar outputs of the learnable functions are therefore rotation invariant, these filters are rotation-equivariant since they borrow their rotational properties from the spherical harmonics.

The tensor-product interaction in (3.33) now employs these filters F, using ℓ_2-order spherical harmonics functions as the kernel, to compute the messages $\mathbf{m}_{j\to i}^{(\ell_3,t)}$ propagated from every node j in $\mathcal{N}(i)$ to the node i. Let $\otimes$ represents the outer-product operation, i.e. $\mathbf{a}\otimes\mathbf{b} = \mathbf{a}\mathbf{b}^T$ and $\text{vec}(\cdot)$ the operation that flattens a matrix to a vector, then the layer interaction is given by:

$$\text{TP}_{\ell_1,\ell_2}^{\ell_3}\left[\mathbf{r}_i - \mathbf{r}_j, \mathbf{h}_j^{(\ell_1,t)}\right] = C_{(\ell_1,\ell_2)}^{\ell_3} \text{vec}\left(R(r_{ij})\mathbf{Y}_{\ell_2}(\hat{\mathbf{r}}_{ij}) \otimes \mathbf{h}_j^{(\ell_1,t)}\right) \tag{3.38}$$

which explicitly written out yields the following layer definition $\mathcal{L}_{icm_3}^{\ell_3}(\mathbf{r}_i - \mathbf{r}_j, h_{icm_1}^{\ell_1})$:

$$\text{TP}_{\ell_1,\ell_2}^{\ell_3}\left[\mathbf{r}_i - \mathbf{r}_j, \mathbf{h}_j^{(\ell_1,t)}\right] = \sum_{m_2,m_1} C_{(\ell_2 m_2)(\ell_1 m_1)}^{(\ell_3,m_3)} \sum_{j\in\mathcal{N}_i} F_{cm_2}^{\ell_2,\ell_1}(\mathbf{r}_{ij}) h_{jcm_1}^{\ell_1} \tag{3.39}$$

where the indices $1, 2, 3$ refer to the input, filter and output indices respectively and $C_{\ell_1,\ell_2}^{\ell_3}$ is the Clebsh-Gordan matrix with $(2\ell_3+1)$ rows and $(2\ell_1+1)\times(2\ell_2+1)$ columns. For a general tensor-product between two vectors $u^{(\ell_1)}$ and $v^{(\ell_2)}$ with rotation order ℓ_1 and ℓ_2 respectively

$$h_{m_3}^{\ell_3} = (u^{(\ell_1)} \otimes v^{(\ell_2)})_{m_3} = \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C_{(\ell_1,m_1),(\ell_2,m_2)}^{(\ell_3,m_3)} u_{m_1}^{(\ell_1)} v_{m_2}^{(\ell_2)} \tag{3.40}$$

the Clebsch-Gordan coefficients indicate which coupling between different m and ℓ for the input node features and filters are possible[10]. To acquire non-zero Clebsch-Gordan

[10]This is in analogy to the addition of angular momenta in quantum mechanics. Clebsch-Gordan coefficients allow us to express the "total angular momentum" basis $|\ell_3, m_3; \ell_1, \ell_2\rangle$ in terms of the direct product basis $|\ell_1, \ell_2; m_1, m_2\rangle = |\ell_1 m_1\rangle \otimes |\ell_2 m_2\rangle$ through the simple expansion (inserting a projection operator)

The Clebsch-Gordan coefficient is given by $\langle \ell_1, \ell_2, m_1, m_2 | \ell_3, m_3, \ell_1, \ell_2\rangle$.

coefficients, the rotation order ℓ_3 for the resulting vector h^{ℓ_3} needs to lie between $|\ell_1 - \ell_2|$ and $|\ell_1 + \ell_2|$ and in most models vectors with $\ell_3 \geq \ell_{\text{max}}$, where ℓ_{max} is a hyperparmeter, are discarded [11].

The distinct combinations of $\ell_1 \otimes \ell_2 \to \ell_3$ can result in multiple output tensor for a given output rotation order ℓ_3 as a result of the different combinations of input ℓ_1, m_1 and filter ℓ_2, m_2. We collect all such possible output tensors with $\ell_3 \leq \ell_{\text{max}}$, classify them by the channel index c, and concatenate them.

The Clebsch-Gordan coefficients for certain channels have very intuitive interpretations with regard to what mathematical operation they are trying to model. For example, $\ell_3 = 0$ we obtain standard scalar and vector operations that yield scalars. These correspond to the dot-product $1 \otimes 1 \to 0$, where two vectors are contracted yielding a scalar denoted by Clebsch-Gordan coefficient $C^{(0,0)}_{(1,i)(1,j)} \propto \delta_{ij}$ and standard scalar-scalar multiplication $0 \otimes 0 \to 0$. When $\ell_3 = 1$, we can obtain the combinations $1 \otimes 1 \to 1$ representing the cross product with

$$C^{(1,i)}_{(1,j),(1,k)} = \epsilon_{ijk} \tag{3.41}$$

where ϵ_{ijk} represents the Levi-Civita symbol and $1 \otimes 0 \to 1$ and $0 \otimes 1 \to 1$ which corresponds to simply scalar multiplication of a vector. As detailed in the beginning of Section (3.5) all these operations adhere to the principles of equivariance, meaning each channel of $h^{\ell_3}_{m_3}$, such as scalar, vector and higher-order features corresponding to $\ell = 0, 1, 2$, preserves its own independent equivariance. This can be summarized generally by the following condition of tensor-products[12]:

$$\text{TP}^{\ell_3}_{\ell_1\ell_2}\left[\mathcal{R}(g)\mathbf{r}_i - \mathcal{R}(g)\mathbf{r}_j, D^{\ell_1}(\mathcal{R})\mathbf{h}^{(\ell_1,t)}_i\right] = D^{\ell_3}(\mathcal{R})\text{TP}^{\ell_3}_{\ell_1,\ell_2}\left[\mathbf{r}_i - \mathbf{r}_j, \mathbf{h}^{(\ell_1,t)}_i\right] \tag{3.42}$$

As such stacking multiple such message-passing layers based on tensor-products preserves equivariance over the interaction part of the network.

Update

The tensor-based MPNN finishes by updating the hidden state $\mathbf{h}^{(\ell_1,t)}_i$ to new node features $\mathbf{h}^{(\ell_1,t+1)}_i$ using the aggregated message, (3.33) according to

$$\mathbf{h}^{(\ell_1,t+1)}_i = f_{\text{UPD}}(\mathbf{h}^{(\ell_1,t)}_i, \mathbf{m}^{(\ell_3,t)}_i) \tag{3.43}$$

where the node feature update function f_{UPD} can contain linear or another TP operation.

In TFN's the update function consist of the aggregation of the messages as in (3.33) followed by another linear self-interaction part.

[11] If we restrict a Nequip model to only include $0 \times 0 \to 0$-type interactions we would retrieve an invariant GNN model similar to SchNet

[12] It is important to mention that the representations of the group operations differ. In this context $\mathcal{R}(g)$ signifies 3D rotation group operations and acts on the 3D coordinates of a geometric graph. Conversely, $D^{\ell}(\mathcal{R})$ represents the generalized rotations that act on the geometric features $\mathbf{h}$ of arbitrary dimension.

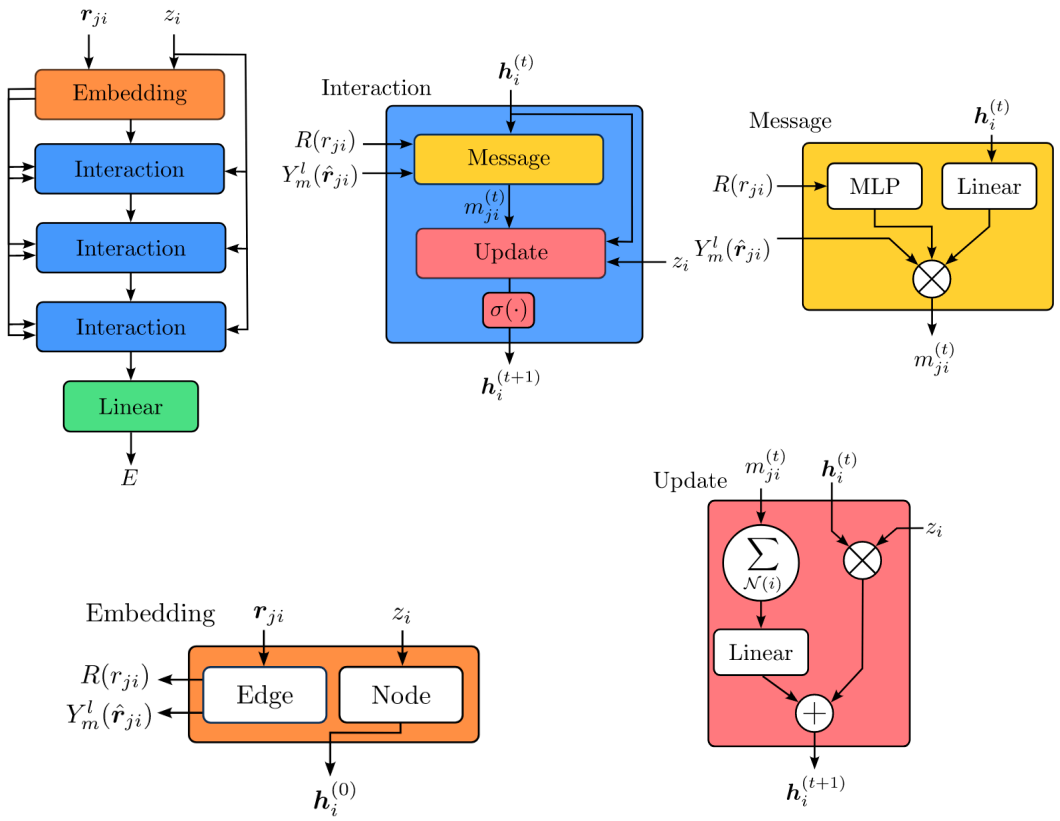


Figure 3.4: Flowchart of the Nequip Architecture. Figure taken from [31].

In NEQUIP a special type of self-interaction operation, absent in TFN's, is computed after each message-passing layer as part of the update step and is reminiscent of residual connections common in convolutional neural networks. This learnable self-connection reinjects the embedded chemical information about the central atom $\boldsymbol{\theta}_i$ and information from the previous layer $\mathbf{h}_i^{(t)}$ to concatenate to the linearly transformed message-aggregation result from TFN. We have

$$h_{icm}^{(\ell_3,t+1)} = h_{icm}^{(\ell_3,t+1)} + \sum_{ac'} W_{cc'a}^{\ell_3} \theta_{i,a} h_{ic'm}^{(\ell_3,t)} \tag{3.44}$$

where $W_{cc'\ell_3 a}$ is a learnable weight matrix of size $[N_{\text{channels}} \times N_{\text{channels}} \times \ell_{\max} \times N_{\text{elements}}]$

Output Non-linearity

As the last step in the message-passing scheme an optional additional non-linearity σ to the tensor-products can be employed. To preserve equivariance these non-linearity functions need to however act as scalar transforms, i.e. $g : \mathbb{R} \to \mathbb{R}$ for each ℓ along the m dimension. In Nequip we employ square-norm gated non-linearity of the form:

$$\sigma(h_{icm}^{\ell_3,t+1)}) = g(\|h_{icm}^{(\ell_3,t+1)}\|) h_{icm}^{(\ell_3,t+1)} \tag{3.45}$$

where g is the SiLU or tanh function.

3.6.3 Output Block

In summary, the nodes of spherical equivariant geometric graph neural networks given by higher-order tensor features $\mathbf{h}_{i,l} \in \mathbb{R}^{(2l+1)\times f}$ with $l = 0, \dots L$ are updated with tensor products $\otimes_{\mathbf{w}}$ consisting of a learnable radial function which takes scalar and vector features with spherical harmonic expansion of displacement $Y_{l_1}^{m_1}(\hat{\mathbf{r}}_{ij})$.

3.7 Higher-body order methods

The equivariant-methods discussed in the last section are only using two-body interactions in their message-passing scheme. Higher-body order equivariant methods such as MACE implement many-body interactions with neighbours. MACE takes inspiration from the Atomic Cluster Expansion and the so-called 'density trick'.

3.7.1 Atomic Cluster Expansion (ACE)

The first step in Atomic Cluster Expansion [19] is to construct the single-bond basis using spherical harmonics and radial function:

$$\phi_{c\ell m}(\mathbf{r}_{ij}) = \langle \mathbf{r}_{ij} | c, \ell, m \rangle = R_c(r_{ij}) Y_\ell^m(\hat{\mathbf{r}_{ij}}) \tag{3.46}$$

For brevity in our expressions we introduce a multi-index $\alpha = (c, \ell, m)$ combining channel c (corresponding to the principal quantum number n), rotation order ℓ (angular momentum) and representation index m (magnetic number).

These functions span a complete and orthonormal basis (depending only on the single bond vector $\mathbf{r_{ij}}$):

$$\int \phi^*_\alpha(\mathbf{r}_{ij})\phi_\beta(\mathbf{r}_{ij})\mathrm{d}\mathbf{r} = \delta_{\alpha\beta} \quad \text{and} \quad \sum_\alpha \phi^*_\alpha(\mathbf{r}_{ij})\phi_\alpha(\mathbf{r}'_{ij}) = \delta(\mathbf{r}_{ij} - \mathbf{r}'_{ij}) \tag{3.47}$$

In the next step, similar to MPNN, we sum over the neighbours j in the surrounding $\mathcal{N}(i)$ of the central atom i to reduce the two-particle basis to the atomic basis (for completeness an additional index relating to the atomic species Z_i

$$A_{i\alpha} = \langle \rho_i^Z | \phi_\alpha \rangle = \sum_{j\in\mathcal{N}(i)} \phi_\alpha(\mathbf{r}_{ij}) \tag{3.48}$$

This is the so-called density trick since we are projecting the one-particle basis onto the neighbourhood density of atom i (with chemical element Z) given by

$$\rho_i^Z(\mathbf{r}) = \sum_j \delta_{ZZ_j}\delta(\mathbf{r} - \mathbf{r}_j) \tag{3.49}$$

This atomic basis is permutation invariant and can in principle represent any function that depends on all neighbours around $\mathcal{N}(i)$. However, it will only be composed of 2-body functions and can therefore only model functions such as $f(\mathbf{r}_i, \mathbf{r}_j)$ but not higher-order multivariate functions such as f_3 from equation (3.10).

To construct higher-body order functions, ACE constructs the atomic product basis, a complete basis of permutation-invariant functions:

$$\mathbf{A}_{i\boldsymbol{\alpha}} = \prod_{\xi=1}^{\nu} A_{i\alpha_\xi} = \prod_{\xi=1}^{\nu} \sum_{j\in\mathcal{N}(i)} \phi_{\alpha_\xi}(\mathbf{r}_{ij}), \quad \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots \alpha_\nu) \tag{3.50}$$

with ν corresponding to the correlation order. We note the similarities to the construction of the cluster correlation function (2.23) for Cluster Expansion.

The product basis can then be used to represent the atomic energy via a body-order expansion as such:

$$E_i = \sum_\alpha c^{(1)}_\alpha A_{i\alpha} + \sum_{\alpha_1,\alpha_2}^{\alpha_1\geq\alpha_2} c^{(2)}_{\alpha_1,\alpha_2} A_{i\alpha_1}A_{i\alpha_2} + \sum_{\alpha_1,\alpha_2,\alpha_3}^{\alpha_1\geq\alpha_2\geq\alpha_3} c_{\alpha_1\alpha_2\alpha_3} A_{i\alpha_1}A_{i\alpha_2}A_{i\alpha_3} + \dots \tag{3.51}$$

The advantage over using the traditional single-bond basis as a body-ordered expansion of the atomic energy:

$$E_i = \sum_{j,\alpha} c^{(1)}_\alpha \phi_\alpha(\mathbf{r}_{ij}) + \frac{1}{2}\sum_{j_1,j_2}^{j_1\neq j_2} \sum_{\alpha_1,\alpha_2} c^{(2)}_{\alpha_1,\alpha_2}\phi_{\alpha_1}(\mathbf{r}_{ij_1})\phi_{\alpha_2}(\mathbf{r}_{ij_2}) + \dots \tag{3.52}$$

is that the expression (3.51) only scales as $\mathcal{O}(n)$ where n represents the number of neighbours rather than exponentially $\mathcal{O}(n^d)$ for (3.52). This is one of the profound advantages of the ACE representation.

3.7.2 MACE

MACE is based on a spherical equivariant message passing scheme. It however additionally employs the ACE framework to construct higher body-order correlation functions that are used in the message update step:

$$\begin{aligned}\mathbf{m}_i^{(t)} = \sum_{j\in\mathcal{N}(i)} f_{\text{UPD}}^{(1,t)}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}) + \sum_{j_1,j_2\in\mathcal{N}(i)} f_{\text{UPD}}^{(2,t)}(\mathbf{h}_i^{(t)}, \mathbf{h}_{j_1}^{(t)}, \mathbf{h}_{j_2}^{(t)}) + \\ \cdots + \sum_{j_1,\ldots j_\nu\in\mathcal{N}(i)} f_{\text{UPD}}^{(\nu,t)}(\mathbf{h}_i^{(t)}, \mathbf{h}_{j_1}^{(t)}, \ldots, \mathbf{h}_{j_\nu}^{(t)})\end{aligned} \tag{3.53}$$

The maximal correlation-order ν is a hyperparameter in MACE.

Now in MACE the bond-basis functions ϕ are used as the convolution filter F, to interact with the internal node features via a tensor-product, where the contraction is again dictated by the Clebsch-Gordan coefficients:

$$A_{icm_3}^{(\ell_3,t)} = \sum_{\ell_1 m_1 \ell_2 m_2} C_{(\ell_1 m_1),(\ell_2 m_2)}^{(\ell_3,m_3)} \sum_{j\in\mathcal{N}(i)} R_c^{(t,\ell_1\ell_2)}(\mathbf{r}_{ij}) Y_{\ell_2}^{m_2}(\hat{\mathbf{r}}_{ij}) \sum_{\tilde{k}} W_{c\tilde{c}}^{(\ell_1,t)} h_{j\tilde{c}m_1}^{(\ell_1,t)} \tag{3.54}$$

The Clebsch-Gordan coefficients ensure that the $A_{icm_3}^{(\ell_3,t)}$ is equivariant [13]. Notice the similarities to the message construction in equation (3.39). In MACE the tensor product is however formed between the convolution filter, taking the same form as in Nequip (see equation (3.34)) and node features linearly transformed by a self-interaction operation controlled by a learnable weight matrix $W_{cc'}^{\ell_2}$ (i.e TP$(F(\cdot), W(\mathbf{h}_j))$ instead of TP$(F(\cdot), \mathbf{h}_j)$).

To get a ν-correlation order basis we could again consider the product basis:

$$\mathbf{A}_{ic\boldsymbol{m}}^{(\boldsymbol{\ell},t)} = \prod_{\xi}^{\nu} A_{icm_\xi}^{(\ell_\xi,t)}, \qquad \boldsymbol{\ell}\mathbf{m} = (\ell_1 m_1, \ldots, \ell_\nu m_\nu) \tag{3.55}$$

Although this product basis is permutation and translational invariant, it breaks the rotational equivariance. As such we can't use this to represent internal features since if we want to retain an equivariant model each layers representation have to be equivariant.

In MACE these rotationally-equivariant higher-order basis functions are obtained by properly symmetrizing the product basis $\mathbf{A}$ via:

$$\mathbf{B}_{i\eta_\nu cM}^{(L,t)} = \sum_{\boldsymbol{\ell},\mathbf{m}} \mathcal{C}_{\eta_\nu,\boldsymbol{\ell}\mathbf{m}}^{L,M} \prod_{\xi=1}^{\nu} \sum_{c'} W_{cc'\ell_\xi}^{(t)} A_{ic'\ell_\xi m_\xi}^{(t)}, \quad \boldsymbol{\ell}\mathbf{m} = (\ell_1 m_1, \ldots, \ell_\nu m_\nu) \tag{3.56}$$

where $\mathcal{C}_{\eta_\nu,\boldsymbol{\ell}\mathbf{m}}^{L,M}$ corresponds to the generalized Clebsch-Gordan coefficients given as the product of Clebsch-Gordan coefficients:

$$\mathcal{C}_{\ell_1 m_1,\ldots\ell_\nu m_\nu}^{LM} = C_{\ell_1 m_1,\ell_2 m_2}^{L_2 M_2} C_{L_2 M_2,\ell_3 m_3}^{L_3 M_3} \ldots C_{L_{\nu-1} M_{\nu-1},\ell_\nu m_\nu}^{L_\nu M_\nu} \tag{3.57}$$

[13] If we were to model invariant features, we could omit the Clebsch-Gordan coefficients.

that ensures our symmetrized product basis is equivariant to the output rotation order L. The index η_ν enumerates the number of possible combinations of input rotation orders $\ell_1, \dots \ell_\nu$ that yield the desired output rotation order $L = (L_2, \dots L_\nu)$. We see this by noting that our values $L_2, \dots, L_\nu$ are restricted to $|\ell_1 - \ell_2| \leq L_2 \leq \ell_1 + \ell_2$ and for all $i \geq 3$, $|L_{i-1} - \ell_1| \leq L_i \leq L_{i-1} + l_i$.

We note that a self-interaction part with weight matrix $W_{cc'}^{(t,\ell_\xi)}$ to mix features $\mathbf{A}_i^{(t)}$ of different channels c' is also included in equation (3.56).

The fully symmetrized basis can now be used to construct the many-body message for each atom by a linear combination of symmetrized basis features with different body-orders:

$$m_{icM}^{(L,t)} = \sum_{\nu,\eta_\nu} W_{c\eta_\nu}^{(L,t)}(\boldsymbol{\theta}_i)\mathbf{B}_{i\eta_\nu c}^{(L,t)} \tag{3.58}$$

Here $W_{c\eta_\nu}^{(L,t)}$ is a learnable weight matrix that embeds the chemical information of the central atom i into the network through a linear operation similar to (3.44).

With those messages, we can then define the update step for our message-passing scheme. Similar to Nequip, we have a residual connection to node features from the previous layer combined with a linear-combination of a feature mixed aggregated message (3.58):

$$h_{icM}^{(L,t+1)} = f_{\text{UPD}}(\mathbf{h}_i^{(L,t)}, \mathbf{m}_i^{(L,t)}) = \sum_{\tilde{c}} W_{cc'}^{(L,t)} m_{icM}^{(L,t)} + \sum_{\tilde{c}} W_{cc'}^{(L,t)}(\boldsymbol{\theta}_i) h_{i\tilde{c}M}^{(L,t)} \tag{3.59}$$

Each feature term in the expansion (3.53) can be specified implicitly through the body order chosen in the linear combination of $\mathbf{B}$ in (3.58).

The output of predicting the total energy consists of using the invariant features $h_{icm=0}^{(L=0,t)}$ to compute the local energy E_i via a hierarchical decomposition by message-passing iteration. For T such iterations we obtain

$$\hat{E} = \sum_i \left[E_i^{(0)} + \sum_{t=1}^{T-1} \sum_c W_c^{(t)} h_{ic0}^{(0,t)} + \text{MLP}\left(\{h_{ic0}^{(0,T)}\}_c \right) \right] \tag{3.60}$$

The zeroth term E_i^0 corresponds to a fixed term solely determined by the atomic type while the other layers $T \geq t \geq 1$ use a learnable node-feature transformation to the atomic site energies.

Chapter 4

Results and Discussion

4.1 Motivation and Preliminaries

Although significant work has gone into the investigation of three-dimensional compounds and the training of machine learning force fields, the exploration of the two-dimensional chemical space remains largely untapped. This is in part because most current two-dimensional datasets are substantially smaller, mostly focus on binary compounds and also don't contain enough diversity in stoichiometry to capture the variety in the two-dimensional compound space, making it challenging to train and obtain a universal two-dimensional machine learning force fields.

To address these issues, Wang. et al. [40] constructed a two-dimensional dataset to find compounds using a systematic-symmetry based approach in combination with a machine-learned universal force-field assisted geometry optimization. This framework has led to finding an additional, 6500 binary and ternary two-dimensional stable compounds previously not present in any other two-dimensional database and yielding a dataset consisting of a total of 30 different stoichiometries of the form A_nB_m and $A_nB_mC_k$ covering compounds made up of elements from across the entire periodic table (excluding radioactive materials, At, Tc, Pr, Pm and rare gases He, Ne, Ar, Kr, Xe, Rn but notably including the lanthanides).

After the initial search and a screening of viable compounds, a UFF (based on M3GNet) was employed to pre-relax the structure. This pre-relaxation using UFFs is done for efficiency purposes, since a complete geometry optimization using DFT is laborious. This pre-relaxation has the additional benefit of further removing unstable pre-relaxed structure (in their case, unstable compounds refers to relaxed structures with energy of 600 meV/atom away from the hull) that do not have to be considered in the subsequent complete electronic structure calculation using VASP, thus saving additional time and resources. The framework evidently makes extensive use of machine learning and depends strongly on the quality of the trained universal machine learning force field. However through this cycle of training a UFF, obtaining new training data by applying the UFF

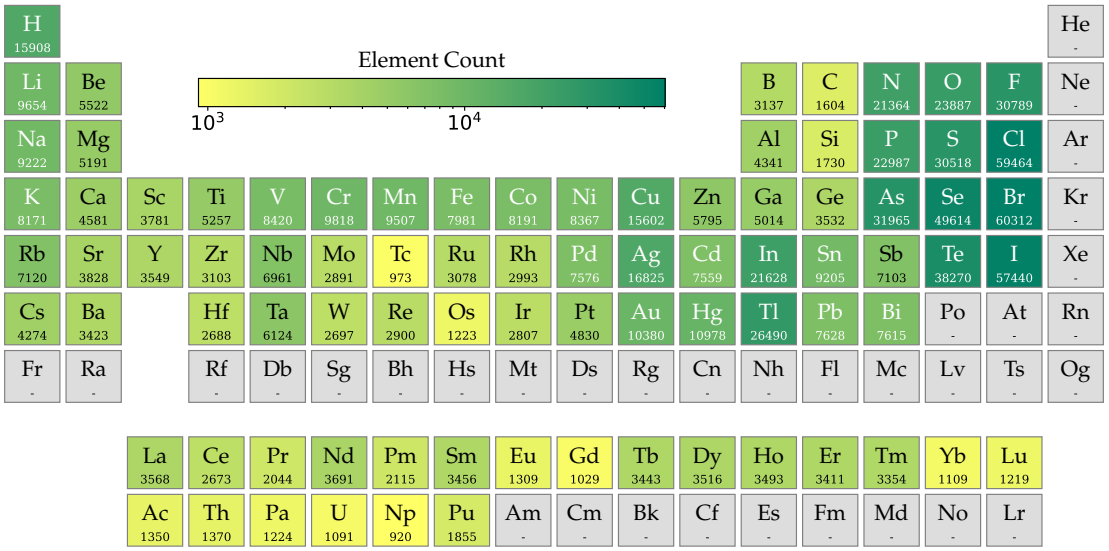


Figure 4.1: Chemical element distribution for the two-dimensional extended compound dataset.

and reusing that new training data to deploy an improved model, the result is that employing this cycle many times over, the ever-improved models will allow for a more and more complete description of all the two-dimensional compounds (going also beyond the restriction of binary and ternary compounds).

In line with this thinking, an extended two-dimensional dataset with chemical element distribution, shown in Figure (4.1) and space group distribution (4.2), was constructed. Concretely, the dataset consists of the DFT geometry optimization data for 88'483 two-dimensional structures. For each step, the respective forces, stresses, and energies were computed using VASP with the PBE functional. The last entry for each run corresponds to the fully relaxed DFT structure with final energy, forces and stress that correspond to the most stable configuration.

An outlier removal was performed on the data by removing structures that contain more than 20 sites as well as structures with isolated atoms, i.e. structures containing atoms without any neighbouring atoms within the cutoff radius. Furthermore, structures possessing extremely large energies, forces and stresses were filtered out. For energies, we chose to remove values outside the range of $E/N > 0$ and $E/N < -20$ (in units of $\mathrm{eV/atom}$). For forces, any values with magnitudes exceeding $20\ \mathrm{eV/Å}$ were discarded. Similarly, stresses with magnitudes surpassing $500\ \mathrm{kbar}$ were also removed from consideration. These cutoffs were deemed sensible and selected based on the full distribution of energies, forces and stresses and yields the following outlier removed distributions

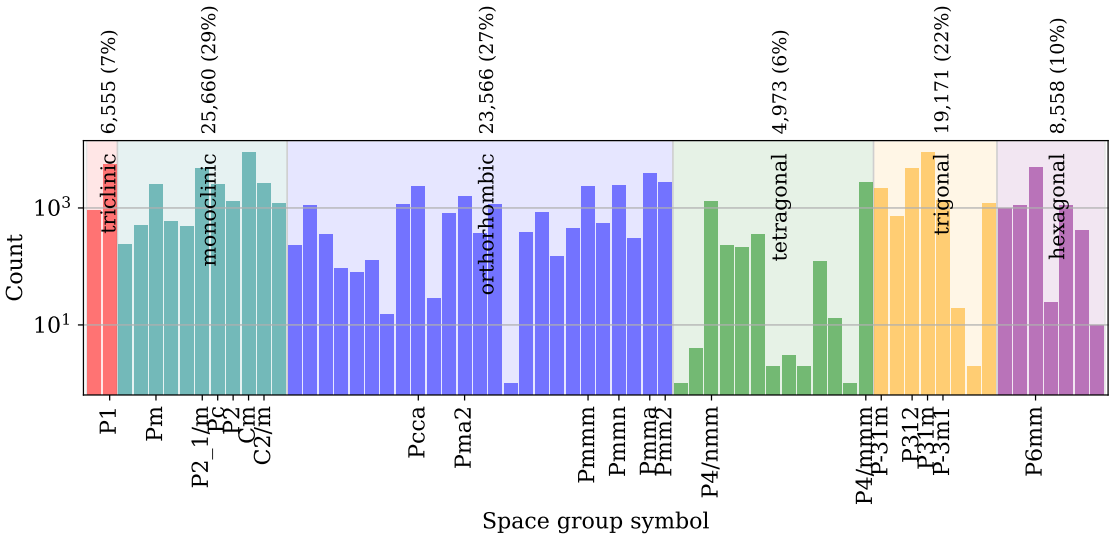


Figure 4.2: Spacegroup distribution of structures contained in the two-dimensional dataset.

shown in Figure (6.2). The stresses

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \tag{4.1}$$

and forces, represented as a multidimensional array $[n_{\text{atoms}} \times [3 \times 3]]$ (each entry corresponding to the individual x, y and z components of the force) for each structure, were flattened to a scalar value by aggregating over all dimensions. Since we are considering two-dimensional structures, all stress components with indices z will be zero. The elimination of outliers from the dataset resulted in a reduction of 3,204 materials, leaving a total of 85,279 two-dimensional materials.

In the original paper by Wang et al. [40], a transfer-learned M3GNet model with base hyperparameters was employed. With this base configuration, they were able to obtain a mean-absolute energy error of $198\ \text{meV/atom}$ and $96\ \text{meV/atom}$ for the median-absolute energy error.

In their case, they only selected the first, last and $N/3$ (where N is the total number of geometry optimizations) step for each material in the geometry optimization as training data.

In this thesis, we will investigate varying the selection of the geometry optimization steps controlled by the cutoff parameter and its effect on the accuracy of our model. Concretely, the cutoff e_{cut} determines by verifying $|E_i/N - E_{i+1}/N| > e_{\text{cut}}$ which steps i in the geometry optimization are retained for our data. Smaller cutoffs e_{cut} evidently results in larger datasets with more geometry optimization steps. Since a significant

portion of the energies turn out to very similar as the geometry optimization converges, we might be concerned that small cutoffs might lead to overfitting of our models to the small energy/force regime. If the cutoff on the other hand is too-large and none of the geometry relaxation steps satisfy the above condition, we resort to the selection of steps $[0, N/2, N]$, which might results in a too small dataset for our networks to effectively train on. To get an estimate of the impact of the cutoff on the dataset sizes, we consider the outlier-removed dataset with a 90:5:5 split. For a cutoff value of 1000000, the dataset contains (230185,12812,12838) geometry optimization steps for training, validation and testing respectively. Conversely, for $e_{\text{cut}} = 0.02$ we obtain (432997, 24337, 23596) data points, essentially retaining double the number of steps in the geometry optimization.

In a first step, we conduct a search for the most effective combinations of hyperparameters and cutoff values for both M3GNet and Mace models. Although Mace has been designed with molecules in mind, it has very recently been applied to material specific problems in the work of Kovacs et al. [47]. However, their evaluation was limited to assessing the direct model predictions errors across two material databases, a metric that does not necessarily provide a direct measure of performance in applications such as for geometry optimizations. Our focus will lie on the latter, as geometry relaxation during high-throughput studies are the main application of universal force-fields so far.

To relax our structures using the machine learning force field, we utilize the Fast Inertial Relaxation Engine (FIRE) algorithm [6]. The relaxation process consist of 1000 iteration steps, with a convergence criterion set to $f = 0.1$. The convergence criterion dictates the required energy difference between two consecutive steps. To consider the relaxation successful, convergence must be achieved at any stage within the 10000 iteration steps. Once convergence is attained, the algorithm proceeds to the next structure and this is repeated until all structures are fully relaxed.

The outputs of the relaxation are both a structure and an energy. An accurate measure for a successful ML force field employed for geometry optimization will thus have to include both an energy and area error comparing the DFT and ML-relaxed structures. Defining an energy error is straightforward, since our energies are already scalar values that can easily be compared. In contrast for condensing a two-dimensional structure to a scalar value, as many viable options exist. In this thesis, we opted to compute the area of a structure computed using the estimated lattice parameters $\mathbf{a}$ and $\mathbf{b}$ (corresponding to the non-periodic components of the material) via:

$$\text{Area} = \|\mathbf{a} \times \mathbf{b}\|/\text{atoms} \tag{4.2}$$

Taking the cross-products of $\mathbf{a}$ and $\mathbf{b}$ yields a vector that is perpendicular to both, representing the area they cover when we calculate its magnitude by taking the norm. This quantity has the units of $\text{Å}^2/\text{atom}$.

It is important to note that the different architectures have different unit conventions. In the case of M3GNet, energies are expressed in electron volts eV, forces in units of $\text{eV}/\text{Å}$ (with N the number of atom in each structure) and stresses in units of gigapascals (GPa). Stresses in VASP are typically expressed in kilobars (kbar) and need to be adjusted to be

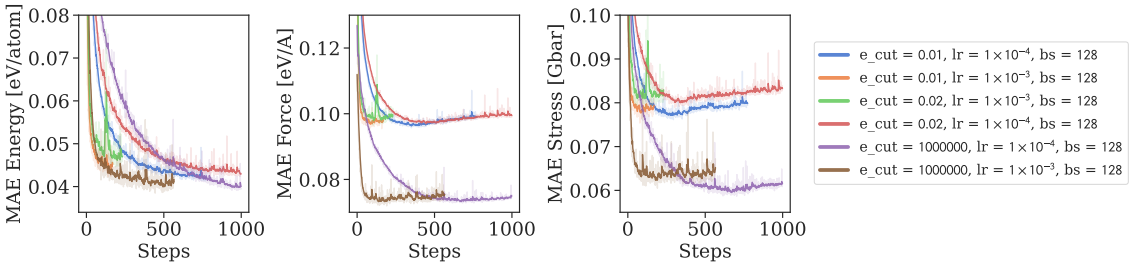


Figure 4.3: The evolution of validation mean-absolute error (MAE) for the energies, forces and stress as a function of steps for M3GNet on the two-dimensional dataset. Cutoff (`ecut`) and learning rates (`lr`) are varied while maintaining a fixed batch size.

compatible with M3GNet. To achieve this, the stresses are multiplied by -0.1 to convert the units to gigabars (Gbar). The negative sign is necessary because M3GNet follows a convention where compressive stresses are considered positive.

On the other hand, both Mace and Nequip employ stresses measured in units of electrons volts per cubic angstrom ($\mathrm{eV/Å^3}$) instead of kbar. To convert VASP stress units of kbar to $\mathrm{eV/Å^3}$, we multiply by -0.1 to obtain the values in gigabars (Gbar) followed by a division of ~ 160.22.

Furthermore, our force field training of Mace and M3GNet involving the two-dimensional structures employs the loss function defined in equation (3.4). This loss function optimizes for energies, forces and stresses. The M3GNet and Mace package are additionally modified to remove any z-stress components (determined to be the non-periodic direction) obtained and used in this loss function. This adjustment ensures that there are no unintended gradients along the z directions that could potentially impact our predictions in undesirable ways.

4.2 UFF for two-dimensional structures

4.2.1 M3GNet

We start by performing a sequential hyperparameter optimization for M3GNet models trained on the extended two-dimensional dataset of varying cutoffs $e_{\text{cut}} = 0.01, 0.02, 1000000$. For each of these cutoffs two models of learning rate 1×10^{-3} and 1×10^{-4} with fixed batch size 128 were trained using a 90:5:5: dataset split.

Figure (4.3) shows the training curves for the validation mean-absolute error of energy, forces and stresses obtained from direct model predictions. We find that across all energies, forces and stresses the models using the cutoff $e_{\text{cut}} = 1000000$ perform best. When we consider shorter training times, $\text{lr} = 1 \times 10^{-3}$ is favored with a validation MAE of $38.47\ \mathrm{meV/atom}$, $71.93\ \mathrm{meV/Å}$ and 60.5×10^{-3} GPa for energies, forces and stresses respectively. However, at longer time scales model $\text{lr} = 1 \times 10^{-4}$ with $\text{bs} = 128$ achieves similar performance with MAEs of $39.14\ \mathrm{meV/atom}$, $72.81\ \mathrm{meV/Å}$ and 58.7×10^{-3} GPa

for energy, forces and stresses respectively. Notably, the forces and stresses are slightly improved over the lr $= 1 \times 10^{-3}$ model at the expense of a minor loss in energy accuracy.

The performance of these two models with learning rate lr $= 1 \times 10^{-4}$ and lr $= 1 \times 10^{-3}$ on the validation set for $e_{\text{cut}} = 10000000$ dataset is then compared to models with the same learning rate but of of different batch sizes 256 and 512. The resulting training curves can be seen in the Figure (4.5) from which we can conclude that for direct predictions of energy, forces and stress the models with batch size 128 and lr $= 1 \times 10^{-4}$, lr $= 1 \times 10^{-3}$ and dataset with $e_{\text{cut}} = 1000000$ achieve the best performance.

As postulated in the beginning of this chapter, we suspect that the inferior performance of models trained on datasets with smaller cutoffs can be traced-back to over-fitting to the low energy sector. This issue is potentially exacerbated by the fact that for smaller cutoffs, certain elements have more geometry optimization steps than others, as depicted in Figure (6.3), which is further amplified by the general class in-balance shown in Figure (4.1).

In order to gauge the effectiveness of the models `lr1e-4bs128` and `lr1e-3bs128` for geometry optimizations, we perform a relaxation of the 4279 test and 4279 validation structures for the $e_{\text{cut}} = 1000000$ dataset using these force fields and cross-reference these with the relaxed DFT structures. We also consider the models with cutoff 0.01 and 0.02. This is because the direct model evaluation performed above cannot give a clear indication which of these models performs best at performing structural relaxations, although we postulate that a small general prediction error is positively correlated with a small relaxation error. The energy and area mean-absolute error for these relaxations can be found listed in Table (4.1). All the models listed have a convergence rate of 97.8% and above. We observe again that the models with $e_{\text{cut}} = 1000000$, particularly model `lr1e-4bs128` displays superior energy performance in both for the validation and test set. The MAE for the energies and area are 76.81 meV/atom and 78.95×10^{-2} Å^2/atom respectively, while the median is given by 33.88 meV/atom and 15.87×10^{-2} Å^2/atom. Our model thus yields more than a two-fold improvement to the model from Wang et al. [40], where a mean-absolute error of 198 meV/atom and median of 98 meV/atom for the energies was achieved. That there is good agreement between the model predictions and the DFT ground truth can also be seen from the Figure (4.4), which show high-linearity and a large R^2 value in the energies.

Regarding the area errors, the `lr1e-3bs128` models produce slightly improved results compared to the `lr1e-4bs128` models, a pattern that is consistent across both the test and validation sets, as well as across different cutoffs. It is also somewhat unexpected to find that the errors for energy and area are worse for the validation data compared to the test data. This discrepancy could just be an unintended consequence of the data-splitting processed, in which keys are randomly assigned for the validation and test sets. One possible approach to mitigate this effect would be for example through k-fold cross-validation.

Lastly, we wish to investigate the accuracy of the models in its relaxation predictions across all the different chemical elements contained in the dataset, to gauge their usefulness as

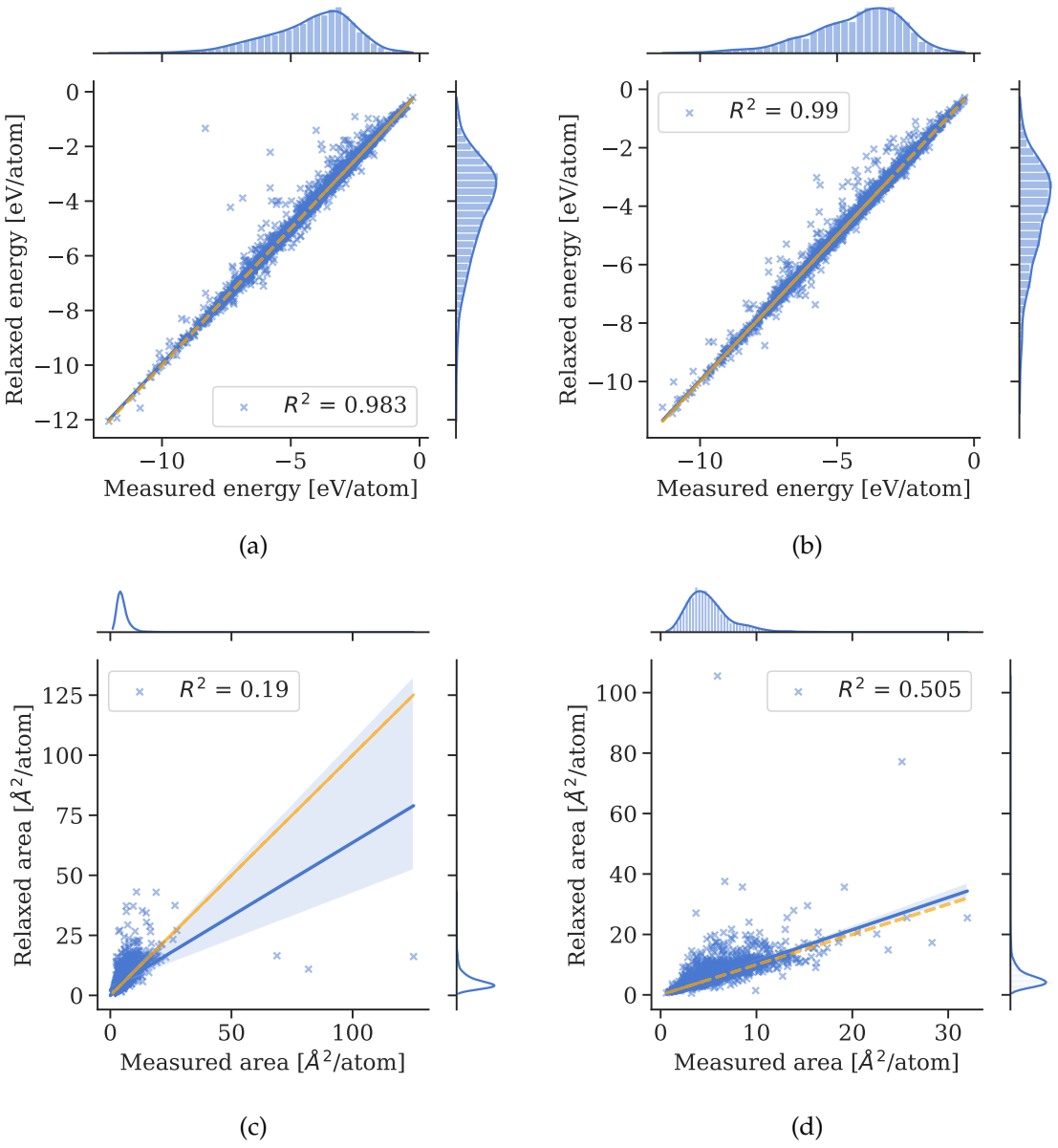


Figure 4.4: Line-Deviation plots for energy and area ofM3GNet with lr $= 1 \times 10^{-4}$ and bs $= 128$ on the validation set for (a) and (c) and test set for (b) and (d) respectively. These models were trained with a 90:5:5 split on the two-dimensional dataset with $e_{\text{cut}} = 1000000$.

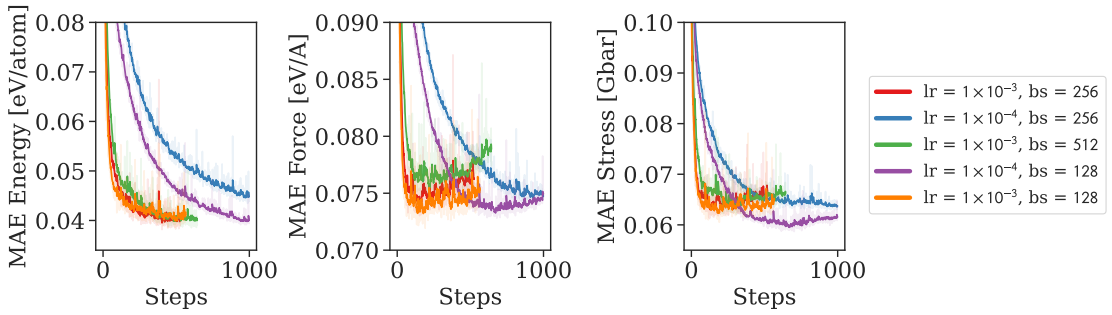


Figure 4.5: Validation MAE evolution with epochs for M3GNet on two-dimensional dataset with $e_{\text{cut}} = 1000000$ for varying batch sizes.

a universal force field. For lr1e-4bs128 with $e_{\text{cut}} = 1000000$ the energy and area MAE distributions over the periodic system can be found in Figure (6.7a) and Figure (6.7b) for the validation data and Figure (4.7a) and Figure (4.7a) for the test data.

In terms of the energies, our findings align with previous results that documented the first-row anomaly [13, 27, 50]. This pattern shows elements boron, carbon, nitrogen, oxygen and fluorine consistently perform below average when compared to other elements[1]. Additionally, we find that metals, particularly transition metals like niobium and tungsten, along with the alkali earth metal barium to possess a higher MAE than other groups across both the validation and test sets. These large errors for the metals can most likely be explained by DFT inadequate representation of magnetic interactions, which consequently complicates model learning. Previous studies, such as those conducted by Schmidt. et al. [27], found similarly large errors for certain metals. In their case, the main culprits were chromium, manganese and iron. However, in our study, these elements appear to yield reasonable results for both the validation and test sets.

Furthermore we observe large fluctuations of the errors across validation and test sets for certain lanthanide and actinide metals (elements such as lanthanum for example have a energy validation MAE of $92.2\ \text{meV/atom}$ while for the test set we obtain $203.4\ \text{meV/atom}$). We suspect the reason is twofold. Firstly, there is comparably fewer data for these groups (we are dealing with an unbalanced dataset), as evident from Figure (4.1). Secondly, the pseudopotentials currently available in DFT frameworks such as VASP for these elements frequently introduce numerical complications that can hinder the convergence of the geometry optimizations [50].

As for the area errors, we observe a similar instability pattern for the lanthanides and actinides. Interestingly, however, no first row anomaly in either the validation and test set can be observed. Instead, the alkali and alkali-earth metals yield large area errors that are present in both the validation and test sets. Particularly noticeable is the disproportionally large validation area MAE of $474.5 \times 10^{-2}\ \text{Å}^2/\text{atom}$ for caesium, which lies well above the second-worst MAE of Rubidium with $269.1 \times 10^{-2}\ \text{Å}^2/\text{atom}$. It is somewhat surprising to

[1]We note that fluorine in our case seems to be performing reasonable well.

Set	Model		Cutoff = 0.01	Cutoff = 0.02	Cutoff = 1000000
Val	lr1e-3bs128	Energy	87.43	88.33	**87.31**
		Area	**93.37**	94.95	93.73
	lr1e-4bs128	Energy	90.62	90.14	**84.38**
		Area	94.41	96.86	**94.75**
Test	lr1e-3bs128	Energy	84.23	83.73	**82.68**
		Area	76.84	77.72	**76.42**
	lr1e-4bs128	Energy	82.31	84.09	**76.81**
		Area	80.17	78.99	**78.95**

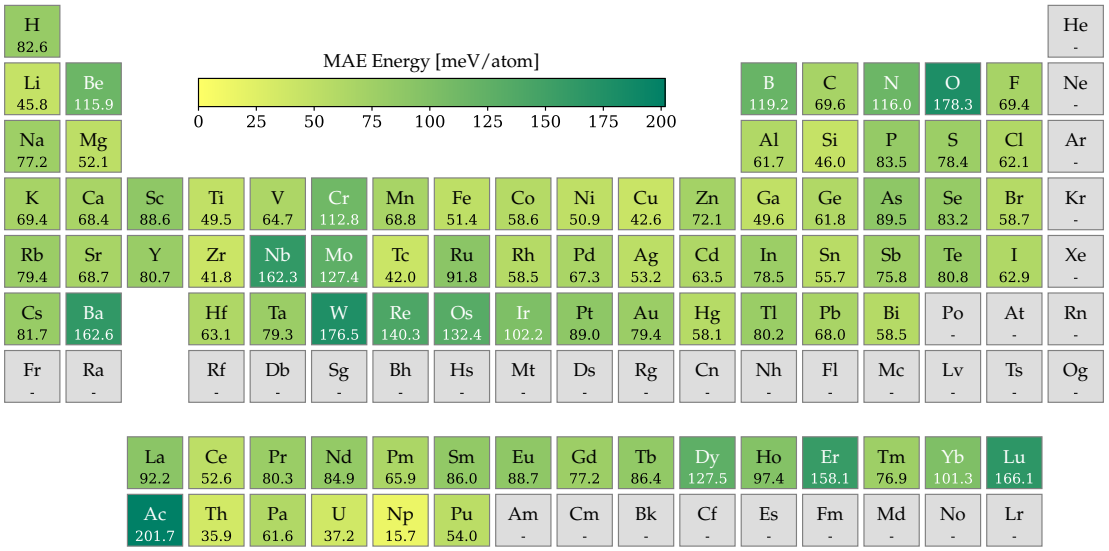
Table 4.1: Mean-absolute error comparison for M3GNET relaxation results on the two-dimensional structures on test and validation structures for models lr1e-3bs128, lr1e-4bs128 on cutoffs $1000000, 0.01, 0.02$. The relaxation were performed using $n_{\text{steps}} = 1000$ and $f_{\text{max}} = 0.1$. All energies are presented in units of [meV/atom] and areas in units of [Å^2/atom $\times$ 100]. The area and energy predictions highlighted in bold represent the best values achieved by each model across various cutoffs, while the second best options are underlined. The color red indicates the best area prediction for a given cutoff, while blue signifies the best energy prediction.

find large errors in the first and second row of the periodic table only. Generally, we would anticipate that larger elements across the entire periodic table would produce greater area errors.

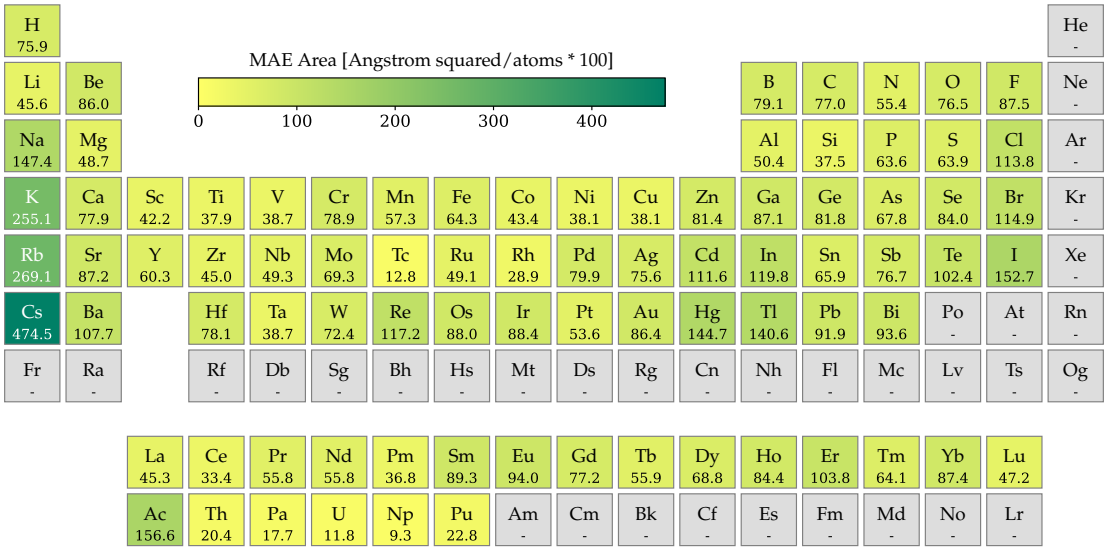
4.2.2 MACE

Adopting the same procedure and analysis as for M3GNET, we next employ MACE to train a universal force field. However, at this stage, we forego the analysis of various cutoffs and concentrate solely on the results with $e_{\text{cut}} = 1000000$. For this particular cutoff, six different MACE force fields of varying batch sizes 64, 128 and 192 and learning rates $0.01, 0.03$ and 0.003 were trained, and their training curve can be seen in Figure (6.9). The models `lr0.001bs64` and `lr0.003bs128` performed best on the validation data. For `lr0.01bs64` we obtain direct MAE prediction errors of 33.78 meV/atom, 76.46 meV/Å and 57.81×10^{-3} eV/Å^3 for energies, forces and stresses, respectively. Similarly, for `lr0.003bs128`, the direct MAE prediction errors were 35.27 meV/atom for energies,73.89 meV/Å for forces and 57.8×10^{-3} eV/Å^3 for stresses.

The results for the subsequent relaxation of the test data using our two trained MACE models can be found in the Table (4.2). We find that the model `lr0.003bs128` outperforms `lr0.01bs64` when used as a force field to relax structures. The distribution of the energy and area errors across the periodic system for this better performing model can be found in the Figure (4.8). We again observe the first-row anomaly in our energies, larger area errors for alkali and alkali earth metals and larger errors for lanthanides and actinides in both the area and energies.

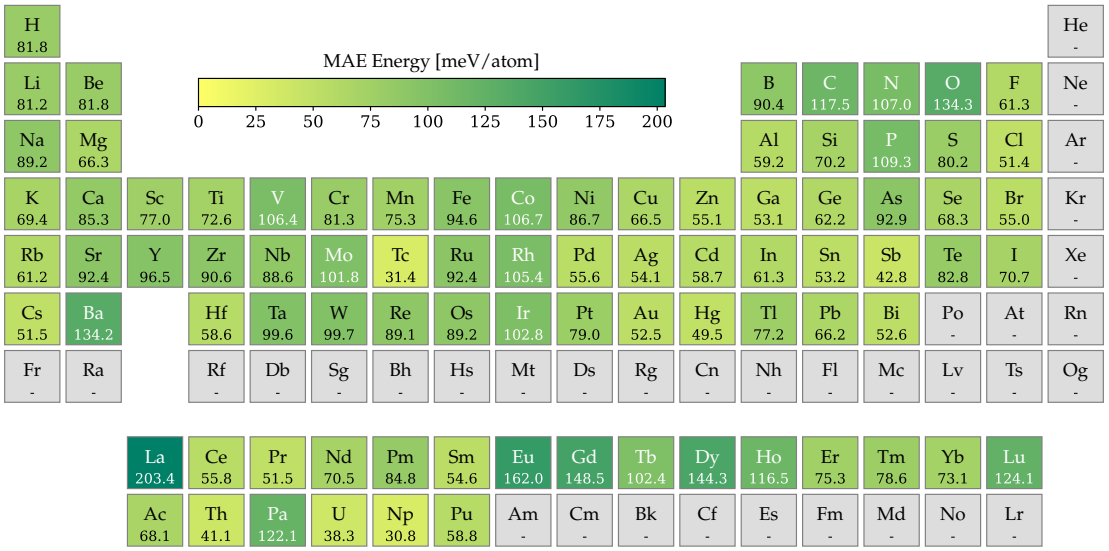


(a)

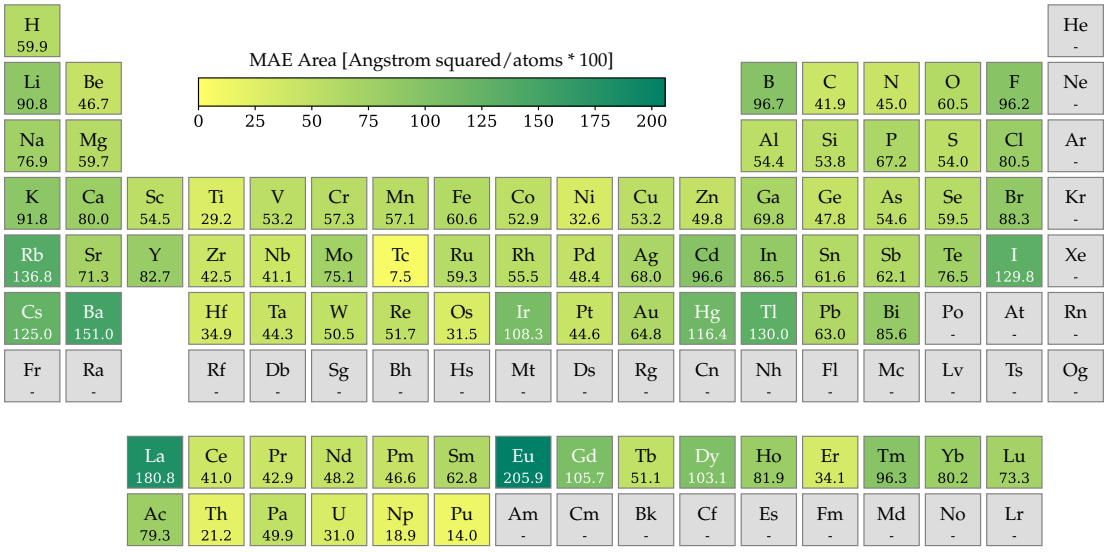


(b)

Figure 4.6: Distribution of the mean absolute error (MAE) in the *validation data* for (a) energy and (b) area, categorized by chemical elements in the periodic system. This is using the **M3GNet** model with a learning rate $\mathrm{lr} = 1 \times 10^{-4}$ and a batch size of $\mathrm{bs} = 128$.



(a)



(b)

Figure 4.7: Distribution of the mean absolute error (MAE) in the *test data* for (a) energy and (b) area, categorized by chemical elements in the periodic system. This is using the **M3GNET** model with a learning rate $\mathrm{lr} = 1 \times 10^{-4}$ and a batch size of $\mathrm{bs} = 128$.

Model		Cutoff = 1000000
lr0.003bs128	Energy	121.65
	Area	68.60
lr0.01bs64	Energy	149.33
	Area	69.59

Table 4.2: MACE relaxation results for the two-dimensional structures on test structures with $n_{\text{steps}} = 1000$ and $f_{\text{max}} = 0.1$ and cutoff=1000000. The energies are given in units of meV/atom and the area in $[\text{Å}^2/\text{atom} \times 100]$.

Although the energy MAE for both models is worse than that of M3GNet (compare to Table (4.1)), it is noteworthy that the area error is considerably small in comparison to M3GNET. This is also evident from the larger area R^2 coefficient the MACE models obtain for the line plot showing the deviations from the DFT ground truths in Figure (6.8) when compared to Figure (4.4).

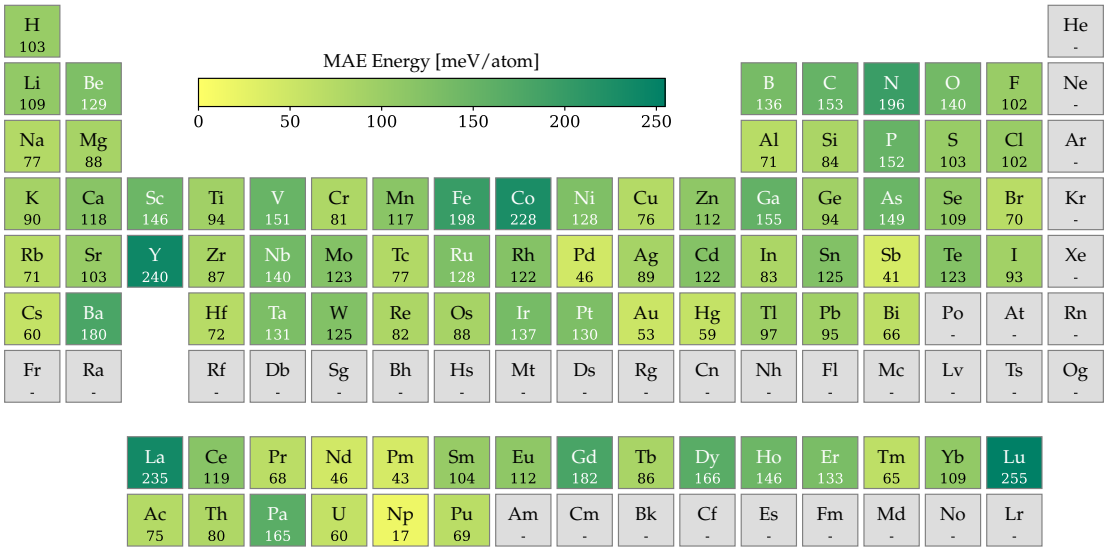
For all trained models, we chose the base MACE configuration with a max equivariance of $L = 2$. We set the number of channels of node feature embedding to be 192 and use the standard number of hidden irreducible representations of 32. Unlike M3GNET, MACE models can employ an adaptive learning scheduler for the weights of the different losses for stresses, forces and energies, referred to as stochastic weight averaging (SWA). At the hyperarameter `start-swa`=50 step[2], the weight of the energy in the loss is increased by `swa-energy-weight` = 10 to yield a better energy prediction at the end of the training procedure. The parameter `swa-lr` = 1×10^{-3} sets the learning rate after the `start-swa` steps. These are only a few of the hyperparameters available to MACE, and we suspect that further work, consisting in a broader exploration of its hyperparameters, could be fruitful in identifying parameters that yield area and energy errors of more comparable magnitude to M3GNET as well as minimise the currently large discrepancy between the two errors.

4.3 Alloy specific force-field

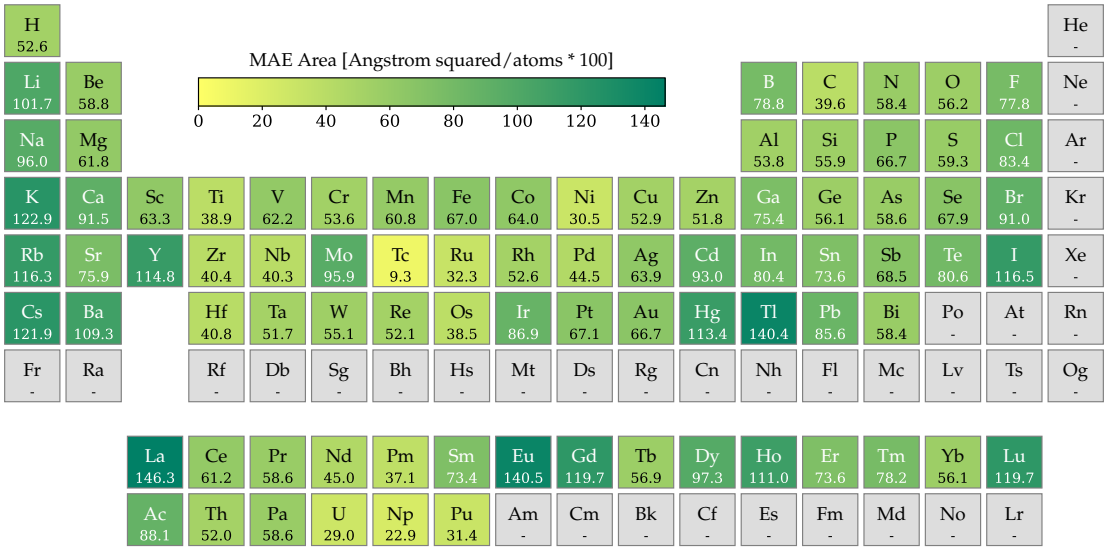
After carrying out the evaluation of the universal machine learning force field in the last section, we aim to determine the potential benefits of transfer learning. Specifically, we will assess whether transfer learning from the two-dimensional universal MACE and M3GNET models to material-specific force fields proves advantageous. To investigate this claim, we will employ a two-dimensional alloy dataset. To begin, we present a brief overview and statistical analysis of this dataset.

This dataset consists of DFT geometry optimizations for a variety of alloyed compounds, namely: iridium ruthenium disulfide ($IrRuS_2$), molybdenum tantalum disulfide ($MoTaS_2$),

[2] At 100 epochs, marked by `start-swa` = 50, a notable change can be observed in the training figures (6.9). The validation energy error exhibits a sudden decrease beyond this point, while the force errors show an increase.



(a)



(b)

Figure 4.8: Distribution of the mean absolute error (MAE) in the *test data* for the two-dimensional dataset for (a) energy and (b) area, categorized by chemical elements in the periodic system. This is using the **MACE** model with a learning rate $\mathrm{lr} = 0.003$ and a batch size of $\mathrm{bs} = 128$.

Model	Energy [meV]	Area [$\text{Å}^2/\text{atom} \times 100$]
Normal, lr = 1e-3, bs=128	**9.82**	1.10
Transfer, lr = 1e-3, bs=128	10.88	**1.05**

Table 4.3: Alloy trained on all systems for M3GNet with Cutoff = 0.01. The transfer model loaded here is the overall best performing model on the two-dimensional data with lr=1e-4, bs=128.

molybdenum tungsten disulfide ($MoWS_2$), niobium molybdenum disulfide ($NbMoS_2$), titanium-iron disulfide ($TiFeS_2$), titanium niobium disulfides ($TiNbS_2$), titanium tantalum disulfide ($TiTaS_2$), titanium vanadium disulfide ($TiVS_2$) and zirconium tantalum disulfide ($ZrTaS_2$). All these alloys fall into the category of transition metal dichalcogenides (TMDs), a class of two-dimensional materials. These materials have gained significant interest due to their potential applications in energy storage [12].

The general formula for TMDs is MX_2 where M represents a transition metal and X is a chalcogen such as sulfur. We consider the subset of transition metal disulfides. For our transition metal disulfides, the composition ratio x yields a ratio between the two sulfides, for example for $MoTaS_2$ we will have $Mo(x)Ta(1-x)S_2$.

The chemical element distribution can be found in Figure (6.10) and the distribution of the space group symmetries in Figure (6.11). We observe that with respect to chemical elements the dataset is unbalanced with zirconium and vanadium appearing the least. The distribution of the total energy per atom of each system, combined total energy, stresses and forces can be found in Figure (6.13a), Figure (6.13b), Figure (6.12a) and Figure (6.12b) respectively.

4.3.1 M3GNet

The training of M3Gnet on separate alloy systems included the use of models with a learning rate lr $= 1 \times 10^{-3}$ and bs $= 128$. The cutoff value was chosen to be 0.01 and the data was split into a ratio of 90:5:5 for training, validation and testing respectively. This training was conducted twice, once using traditional learning and once using transfer learning from the two-dimensional dataset. The MAE results for the relaxation using the test set can be found in the Table (4.4). The line and error plots are shown in the Figures (6.14), (6.15), (6.16), (6.17) in the appendix. As expected, the individual system errors are much lower compared to the MAE for the energy and area of the universal force fields from the last section. Surprisingly, however, the transfer model performs worse for the energies on average than the model with normal training. Exceptions include alloys that contain molybdenum, i.e $MoTaS_2$, $MoWS_2$, $NbMoS_2$. In particular, the area errors are substantially worse for all alloy systems, except $NbMoS_2$. We suspect that by lowering the learning rate we might mitigate so-called catastrophic forgetting [25] in which the prior learned information from the model is 'forgotten' when trained on new data. It could however also indicate unstable training.

Alloy System		Cutoff = 0.01	
		lr=1e-3, normal	lr=1e-3, transfer
$IrRuS_2$	Energy	**15.98**	53.13
	Area	**1.47**	26.08
$MoTaS_2$	Energy	2.46	**1.72**
	Area	**0.45**	0.49
$MoWS_2$	Energy	0.51	**0.36**
	Area	**0.51**	0.948
$NbMoS_2$	Energy	4.90	**1.26**
	Area	**0.31**	0.39
$TiNbS_2$	Energy	**16.58**	28.84
	Area	**4.61**	52.29
$TiTaS_2$	Energy	**6.98**	33.90
	Area	**26.85**	41.17
$TiVS_2$	Energy	**19.08**	28.83
	Area	**1.44**	45.07
$ZrTaS_2$	Energy	24.73	**14.96**
	Area	**1.69**	8.69

Table 4.4: Relaxation for M3GNet for datasplit 9055 with batch size 128 on the two-dimensional alloy dataset. The transfer model used here is the UFF trained on all two-dimensional data with lr=1e-4 and bs=128. The relaxations were performed using the *test set*. The energies are given in units of $\mathrm{meV/atom}$ and the area in units of $\mathrm{\AA}^2/\mathrm{atom} \times 100$

Furthermore, we combined all alloy systems into a combined dataset to train a specific TMD force field again using normal and transfer learning. The relaxation results can be found in Table (4.3) and visualized in Figure (6.18). Similarly to before, we suspect we can attribute the slightly worse performance of the transfer model compared to the normal model, to the use of a too large learning rate.

4.3.2 MACE

Using MACE we perform a more exhaustive evaluation of the alloy dataset, where now in addition to a 90:5:5 dataset split, we also consider an 80:10:10 ratio with cutoffs 0.01 and, 1000000. This will give us better holdout estimates, especially in settings where the data is scarce. The results, shown in Table (4.6) for the 80:10:10 datasplit and Table (4.5) for the 90:5:5 ratio, were obtained by a structure relaxation using the test data. Contrary to M3GNet, these results indicate that transfer learning yields at least similar if not improved accuracy. Furthermore, the increased geometry optimization steps in the Cutoff=0.001 dataset compared to the Cutoff=1000000 dataset, seems to benefit performance for the transfer models across all alloy systems. We suspect that this in part because MACE is a more expressive model compared to M3GNet that benefits positively from increased data without the unwanted side effects of overfitting. However, evident from a comparison to

the M3Gnet results in the Table (4.4) it seems MACE fails to obtain relaxation results for multiple alloy systems such as $IrRuS_2$ and $NbMoS_2$, indicating a more unstable training. Another factor that could contribute to this failure is the dataset itself. In the instance of $IrRuS_2$ with the cutoff = 0.001 dataset for both splits, the inability to relax the system is observed in both the normal and transfer learned models, hinting at some underlying problem in the dataset.

Additionally, the increased GPU memory requirements of MACE limited the ability to increase the batch size of the models beyond the lower double digits. We suspect that this could be an additional reason for the instability and sometimes inferior accuracy for certain alloys compared to M3GNet.

Despite these drawbacks, the successful relaxations yielded models with much improved energies compared to M3GNet, with the sole exception of $ZrTaS_2$ as can be seen from the Table (4.7). This outcome differs significantly from our observations for the universal force fields, where MACE produced inferior energy outputs but achieved better area error results.

Alloy System		Cutoff = 0.001		Cutoff = 100000	
		lr=0.0001, transfer	lr=0.003, normal	lr=0.0001, transfer	lr=0.003, normal
$IrRuS_2$	Energy	-	-	**12.95**	17.74
	Area	-	-	**8.11**	24.74
$MoTaS_2$	Energy	1.97	**1.75**	2.80	2.98
	Area	**2.47**	4.24	4.64	5.59
$MoWS_2$	Energy	**0.49**	1.73	1.96	5.00
	Area	**2.46**	5.03	6.36	6.27
$NbMoS_2$	Energy	**1.84**	-	3.62	-
	Area	**2.10**	-	6.18	-
$TiNbS_2$	Energy	**2.66**	3.04	2.78	89.3
	Area	4.20	5.52	5.23	**3.75**
$TiTaS_2$	Energy	**2.22**	13.84	3.02	2.79
	Area	**3.42**	6.15	5.89	7.46
$TiVS_2$	Energy	4.16	**2.62**	-	-
	Area	12.46	**8.41**	-	-
$ZrTaS_2$	Energy	49.5	61.92	46.39	**44.98**
	Area	**27.46**	32.22	28.04	29.04

Table 4.5: Relaxation MAE results for MACE with batch size 16 using the test dataset obtained from a 90:5:5 dataset split. Energies are given in units of $\mathrm{meV/atom}$ with area errors in units of $[\mathrm{Å}^2/\mathrm{atom} \times 100]$.

4.4 Formation Energy training

Lastly, we will explore direct formation energy training from our models trained on the alloy dataset. As elaborated on in section (2.4), the formation energy plays a pivotal role in understanding and predicting the properties and stability of various compounds by quantifying the energy required for the creation of a compound from its constituent elements. Traditionally, CE models are fitted to DFT data to model alloys. These are

Alloy System		Cutoff = 0.001		Cutoff = 1000000	
		lr=0.0001, transfer	lr=0.003, normal	lr=0.0001, transfer	lr=0.003, normal
$IrRuS_2$	Energy	-	-	13.95	**2.48**
	Area	-	-	95.52	**26.22**
$MoTaS_2$	Energy	**2.14**	-	2.31	2.20
	Area	**2.77**	-	4.15	4.84
$MoWS_2$	Energy	**1.27**	1.85	1.96	5.7
	Area	24.59	**4.35**	6.56	6.77
$NbMoS_2$	Energy	3.11	-	2.29	**1.98**
	Area	5.02	-	5.85	**4.90**
$TiNbS_2$	Energy	2.91	6.74	**2.23**	30.68
	Area	4.95	7.50	3.40	**1.85**
$TiTaS_2$	Energy	2.78	3.60	**2.58**	3.96
	Area	**4.66**	4.80	6.38	8.34
$TiVS_2$	Energy	3.39	4.44	**1.63**	2.66
	Area	11.61	**10.77**	11.00	12.82
$ZrTaS_2$	Energy	**2.43**	7.02	-	27.99
	Area	**9.84**	10.12	-	20.41

Table 4.6: Relaxation MAE results for MACE with batch size 16 using the test dataset obtained from a 80:10:10 dataset split.

required to treat larger unit cells needed to model smaller percentages when alloying, to treat disorder and temperature in, e.g., Monte Carlo simulations.

In this context, we will explore whether machine learning force fields can serve as a viable replacement for Cluster Expansion in formation energy predictions. The formation energy dataset, originating from the work of Silva. et al. [37, 38], consists of Cluster Expansion for comparison and DFT formation energy predictions for training obtained using CASM (Cluster Approach to Statistical Mechanics) [52] and VASP [3, 4] respectively, of the transition metal disulfides $MoTaS_2$, $MoWS_2$, $NbMoS_2$ and $TiTaS_2$ with the same structures as in the two-dimensional alloy dataset. We clarify that we have predictions of alloys for various compositions along the tie-line, $A(x)B(_1-x)S_2$ including the pristine end-members AS_2 of $x = 0$ and BS_2 of $x = 1$.

It is important to note that the formation energies are calculated "per substitutional lattice sites", which refers to the number of transition metals in the system. All the prototypes for which we have parameterized a Cluster Expansion possess one transition metal in the unit cell. Therefore, the per atom formation energies amount to one-third of these values.

The distribution of formation energies is illustrated in Figure (4.10). The vertical lines mark $E_F = 0$, indicating the threshold for the formation of stable compounds. We note that all combinations of $Mo(x)W(1-x)S_2$ exhibit negative formation energies, indicating their stability across all alloy configurations included in the dataset. On the contrary, all $NbMoS_2$ alloys possess positive formation energy and thus don't form under normal conditions.

Before we delve into the results obtained from the machine learning force fields, we briefly review the formation energy predictions derived from Cluster Expansion. This

Alloy System		MACE Cut=0.001, lr=0.0001, transfer	M3GNet Cut=0.01, lr=1e-3, normal
$IrRuS_2$	Energy	-	**15.98**
	Area	-	**1.47**
$MoTaS_2$	Energy	**1.97**	2.46
	Area	2.47	**0.45**
$MoWS_2$	Energy	**0.49**	0.51
	Area	2.46	**0.51**
$NbMoS_2$	Energy	**1.84**	4.90
	Area	2.10	**0.31**
$TiNbS_2$	Energy	**2.66**	16.58
	Area	**4.20**	4.61
$TiTaS_2$	Energy	**2.22**	6.98
	Area	**3.42**	26.85
$TiVS_2$	Energy	**4.16**	19.08
	Area	12.46	**1.44**
$ZrTaS_2$	Energy	49.5	**24.73**
	Area	27.46	**1.69**

Table 4.7: Model comparison between best-performing MACE and M3GNET models on the 90:5:5 split test dataset.

will provide us with a benchmark required of our machine learning models in order to outperform Cluster Expansion.

Figure (4.11) presents the relationship between the formation energy prediction obtained from Cluster Expansion and the 'ground truth' predictions from DFT for the four transition metal dichalcogenides available in our dataset. It is evident that Cluster Expansion yields impressive results, with an almost perfect R^2-coefficient for all systems. Some variations do occur, with the best performance achieved by the alloy system $MoWS_2$ with the smallest variance in the formation energy across the dataset. To get a grasp on an overall alloy-specific Cluster Expansion error, the system-specific formation energy datasets were aggregated and yielded the results shown in Figure (4.10b).

To get an accurate measure of comparison for the machine learning models, we have to similarly split this formation energy dataset into training, validation and test sets and calculate the mean-absolute error of the Cluster Expansion for each of these subsets. This is shown in the Table (4.9). We find for individual alloy subsets the errors are quite evenly distributed across the training, validation and test sets. It is important to point out that this data splitting takes place after the Cluster Expansion model has been trained on the whole dataset, thereby not considering a separate holdout for validation and testing. This is evident from $MoTaS_2$, where the test MAE is significantly lower than the training MAE[3]. This approach could possibly skew the comparison with the error rates that we obtain from the machine learning models, and fails to give us a means to quantify the generalization capability of the Cluster Expansion approach. In the future, we aim to produce a more accurate error comparison between the models by properly considering

[3]This scenario is unlikely to arise if only the training set is used to fit the Cluster Expansion model

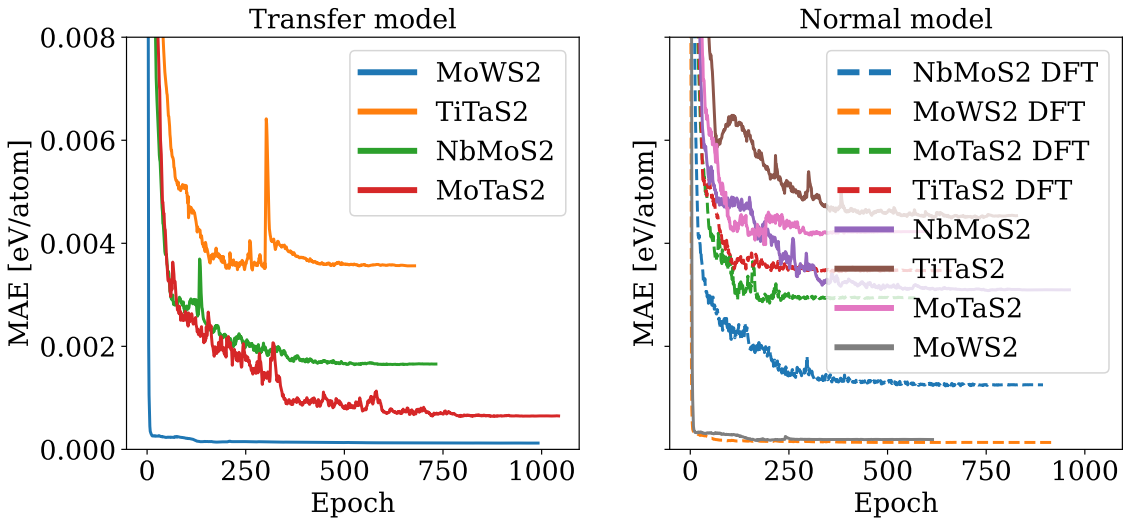


Figure 4.9: Both NEQUIP models aiming at predicting the formation energy validation MAE were trained with the dataset with a cutoff of 1000000 and data split 801010. The structures used for training the NEQUIP model were obtained from (left) MACE transfer model with lr=0.0001 and bs = 16 (right) MACE model trained normally with lr=0.003 and bs=16 from the ML model.

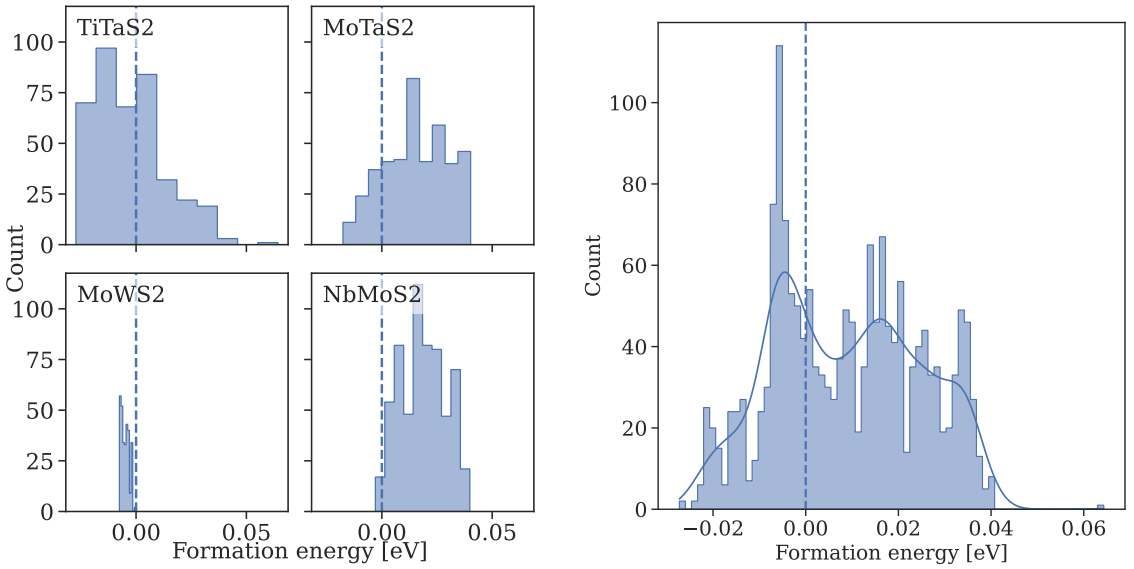
the necessary data separation for the Cluster Expansion models. Nevertheless, these errors can give us a good measure to the lower-bound of the test and validation errors we can obtain for Cluster Expansion.

For the training of formation energy using machine learning force fields, we turn to NEQUIP due to its convenient adaptation of an energy-only training[4]. We will employ a model with a learning rate 0.0075 and a batch size of 16.

We train these NEQUIP models using three distinct set of inputs. The first dataset, that will serve as input to model ML1, is assembled from the formation energy and fully relaxed training structures obtained from the MACE model of learning rate 0.003 and batch size 16 of the two-dimensional alloy dataset. Our second dataset, designed for the ML2 model, is similarly constructed using the relaxed structures obtained from the transfer learned MACE model of learning rate 0.001 and batch size 16. Finally, our third dataset serves as a comparison and involves using the fully relaxed DFT structures. In principle these structures should result in the best formation energy predictions since these correspond to the best relaxed structures.

The training evolution for all three model inputs can be seen in Figure (4.9). The model test and validation predictions for each of these across the four alloys is shown in the Table (4.8) with the Cluster Expansion errors shown for comparison. Inline with our expectation, we find that the transfer model ML2 performs better than the one using the

[4]Attempts by me have been made to adapt the MACE code to such an energy-only training, to no avail however.



(a) Distribution Energy per atom for the individual systems.

(b) Energy distribution for all the systems combined.

Figure 4.10: Formation energy distribution for alloys.

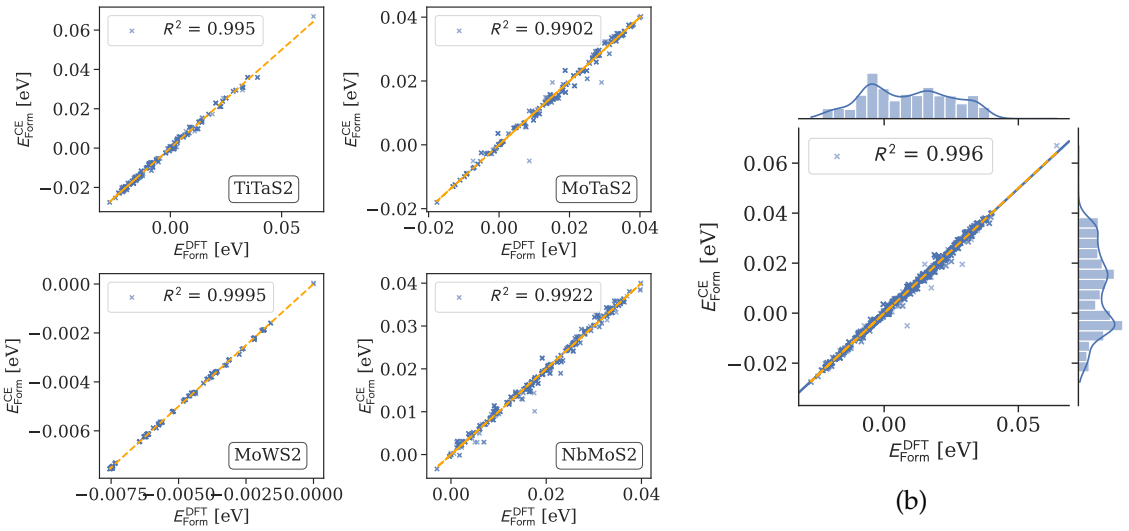
relaxed structures for the normal trained MACE model ML1, except for$TiTaS_2$ in the test set. As anticipated, the ML model utilizing the DFT relaxed structures performs consistently the best. However, it is noteworthy that in specific instances, such as $NbMoS_2$ and MoWS, ML2 has matching errors.

Importantly, our results demonstrate, that ML2 outperforms Cluster Expansion in terms of accuracy for all alloys across both the validation and test sets and the improvements can reach as much as 0.12meV for $MoWS_2$.

4.4.1 Stoichometry models

We also explored the use of stoichometry models such as ROOST [22] and CRABNET [39]. These models disregard the exact spatial arrangement of atoms as model input and instead only use the materials stoichometry, which is the ratio of different elements in a material. However we found that these models do not yield satisfactory results for determining the formation energy of alloys. One obvious inherent limitations of stoichometry models is that they cannot differentiate between alloys that have the same composition but a different overall spatial arrangement of atoms[5]. This would coincide with earlier findings such as [27], where ROOST was trained on perovskite data with poor results.

[5]For example, the same two elements can form different alloy phases with distinct properties depending on the ratio and arrangement of atoms.



(a) (b)

Figure 4.11: Deviation line plot for DFT formation energy and Cluster Expansion formation energy for (a) the individual systems separately and (b) all the systems combined.

Set	Model	$NbMoS_2$	$MoWS_2$	$MoTaS_2$	$TiTaS_2$
Test	Clex	0.161	0.011	0.219	0.284
	ML (relaxed DFT)	**0.067**	**0.006**	**0.065**	**0.208**
	ML 1 (normal)	0.098	0.012	0.149	0.230
	ML 2 (transfer)	**0.067**	**0.006**	0.099	0.259
Val	Clex	0.158	0.010	0.266	0.278
	ML (relaxed DFT)	**0.060**	0.072	0.138	**0.192**
	ML 1 (normal)	0.144	0.010	0.198	0.250
	ML 2 (transfer)	0.084	**0.007**	**0.031**	0.198

Table 4.8: Comparison of Mean Absolute Error (MAE) in formation energy [meV/atom] for both validation and test sets. The comparison includes the results for Cluster Expansion (Clex) and three NEQUIP models, which were trained with a learning rate of 0.0075 and a batch size of 16, for the alloys $NbMoS_2$, $MoWS_2$, $MoTaS_2$, and $TiTaS_2$. The test and validation sets were obtained from a 80:10:10 dataset split. ML1 Model uses the final relaxed structures from the normally trained MACE model with learning rate 0.003 and batch size 16 as input. ML 2 uses the structures from the transfer learned MACE model with lr=0.0001 and bs = 16. Both MACE models were trained with the cutoff of 1000000. ML (relaxed DFT) uses the fully relaxed DFT structures as inputs. The best performing models for the validation and test sets separately for each of the alloys, are denoted in bold, while second-best models are underlined.

Alloy	Dataset	MAE Clex [meV/atom]	RMSE Clex [meV/atom]
$MoTaS_2$	train	0.267	0.494
$MoTaS_2$	val	0.266	0.395
$MoTaS_2$	test	0.219	0.343
$MoWS_2$	train	0.011	0.014
$MoWS_2$	val	0.009	0.013
$MoWS_2$	test	0.011	0.014
$NbMoS_2$	train	0.190	0.307
$NbMoS_2$	val	0.158	0.273
$NbMoS_2$	test	0.161	0.238
$TiTaS_2$	train	0.277	0.355
$TiTaS_2$	val	0.278	0.358
$TiTaS_2$	test	0.284	0.373

Table 4.9: Mean absolute and root-mean squared error for the Cluster Expansion for each of the datasets where the ground truths here are the computed DFT energies. The train, val and test keys correspond to the keys of the two-dimensional alloy dataset of cutoff=1000000 with an 80:10:10 split.

Chapter 5

Conclusion and Outlook

In this thesis we have demonstrated the potential of using machine-learned force fields for various application in material science. Through a systematic exploration of various model architectures and training schemes, it has been shown that these computational techniques offer a viable alternative to traditional method such as Density Functional Theory and Cluster Expansion.

The universal force field training on a two-dimensional dataset showed a notable, more than two-fold improvement in energy predictions compared to a previous model. These UFF models will open the opportunity for more efficient and accurate high-throughput searches of new two-dimensional materials and the creation of a larger dataset spanning the two-dimensional compound space. In fact, the creation of a large dataset of ML predicted structures in the vein of Matterverse (https://matterverse.ai/) for two-dimensional structures using the developed force field was just started by the Marques group in Bochum.

Furthermore, these machine-learning force fields can serve as an effective pre-relaxation technique, reducing computational overhead and time requirements before employing full DFT geometry optimizations.

The advent of recent notable improvements to graph neural network models for force fields has the potential to expedite these advancements. In this thesis, we have utilized the novel new Mace architecture on the two-dimensional dataset and compared its performance to the previously employed M3GNet model and found promising results with regards to the structure prediction. However, further investigation is necessary to definitely ascertain the advantages of Mace over M3Gnet in predicting energies.

Furthermore, we have been able to verify the effectiveness of transfer learning from our trained universal force fields to material specific force fields. Although the improvements in transfer learning to an alloy dataset were marginal in some cases, we anticipate the gap between regular learning and transfer learning to become much more pronounced when data for the specific material are limited. In these scenarios, transfer learning will prove

particularly beneficial, enabling us to achieve results that are more comparable to DFT with considerably reduced time and resource requirements.

Lastly, we have applied a graph neural network to directly predict the formation energies of alloys from structures relaxed using our alloy-specific force field. Traditionally, alloy formation energy predictions involve the use of Cluster Expansion, which requires the creation of a material specific DFT dataset to fit the Cluster Expansion model to. However, this method lacks transferability to other materials, limiting its generalizability despite the somewhat large time investment it requires. We were able to demonstrate improved performance, ranging up to 0.12 $\mathrm{meV/atom}$, for machine learning models using ML-relaxed structures, offering the additional advantages of knowledge transfer and generalization across materials, as well as faster inference time compared to Cluster Expansion.

In the next phase of our work, we plan to undertake the final evaluation of numerous force field models - both universal and material specific - which have been developed. Further investigation is also required to understand the reasons behind the failure of certain models, such as the models trained on the $IrRuS_2$ dataset for $e_{\text{cut}} = 0.01$ when trained separately. Additionally, we will be actively investigating the factors contributing to the suboptimal performance for the MACE models when utilized as a Universal Force Field and M3GNet used as a transfer model.

While this thesis provides some insights into suitable parameter choices, a more thorough analysis of the effectiveness of a smaller cutoff parameter, for example, still awaits. Further works could also focus on implementing sensible cutoffs for our energy and area model deviations from DFT after the relaxation. This would alleviate some of the 'outliers' or unsuccessful relaxations seen in some of our results. Additionally, one could consider benchmarking other machine learning force field architectures.

We hope that with our results we can inspire further works in creating better universal and material-specific force fields, opening the doors to hopefully getting us one step closer to resolving the century-long quest initially embarked on by Dirac.

Bibliography

[1] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Physical Review* 140 (4A Nov. 15, 1965), A1133–A1138. DOI: 10.1103/PhysRev.140.A1133. URL: https://link.aps.org/doi/10.1103/PhysRev.140.A1133.

[2] J.M. Sanchez, F. Ducastelle, and D. Gratias. "Generalized Cluster Description of Multicomponent Systems". In: *Physica A: Statistical Mechanics and its Applications* 128.1-2 (Nov. 1984), pp. 334–350. ISSN: 03784371. DOI: 10.1016/0378-4371(84)90096-7. URL: https://linkinghub.elsevier.com/retrieve/pii/0378437184900967.

[3] G. Kresse and J. Furthmüller. "Efficiency of Ab-Initio Total Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set". In: *Computational Materials Science* 6.1 (July 1, 1996), pp. 15–50. ISSN: 0927-0256. DOI: 10.1016/0927-0256(96)00008-0. URL: https://www.sciencedirect.com/science/article/pii/0927025696000080.

[4] G. Kresse and J. Furthmüller. "Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set". In: *Physical Review B* 54.16 (Oct. 15, 1996), pp. 11169–11186. DOI: 10.1103/PhysRevB.54.11169. URL: https://link.aps.org/doi/10.1103/PhysRevB.54.11169.

[5] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. "Generalized Gradient Approximation Made Simple". In: *Physical Review Letters* 77.18 (Oct. 28, 1996), pp. 3865–3868. DOI: 10.1103/PhysRevLett.77.3865. URL: https://link.aps.org/doi/10.1103/PhysRevLett.77.3865.

[6] Erik Bitzek et al. "Structural Relaxation Made Simple". In: *Physical Review Letters* 97.17 (Oct. 27, 2006), p. 170201. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.97.170201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.97.170201.

[7] Jörg Behler and Michele Parrinello. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Physical Review Letters* 98.14 (Apr. 2, 2007), p. 146401. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.98.146401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.98.146401.

[8] Jörg Behler. "Atom-Centered Symmetry Functions for Constructing High-Dimensional Neural Network Potentials". In: *The Journal of Chemical Physics* 134.7 (Feb. 21, 2011), p. 074106. ISSN: 0021-9606. DOI: 10.1063/1.3553717. URL: https://aip.scitation.org/doi/10.1063/1.3553717.

[9] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[10] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. June 12, 2017. DOI: 10.48550/arXiv.1704.01212. arXiv: 1704.01212 [cs]. URL: http://arxiv.org/abs/1704.01212. preprint.

[11] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. June 12, 2017. arXiv: 1704.01212 [cs]. URL: http://arxiv.org/abs/1704.01212. preprint.

[12] Sajedeh Manzeli et al. "2D Transition Metal Dichalcogenides". In: *Nature Reviews Materials* 2.8 (8 June 13, 2017), pp. 1–15. ISSN: 2058-8437. DOI: 10.1038/natrevmats.2017.33. URL: https://www.nature.com/articles/natrevmats201733.

[13] Jonathan Schmidt et al. "Predicting the Thermodynamic Stability of Solids Combining Density Functional Theory and Machine Learning". In: *Chemistry of Materials* 29.12 (June 27, 2017), pp. 5090–5103. ISSN: 0897-4756. DOI: 10.1021/acs.chemmater.7b00156. URL: https://doi.org/10.1021/acs.chemmater.7b00156.

[14] Kristof T. Schütt et al. *SchNet: A Continuous-Filter Convolutional Neural Network for Modeling Quantum Interactions*. Dec. 19, 2017. DOI: 10.48550/arXiv.1706.08566. arXiv: 1706.08566 [physics, stat]. URL: http://arxiv.org/abs/1706.08566. preprint.

[15] Nathaniel Thomas et al. *Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds*. May 18, 2018. arXiv: 1802.08219 [cs]. URL: http://arxiv.org/abs/1802.08219. preprint.

[16] Maurice Weiler et al. *3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data*. Oct. 27, 2018. arXiv: 1807.02547 [cs, stat]. URL: http://arxiv.org/abs/1807.02547. preprint.

[17] Tian Xie and Jeffrey C. Grossman. "Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties". In: *Physical Review Letters* 120.14 (Apr. 6, 2018), p. 145301. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.120.145301. arXiv: 1710.10324 [cond-mat]. URL: http://arxiv.org/abs/1710.10324.

[18] Chi Chen et al. "Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals". In: *Chemistry of Materials* 31.9 (May 14, 2019), pp. 3564–3572. ISSN: 0897-4756, 1520-5002. DOI: 10.1021/acs.chemmater.9b01294. arXiv: 1812.05055 [cond-mat, physics:physics]. URL: http://arxiv.org/abs/1812.05055.

[19] Ralf Drautz. "Atomic Cluster Expansion for Accurate and Transferable Interatomic Potentials". In: *Physical Review B* 99.1 (Jan. 8, 2019), p. 014104. ISSN: 2469-9950, 2469-9969. DOI: 10.1103/PhysRevB.99.014104. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.014104.

[20] Steven M. Girvin and Kun Yang. *Modern Condensed Matter Physics*. Cambridge ; New York, NY: Cambridge University Press, 2019. ISBN: 978-1-107-13739-4.

[21] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[22] Rhys E. A. Goodall and Alpha A. Lee. "Predicting Materials Properties without Crystal Structure: Deep Representation Learning from Stoichiometry". In: *Nature Communications* 11.1 (Dec. 8, 2020), p. 6280. ISSN: 2041-1723. DOI: 10.1038/s41467-020-19964-7. arXiv: 1910.00617 [cond-mat, physics:physics]. URL: http://arxiv.org/abs/1910.00617.

[23] Gus L. W. Hart et al. "Machine Learning for Alloys". In: *Nature Reviews Materials* 6.8 (8 Aug. 2021), pp. 730–755. ISSN: 2058-8437. DOI: 10.1038/s41578-021-00340-w. URL: https://www.nature.com/articles/s41578-021-00340-w.

[24] Sara Kadkhodaei and Jorge A. Muñoz. "Cluster Expansion of Alloy Theory: A Review of Historical Development and Modern Innovations". In: *JOM* 73.11 (Nov. 1, 2021), pp. 3326–3346. ISSN: 1543-1851. DOI: 10.1007/s11837-021-04840-6. URL: https://doi.org/10.1007/s11837-021-04840-6.

[25] Prakhar Kaushik et al. *Understanding Catastrophic Forgetting and Remembering in Continual Learning with Optimal Relevance Mapping*. Feb. 22, 2021. DOI: 10.48550/arXiv.2102.11343. arXiv: 2102.11343 [cs]. URL: http://arxiv.org/abs/2102.11343. preprint.

[26] Conrad W. Rosenbrock et al. "Machine-Learned Interatomic Potentials for Alloys and Alloy Phase Diagrams". In: *npj Computational Materials* 7.1 (1 Jan. 29, 2021), pp. 1–9. ISSN: 2057-3960. DOI: 10.1038/s41524-020-00477-2. URL: https://www.nature.com/articles/s41524-020-00477-2.

[27] Jonathan Schmidt et al. "Crystal Graph Attention Networks for the Prediction of Stable Materials". In: *Science Advances* 7.49 (Dec. 3, 2021), eabi7948. DOI: 10.1126/sciadv.abi7948. URL: https://www.science.org/doi/full/10.1126/sciadv.abi7948.

[28] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. *Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra*. June 7, 2021. arXiv: 2102.03150 [physics]. URL: http://arxiv.org/abs/2102.03150. preprint.

[29] Oliver T. Unke et al. "Machine Learning Force Fields". In: *Chemical Reviews* 121.16 (Aug. 25, 2021), pp. 10142–10186. ISSN: 0009-2665, 1520-6890. DOI: 10.1021/acs.chemrev.0c01111. URL: https://pubs.acs.org/doi/10.1021/acs.chemrev.0c01111.

[30] Ilyes Batatia et al. *The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials*. Nov. 24, 2022. arXiv: 2205.06643 [cond-mat, physics:physics, stat]. URL: http://arxiv.org/abs/2205.06643. preprint.

[31] Ilyes Batatia et al. *The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials*. Nov. 24, 2022. arXiv: 2205.06643 [cond-mat, physics:physics, stat]. URL: http://arxiv.org/abs/2205.06643. preprint.

[32] Simon Batzner et al. "E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials". In: *Nature Communications* 13.1 (May 4, 2022), p. 2453. ISSN: 2041-1723. DOI: 10.1038/s41467-022-29939-5. arXiv: 2101.03164 [cond-mat, physics:physics]. URL: http://arxiv.org/abs/2101.03164.

[33] Chi Chen and Shyue Ping Ong. "A Universal Graph Deep Learning Interatomic Potential for the Periodic Table". In: *Nature Computational Science* 2.11 (11 Nov. 2022), pp. 718–728. ISSN: 2662-8457. DOI: 10.1038/s43588-022-00349-3. URL: https://www.nature.com/articles/s43588-022-00349-3.

[34] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. *Directional Message Passing for Molecular Graphs*. Apr. 5, 2022. arXiv: 2003.03123 [physics, stat]. URL: http://arxiv.org/abs/2003.03123. preprint.

[35] Mario Geiger and Tess Smidt. *E3nn: Euclidean Neural Networks*. July 18, 2022. arXiv: 2207.09453 [cs]. URL: http://arxiv.org/abs/2207.09453. preprint.

[36] Jiaqi Han et al. *Geometrically Equivariant Graph Neural Networks: A Survey*. Feb. 21, 2022. arXiv: 2202.07230 [cs]. URL: http://arxiv.org/abs/2202.07230. preprint.

[37] Andrea Silva et al. "Design Guidelines for Two-Dimensional Transition Metal Dichalcogenide Alloys". In: *Chemistry of Materials* 34.23 (Dec. 13, 2022), pp. 10279–10290. ISSN: 0897-4756, 1520-5002. DOI: 10.1021/acs.chemmater.2c01390. URL: https://pubs.acs.org/doi/10.1021/acs.chemmater.2c01390.

[38] Andrea Silva et al. "Pettifor Maps of Complex Ternary Two-Dimensional Transition Metal Sulfides". In: *npj Computational Materials* 8.1 (1 Aug. 20, 2022), pp. 1–11. ISSN: 2057-3960. DOI: 10.1038/s41524-022-00868-7. URL: https://www.nature.com/articles/s41524-022-00868-7.

[39] Anthony Yu-Tung Wang et al. "CrabNet for Explainable Deep Learning in Materials Science: Bridging the Gap Between Academia and Industry". In: *Integrating Materials and Manufacturing Innovation* 11.1 (Mar. 2022), pp. 41–56. ISSN: 2193-9764, 2193-9772. DOI: 10.1007/s40192-021-00247-y. URL: https://link.springer.com/10.1007/s40192-021-00247-y.

[40] Hai-Chen Wang et al. *Symmetry-Based Computational Search for Novel Binary and Ternary 2D Materials*. Dec. 7, 2022. DOI: 10.48550/arXiv.2212.03975. arXiv: 2212.03975 [cond-mat]. URL: http://arxiv.org/abs/2212.03975. preprint.

[41] Jun-Zhong Xie, Xu-Yuan Zhou, and Hong Jiang. "Perspective on Optimal Strategies of Building Cluster Expansion Models for Configurationally Disordered Materials". In: *The Journal of Chemical Physics* 157.20 (Nov. 28, 2022), p. 200901. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/5.0106788. URL: https://pubs.aip.org/jcp/article/157/20/200901/2842100/Perspective-on-optimal-strategies-of-building.

[42] Keqiang Yan et al. *Periodic Graph Transformers for Crystal Material Property Prediction*. Sept. 23, 2022. arXiv: 2209.11807 [cs]. URL: http://arxiv.org/abs/2209.11807. preprint.

[43] C. Lawrence Zitnick et al. *Spherical Channels for Modeling Atomic Interactions*. Oct. 13, 2022. arXiv: 2206.14331 [physics]. URL: http://arxiv.org/abs/2206.14331. preprint.

[44] Ilyes Batatia et al. *MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields*. Jan. 26, 2023. arXiv: 2206.07697 [cond-mat, physics:physics, stat]. URL: http://arxiv.org/abs/2206.07697. preprint.

[45] Chaitanya K. Joshi et al. *On the Expressive Power of Geometric Graph Neural Networks*. Jan. 23, 2023. arXiv: 2301.09308 [cs, math, stat]. URL: http://arxiv.org/abs/2301.09308. preprint.

[46] Arthur Kosmala et al. *Ewald-Based Long-Range Message Passing for Molecular Graphs*. June 6, 2023. arXiv: 2303.04791 [cond-mat, physics:physics]. URL: http://arxiv.org/abs/2303.04791. preprint.

[47] David Peter Kovacs et al. "Evaluation of the MACE Force Field Architecture: From Medicinal Chemistry to Materials Science". In: *The Journal of Chemical Physics* 159.4 (July 28, 2023), p. 044118. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/5.0155322. arXiv: 2305.14247 [physics, stat]. URL: http://arxiv.org/abs/2305.14247.

[48] Yi-Lun Liao and Tess Smidt. *Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs*. Feb. 27, 2023. DOI: 10.48550/arXiv.2206.11990. arXiv: 2206.11990 [physics]. URL: http://arxiv.org/abs/2206.11990. preprint.

[49] Yuchao Lin et al. *Efficient Approximations of Complete Interatomic Potentials for Crystal Property Prediction*. Aug. 1, 2023. arXiv: 2306.10045 [physics]. URL: http://arxiv.org/abs/2306.10045. preprint.

[50] Jonathan Schmidt et al. "Machine-Learning-Assisted Determination of the Global Zero-Temperature Phase Diagram of Materials". In: *Advanced Materials* 35.22 (2023), p. 2210788. ISSN: 1521-4095. DOI: 10.1002/adma.202210788. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202210788.

[51] Xuan Zhang et al. *Artificial Intelligence for Science in Quantum, Atomistic, and Continuum Systems*. July 17, 2023. arXiv: 2307.08423 [physics]. URL: http://arxiv.org/abs/2307.08423. preprint.

[52] CASM. *CASM*. CASM. URL: https://prisms-center.github.io/CASMcode_docs/.

Chapter 6

Appendix

6.1 Convex Optimization

For many machine learning models the optimisation criterion is convex which makes the optimisation a lot easier since the convex functions only have a single global minima and not potentially many local minima as for non-convex functions (i.e every local minimum is a global minima).

Optimisation criterion's for neural networks are often not convex but in practice often local minima's suffice. Recall that a convex function satisfies

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \tag{6.1}$$

with $0 \leq \lambda \leq 1$ and $\mathbf{x}, \mathbf{y} \in \mathcal{S}$. They are often not complex because one can show that in general a function f is convex iff the Hessian $\mathbf{H} = \nabla^2 f(\mathbf{x})$ is positive semi-definite and strictly convex if $\mathbf{H}$ positive definite. So we require the Hessian of the loss function $\mathcal{L}(\boldsymbol{\theta})$ to be positive definite to have a convex optimisation problem.

In principle here we start at some random point and hope that we can end up in a useful minima. In practice this is not always as easy as it sounds.

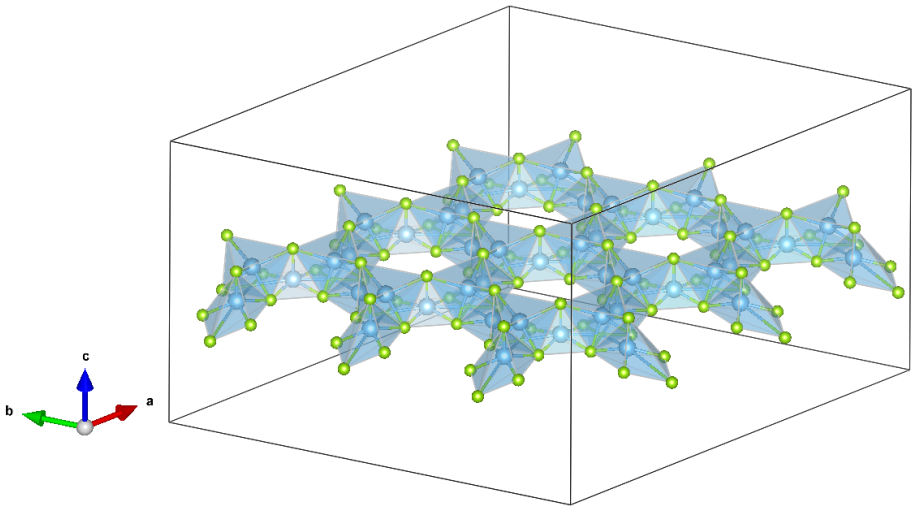
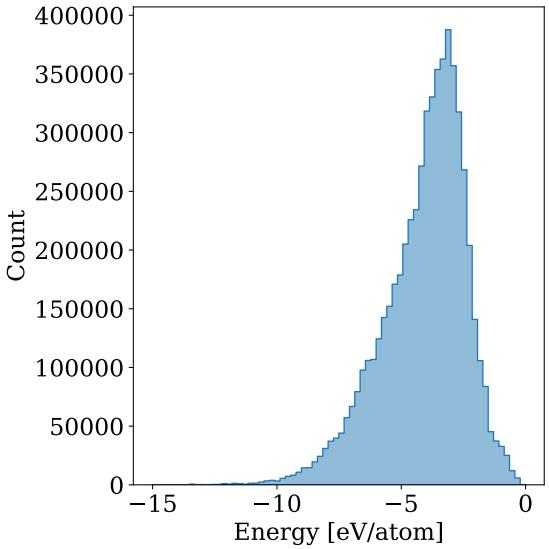
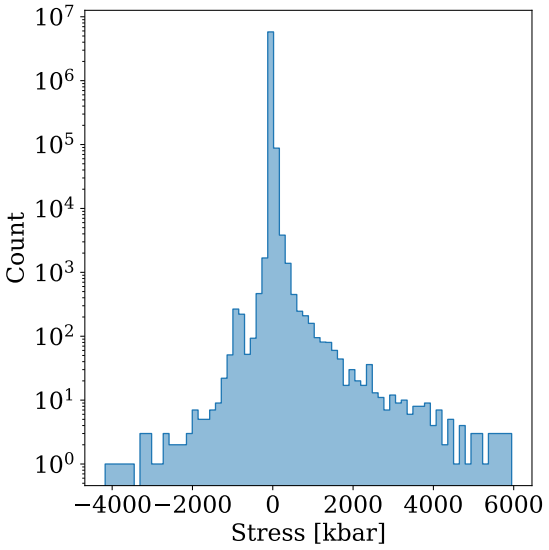


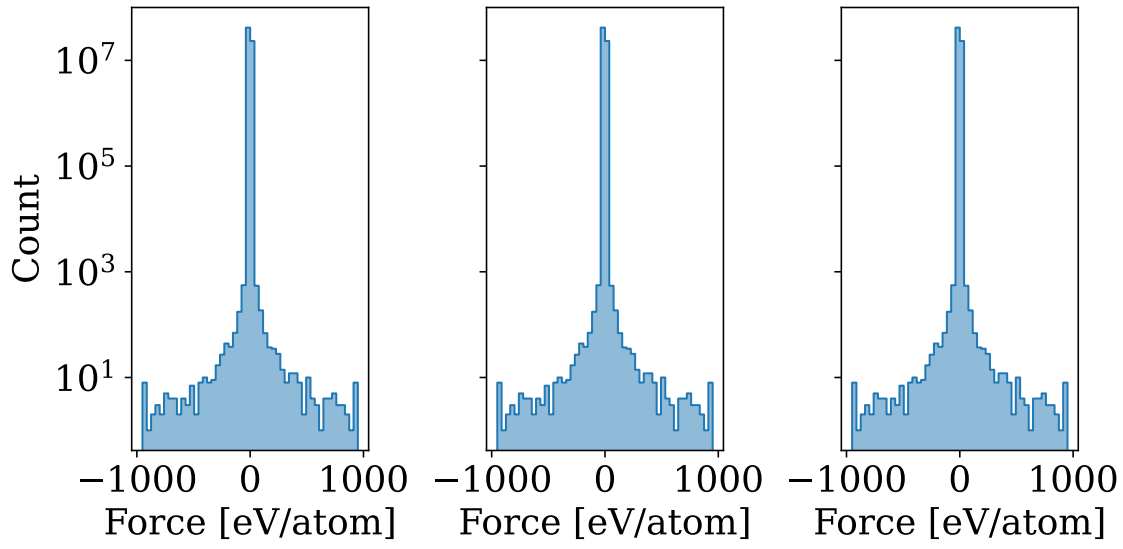
Figure 6.1: Example Structure Sn4Br8. We see the periodicity in the x and y direction while there is no periodicity in the z direction. This is a supercell consisting of 3 units cells in x and y direction.



(a) Energy per atom distribution for 2D dataset



(b) Stress distribution for the 2D dataset



(c) Force distribution for the x, y and z component of the Force vector $\mathbf{F}$ for each atom

Figure 6.2: Distribution over energies, stresses and forces for the 2D Dataset. Note the logarithmic scaling of the y-axis for both forces and stresses.

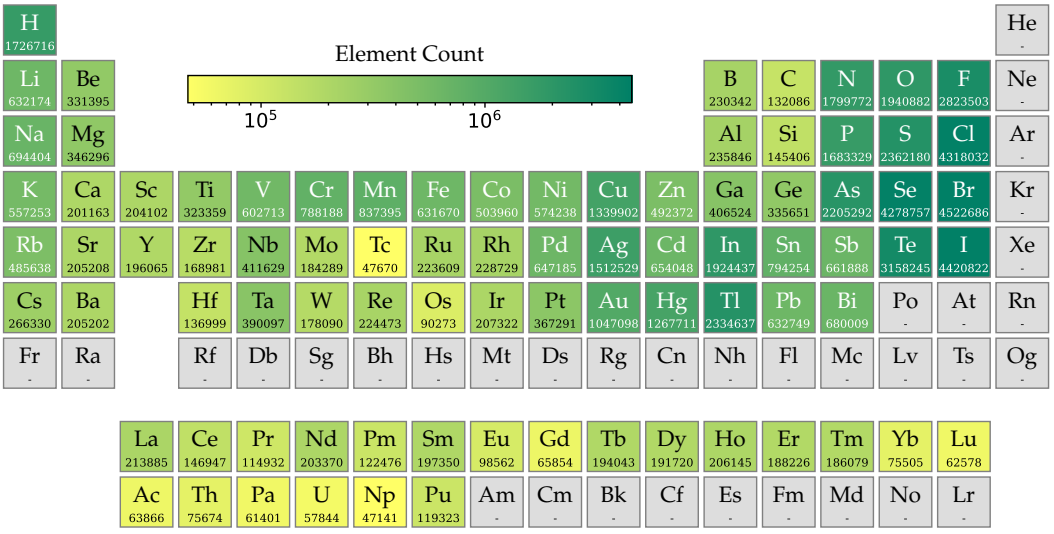
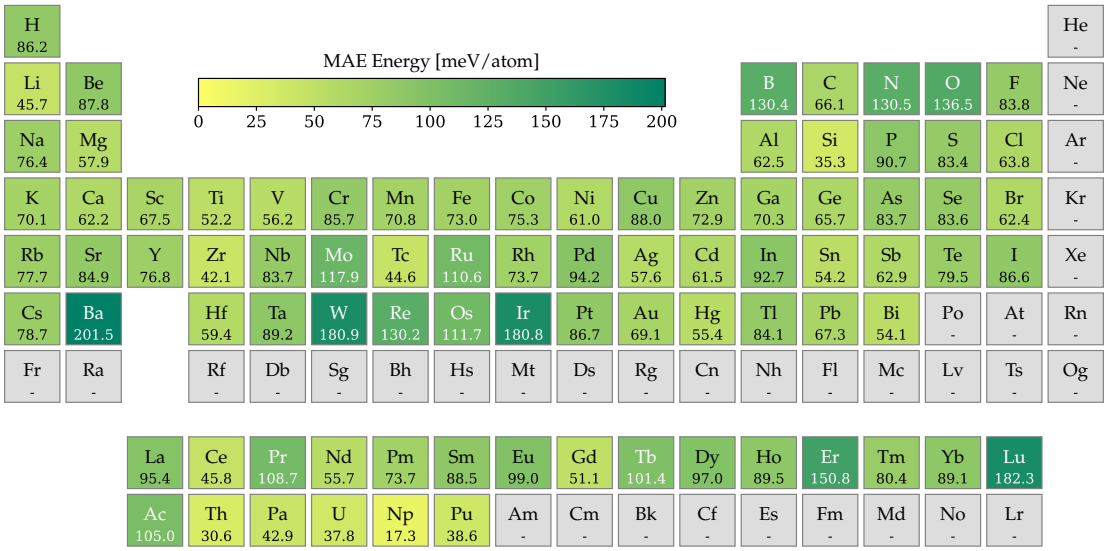
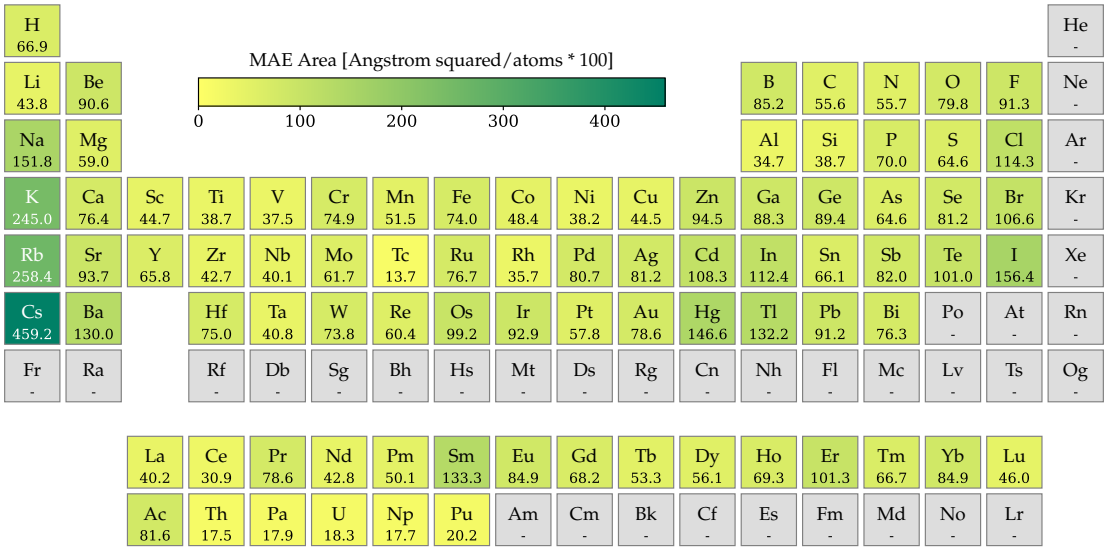


Figure 6.3: Chemical element distribution of all geometry optimization steps for the two-dimensional extended compound dataset.

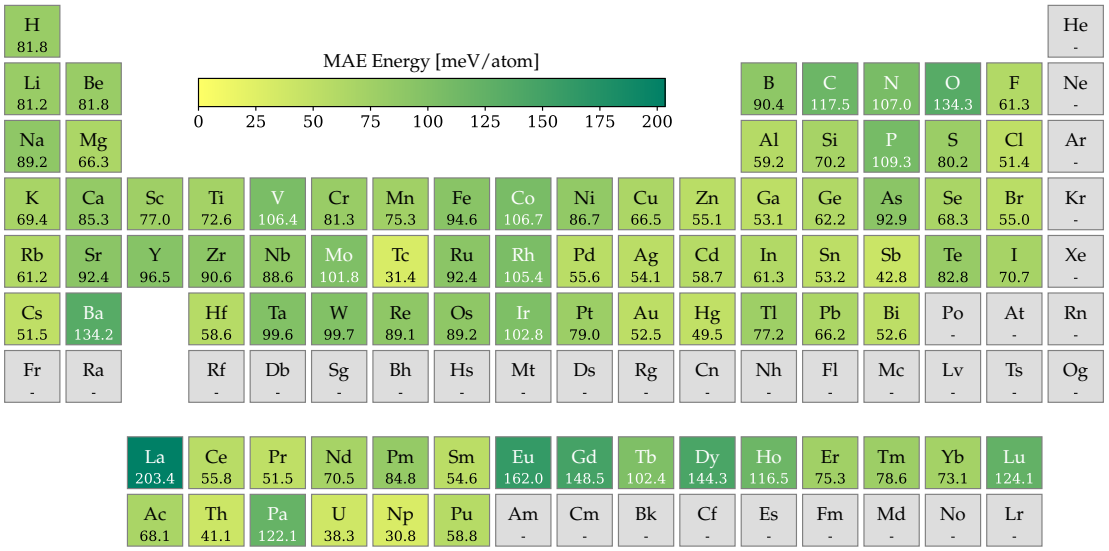


(a)

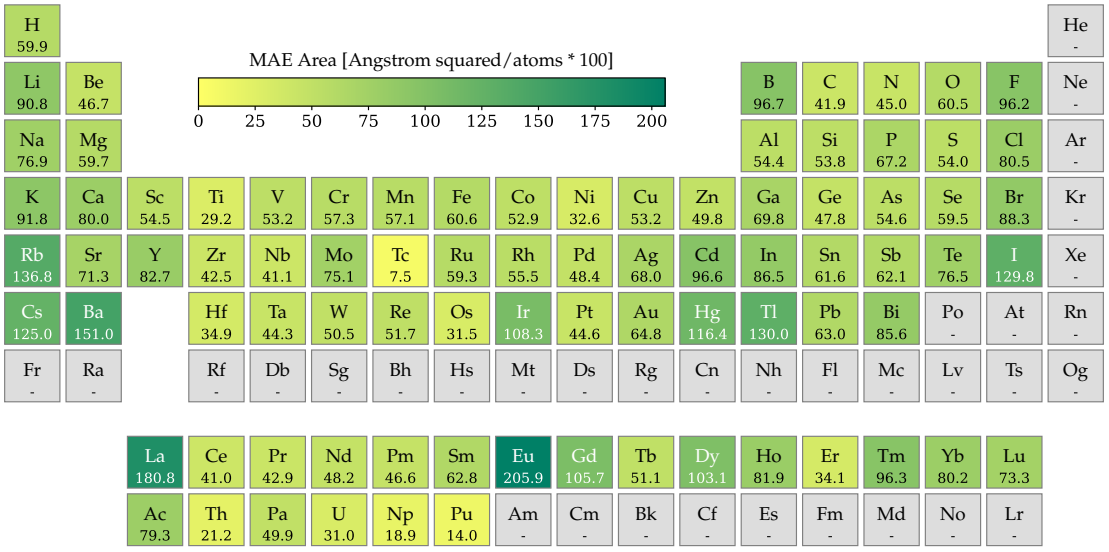


(b)

Figure 6.4: Distribution of the mean absolute error (MAE) in the *validation data* for (a) energy and (b) area, categorized by chemical elements in the periodic system. This is using the **M3GNet** model with a learning rate $\text{lr} = 1 \times 10^{-3}$ and a batch size of $\text{bs} = 128$.



(a)



(b)

Figure 6.5: Distribution of the mean absolute error (MAE) in the *validation data* for (a) energy and (b) area, categorized by chemical elements in the periodic system. This is using the **M3GNet** model with a learning rate $\mathrm{lr} = 1 \times 10^{-3}$ and a batch size of $\mathrm{bs} = 128$.

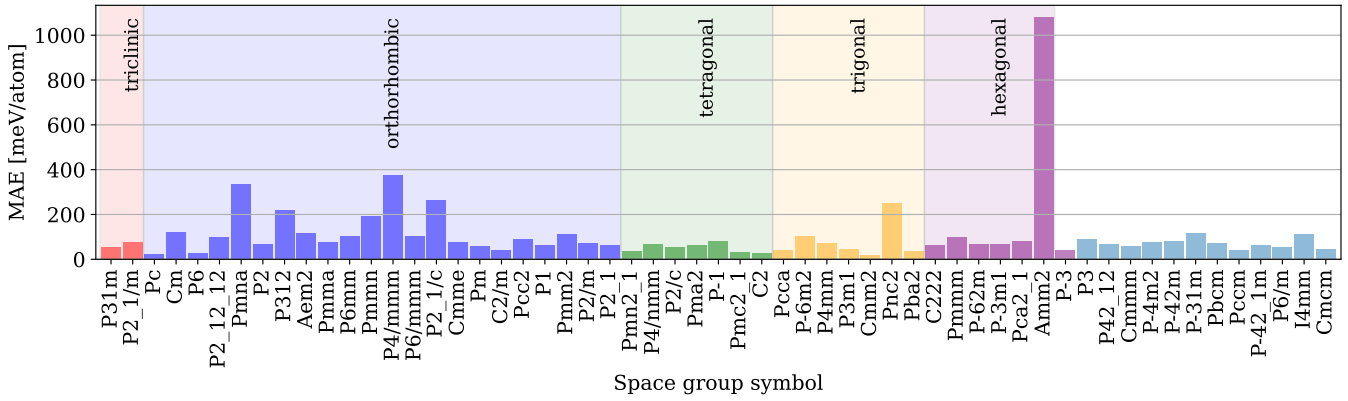
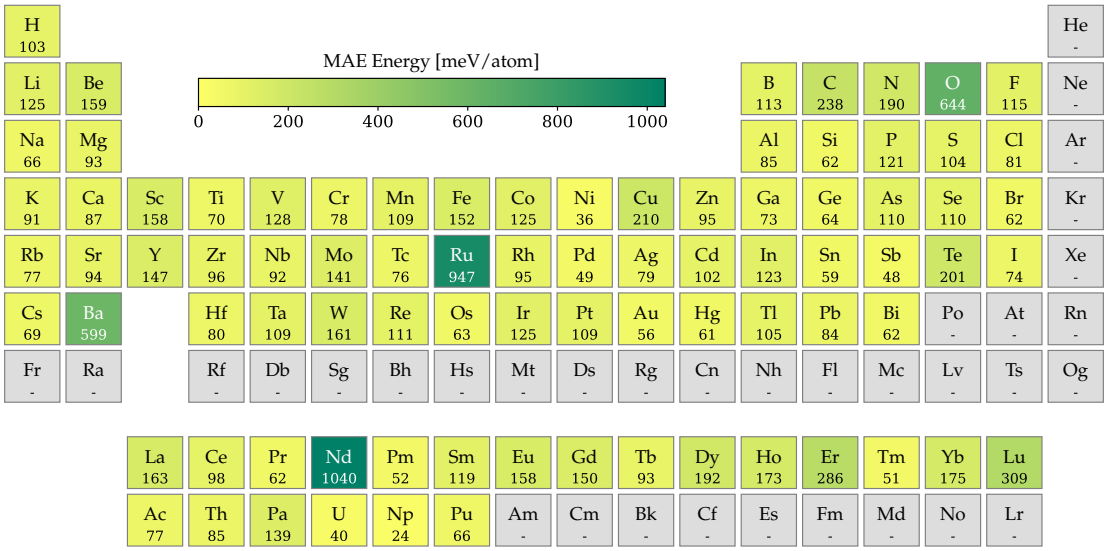


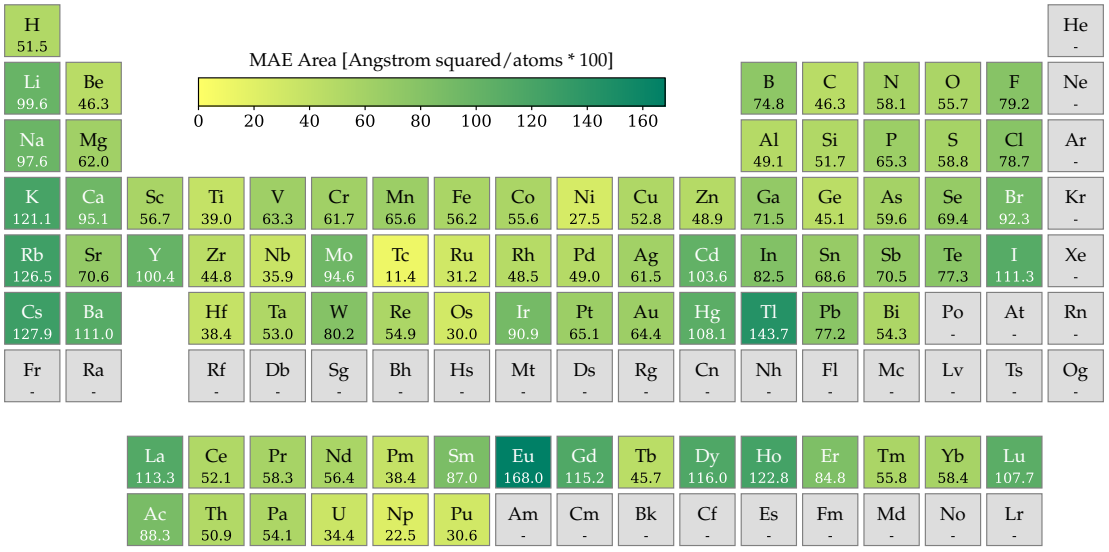
Figure 6.6: Space group energy MAE distribution of test set for model lr1e-4bs128 with $e_{\text{cut}} = 1000000$. We find an exceedingly large energy error resulting from structures belonging to the hexagonal space group Amm2.

Alloy System		Cutoff = 0.01 lr=1e-3, normal	lr=1e-3, transfer
$IrRuS_2$	Energy	**17.65**	61.11
	Area	**2.19**	25.30
$MoTaS_2$	Energy	2.44	**1.41**
	Area	**0.36**	0.51
$MoWS_2$	Energy	0.62	**0.38**
	Area	**0.52**	1.03
$NbMoS_2$	Energy	4.02	**1.47**
	Area	0.33	**0.31**
$TiNbS_2$	Energy	**19.03**	53.52
	Area	**1.86**	52.68
$TiTaS_2$	Energy	**7.85**	31.79
	Area	**33.45**	41.38
$TiVS_2$	Energy	**16.33**	43.06
	Area	**0.47**	43.54
$ZrTaS_2$	Energy	**7.30**	39.02
	Area	**0.51**	42.88

Table 6.1: Relaxation for M3GNet for datasplit 9055 with batch size 128 on the two-dimensional alloy dataset. The transfer model used here is the UFF trained on all two-dimensional data with lr=1e-4 and bs=128. The relaxations were performed using the *validation set*. The energies are given in units of meV/atom and the area in units of $\text{Å}^2/\text{atom} \times 100$



(a)



(b)

Figure 6.7: Distribution of the mean absolute error (MAE) for (a) energy and (b) area, categorized by chemical elements in the periodic system. These errors are obtained by using the **MACE** model with a learning rate $\text{lr} = 1 \times 10^{-3}$ and a batch size of $\text{bs} = 64$ to relax the *test data*.

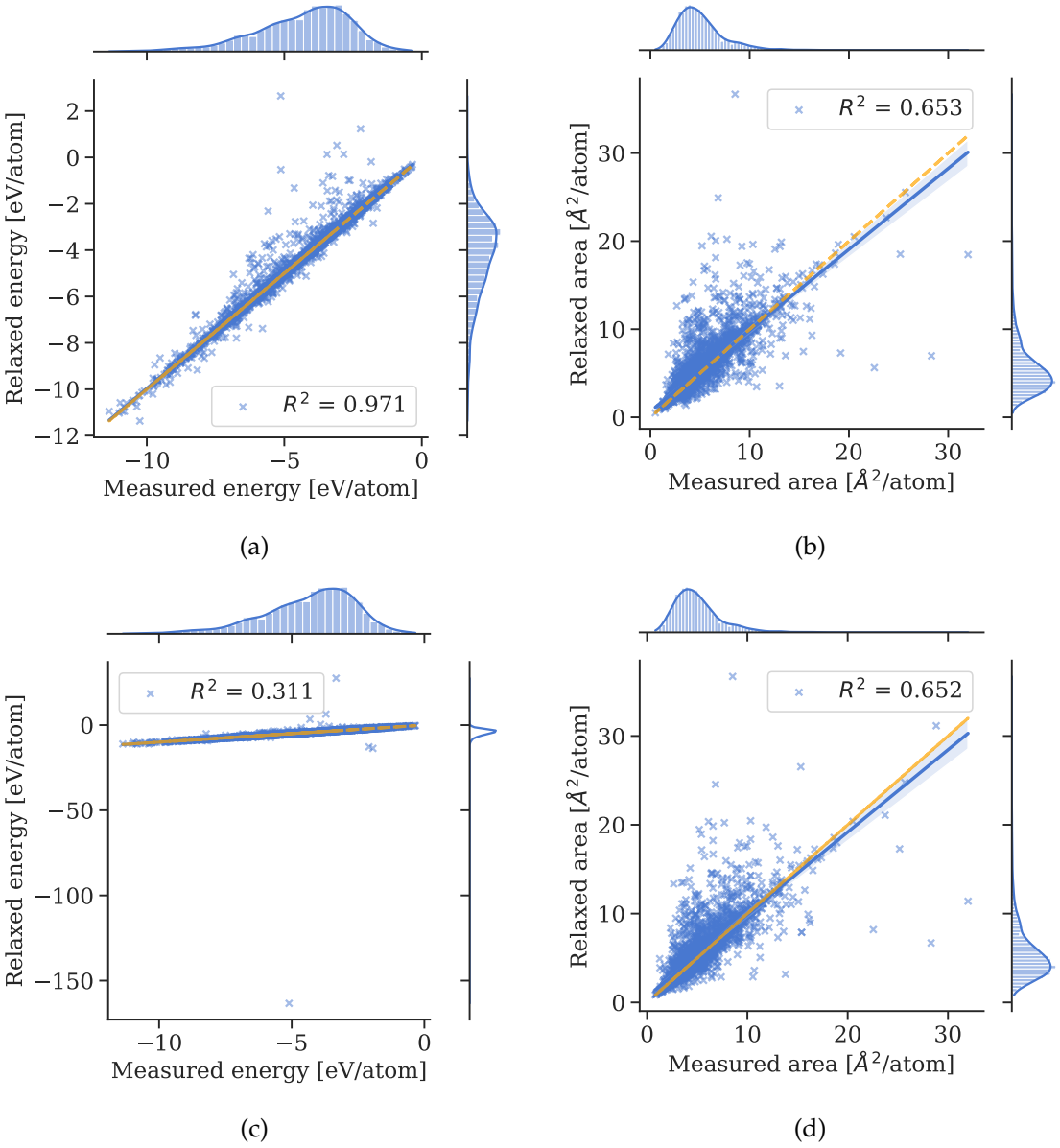


Figure 6.8: Test set MAE errors for (a) energy (b) area of MACE model lr $= 0.003$, bs $= 128$ and (c) energy and (d) area of model lr $= 1 \times 10^{-3}$, bs $= 64$. From plot (c) we observe one clear outlier in the model.

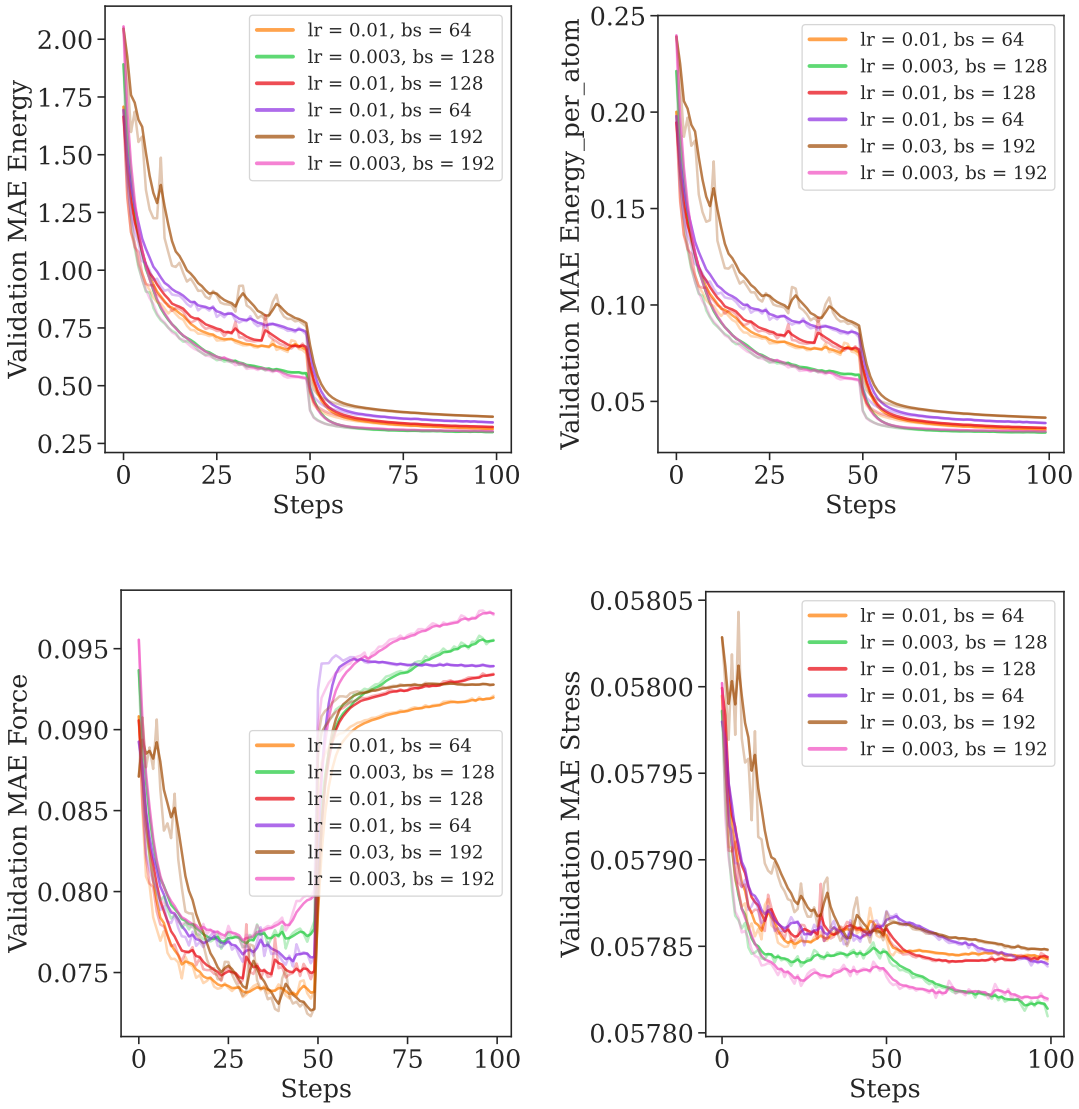


Figure 6.9: Training curves for UFF MACE models on the two-dimensional data with $e_{\text{cut}} = 1000000$.

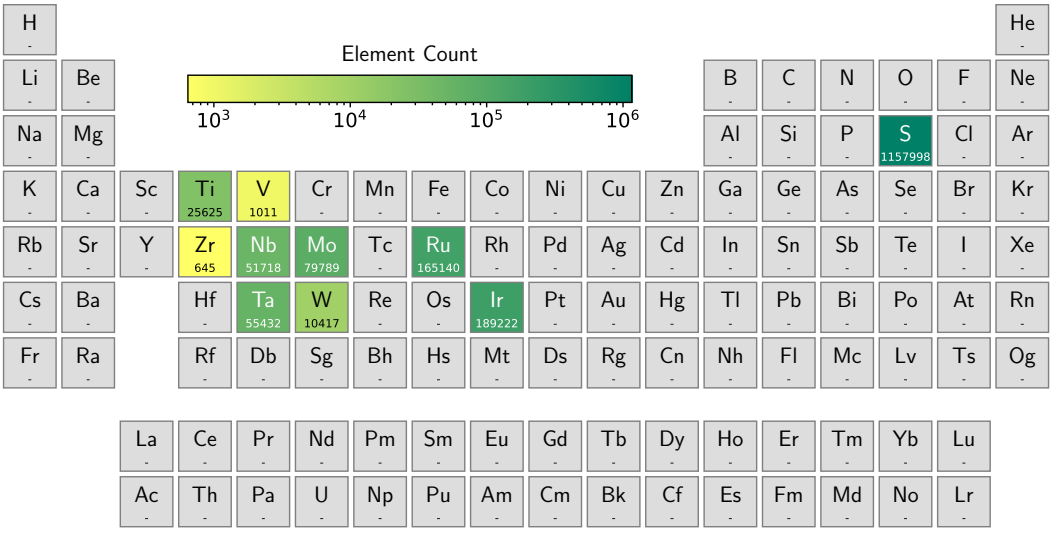


Figure 6.10: Alloy dataset chemical element distribution.

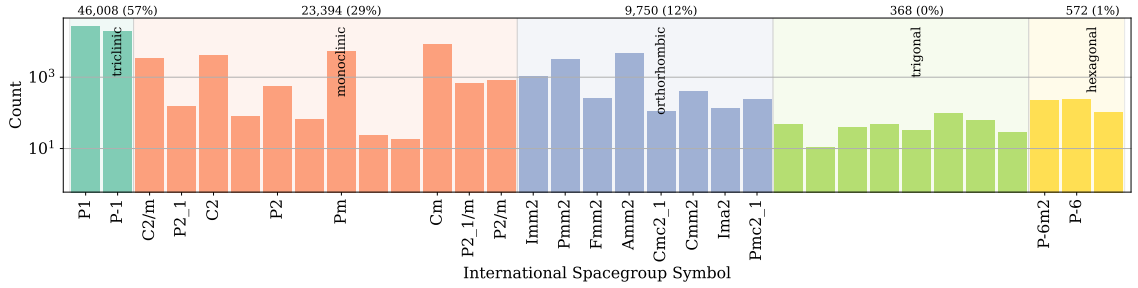
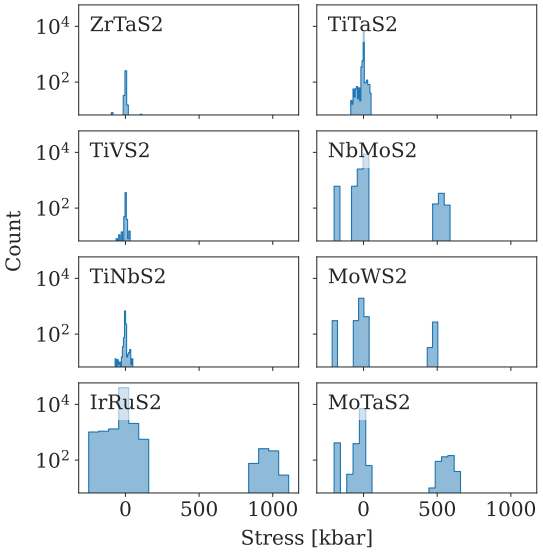


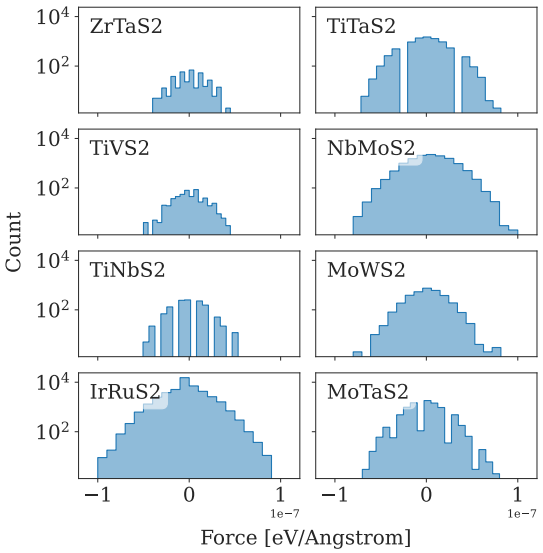
Figure 6.11: Distribution of space group symmetries for the alloy dataset.

	$NbMoS_2$	$MoWS_2$	$MoTaS_2$	$TiTaS_2$
DFT	1.23	0.13	0.28	3.42
ML 1	3.08	0.17	0.39	4.48
ML 2	1.64	0.12	0.63	3.48

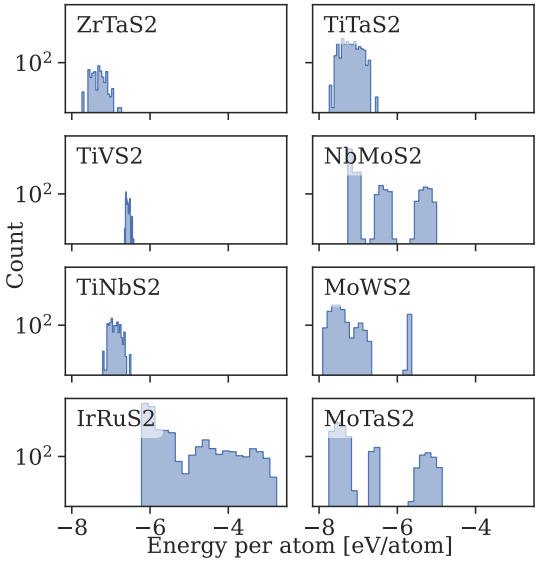
Table 6.2: Formation energy MAE [meV] for the validation set. ML1 Model from which the relaxed structures are used from is the MACE 801010 normal training lr=0.003, bs=16 model. ML 2 is MACE transfer model with 801010 ecutoff = 1000000 and lr=0.0001 and bs = 16. The model here used with that data is a NEQUIP model with a lr=0.0075 and bs=16. The energy MAE errors are given in units of meV.



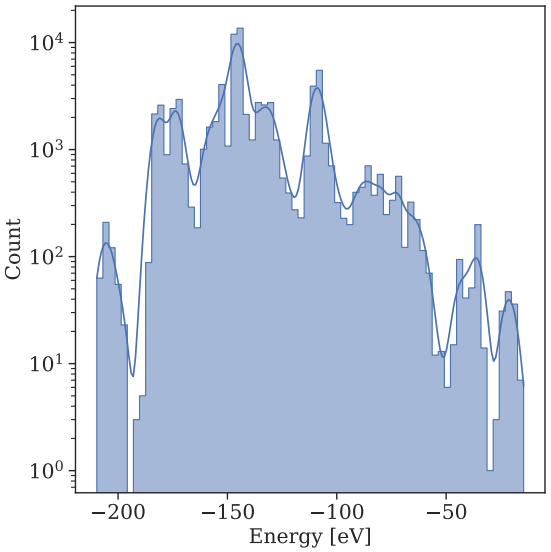
(a) Distribution stress for alloys.



(b) Distribution force for alloys.



(a) Distribution Energy per atom for the individual alloy systems.



(b) Energy distribution for all the alloy systems combined.

Figure 6.13: Energy distributions alloys

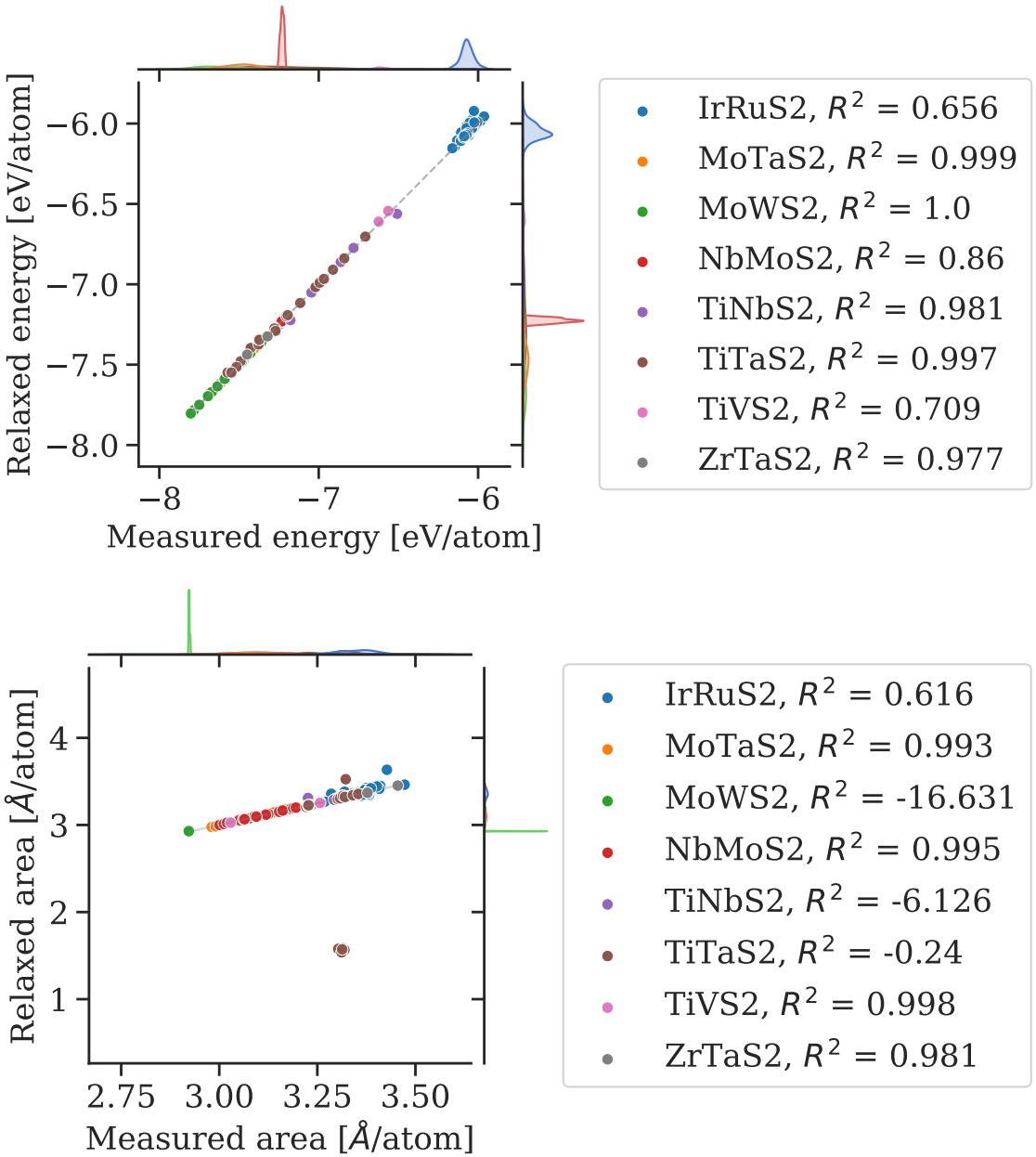


Figure 6.14: Line plot for M3GNet with datasplit=9055, normal training, cutoff = 0.01, lr = 0.001, bs=128

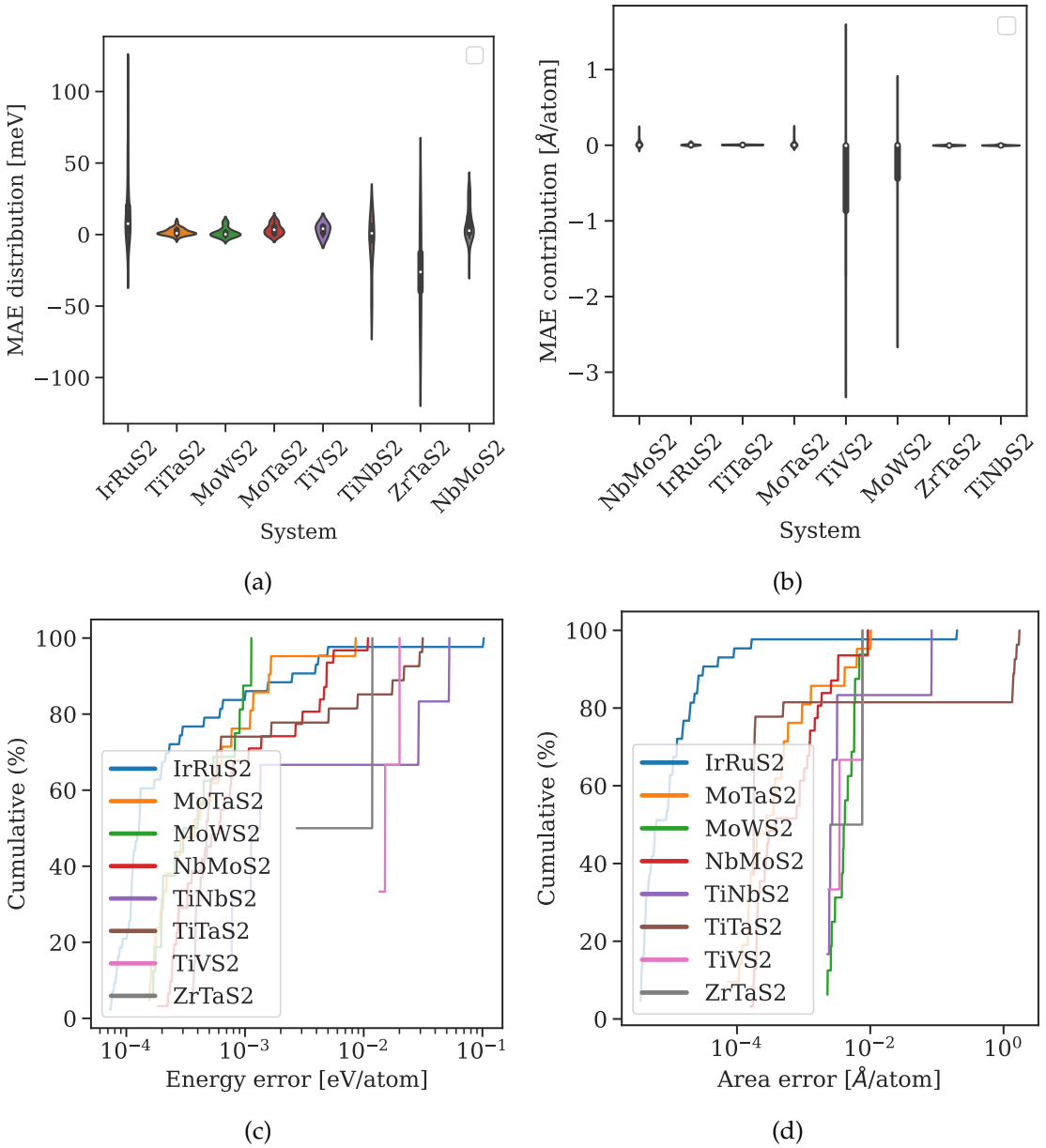


Figure 6.15: Error distribution and cumulative error histograms for M3GNet model with datasplit=9055, cutoff = 0.01, normal learning, lr=0.001, bs=128.

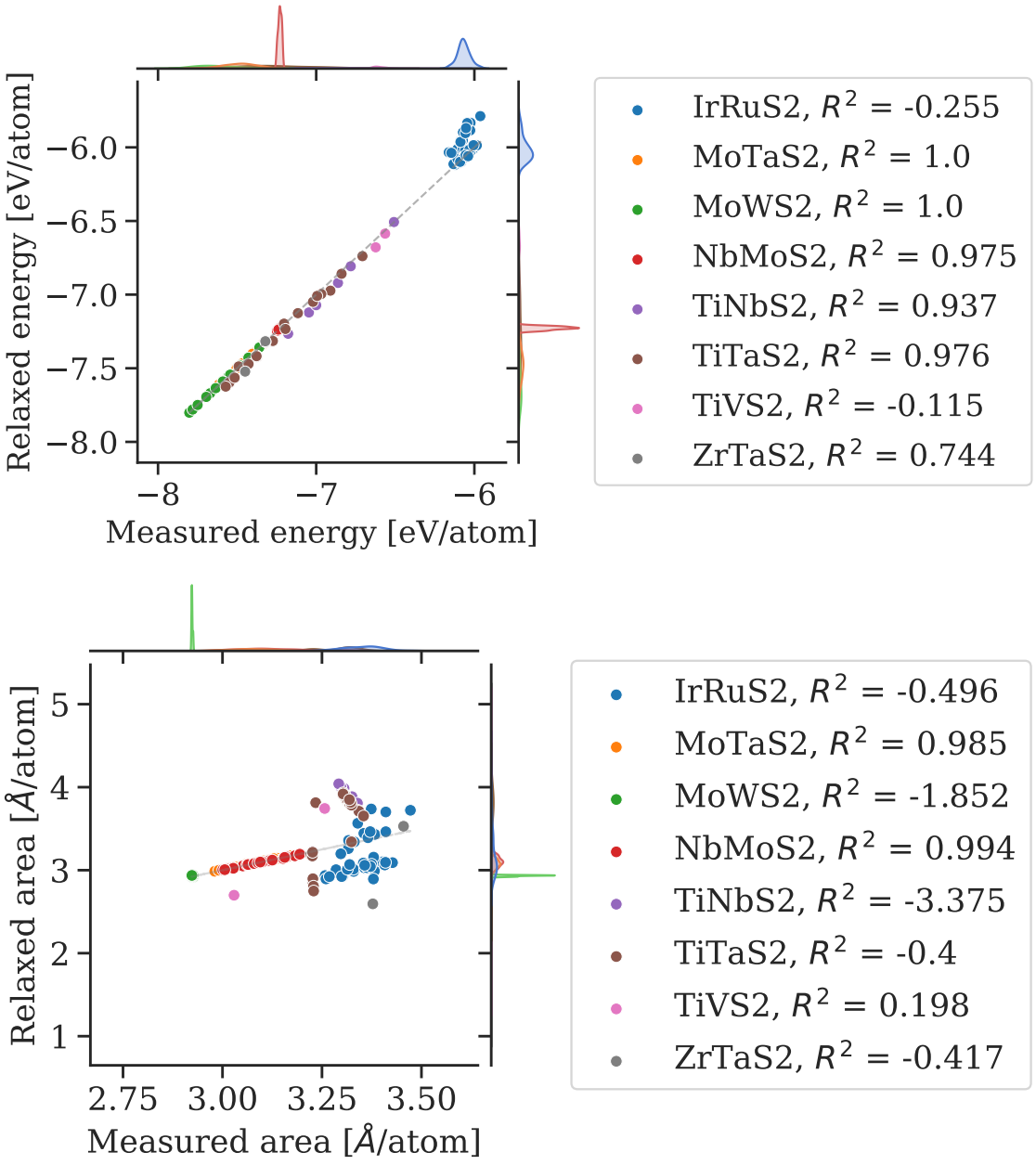


Figure 6.16: Line plot for M3GNet with datasplit=9055, transfer training, cutoff = 0.01, lr=0.001, bs=128

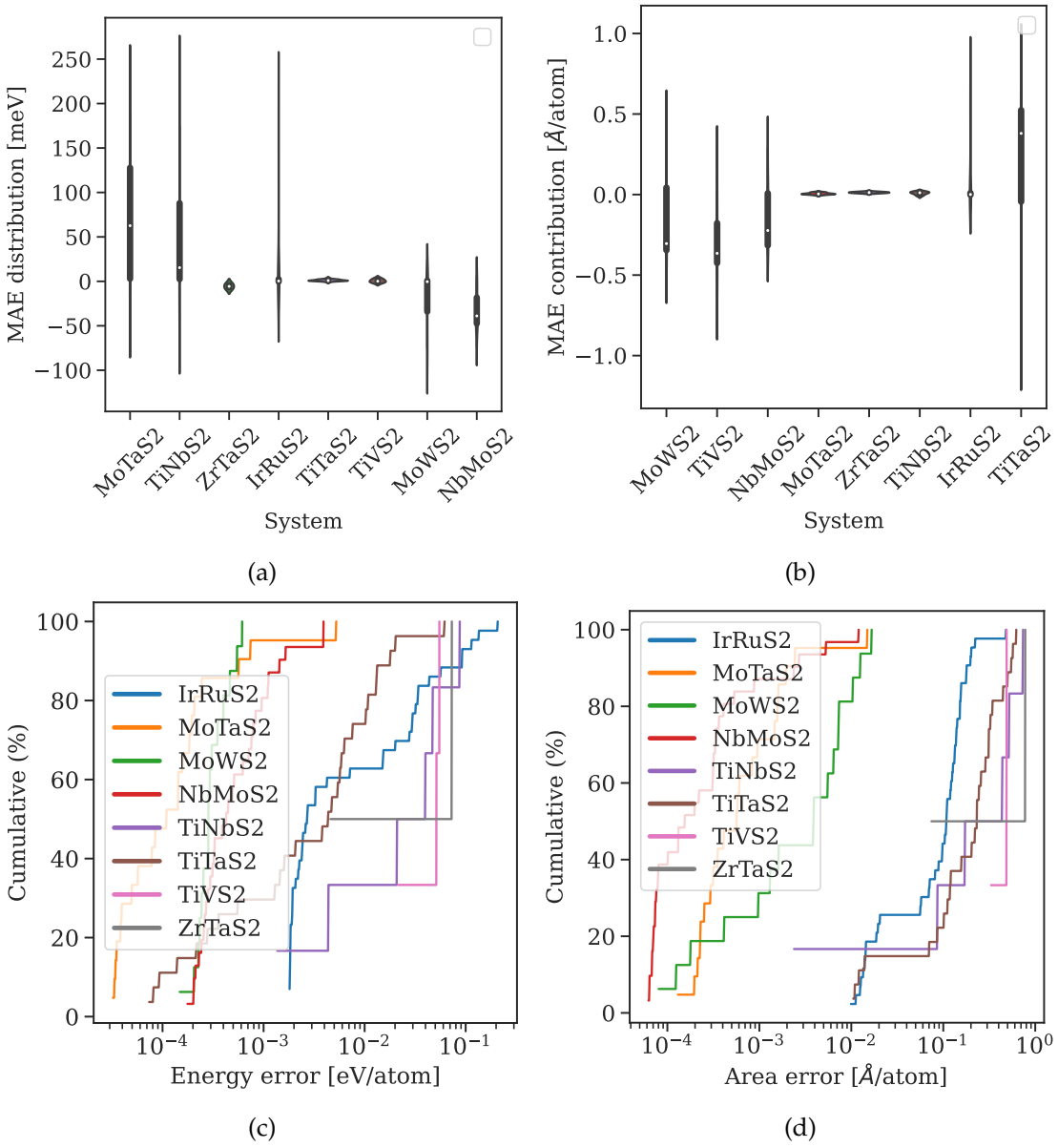


Figure 6.17: Error distribution and cumulative error histograms for M3GNet model with datasplit=9055, cutoff = 0.01, transfer learning, lr=0.001, bs=128.

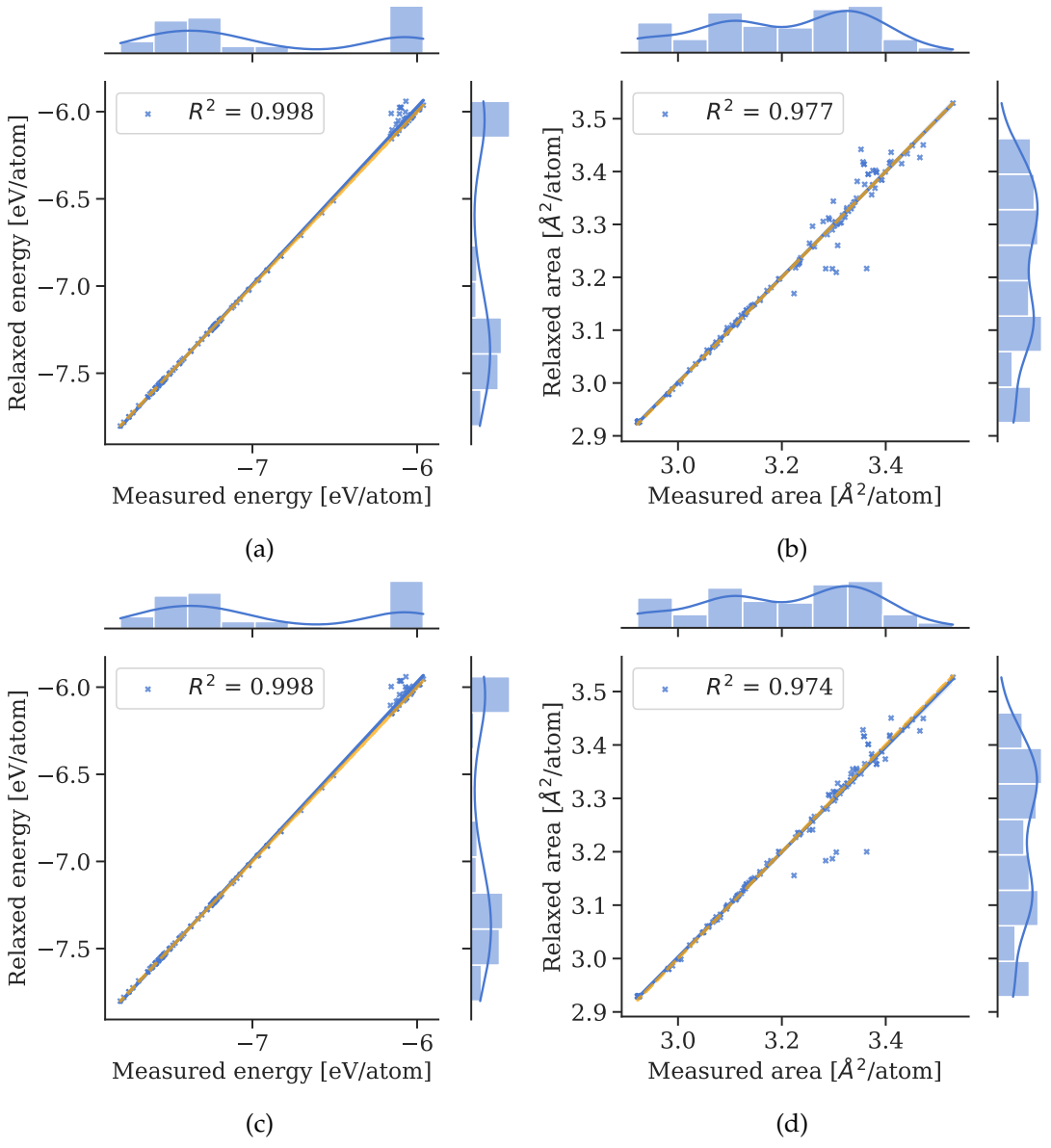
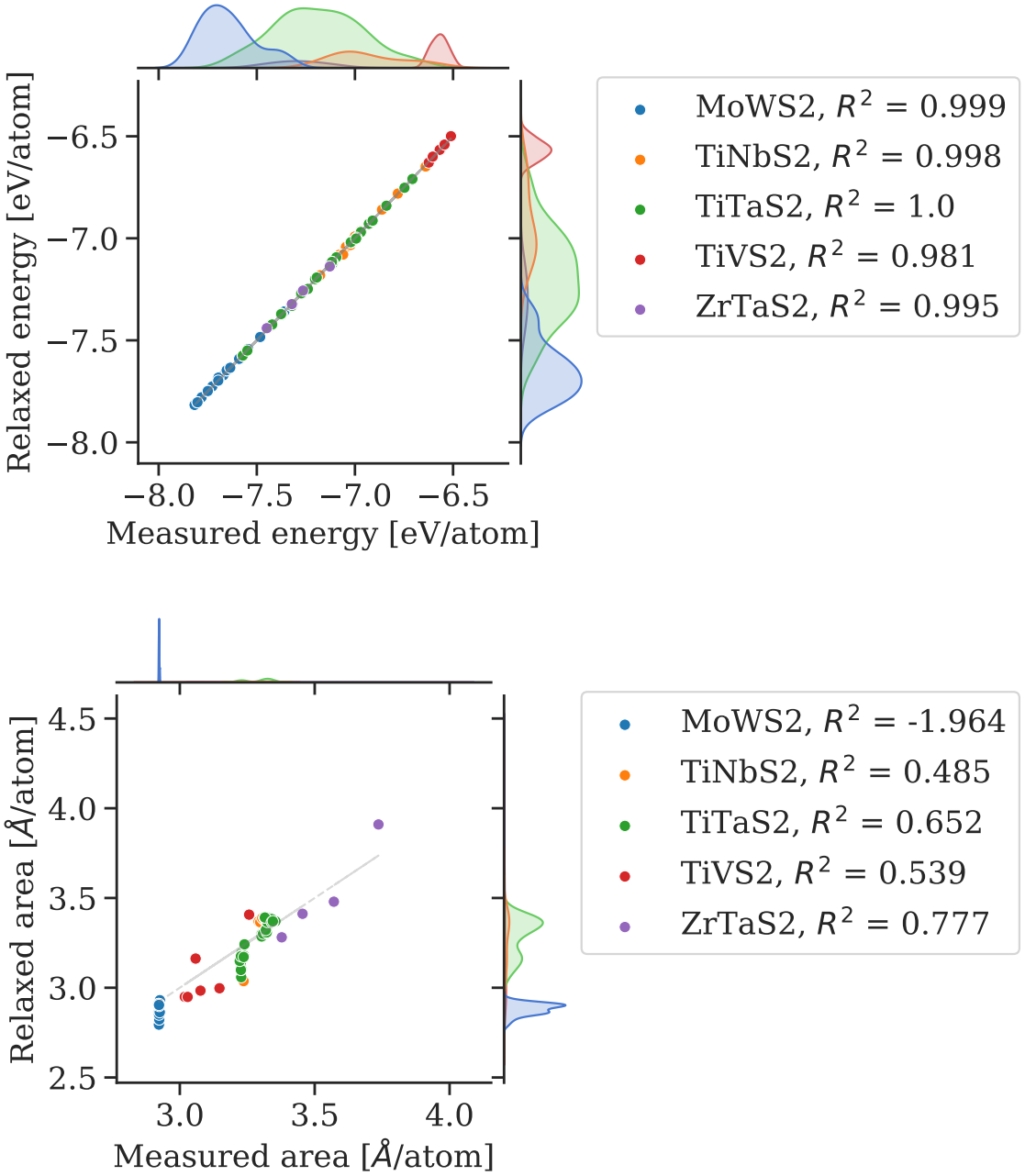


Figure 6.18: (a) Energy MAE (b) Area MAE for normally trained M3GNet model with cutoff 0.01 on the complete alloy dataset. Similarly for (c) and (d) we have the energy and area MAE for the transfer learned model lr1e-3bs128 from the UFF model lr1e-4bs128.



(a) Line plot for MACE with datasplit=801010, normal training, cutoff = 0.001, lr = 0.003, bs=16.

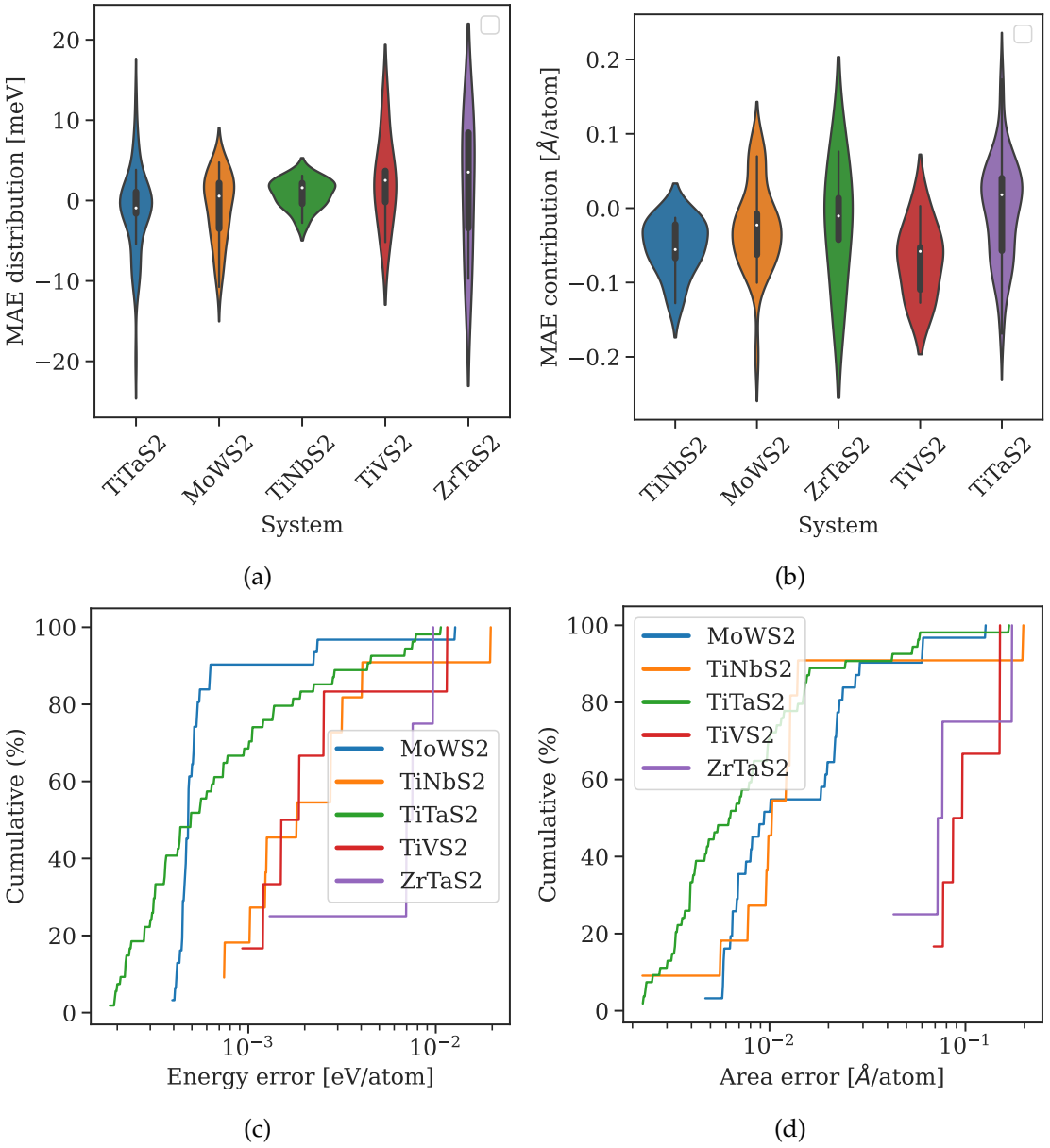
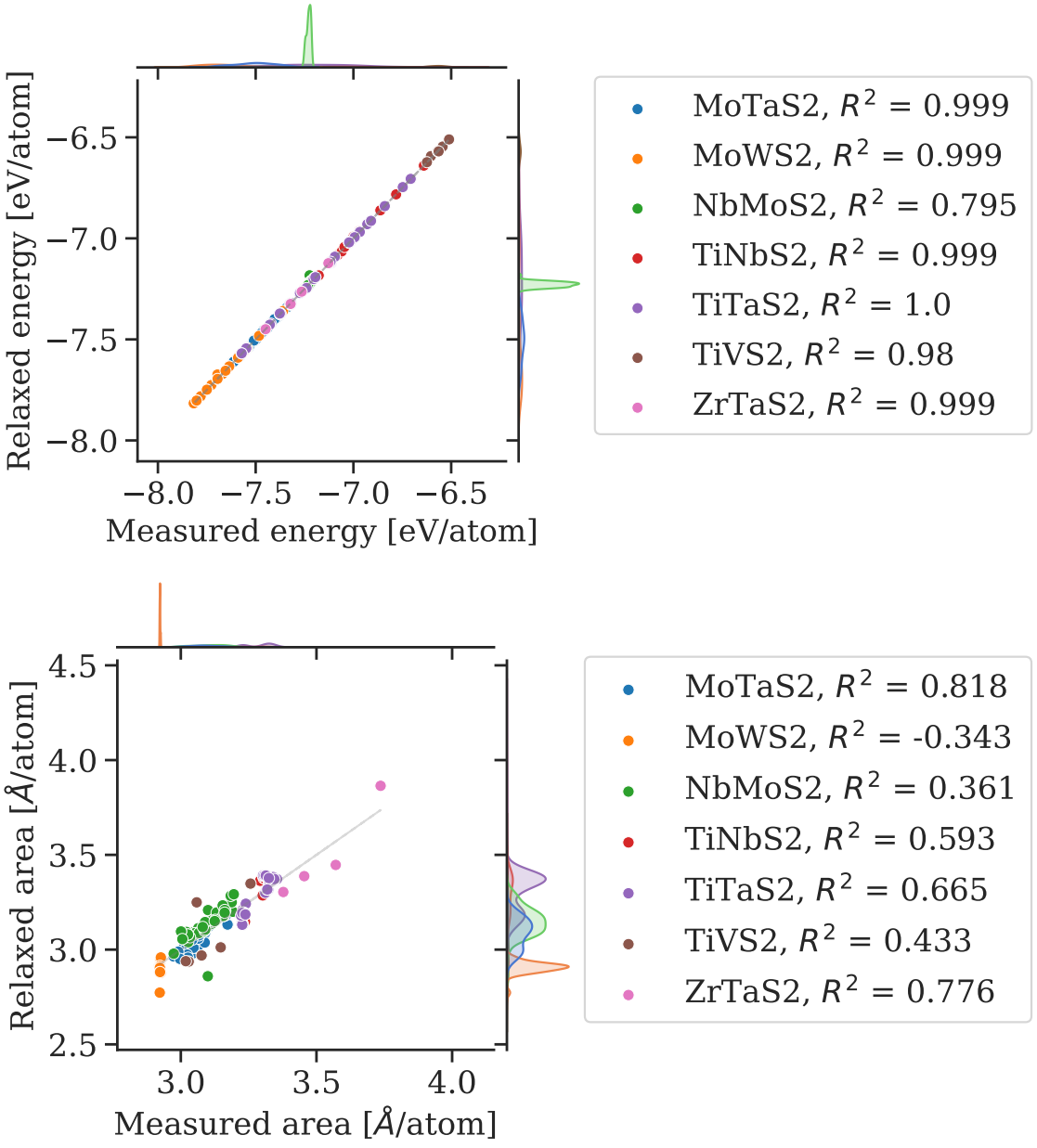


Figure 6.20: Error distribution and cumulative error histograms for MACE model with datasplit=801010, cutoff = 0.001, normal learning, lr=0.003, bs=16.



(a) Line plot for MACE with datasplit=801010, transfer training, cutoff = 0.001, lr = 0.0001, bs=16.

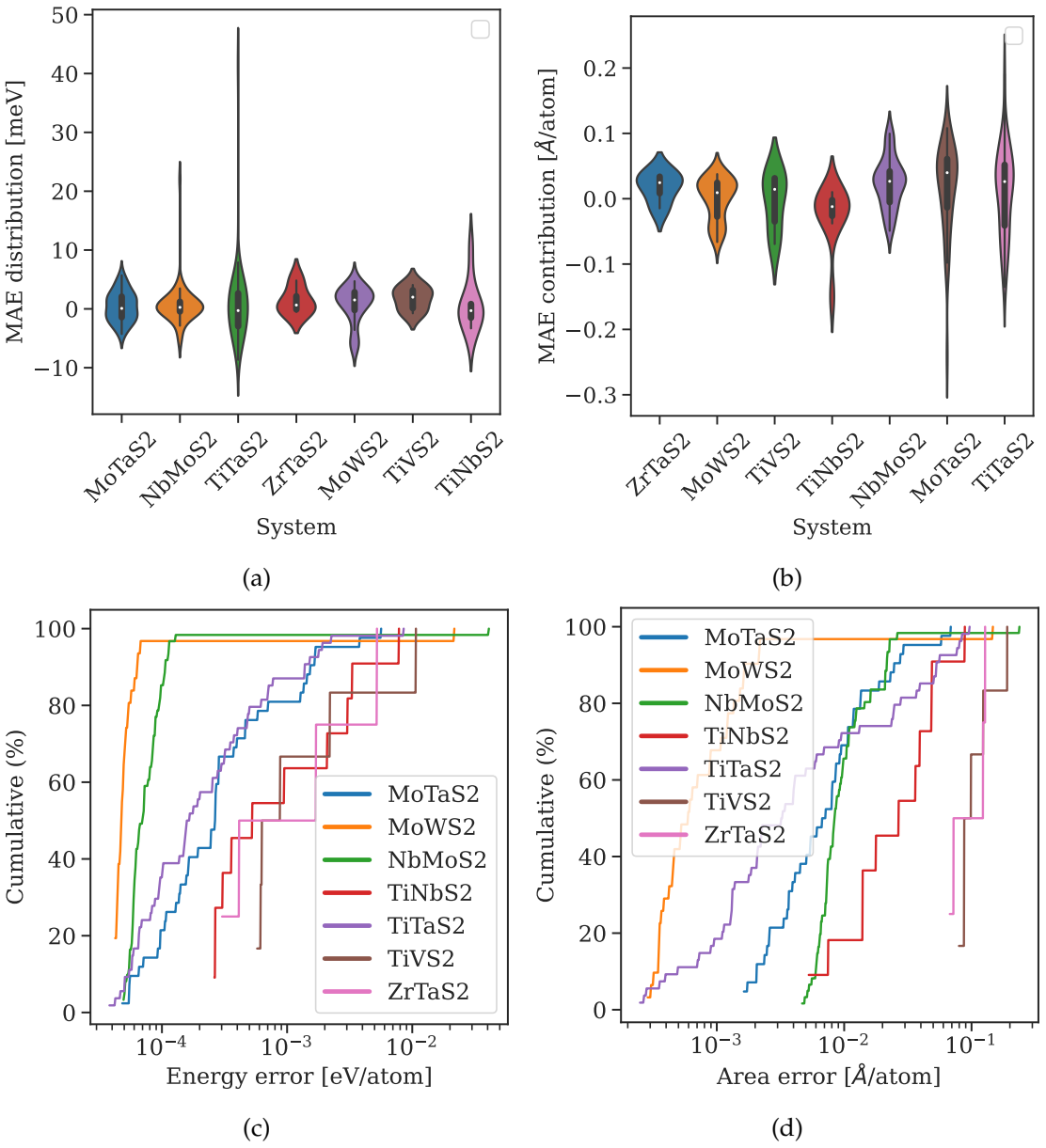
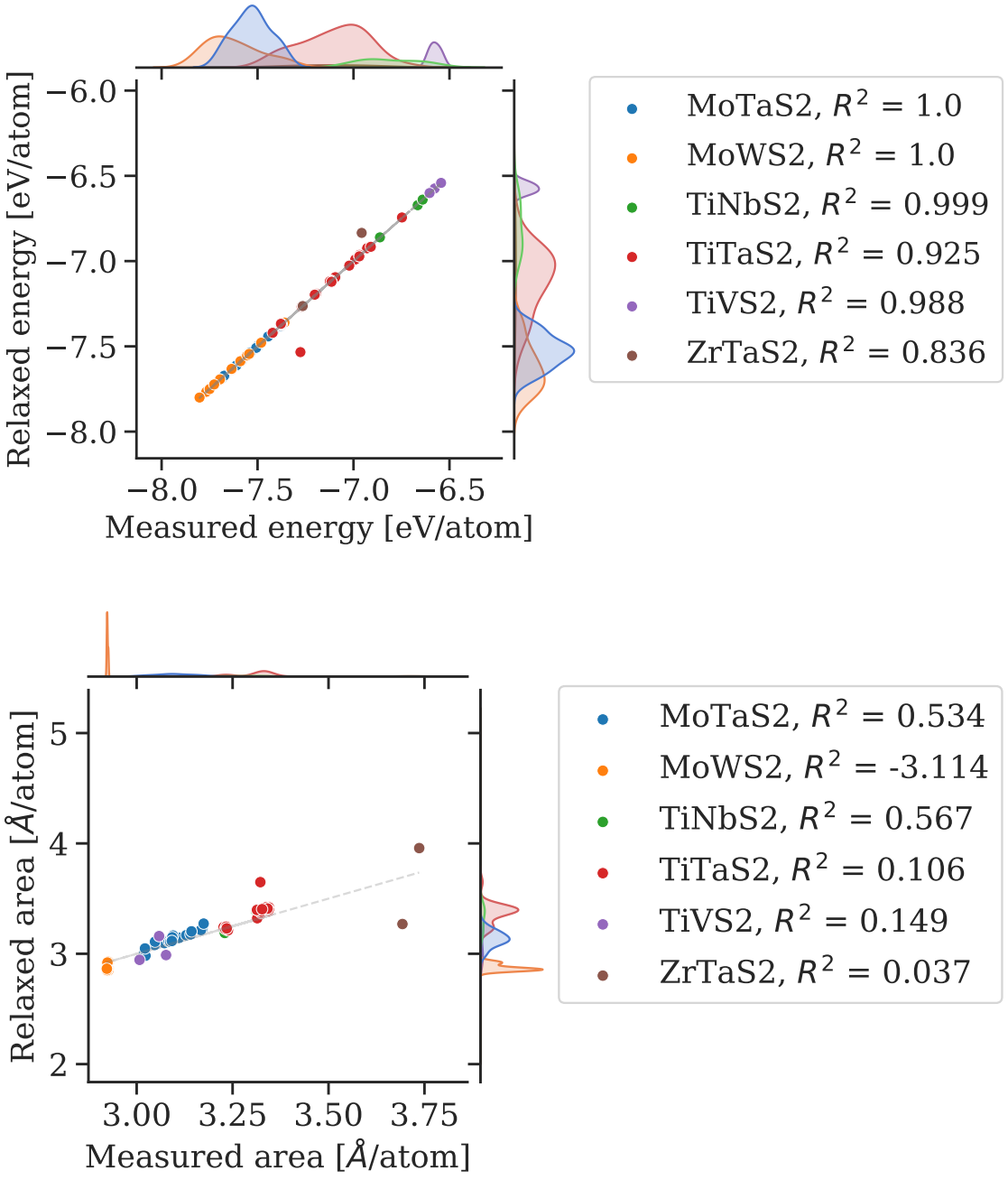


Figure 6.22: Error distribution and cumulative error histograms for MACE model with datasplit=801010, cutoff = 0.001, transfer learning, lr=0.0001, bs=16.



(a) Line plot for MACE with datasplit=9055, normal training, cutoff = 0.001, lr = 0.003, bs=16.

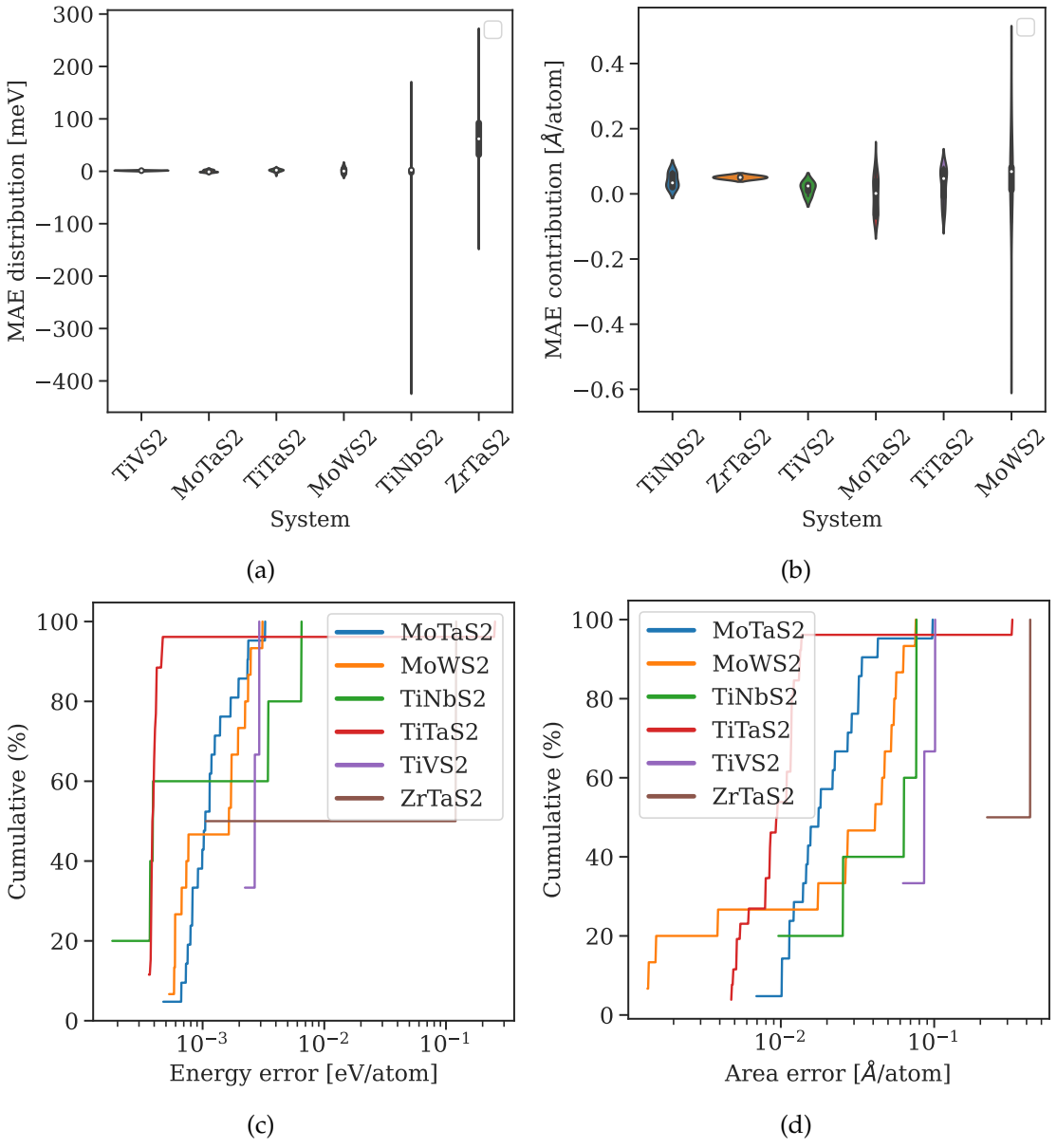
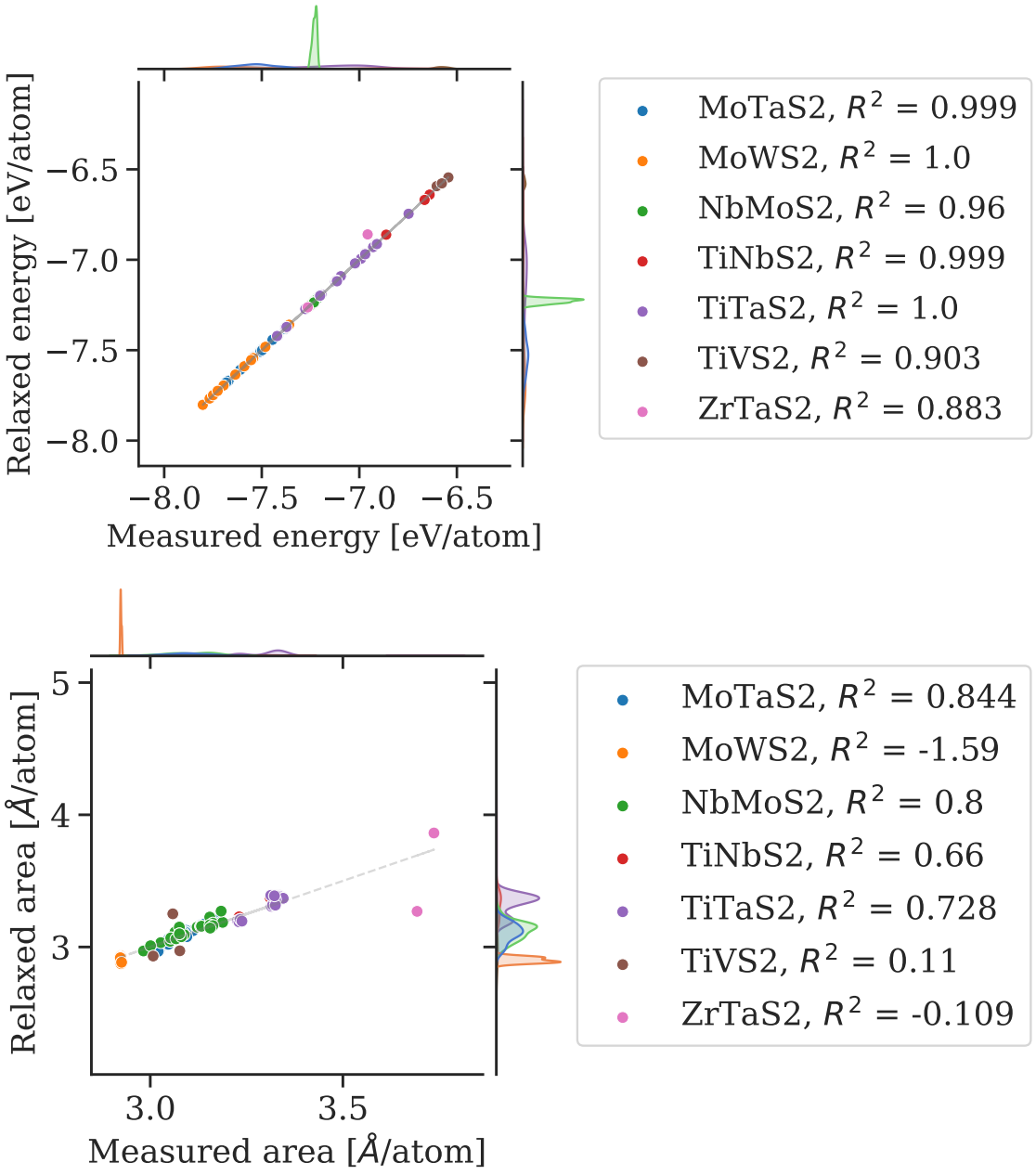


Figure 6.24: Error distribution and cumulative error histograms for MACE model with datasplit=9055, normal learning, lr=0.03, bs=16.



(a) Line plot for MACE with datasplit=9055, transfer training, cutoff = 0.001, lr = 0.0001, bs=16.

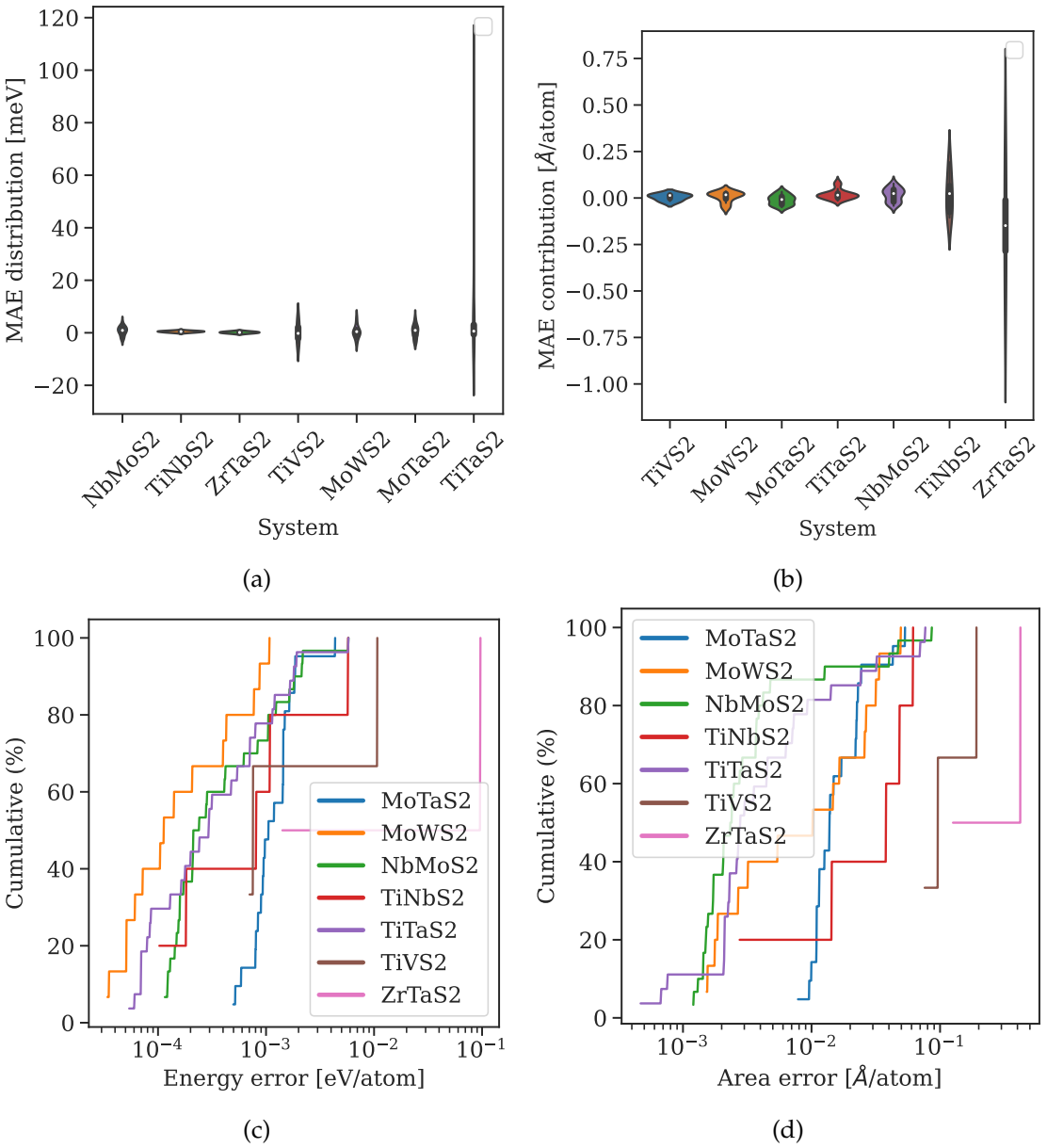


Figure 6.26: Error distribution and cumulative error histograms for MACE model with datasplit=9055, transfer learning, lr=0.03, bs=16.