Master's thesis

Electrostatic Interactions in Neural-Network Force Fields

Carried out at the

Institute of Materials Chemistry

Under supervision of

Univ.Prof. Georg K. H. Madsen, PhD

and

Senior Scientist Jesús Carrete Montaña, PhD

By

Johannes Schörghuber, BSc

Vienna, 9th May, 2023

Johannes Schörghuber, BSc

Abstract

Neural-network force fields provide a computationally efficient, flexible and transferable way to compute forces and further quantities with ab-initio-like accuracy, for example for use in high-throughput applications such as molecular dynamics. Such models often work in part by encoding the absolute coordinates using descriptors which capture the immediate environment of an atom [1]. These are in most cases defined via a cutoff radius, making them local by definition. While such networks already produce excellent results using these descriptors [2, 3, 4], an accurate modelling of long-range interactions is desirable to move towards more physical and transferable models of the systems of interest [5].

In the present work, a methodology to model electrostatic interactions using a neural-network force field has been developed. Training data that represents only the electrostatic interactions between the atoms in the system is generated via density functional theory calculations and the performance impact of different training data choices are explored. Atomic charges are predicted using a modified version of the electron-passing neural-network introduced in Ref. [6], which is implemented to be combined with the neural-network force field NeuralIL [4]. The applicability of the approach is demonstrated on surface reconstructions and liquids. It is shown that atomic charges can be predicted to accurately reproduce the electrostatic energy of the system by training the network on the total electrostatic energies of the reference systems. However, the need for additional local training information for each atom is highlighted by the poor performance in force predictions when integrating models trained using this methodology with NeuralIL.

Zusammenfassung

Kraftfelder basierend auf neuronalen Netzwerken bieten eine effiziente, flexible und übertragbare Möglichkeit zur Berechnung von Kräften und weiterer Größen mit ab-initio-ähnlicher Genauigkeit, beispielsweise für den Einsatz in Anwendungen, die eine hohe Anzahl an Evaluationen erfordern, wie der Molekulardynamik. Oftmals werden in solchen Modellen die absoluten Positionen mithilfe von Deskriptoren, welche die unmittelbare Umgebung der Atome beschreiben, dargestellt [1]. Diese sind in den meisten Fällen über einen maximalen Grenzradius definiert und beschreiben daher nur das lokale Umfeld eines Atoms. Während Netzwerke basierend auf solchen Deskriptoren bereits Resultate mit sehr hoher Genauigkeit liefern [2, 3, 4], ist eine akkurate Beschreibung von weitreichenden Wechselwirkungen für physikalisch vollständigere und besser transferable Modelle von Interesse [5].

In der vorliegenden Arbeit wurde eine Methodik für die Beschreibung elektrostatischer Wechselwirkungen in Kraftfeldern basierend auf neuronalen Netzwerken entwickelt. Trainingsdaten, welche ausschließlich elektrostatische Interaktionen beschreiben, werden mithilfe von Dichtefunktionaltheorie generiert und der Einfluss unterschiedlicher Arten von Trainingsdaten untersucht. Atomare Teilladungen werden mithilfe einer modifizierten Version des electron-passing neural network [6], welche kompatibel mit NeuralIL [4] implementiert ist, modelliert. Die Anwendbarkeit wird anhand von Oberflächenrekonstruktionen und Flüssigkeiten demonstriert. Es wird gezeigt, dass atomare Teilladungen, welche die elektrostatischen Energien der untersuchten Systeme reproduzieren, mit hoher Genauigkeit berechenbar sind. Jedoch zeigt die Integration des Modells in NeuralIL, dass zusätzliche, lokale Trainingsinformationen notwendig sind, um auch genaue Kräfte berechnen zu können.

Contents

1 Introduction

Atomic-level simulations have become an integral tool in the understanding of atomic structure, transport phenomena and reactivity. While first-principles electronic structure methods provide highly accurate descriptions of a vast amount of systems, they are often limited to smaller systems and fewer configurations due to the high computational costs. To be able to facilitate the studies of more complicated systems such as alloys, surfaces and interfaces and to allow for the use of methods such as molecular dynamics and Monte Carlo simulations, a different way to model the potential energy hypersurface is needed.

A computationally more efficient alternative to ab-initio methods are force fields (FFs), which consider the atoms as point-like particles and model the energy and forces in a more classical fashion. Classical FFs, such as OPLS-AA [7] and ReaxFF [8], which also includes terms for accounting for phenomena such as polarization and is considered the state-of-the-art FF for molecular mechanics, are usually constructed via the parametrization of functional forms built up using physically inspired terms. However, these FFs require at times difficult parametrizations and are still limited in terms of scalability and transferability.

In recent years, machine-learning force fields have been introduced as a more general and flexible alternative to classical FF parametrizations [1, 2]. These have been shown to achieve DFT-like accuracy while keeping computational costs low [3]. However, neural-network force fields, which are constructed adhering to the architecture introduced in Ref. [1], do by definition not account for long-range interactions beyond a pre-defined cutoff radius. One type of long-range interactions that is not accounted for explicitly are electrostatic interactions between (charged) particles: Modelling of these interactions has been found to be important for a physically accurate description of a variety of phenomena in different systems, from the structure and dynamical behaviour of proteins [9] to interface properties [10] to water in different phases and configurations [11].

In this work, a way of integrating electrostatic interactions with the neural-network force field NeuralIL [4] is explored. The interactions are modelled explicitly by extending the total energy expression by a Coulomb term with charges predicted by a separate, auxiliary electron-passing neural network [6]. A modified version of this graph-based neural network is implemented in a programming framework compatible with NeuralIL. Different training data choices, methods to obtain training data from density functional theory calculations, and their impact on the performance of the augmented NeuralIL model are investigated to move towards a more phyiscally accurate description of atomic systems.

2 The Projector Augmented Wave (PAW) Method

2.1 Density Functional Theory

The central equation in electronic structure calculations is the many-body problem, which refers to the task of solving the Schrödinger equations for a system containing multiple particles. The time-independent Schrödinger equation is given by

$$\hat{H}|\Psi\rangle = E|\Psi\rangle \tag{2.1}$$

Denoting electrons with the subscripts e and e' and nuclei with the subscripts a and a', the Hamiltonian $\hat{H}$ is defined, in an atomic system of units with the electronic charge e = 1, the electron mass $m_e = 1$ and $\hbar = 1$, as

$$\hat{H} = -\frac{1}{2}\sum_{e}\nabla^2 + \frac{1}{2}\sum_{e\neq e'}\frac{1}{|\boldsymbol{r}_e - \boldsymbol{r}_{e'}|} - \sum_{ea}\frac{\mathcal{Z}_a}{|\boldsymbol{r}_e - \boldsymbol{R}_a|} + \frac{1}{2}\sum_{a\neq a'}\frac{\mathcal{Z}_a\mathcal{Z}_{a'}}{|\boldsymbol{R}_a - \boldsymbol{R}_{a'}|} \tag{2.2}$$

Density Functional Theory (DFT) is an approach for expressing the energy of an interacting many-body system introduced by Hohenberg and Kohn in 1964 [12]. The method deals with the exploding complexity of the problem by reformulating it in terms of a functional of the electron density $n(\boldsymbol{r})$ instead of treating each particle explicitly and is exact in its most general form, as shown in Ref. [12]. DFT is based on two central concepts: First, under the Born-Oppenheimer approximation, the nuclei parametrize an external potential $v_{\text{ext}}(\boldsymbol{r}) = -\sum_a \frac{\mathcal{Z}_a}{|\boldsymbol{r}-\boldsymbol{R}_a|}$, which uniquely determines the ground state density $n_0(\boldsymbol{r})$ [12]. (The nucleus-nucleus interaction can be seen as an additive constant to the energy for a given set of atomic positions.) The Born-Oppenheimer approximation assumes an instantaneous response of the electrons to a change in the positions of the nuclei, justified by the significantly heavier nature of the nuclei compared to the electrons. Second, the ground state electronic density can be determined by minimizing an energy functional $E[n]$.

However, this requires the explicit form of the functional $E[n]$ to be known exactly, which is almost never the case. A self-consistent, iterative formulation of DFT, which lends itself to finding well-defined approximations for practical calculations, was introduced by Kohn and Sham in 1965 [13]. The resulting Kohn-Sham Density Functional Theory approach (KS-DFT) will be the focus of this section. It is mentioned that while methods to solve the time-dependent problem (TDDFT methods) are also in use [14], only the time-independent problem is considered here.

In KS-DFT, the interacting many-body problem, the solution of which is denoted by the wave function $|\Psi\rangle$, is expressed as a system of non-interacting electrons characterized by the wave functions $|\psi_n\rangle$. The density is formulated in terms of the non-interaction wave functions as $n(\boldsymbol{r}) = \sum_n |\psi_n(\boldsymbol{r})|^2$, where $\psi_n(\boldsymbol{r}) = \langle \boldsymbol{r}|\psi_n\rangle$. Using this expression for the density, the KS energy functional is then defined as

$$E[n] = T_s[\{\psi_n\}] + E_H[n] + \int v_{\text{ext}}(\boldsymbol{r})n(\boldsymbol{r})d^3\boldsymbol{r} + E_{xc}[n] \tag{2.3}$$

where

$$T_s[\{\psi_n\}] = \sum_n \langle\psi_n| - \frac{\nabla^2}{2}|\psi_n\rangle \tag{2.4}$$

denotes the non-interacting kinetic energy and

$$E_H[n] = \frac{1}{2}\int \frac{n(\boldsymbol{r})n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' \tag{2.5}$$

denotes the Hartree energy. The functional E_{xc} is referred to as the exchange-correlation functional and accounts for the error introduced by the reformulation in terms of non-interacting electrons. Thus, it restores the electron-electron interaction energy of the interacting system $E_{ee} = \frac{1}{2}\sum_{e\neq e'} |\boldsymbol{r}_e - \boldsymbol{r}_{e'}|^{-1}$ and the kinetic energy of the interacting system T. Formally, E_{xc} is therefore defined as

$$E_{xc} = T - T_s + E_{ee} - E_H \tag{2.6}$$

With the functionals defined, the ground state of the system is then determined by solving the KS equations (2.7). While these equations are in theory exact, the form of E_{xc} is not known in the general case and has to be approximated. There exists a broad range of ways to construct exchange-correlation functionals, details of which will not be discussed in this work.

$$\left[-\frac{1}{2}\nabla^2 + \int \frac{n(\boldsymbol{r})}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}' + v_{\text{ext}}(\boldsymbol{r}) + \frac{\delta E_{\text{xc}}}{\delta n}(\boldsymbol{r})\right]\psi_n(\boldsymbol{r}) = \varepsilon_n\psi_n(\boldsymbol{r}) \tag{2.7}$$

While equation (2.7) describes a practical way to obtain ground-state solutions, challenges related to the implementation in software still arise. The wave functions are

required to be orthogonal, which in turn leads to highly oscillatory behaviour close to the nuclei. This in turn requires a large set of plane wave basis functions to achieve a sufficiently accurate representation. Multiple approaches to solve this problem exist:

Pseudo-potential methods rely on a smoother potential representation for the core electrons to avoid the oscillatory behaviour. The KS equations are then only solved for the valence electrons [15]. While this greatly reduces computational cost, it relies on finding a suitable parametrization of the pseudo potential for accurate results. Furthermore, even if an accurate result is obtained, information about the all-electron wave function is lost in the regions surrounding the nuclei.

The (linearized) Augmented Plane Wave method ((L-)APW) provides a different approach by modifying the plane wave basis set in the core regions. In these regions the wave functions are modelled as partial waves, which are then matched with the plane waves at the core-interstitial boundary.

Another framework, which will be discussed in detail in the following sections, is given by the Projector Augmented Wave (PAW) method first introduced by Blöchl [16], of which the pseudo-potential method has been shown to be a well-defined approximation [17].

2.2 The PAW Transformation

As discussed in section 2.1, the main problem regarding solving the KS equations for all electrons lies in the highly oscillatory behaviour close to the nuclei. The central idea of the PAW method lies in the definition of a transformation $\hat{\mathcal{T}}$ that maps a smooth auxiliary wave function $|\tilde{\psi}_n\rangle$ to the all-electron single particle wave function $|\psi_n\rangle$, which yields a transformed set of equations that take the place of the KS equations given in equation (2.7).

$$\hat{\mathcal{T}}^* \hat{H} \hat{\mathcal{T}} |\tilde{\psi}_n\rangle = \varepsilon_n \hat{\mathcal{T}}^* \hat{\mathcal{T}} |\tilde{\psi}_n\rangle \tag{2.8}$$

It is desirable for the PAW transformation operator $\hat{\mathcal{T}}$ to be linear and, since the wave functions are well-behaved away from the nuclei, reduce to an identity transformation outside a cutoff radius r_c^a away from the nucleus a, as given in equation (2.9). The cutoff radii should be defined in such a way that the augmentation spheres do not overlap.

$$\hat{\mathcal{T}} = 1 + \sum_a \hat{\mathcal{T}}^a \quad \text{with} \quad \hat{\mathcal{T}}^a = 0 \quad \text{for} \quad |\boldsymbol{r} - \boldsymbol{R}^a| \geq r_c^a \tag{2.9}$$

In the region of distances smaller than the cutoff radius r_c^a the all-electron wave function is expanded using a set of partial waves $\{\phi_i^a\}$. The choice of the functional form of these waves is a parameter for the implementation: A natural choice for molecular systems are the solutions of the radial Schrödinger equation for isolated atoms [16]. Similar to the treatment of the full wave function there is a set of smooth partial waves $\{\tilde{\phi}_i^a\}$ which transform according to

$$|\phi_i^a\rangle = (1 + \hat{\mathcal{T}}^a)|\tilde{\phi}_i^a\rangle \quad \forall\, i, a \tag{2.10}$$

For given sets of (smooth) partial waves $\{\phi_i\}$ and $\{\tilde{\phi}_i\}$ this yields a complete definition of the PAW transformation operator $\hat{\mathcal{T}}$. If the set $\{\tilde{\phi}_i\}$ is complete inside the augmentation region this allows for the expansion of the smooth wave function according to

$$|\tilde{\psi}_n\rangle = \sum_i P_{ni}^a |\tilde{\phi}_i\rangle \quad \text{for} \quad |\boldsymbol{r} - \boldsymbol{R}^a| < r_c^a \tag{2.11}$$

Due to equation (2.10) the expansion of the full all-electron wave function $|\psi_n\rangle$ via the set $\{\phi_i\}$ takes the same form as equation (2.11) with the same expansion coefficients P_{ni}^a. The linearity requirement for $\hat{\mathcal{T}}$ leads to the expansion coefficients being linear functionals of the smooth wave function $|\tilde{\psi}_n\rangle$.

$$P_{ni}^a = \langle \tilde{p}_i^a | \tilde{\psi}_n \rangle \tag{2.12}$$

where $\langle \tilde{p}_i^a|$ are the so-called projector functions, which have to fulfill the completeness condition given in equation (2.13) in order for the equality $\sum_i |\tilde{\phi}_i^a\rangle\langle\tilde{p}_i^a|\tilde{\psi}_n\rangle = |\tilde{\psi}_n\rangle$ to hold inside the augmentation regions.

$$\sum_i |\tilde{\phi}_i^a\rangle\langle\tilde{p}_i^a| = 1 \quad \text{for} \quad |\boldsymbol{r} - \boldsymbol{R}^a| < r_c^a \tag{2.13}$$

This further implies a duality relation between the projector functions and the smooth partial waves:

$$\langle \tilde{p}_{i_1}^a | \tilde{\phi}_{i_2}^a \rangle = \delta_{i_1 i_2} \quad \text{for} \quad |\boldsymbol{r} - \boldsymbol{R}^a| < r_c^a \tag{2.14}$$

Summarizing, the definitions of (smooth) partial waves and projector functions in turn fully define a linear transformation operator

$$\hat{\mathcal{T}} = 1 + \sum_a \sum_i (|\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle)\langle \tilde{p}_i^a| \tag{2.15}$$

with which the full, all-electron wave function $|\psi_n\rangle$ can be expressed as

$$|\psi_n\rangle = |\tilde{\psi}_n\rangle + \sum_a \sum_i (|\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle)\langle \tilde{p}_i^a|\tilde{\psi}_n\rangle \tag{2.16}$$

where the sets of (smooth) partial waves $\{\phi_i\}$ and $\{\tilde{\phi}_i\}$ and the set of projector functions $\{\tilde{p}_i\}$ are independent of the system and can thus be precalculated. The transformation allows for a full reconstruction of the all-electron wave function $|\psi_n\rangle$ from the smooth auxiliary wave functions $|\tilde{\psi}_n\rangle$, which is found by solving the transformed Kohn-Sham equations. This overcomes the problem of the highly oscillatory behaviour near the nuclei when solving while also avoiding a loss of information in the core regions, as is the case with pseudopotential methods.

While the transformation gives a well defined mapping of the smooth and all-electron wave functions, a central approximation is customary to obtain a computationally efficient method: The frozen core approximation assumes a localization of the core states within the augmentation spheres and that they are not altered by the formation or breaking of chemical bonds. This has been found to be a valid approximation that only has a small effect on accuracy in Ref. [18]. As a consequence the core Kohn-Sham states are directly given by the partial waves in that region:

$$|\psi_n^{a,\text{core}}\rangle = |\phi_\alpha^{a,\text{core}}\rangle \tag{2.17}$$

Extensions of the PAW method beyond the frozen core approximation are possible, but often forgone in implementations such as GPAW due to the high accuracy of calculations within the approximation [19, 20].

2.3 (Semi-)Local Operators

As discussed in section 2.2, the smooth auxiliary wave function takes the role of the variational parameter in PAW, wherefore expressions for the expectation values of operators have to be reformulated in terms of $|\tilde{\psi}_n\rangle$. In general an expectation value of an operator $\hat{A}$ in the frozen core approximation is given by

$$\langle \hat{A} \rangle = \sum_{n}^{\text{valence}} f_n \langle \psi_n | \hat{A} | \psi_n \rangle + \sum_{a} \sum_{\alpha}^{\text{core}} \langle \phi_\alpha^{a,\text{core}} | \hat{A} | \phi_\alpha^{a,\text{core}} \rangle \tag{2.18}$$

where f_n denotes the occupation of a state n. Introducing the PAW transformation defined in equation (2.15) to expand $|\psi_n^a\rangle$ using partial waves yields

$$\sum_{n}^{\text{valence}} f_n \langle \psi_n^a | \hat{A} | \psi_n^a \rangle = \sum_{i_1 i_2} \langle \phi_{i_1}^a | \hat{A} | \phi_{i_2}^a \rangle \sum_{n}^{\text{valence}} f_n P_{n i_1}^{a*} P_{n i_2}^a \tag{2.19}$$

The last sum in equation (2.19) is defined as the one-center density matrix $D_{i_1 i_2}^a$ in equation (2.20).

$$D_{i_1 i_2}^a = \sum_n f_n P_{n i_1}^{a*} P_{n i_2}^a = \sum_n f_n \langle \tilde{\psi}_n | \tilde{p}_{i_1}^a \rangle \langle \tilde{p}_{i_2}^a | \tilde{\psi}_n \rangle \tag{2.20}$$

Finally, over the whole region the above transformations yield equation (2.21) as a general expression for the expectation value of a (semi-)local operator depending on the smooth auxiliary wave function.

$$\begin{aligned} \langle \hat{A} \rangle = & \sum_{n}^{\text{valence}} f_n \langle \tilde{\psi}_n | \hat{A} | \tilde{\psi}_n \rangle \\ & + \sum_a \sum_{i_1 i_2} (\langle \phi_{i_1}^a | \hat{A} | \phi_{i_2}^a \rangle - \langle \tilde{\phi}_{i_1}^a | \hat{A} | \tilde{\phi}_{i_2}^a \rangle) D_{i_1 i_2}^a \\ & + \sum_a \sum_{\alpha}^{\text{core}} \langle \phi_\alpha^{a,\text{core}} | \hat{A} | \phi_\alpha^{a,\text{core}} \rangle \end{aligned} \tag{2.21}$$

This directly yields an expression for the expectation value of the real-space projection operator $|\boldsymbol{r}\rangle\langle\boldsymbol{r}|$, which gives the electron density $n(\boldsymbol{r})$:

$$n(\boldsymbol{r}) = \sum_{n}^{\text{valence}} f_n |\tilde{\psi}_n|^2 + \sum_a \sum_{i_1 i_2} (\phi_{i_1}^a \phi_{i_2}^a - \tilde{\phi}_{i_1}^a \tilde{\phi}_{i_2}^a) D_{i_1 i_2}^a + \sum_a \sum_{\alpha}^{\text{core}} |\phi_\alpha^{a,\text{core}}|^2 \tag{2.22}$$

In practice, a smooth core density $\tilde{n}_c^a(\boldsymbol{r})$, which is equal to the core density $n_c^a(\boldsymbol{r})$ for $|\boldsymbol{r} - \boldsymbol{R}^a| \geq r_c^a$, is constructed [16, 19, 20]. This gives the one-center expansions

$$
\begin{aligned}
n^a(\boldsymbol{r}) &= \sum_{i_1 i_2} \phi^a_{i_1}(\boldsymbol{r}) \phi^a_{i_2}(\boldsymbol{r}) D^a_{i_1 i_2} + n^a_c(\boldsymbol{r}) \\
\tilde{n}^a(\boldsymbol{r}) &= \sum_{i_1 i_2} \tilde{\phi}^a_{i_1}(\boldsymbol{r}) \tilde{\phi}^a_{i_2}(\boldsymbol{r}) D^a_{i_1 i_2} + \tilde{n}^a_c(\boldsymbol{r})
\end{aligned}
\tag{2.23}
$$

which allow for expressing the all-electron density as a sum of a smooth contribution evaluated on the whole space and atom-centered corrections.

$$
n(\boldsymbol{r}) = \tilde{n}(\boldsymbol{r}) + \sum_a (n^a(\boldsymbol{r}) - \tilde{n}^a(\boldsymbol{r})) \quad \text{with} \quad \tilde{n}(\boldsymbol{r}) = \sum_n^{\text{valence}} f_n |\tilde{\psi}_n(\boldsymbol{r})|^2 + \sum_a \tilde{n}^a_c(\boldsymbol{r}) \tag{2.24}
$$

Similarly to the density, the expectation value of the kinetic energy operator can also be expressed as a sum of smooth contributions and atom-centered corrections using equation (2.21):

$$
T_s[\{\psi_n\}] = \sum_n^{\text{valence}} f_n \langle \tilde{\psi}_n | -\frac{1}{2}\nabla^2 | \tilde{\psi}_n \rangle + \sum_a \Delta T^a_s[\{D^a_{i_1 i_2}\}] \tag{2.25}
$$

with the local corrections

$$
\begin{aligned}
\Delta T^a_s[\{D^a_{i_1 i_2}\}] = &\sum_{i_1 i_2} (\langle \phi^a_{i_1} | -\frac{1}{2}\nabla^2 | \phi^a_{i_2} \rangle - \langle \tilde{\phi}^a_{i_1} | -\frac{1}{2}\nabla^2 | \tilde{\phi}^a_{i_2} \rangle) D^a_{i_1 i_2} \\
&+ \sum_a \sum_\alpha^{\text{core}} \langle \phi^{a,\text{core}}_\alpha | -\frac{1}{2}\nabla^2 | \phi^{a,\text{core}}_\alpha \rangle
\end{aligned}
\tag{2.26}
$$

The exchange-correlation energy for (semi-)local XC functionals such as LDA or GGA-type functionals can, at least conceptually, be obtained in an analogous way using equation (2.21) [19, 20].

2.4 Electrostatic Energy

Due to the non-linear and non-local nature of the Hartree term, the introduction of the PAW transformation requires a more involved derivation in this case, starting from the classical electrostatic energy given in equation (2.27).

$$E_C[n] = \frac{1}{2}\int \frac{n(\boldsymbol{r})n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' + \int \frac{n(\boldsymbol{r})\sum_a Z^a(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' + \frac{1}{2}\sum_{a\neq a'}\int \frac{Z^a(\boldsymbol{r})Z^{a'}(\boldsymbol{r})}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' \tag{2.27}$$

where $Z^a(\boldsymbol{r}) = -\mathcal{Z}^a\delta(\boldsymbol{r}-\boldsymbol{R}^a)$ is the charge density of the nucleus a with $\mathcal{Z}$ denoting the atomic number. This expression does not include nuclear self-interaction, which is the case for the Hartree energy formulated in terms of the charge density

$$U_H[\rho] = \frac{1}{2}\int \frac{\tilde{\rho}(\boldsymbol{r})\tilde{\rho}(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' + \sum_a \frac{1}{2}\int \frac{\rho^a(\boldsymbol{r})\rho^a(\boldsymbol{r}') - \tilde{\rho}^a(\boldsymbol{r})\tilde{\rho}^a(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d^3\boldsymbol{r}d^3\boldsymbol{r}' \tag{2.28}$$

with the charge density $\rho(\boldsymbol{r})$ being defined in equation (2.29). This quantity is introduced due to numerical problems with calculating the Hartree energy of a density with non-zero local charge. An exact definition and more detailed discussion of the charge density ρ and its' smooth counterpart $\tilde{\rho}$ is given in the following paragraphs.

$$\rho(\boldsymbol{r}) = n(\boldsymbol{r}) + \sum_a Z^a(\boldsymbol{r}) \tag{2.29}$$

Evaluating $U_H[\rho]$ with the above defined charge density is computationally infeasible: It would entail a double summation over all nuclei, as in the last term of equation (2.27), which is $\mathcal{O}(N^2_{\text{atoms}})$ in a direct implementation. Furthermore, terms lying on different grids (radial grids for the atom-centered terms and equally spaced coarse grids for the smooth terms) would have to be treated in the same integrals. The solution lies in the definition of one-center all-electron and smooth charge densities [20], as given in equation (2.30).

$$\begin{aligned}\rho^a(\boldsymbol{r}) &= n^a(\boldsymbol{r}) - \mathcal{Z}^a\delta(\boldsymbol{r}-\boldsymbol{R}^a)\\ \tilde{\rho}^a(\boldsymbol{r}) &= \tilde{n}^a(\boldsymbol{r}) + \sum_{lm} Q^a_{lm}\tilde{g}^a_{lm}(\boldsymbol{r})\end{aligned} \tag{2.30}$$

where $\tilde{g}^a_{lm}(\boldsymbol{r})$ are functions localized in the augmentation sphere of atom a, which are normalized according to

$$\int r^l Y_{lm}(\widehat{\boldsymbol{r}-\boldsymbol{R}^a})\tilde{g}^a_{l'm'}(\boldsymbol{r})d^3\boldsymbol{r} = \delta_{ll'}\delta_{mm'} \quad \text{for} \quad |\boldsymbol{r}-\boldsymbol{R}^a| < r^a_c \tag{2.31}$$

The compensation charges in the last term of $\tilde{\rho}^a(\boldsymbol{r})$ in equation (2.30) are defined in such a way that the difference between the all-electron charge density $\rho(\boldsymbol{r})$ and the smooth charge density $\tilde{\rho}(\boldsymbol{r})$ has multipole moments equal to zero, in order to electrostatically decouple the augmentation spheres [20]. This fully defines the expansion coefficients Q^a_{lm} by equation (2.32).

$$Q^a_{lm} = \int r^l Y_{lm}(\hat{\boldsymbol{r}}) \left[n^a(\boldsymbol{r}) - \tilde{n}^a(\boldsymbol{r}) + Z^a(\boldsymbol{r})\right] d^3\boldsymbol{r} = \Delta^a \delta_{l0} + \sum_{i_1 i_2} \Delta^a_{lm,i_1 i_2} D^a_{i_1 i_2} \tag{2.32}$$

with

$$\begin{aligned} \Delta^a &= \int Y_{00}(\hat{\boldsymbol{r}}) \left[n^a_c(\boldsymbol{r}) - \tilde{n}^a_c(\boldsymbol{r}) - \mathcal{Z}^a \delta(\boldsymbol{r})\right] d^3\boldsymbol{r} \\ \Delta^a_{lm,i_1 i_2} &= \int r^l Y_{lm}(\hat{\boldsymbol{r}}) \left[\phi^a_{i_1}(\boldsymbol{r})\phi^a_{i_2}(\boldsymbol{r}) - \tilde{\phi}^a_{i_1}(\boldsymbol{r})\tilde{\phi}^a_{i_2}(\boldsymbol{r})\right] d^3\boldsymbol{r} \end{aligned} \tag{2.33}$$

The compensation charge densities $\tilde{Z}^a(\boldsymbol{r}) = \sum_{lm} Q^a_{lm} \tilde{g}^a_{lm}(\boldsymbol{r})$ are thus also fully defined, so a smooth total charge density can be formulated.

$$\tilde{\rho}(\boldsymbol{r}) = \tilde{n}(\boldsymbol{r}) + \sum_a \sum_{lm} Q^a_{lm} \tilde{g}^a_{lm}(\boldsymbol{r}) \tag{2.34}$$

which in turn yields the expression $\rho(\boldsymbol{r}) = \tilde{\rho}(\boldsymbol{r}) + \sum_a \left[\rho^a(\boldsymbol{r}) - \tilde{\rho}^a(\boldsymbol{r})\right]$ for the all-electron charge density. With the definition of the all-electron and smooth charge densities $\rho(\boldsymbol{r})$ and $\tilde{\rho}(\boldsymbol{r})$ as well as the atom-centered forms $\rho^a(\boldsymbol{r})$ and $\tilde{\rho}^a(\boldsymbol{r})$ the electrostatic energy can finally be evaluated using equation (2.28).

The term in the last sum over all atoms a can again be expressed as an atom-centered correction tensor $\Delta E^a_C[\{D^a_{i_1 i_2}\}]$, analogous to the expressions in section 2.3. For purposes of a compact notation of the tensor the notation $(f|g)$ is used to denote the Coulomb integral for the functions $f(\boldsymbol{r})$ and $g(\boldsymbol{r})$ and $((f)) = (f|f)$:

$$(f|g) = \int \frac{f^*(\boldsymbol{r}) g(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} d^3\boldsymbol{r} d^3\boldsymbol{r}' \tag{2.35}$$

$$
\begin{aligned}
\Delta E_C^a[\{D_{i_1 i_2}^a\}] = & \frac{1}{2}((n^a)) + (n^a|Z^a) - \frac{1}{2}((\tilde{n}^a)) - (\tilde{n}^a|\tilde{Z}^a) - \frac{1}{2}((\tilde{Z}^a)) \\
= & \frac{1}{2}[((n_c^a)) - ((\tilde{n}_c^a))] - \mathcal{Z}^a \int \frac{n_c^a(\boldsymbol{r})}{r} d\boldsymbol{r} - \sum_{lm} Q_{lm}^a (\tilde{n}_c^a | \tilde{g}_{lm}^a) \\
& + \sum_{i_1 i_2} D_{i_1 i_2}^{a*} \Bigg[(\phi_{i_1}^a \phi_{i_2}^a | n_c^a) - (\tilde{\phi}_{i_1}^a \tilde{\phi}_{i_2}^a | \tilde{n}_c^a) \\
& \qquad - \mathcal{Z}^a \int \frac{\phi_{i_1}^a(\boldsymbol{r}) \phi_{i_2}^a(\boldsymbol{r})}{r} d\boldsymbol{r} - \sum_{lm} Q_{lm}^a (\tilde{\phi}_{i_1}^a \tilde{\phi}_{i_2}^a | \tilde{g}_{lm}^a) \Bigg] \\
& + \frac{1}{2} \sum_{i_1 i_2 i_3 i_4} D_{i_1 i_2}^{a*} \left[(\phi_{i_1}^a \phi_{i_2}^a | \phi_{i_3}^a \phi_{i_4}^a) - (\tilde{\phi}_{i_1}^a \tilde{\phi}_{i_2}^a | \tilde{\phi}_{i_3}^a \tilde{\phi}_{i_4}^a) \right] D_{i_3 i_4}^a \\
& - \frac{1}{2} \sum_{lml'm'} Q_{lm}^a Q_{l'm'}^a (\tilde{g}_{lm}^a | \tilde{g}_{l'm'}^a)
\end{aligned}
\tag{2.36}
$$

where all terms in ΔE_C^a only need to be treated on the radial atom-centered grids and not the full space.

2.5 The Kohn-Sham Equations in PAW

In sections 2.3 and 2.4 a framework was derived to evaluate the different energy contributions to the total energy of an atomic system based on the smooth expression for the densities $\tilde{n}(\boldsymbol{r})$ and $\tilde{\rho}(\boldsymbol{r})$ and thus the underlying Kohn-Sham wave functions $|\tilde{\psi}_n\rangle$. These quantitites are obtained by solving the transformed Kohn-Sham equation (2.8). With the results of the previous sections it is now possible to fully define the terms $\hat{\tilde{H}} = \mathcal{T}^* \hat{H} \mathcal{T}$ and $\hat{S} = \mathcal{T}^* \mathcal{T}$.

Using the definition of the PAW transformation in equation (2.15) the overlap operator $\hat{S}$ is expressed directly via equation (2.37). Due to the transformation of the KS equations the original orthogonality condition $\langle \psi_n | \psi_m \rangle = \delta_{nm}$ does not hold for the smooth wave functions, which are orthogonal with respect to the aforementioned overlap operator instead $\langle \tilde{\psi}_n | \hat{S} | \tilde{\psi}_m \rangle = \delta_{nm}$.

$$
\begin{aligned}
\hat{S} &= \mathcal{T}^* \mathcal{T} \\
&= 1 + \sum_a \sum_{i_1 i_2} |\tilde{p}_{i_1}^a\rangle (\langle \phi_{i_1}^a | \phi_{i_2}^a \rangle - \langle \tilde{\phi}_{i_1}^a | \tilde{\phi}_{i_2}^a \rangle) \langle \tilde{p}_{i_2}^a | \\
&= 1 + \sum_a \sum_{i_1 i_2} |\tilde{p}_{i_1}^a\rangle \sqrt{4\pi} \Delta_{00, i_1 i_2}^a \langle \tilde{p}_{i_2}^a |
\end{aligned}
\tag{2.37}
$$

While the Hamiltonian could be similarly deduced by applying the PAW transformation to the operator directly, it is beneficial to take advantage of the expression for the electrostatic energy $U_H[\rho]$ derived in section 2.4. Using the relation

$$\frac{\partial E[n]}{\partial \langle \tilde{\psi}_n |} = f_n \hat{\tilde{H}} |\tilde{\psi}_n\rangle \tag{2.38}$$

for the Kohn-Sham total energy $E[n] = T_s[\{\psi_n\}] + U_H[\rho] + E_{xc}[n]$ and defining the sum of all local corrections to the energy contributions as $\Delta E^a = \Delta T_s^a + \Delta E_{xc}^a + \Delta E_C^a$ and inserting the energy expressions derived previously yields

$$\begin{aligned}
\frac{\delta E[n]}{\delta \tilde{\psi}_n^*(\boldsymbol{r})} =& \frac{\delta}{\delta \tilde{\psi}_n^*(\boldsymbol{r})} \left[T_s[\{\tilde{\psi}_n\}] + E_{xc}[\tilde{n}] + U_H[\tilde{\rho}] + \Delta E^a[\{D^a_{i_1 i_2}\}] \right] \\
=& \; f_n(-\frac{1}{2}\nabla^2)\tilde{\psi}_n(\boldsymbol{r}) \\
&+ \int f_n[v_{xc}[\tilde{n}](\boldsymbol{r}') + u_H[\tilde{\rho}](\boldsymbol{r}')]\delta(\boldsymbol{r} - \boldsymbol{r}')\tilde{\psi}_n(\boldsymbol{r}')d^3\boldsymbol{r}' \\
&+ \sum_a \sum_{i_1 i_2} f_n \left[\int u_H[\tilde{\rho}](\boldsymbol{r}') \sum_{lm} \Delta^a_{lm,i_1 i_2} \tilde{g}^a_{lm}(\boldsymbol{r}') + \frac{\delta \Delta E^a}{\delta D^a_{i_1 i_2}} d^3\boldsymbol{r}' \right] \tilde{p}^a_{i_1}(\boldsymbol{r}) P^a_{n i_2}
\end{aligned} \tag{2.39}$$

with $v_{xc}[\tilde{n}](\boldsymbol{r}) = \frac{\delta E_{xc}[\tilde{n}]}{\delta \tilde{n}(\boldsymbol{r})}$ being a (semi-)local exchange correlation potential. From this result the PAW transformed Hamiltonian can again be extracted in a form that involves only evaluation of smooth terms on the whole space and atom-centered corrections:

$$\hat{\tilde{H}} = T_s[\{\tilde{\psi}_n\}] + v_{xc}[\tilde{n}] + u_H[\tilde{\rho}] + \sum_a \sum_{i_1 i_2} |\tilde{p}^a_{i_1}\rangle \Delta H^a_{i_1 i_2} \langle \tilde{p}^a_{i_2}| \tag{2.40}$$

where the one-center correction tensor $\Delta H^a_{i_1 i_2}$ is given by

$$\Delta H^a_{i_1 i_2} = \sum_{lm} \Delta^a_{lm,i_1 i_2} \int u_H[\tilde{\rho}](\boldsymbol{r}) \tilde{g}^a_{lm}(\boldsymbol{r}) d\boldsymbol{r} + \frac{\delta \Delta E^a}{\delta D^a_{i_1 i_2}} \tag{2.41}$$

2.6 Forces

As discussed in Ref. [16], multiple components have to be considered when calculating the total forces acting on the nuclei: First, the gradient of $E[\tilde{n}]$ with respect to atomic positions (the Hellmann-Feynman forces). Second, a contribution due to a change in the all-electron wave function with respect to atomic positions for fixed smooth wave functions to account for the $\boldsymbol{R}^a$-dependency of the augmentations. Third, since the calculations are perfomed in the frozen core approximation, Pulay forces for the core electrons have to be considered [21].

When calculating forces due to a shift of the wave function it has to be ensured that the same subspace of the Hilbert spaced is spanned by the occupied wave functions, as well as the orthogonality condition not being violated. In the following, primed quantities represent linear expansions of the shifted wave functions.

The condition of conserving the subspace of the Hilbert space spanned allows expressing the shifted wave function via the linear combination

$$|\tilde{\psi}'_n\rangle = |\tilde{\psi}_n\rangle + \sum_m |\tilde{\psi}_m\rangle \Lambda_{mn} \tag{2.42}$$

Combining equation (2.42) with the orthogonality condition $\langle\tilde{\psi}'_n|\hat{S}'|\tilde{\psi}'_m\rangle = \delta_{nm}$ fully determines the expansion coefficients Λ_{nm} [16].

$$\Lambda_{nm} + \Lambda^*_{nm} = -\langle\tilde{\psi}_n|\nabla_{\boldsymbol{R}^a}\hat{S}|\tilde{\psi}_m\rangle \tag{2.43}$$

where $\nabla_{\boldsymbol{R}^a}$ denotes the gradient with respect to nuclear coordinates. Similar to the invariance of the total energy with respect to a unitary transformation, an arbitrary anti-symmetric matrix can be added to Λ [16]. A first-order Taylor expansion of the general form of the unitary matrix $\Lambda = \exp(B)$ with $B = -B^*$ (anti-Hermitian) yields $\Lambda = 1 + B$. From the aforementioned invariance of the total energy it thus follows that the forces are also invariant with respect to the anti-symmetric part of Λ [16], wherefore

$$\nabla_{\boldsymbol{R}^a}|\tilde{\psi}_n\rangle = -\frac{1}{2}\sum_m |\tilde{\psi}_m\rangle \left[\langle\tilde{\psi}_m|\nabla_{\boldsymbol{R}^a}\hat{S}|\tilde{\psi}_n\rangle + B_{mn}\right] \tag{2.44}$$

Combining equation (2.44) and the Hellmann-Feynman forces finally yields equation (2.45) as an expression for the total forces in PAW.

$$
\begin{aligned}
\boldsymbol{f}^a_{\text{total}} = &- \sum_n f_n \langle \tilde{\psi}_n | \nabla_{\boldsymbol{R}^a} \hat{\tilde{H}} | \tilde{\psi}_n \rangle \\
&+ \sum_{nm} \frac{f_n + f_m}{2} \langle \tilde{\psi}_m | \nabla_{\boldsymbol{R}^a} \hat{S} | \tilde{\psi}_n \rangle \langle \tilde{\psi}_n | \hat{\tilde{H}} | \tilde{\psi}_m \rangle \\
&+ \sum_{nm} \frac{f_n - f_m}{2} B_{mn} \langle \tilde{\psi}_n | \hat{\tilde{H}} | \tilde{\psi}_m \rangle
\end{aligned}
\tag{2.45}
$$

The gradient with respect to atomic positions can be directly applied to the expression of the overlap operator given in equation (2.37):

$$
\nabla_{\boldsymbol{R}^a} \hat{S} = \sum_{i_1 i_2} \Delta S^a_{i_1 i_2} \left(\nabla_{\boldsymbol{R}^a} |\tilde{p}^a_{i_1}\rangle \langle \tilde{p}^a_{i_2}| + \text{c.c.} \right) \tag{2.46}
$$

with $\Delta S^a_{i_1 i_2} = \langle \phi^a_{i_1} | \phi^a_{i_2} \rangle - \langle \tilde{\phi}^a_{i_1} | \tilde{\phi}^a_{i_2} \rangle$. The last term in equation (2.45) accounts for electronic excitations due to a unitary transformation between occupied and unoccupied states and in general cannot be further specified [16]. In the special case of the matrix $\langle \tilde{\psi}_n | \hat{\tilde{H}} | \tilde{\psi}_m \rangle$ commuting with the occupations, this term is exactly equal to zero. This is fulfilled for the Hamiltonian for converged KS wave functions, due to the resulting matrix being diagonal. In this case, the second term also involves a diagonal matrix, wherefore only a single sum has to be computed. Thus, equation (2.45) is simplified to a force expression like the one given in Ref. [19]:

$$
\boldsymbol{f}^a_{\text{total}} = -\nabla_{\boldsymbol{R}^a} E[\tilde{n}] + \sum_n f_n \varepsilon_n \sum_{i_1 i_2} \Delta S^a_{i_1 i_2} \left(P^{a*}_{n i_1} \langle \nabla_{\boldsymbol{R}^a} \tilde{p}^a_{i_2} | \tilde{\psi}_n \rangle + \langle \tilde{\psi}_n | \nabla_{\boldsymbol{R}^a} \tilde{p}^a_{i_1} \rangle P^a_{n i_2} \right) \tag{2.47}
$$

3 Neural-Network Force Fields

Neural networks tackle the problem of parametrizing a force field by forgoing a classical scheme of fitting the parameters of a pre-defined functional form in favor of a more flexible form with universal approximation properties. In this problem the Cartesian coordinates $\boldsymbol{r}_i$ of the atoms are taken as inputs and are mapped onto outputs that represent physical quantities, such as the potential energy E_{pot} and the forces on atoms $\boldsymbol{f}_a$. Parameters, which provide an optimal fit to the data but still define a model as general as possible, are found via machine learning.

In their most basic form, neural networks transform the inputs $\boldsymbol{x}$ by computing a linear combination with an added constant and applying a non-linear activation function f_{act}. The simplest possible network is a single-layer perceptron, which performs exactly this transformation to compute the output y:

$$y = f_{\text{act}}\left(\boldsymbol{w} \cdot \boldsymbol{x} + b\right) \tag{3.1}$$

The individual elements of the vector $\boldsymbol{w}$ are referred to as the weights and b the bias, which are trained parameters. To achieve a sufficiently flexible and general model, this basic idea is applied in a nested manner, where multiple so-called hidden layers containing multiple neurons, which compute their individual outputs according to equation (3.1), lie between the input and output layer. A network built according to this architecture is known as a multi-layer perceptron (MLP), which also serves as one of the main building blocks for further neural networks, for example convolutional neural networks. MLPs are dense feed-forward neural networks, which means that a single neuron takes the outputs of all neurons in the previous layer as input. For each layer there are $n_{\text{inputs}} \times n_{\text{neurons}}$ weights and n_{neurons} biases. Collecting the weights and biases in the matrices $\boldsymbol{W} \in \mathbb{R}^{n_{\text{inputs}} \times n_{\text{neurons}}}$ and $\boldsymbol{b} \in \mathbb{R}^{n_{\text{neurons}}}$, the outputs of a single layer are given by

$$\boldsymbol{y} = f_{\text{act}}\left(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}\right) \tag{3.2}$$

and the output of the whole network by repeated application of these matrix-vector products. The sets of all weight matrices $\{\boldsymbol{W}_m\}$ and bias vectors $\{\boldsymbol{b}_m\}$, where m refers to the index of a specific layer, combined with a defined sequence of layers with given widths and a chosen activation function are referred to as the model.

The parameters of a model are found via the following training routine: A set of reference data, which represents the ground truth the model is supposed to approximate, is divided in non-overlapping training and validation sets. The model is evaluated on the training

set and the predicted outputs and underlying ground truths are used to evaluate a loss function $\mathcal{L}$, which models the error made in the predictions. This loss function is minimized in repeated applications of this scheme, a single step of which is referred to as an epoch, to find optimal values for the weights and biases. To prevent the phenomenon of overfitting, validation statistics are computed using the validation set. Since this set is non-overlapping with the training set, predictions are evaluated using information not used for fitting the parameters. Only parameters that achieve minimal errors in those statistics are then accepted.

Neural networks are a promising tool for the parametrization of force fields due to the general and flexible design. The following will be concerned with neural-network force fields following the general architecture first described in Ref. [1] and is a popular choice for the construction of a multitude of force fields [2, 4, 22]. Other approaches such as Gaussian approximation potentials (GAPs) [23] and moment tensor potentials (MTPs) [24] have also been applied, but will not be discussed in this work.

For the force field to be useful it is necessary that it supports an arbitrary number of atoms and that the labelling order of the atoms does not affect the result. One way to achieve this is to compute the energy E_{pot} as a sum of atomic contributions. In the present work $E_{\text{pot}} = E_{\text{BOA}}$, that is, the potential energy of the system under the Born-Oppenheimer approximation is considered as the true, underlying information.

$$E_{\text{pot}}(\{\boldsymbol{R}_a\}) = \sum_{a=1}^{N_{\text{atoms}}} \phi_a(\{\boldsymbol{R}_a\}) \tag{3.3}$$

where ϕ_a represents the, for now not further specified, function that computes the atomic energy contribution of atom a evaluated using the set of Cartesian coordinates of the atoms in the system.

However, directly using the absolute coordinates of atoms as inputs leads to a fundamental problem: Information about the underlying coordinate system would also be used in training the network. This is undesirable in networks built on the architecture described in Ref. [1], since translational and rotational symmetries have to be preserved, making a transformation of the inputs before passing them to the network necessary. This takes the form of encoding information about the immediate environments of the atoms in the system in atom-centered descriptors. There have been a wide variety of proposals for encoding atomic environments, for example atom-centered symmetry functions, Coulomb- or Ewald sum-matrices, and smooth overlap of atomic positions (SOAP) [25, 26, 27]. The focus here will however lie on the second-generation spherical Bessel descriptors described in [28], due to them being the descriptor of choice in NeuralIL [4].

After encoding information in the atom-centered descriptors the potential energy is then

calculated as a sum of atomic contributions that do not depend on the positions directly, but on the descriptors. An embedding vector $\boldsymbol{e}$ can be added to include additional information such as atom types.

$$E_{\text{pot}}(\{\boldsymbol{d}_\alpha, \boldsymbol{e}_\beta\}) = \sum_{a=1}^{N_{\text{atoms}}} \Omega(\boldsymbol{d}_a, \boldsymbol{e}_a) \tag{3.4}$$

where α and β are collective indices running over all descriptors and all embedding coefficients for all atoms respectively. The abstract function ϕ_a in equation (3.3) is replaced by the function Ω, which is given by the network model evaluating the inputs for an atom a.

3.1 Spherical Bessel Descriptors

To transform the $3N_{\text{atoms}}$ Cartesian coordinates of all atoms in the system into atom-centered descriptors, a neighbor density function $\eta_{aA}(\boldsymbol{r})$ representing the local density of each chemical element A around an atom a within a given cutoff radius r_c is constructed in a first step.

$$\eta_{aA}(\boldsymbol{r}) = \sum_{\substack{a' \in A \\ R_{aa'} < r_c \\ a' \neq a}} \delta\left(\boldsymbol{r} - \boldsymbol{R}_{aa'}\right) \tag{3.5}$$

This neighbour density is then projected onto a set of orthonormal basis functions

$$B_{nlm}(\boldsymbol{r}) = g_{n-l,l}(r) Y_l^m(\hat{\boldsymbol{r}}) \tag{3.6}$$

which in total gives an expansion of the form

$$\eta(\boldsymbol{r}) = \sum_{n=0}^{n_{\max}} \sum_{l=0}^{n} \sum_{m=-l}^{l} c_{nlm} g_{n-l,l}(r) Y_l^m(\hat{\boldsymbol{r}}) \tag{3.7}$$

[28], with the integers $0 \leq n \leq n_{\max}$, $0 \leq l \leq n$ and $-l \leq m \leq l$. The parameter $n_{\max}$ specifies the total number of descriptors $d_d = \frac{(n_{\max}+1)(n_{\max}+2)}{2}$ used to construct the descriptor, which in turn determines the granularity of the encoding [4]. Spherical

harmonics are a naturally arising choice for the spherical terms Y_l^m and are used in both the construction of both SOAP and spherical Bessel descriptors [27, 28].

It is left to define the radial basis functions $g_{n-l,l}$ in equations (3.6) and (3.7). There is no simple, straightforward choice for these, as was the case for the angular term. However, multiple constraints that either have to be, or a desirable to be, satisfied, give information on how to choose the radial basis functions. As discussed in section 3, the descriptors have to conserve the fundamental symmetries of mechanics. Furthermore, the completeness property of a set of descriptors is desirable: The embedding $f : X \hookrightarrow D$ of the space of physically distinct atomic environments X to the space of descriptors D should be smooth [28]. This translates to the notion that similar, but distinct, atomic environments should also lead to similar, but distinct, representations in the space of descriptors. A further constraint can be put on this condition leading to optimal completeness, whereby a complete description is achieved by a minimal number of radial basis functions [28]. This leads to obvious computational advantages since it necessitates fewer operations in the evaluation procedure.

The radial basis functions are built on the spherical Bessel functions in such a way that they are orthonormal and decay to zero at the cutoff radius r_c, as well as their first and second derivatives. As a final step, the angular parts of the c_{aAnlm} projections of ρ_{aA} are then projected onto all elements $B_{nlm}(\boldsymbol{r})$ of the basis set:

$$d_{aAA'nl} = \sum_{m=-l}^{l} c_{aAnlm} c^*_{aA'nlm} = \frac{2l+1}{4\pi} \sum_{\substack{a',a''\in A,A' \\ a',a''\neq a}} g_{n-l,l}(R_{aa'}) g_{n-l,l}(R_{aa''}) P_l(\cos\gamma_{aa'a''}) \tag{3.8}$$

with $\gamma_{aa'a''}$ denoting the angle defined by the three atoms a, a' and a'' with a at the vertex and P_l denoting the Legendre polynomial of degree l. Since spherical harmonics are orthogonal by their definition and the radial basis functions explicitly orthonormalized this scheme yields a set of mutually orthonormal basis functions.

3.2 Electrostatic Interactions in NNFFs

While the encoding of atomic neighbourhood environments in atom-centered descriptors solves the problem of achieving a representation that conserves the fundamental invariancies, they only capture the immediate environment of an atom a within the cutoff radius r_c. Thus, only the local environment defined by r_c is available information for the network and, consequently, long-range interactions are not fully captured by construction.

The most basic approach to considering long-range interactions lies in observing that the system to be modelled is well described by an NNFF as is, which is often the case for a small number of different elements and short screening distances [29]. In such cases a minimal accuracy loss is to be expected when relying only on the descriptors, as has been demonstrated across a wide range of different systems [4, 30, 31, 32].

However, for systems where a model built using only the local descriptions is insufficient, long-range interactions have to be integrated on top of the existing framework [5]. In the context of electrostatic interactions, Ref. [29] describes multiple strategies to incorporate them in a NNFF. All of them have in common that they introduce an additional electrostatic energy term in equation (3.4):

$$E_{\text{pot}}(\{\boldsymbol{d}_\alpha, \boldsymbol{e}_\beta, q_a\}) = \sum_{a=1}^{N_{\text{atoms}}} \Omega(\boldsymbol{d}_a, \boldsymbol{e}_a) + \frac{1}{2} \sum_{a \neq a'} \frac{1}{4\pi\varepsilon_0} \frac{q_a q_{a'}}{|\boldsymbol{R}_a - \boldsymbol{R}_{a'}|} \tag{3.9}$$

where $\boldsymbol{q} = \{q_a \ \forall a \in 1, \ldots, N_{\text{atoms}}\}$ is a vector of charges for each atom. (Note the use of SI units from equation (3.9) onwards.) The conventional way to compute the double sum is to reformulate the term as an Ewald summation to improve computational efficiency [33].

Methods to include electrostatic contributions in NNFFs mainly differ in how they determine the charges q_a and in how they perform the computation of the additional term itself: Models such as those described in Refs. [34, 35] predict atomic charges based on the atom-centered descriptors themselves. For such models, the condition of charge conservation has to be enforced explicitly via charge normalization or re-weighting schemes, which may negatively effect transferability of the model [29]. A different approach to predicting atomic charges is implemented in Ref. [22], where a NNFF is combined with the CENT charge equilibration scheme [36], which infers charges using electronegativity parameters based on the local arrangement of atoms. Variants based on (maximally) localized Wannier functions reformulate the Coulomb term not to be computed using point charges, but as integrals parametrized by the Wannier centers [37, 38]. A significantly different route of computing the Coulomb term in equation (3.9) is taken in Ref. [39], where machine learning is used to compute the Ewald summation kernels for the long-range contributions.

The choice of modelling approach for this work is based on predicting atomic point charges based not directly on the descriptors or an charge equilibration scheme, but using a specially constructed neural network called electron-passing neural network (EPNN), which is discussed in detail in section 4.

4 Atomic Charge Prediction using Neural Networks

In order to model an electrostatic contribution to the energy using a point-charge model it is necessary to determine atomic charges for all nuclei. In the present work, this is achieved using a neural-network based approach named electron-passing neural network (EPNN) introduced in Ref. [6], which is built on the general framework of message-passing neural networks (MPNN) described in Ref. [40].

4.1 Graph Representation of Atomic Systems

MPNNs belong to the class of graph neural networks (GNN), which emerged due to the difficulty of dealing with data that is not represented in a Euclidean space in methods such as convolutional neural networks (CNN) [41, 42]. Instead of working with data represented, for example, on regular grids, such networks work on data represented on a graph. A graph is defined as a tuple $\mathcal{G} = (V, E)$ where $V = \{v_a\}$ is the set of all nodes (or vertices) in the graph and $E = \{e_{aa'}\}$ the set of all edges, which represent the connectivity between nodes. In the case of directed graphs, edges point from one node to another without an edge in the reverse direction being defined [41]. In contrast, graphs representing atomic structures will, in general, be un-directed, which means that two nodes v_a and $v_{a'}$ will be connected by two opposingly directed edges $e_{aa'}$ and $e_{a'a}$. The neighbourhood of a node v_a is given by the subset $N(v_a) = \{v_{a'} \in V | (v_a, v_{a'}) \in E\}$ of all nodes $v_{a'}$ that share an edge with v_a. In the context of MPNNs, each node is equipped with a feature vector $\boldsymbol{v}_a$ of dimension d_v. Thus the set of nodes $V \in \mathbb{R}^{N \times d_v}$, where N denotes the total number of nodes. Similarly, each edge is equipped with an edge feature vector $\boldsymbol{e}_a$ of dimension d_e, so the set of edges $E \in \mathbb{R}^{N \times N \times d_e}$ [40]. Whether two nodes are connected by an edge and thus directly communicate with each other is dependent on the underlying problem defining the graph structure and is thus dependent on the application.

4.2 Message-Passing Neural Networks

The core idea of MPNNs lies in a learned representation of the neighbourhood of a node $\boldsymbol{v}_a$ by iteratively updating the node states $\boldsymbol{v}_a^t$ using messages $\boldsymbol{m}_a^{t+1}$ with the help of an update function U_t, as given in equations (4.1) and (4.2). In the present MPNN, the message generation functions M_t and update functions U_t are given by feed-forward neural networks [40, 6], in contrast to previous work, where M_t is given simply by concatenation and a U_t by a sigmoid function [43].

$$\boldsymbol{m}_a^{t+1} = \sum_{v_{a'} \in N(v_a)} M_t(\boldsymbol{v}_a^t, \boldsymbol{v}_{a'}^t, \boldsymbol{e}_{aa'}) \tag{4.1}$$

$$\boldsymbol{v}_a^{t+1} = U_t(\boldsymbol{v}_a^t, \boldsymbol{m}_a^{t+1}) \tag{4.2}$$

where the superscript t is an integer representing index of the message passing step. The total number T of message passing steps can be freely chosen and typically is a hyperparameter of the network.

The combination of the message and update functions for each step t define a convolution-like operation on all neighboring inputs on the graph. Thus, MPNNs can be classified as spatial-based graph convolutional networks (GCN) [42], which generalize the framework of CNNs to non-Euclidean data. These networks were individually introduced as such in Ref. [44] as a method to perform spectral convolutions on graphs. MPNNs can also be used in similar fashion to spectral GCNs, as for example the Ewald-based message passing described in Ref. [39], which makes use of the formal correspondence between the formalism of electrostatic interactions and continuous-filter convolutions [45].

In MPNNs, a desired quantity is computed from the learned node states via a readout function $R(\{\boldsymbol{v}_a\})$. In the case of the total energy such a readout can have the form

$$E = \sum_{v_a \in V} \mathrm{NN}^R(\boldsymbol{v}_a^T) \quad E \in \mathbb{R} \tag{4.3}$$

where NN^R again represents a feed-forward neural network [40]. This defines the total energy as a sum of atomic contributions, similar to the NNFFs described in section 3.

4.3 Electron-Passing Neural Networks

The EPNN (algorithm 1) foregoes the readout step described in section 4.2 in favour of using the learned description of the environment of the nodes to predict atomic charges while conserving the total charge of the system [6]. To achieve this, the graph tuple $\mathcal{G} = (V, E)$ is extended by a set $Q = \{q_a\} \in \mathbb{R}^N$, which are assigned to the nodes and initialized such that the total charge of the whole system $Q^{\mathcal{G}}$ is correct:

$$\sum_{a}^{N_{\text{atoms}}} q_a = Q^{\mathcal{G}} \tag{4.4}$$

Lines 2 to 5 of algorithm 1 constitute the message-passing phase described in section 4.2. In this phase the node states $\boldsymbol{v}_a$ are iteratively updated to form a description of the local environment of a single node a, which is illustrated in the upper panel of figure 1. The description of the local environment is then used to compute the electron updates.

Algorithm 1 Electron-passing neural network (original version)

Require: Graph $\mathcal{G} = (V, E, Q)$

1: Initialize $\{\boldsymbol{v}_a\}$, $\{q_a\}$ such that $\sum_a^{N_\text{atoms}} q_a = Q^\mathcal{G}$
2: **for** $t \leftarrow 0$ to T **do** — Message passing phase
3: $\quad \boldsymbol{m}_a^{t+1} \leftarrow \sum_{v_{a'} \in N(v_a)} M_t(\boldsymbol{v}_a^t, \boldsymbol{v}_{a'}^t, \boldsymbol{e}_{aa'})$ — Pass messages
4: $\quad \boldsymbol{v}_a^{t+1} \leftarrow U_t(\boldsymbol{v}_a^t, \boldsymbol{m}_a^{t+1})$ — Update node states
5: **end for**
6: **for** $s \leftarrow 0$ to S **do** — Electron passing phase
7: $\quad \sigma^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) \leftarrow \text{NN}^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) - \text{NN}^s(q_{a'}^s, q_a^s, \boldsymbol{v}_{a'}^T, \boldsymbol{v}_a^T, \boldsymbol{e}_{a'a})$
8: $\quad q_a^{s+1} \leftarrow \sum_{\boldsymbol{v}_{a'} \in N(\boldsymbol{v}_a)} \sigma^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) f_c(\boldsymbol{e}_{aa'})$ — Update charges
9: **end for**

Similar to the message passing phase, the atomic charges q_a are updated in electron passes (lower panel in figure 1). In order to preserve the total charge $Q^\mathcal{G}$ the updates σ^s computed in line 7 of algorithm 1 have to be anti-symmetric with respect to permutation of the input indices:

$$\sigma^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) = -\sigma^s(q_{a'}^s, q_a^s, \boldsymbol{v}_{a'}^T, \boldsymbol{v}_a^T, \boldsymbol{e}_{a'a}) \tag{4.5}$$

Anti-symmetry is guaranteed by defining the updates as the difference of the outputs of an update generation function for permuted indices. As is the case for the message passing phase, the method for generating the individual terms in equation (4.6) is again a design choice. In Ref. [6], feed-forward neural networks NN^s for each individual electron passing step are chosen.

$$\sigma^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) = \text{NN}^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) - \text{NN}^s(q_{a'}^s, q_a^s, \boldsymbol{v}_{a'}^T, \boldsymbol{v}_a^T, \boldsymbol{e}_{a'a}) \tag{4.6}$$

The messages σ^s are then used to update the atomic charges via equation (4.7) (line 8 of algorithm 1). Note that, in contrast to equation (4.2), no update function is applied to the collected messages to ensure charge preservation.

$$q_a^{s+1} = \sum_{v_{a'} \in N(v_a)} \sigma^s(q_a^s, q_{a'}^s, \boldsymbol{v}_a^T, \boldsymbol{v}_{a'}^T, \boldsymbol{e}_{aa'}) f_c(\boldsymbol{e}_{aa'}) \tag{4.7}$$

The function f_c serves as a cutoff function to ensure possible discontinuities arising from atoms entering and leaving the cutoff range defined by r_c are smoothed out. Ref. [6] suggests a simple linear cutoff function $f_c = \max(0, \boldsymbol{e}_{aa'} - \epsilon)$ with a tolerance $\epsilon = 10^{-5}$. As is the case for the total number of message passing steps T, the total number of electron passing steps S is again a network hyperparameter. With the update functions and step counts chosen, the network is fully defined.

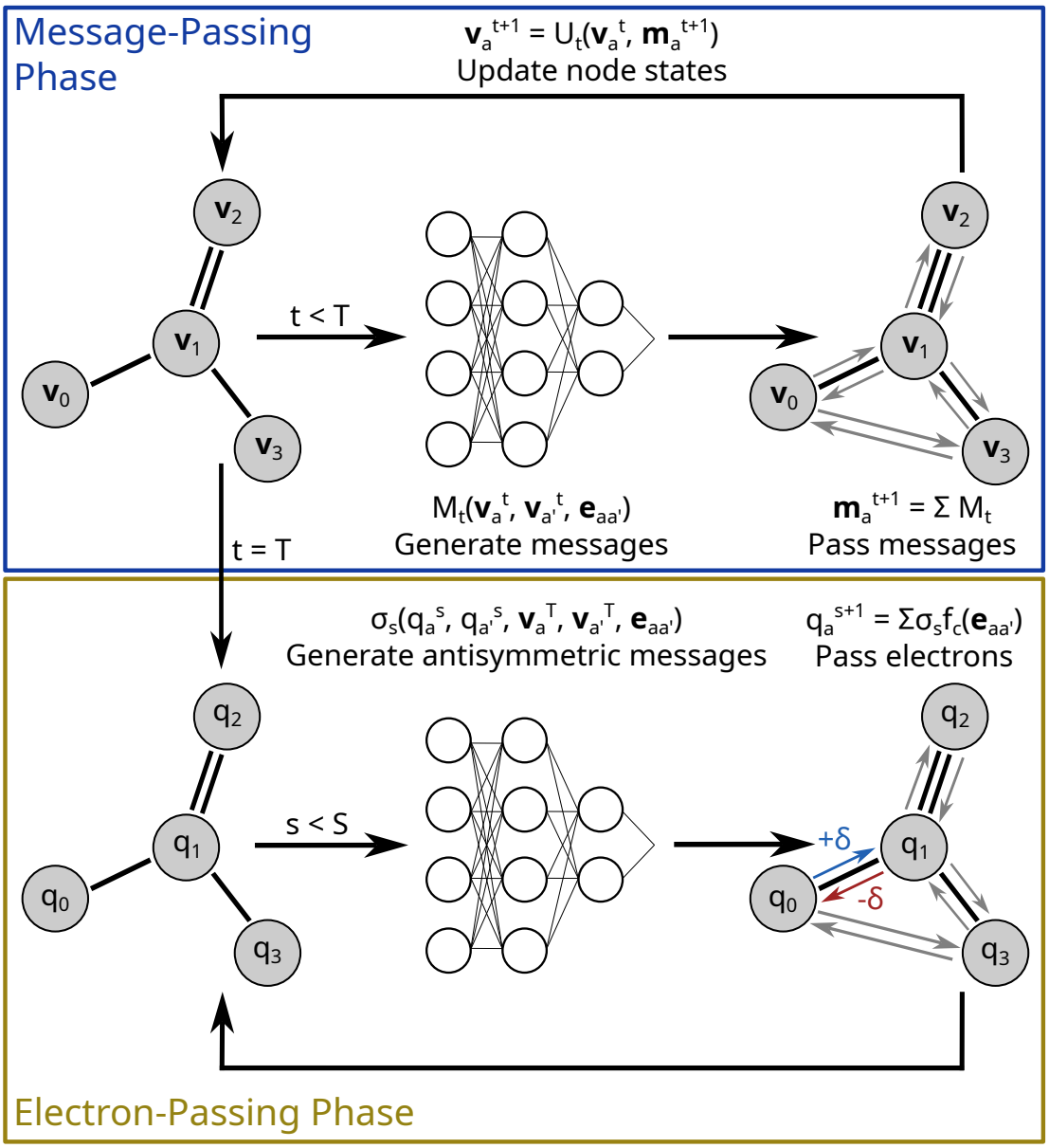


Figure 1: Schematic visualization of the EPNN

5 Methodology

5.1 Training Data

Three datasets from different areas of applications have been used to provide training data for the neural networks in this work. First, the ethylammonium nitrate (EAN) database originally compiled for Ref. [4], which contains configurations sampled from a classical MD run using GROMACS [46] with an OPLS-AA FF parametrization for ionic liquids [7, 47]. The configurations subsampled from the run are used as inputs for a DFT calculations. A subset of 373 of the configurations have been subjected to five steps of the ASE quasi-Newton minimizer to ensure good sampling around local minima [4]. This dataset represents the typical use case of a NNFF, namely the improvement of MD simulation results at finite temperature. Second, a subset of 500 $SrTiO_3$(110) surface layer reconstructions (also known as STO) generated and reported in Ref. [48]. Third, a set of 76 water cluster configurations containing between 4 and 12 molecules [49]. All DFT runs have been performed using GPAW [19, 20], the parameters are compiled in table 1.

Table 1: DFT parameters for the generation of training datasets

Dataset	Mode	Basis/Cutoff	XC	BZ Sampling
EAN	LCAO	Double-ζ polarized	LDA [13]	Γ-only
STO	LCAO	Double-ζ polarized	PBE [50]	(2, 2, 1)
H_2O	PAW	GPAW 0.9.20000/340 eV	RPBE [51]	Γ-only

To evaluate the quality of the implemented version of the EPNN, atomic charges have been obtained by partitioning of the all-electron density using the DDEC6 approach [52] implemented in Chargemol [53].

5.2 Neural-Network Force Field

The energies and forces of the systems are modelled using an updated version of the NeuralIL implementation originally described in Ref. [4]. This NNFF is built on JAX [54] and utilizes the just-in-time (JIT) compilation, vectorization and automatic differentiation features provided to achieve fast training on N_{atoms} energies and $3N_{\text{atoms}}$ forces for each structure. Two major improvements to the original version are implemented: First, the MLP, which originally served as the core model, is replaced by a JAX-conforming implementation of a deep residual network architecture (ResNet) [55]. Second, in place of the original one-cycle minimizer schedule based on the adamw optimizer, the non-linear, learned optimizer VeLO is used [56]. These improvements have been implemented after

the original publication and are reported in Ref. [57]. If not noted otherwise, models have been constructed using a cutoff radius of $r_c = 4.0\,\text{Å}$, $n_{\max} = 4$ and a sequence of hidden layers with widths 64:32:16.

5.3 Electron-Passing Neural Network

5.3.1 Programming Framework

A version of the EPNN based on Ref. [6] is implemented in Python 3.9 based on the Google JAX framework [54]. Since the EPNN works on graph-structured data, the graph neural network library JRAPH [58] is used to build representations of the inputs. Neural network modules are built on top of FLAX [59]. Training is handled using the non-linear, learned VeLO optimizer [56], which is implemented using Deepmind's OPTAX [60]. This combination of modules allows to make full use of the core features of JAX, namely just-in-time (JIT) compilation, automatic differentiation (AD) and scalability to parallel systems.

Since native Python is an interpreted language, compile-time optimizations cannot be utilized. To improve performance of repeatedly called code, JAX uses the XLA (Accelerated Linear Algebra) compiler to perform optimizations on sections when they are first called. This requires the functions to be pure, thus not relying on global state or data dependent shapes or conditionals.

The AD algorithm allows fast and accurate calculations of derivatives and gradients of pure functions and is essential in the field of machine learning to determine the gradient of the loss function in order to optimize the parameters [61]. Furthermore, models can be constructed to be end-to-end differentiable, as is utilized in Ref. [4] to obtain forces at minimal computational cost. Similar techniques can be used in combination with the EPNN implementation described in this section to determine quantities such as, for example, Born effective charges [62] from the predicted outputs.

Neural networks are used to work with large amounts of data and often perform the same operations on different data, making them ideal candidates for parallelization using accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs). Libraries such as JAX use specially implemented array types and algorithms on top of these to allow for simpler scaling to larger, parallel systems, which is utilized to work with the EPNN on GPUs.

Function transformations such as AD and JIT-compilations are handled in JAX using the concept of a PyTree. This abstraction of data structures divides objects into two classes: Container-like Python objects, namely lists, tuples and dictionaries, are referred

to as PyTrees and are used to build a tree structure with any non-PyTree objects as leaves. Internally, two functions are associated with PyTrees [54]: The first allows a conversion of the tree structure to a (children, metadata) pair and the second defines a conversion back to the original PyTree structure. (It is possible to manually register any Python object as a PyTree as long as it provides these two functions.) This enables a consistent mapping of functions and function transformations onto any combination of data structures, which is crucial for tracing functions based solely on the shapes of the inputs.

5.3.2 Graph Representation

As discussed in section 4.1, the atomic system is represented as an un-directed graph, with the nuclei as nodes. The graph is built using the JRAPH module, which implements a general framework for graph datastructures as a tuple consisting of node features, edge features, sender and receiver arrays, optional global information and the total numbers of nodes and edges [58]. The sender and receiver arrays store the node indices of communicating nodes and are of equal length with corresponding indices, wherefore they form an edge list representation of the graph. The nodes $\{v_{a'}\}$ a node v_a communicates with is given by its neighbourhood, which is given by $N(v_a) = \{v_{a'} \in V \,|\, R_{aa'} < r_c\}$ with $R_{aa'} = |\boldsymbol{R}_a - \boldsymbol{R}_{a'}|$ and the cutoff radius r_c being a network hyperparameter to be chosen.

Nodes are implemented in the EPNN using a dictionary, a PyTree with a root and leaves only, to allow for different features and easier extensions. At minimum it contains a field for storing the set of predicted outputs Q and the node states V. In contrast to Ref. [6], types are not stored in a one-hot encoding concatenated to the node states V, but as atomic numbers in a separate field, since they are later used to as inputs to the embedding layer. Additionally, the spherical Bessel descriptors discussed in section 3.1 are added as a field to the node features to include the local environment description used in NEURALIL on top of the learned node states.

The edge description also differs to Ref. [6] in that they are not initialized to a set of Gaussian functions encoding the distances $R_{aa'}$, but simply store the Euclidean distances directly. This choice is made since the local environment is already captured by the spherical Bessel descriptors added to the node features. Further, the cutoff masks used to implement the cutoff function f_c in equation (4.7) are stored as edge features.

Since functions are JIT-compiled based upon the shapes of the inputs when first calling the method, the input data are padded upon construction of the graphs to be of same shape, thus avoiding multiple JIT-compilations to be necessary. Padding and un-padding after calculations are finished are performed using JRAPH functionalities which ensure the padding does not influence the results in any way. This is achieved by creating

an extra "black hole" node for each graph. Extra edges needed to pad the arrays to the same size are directed to these nodes such that computations on the graph are not affected by the padding. Performance is further increased by splitting the data in small batches such that the individual graph objects constructed hold information about multiple configurations. The batch size is a free parameter that is chosen to optimize the trade-off between memory requirements and keeping vectorization pipelines occupied.

5.3.3 Network Architecture

The network implements a modified version of algorithm 1 on top of FLAX [59] and utilizes individual feed-forward neural networks for each message generation, update and electron pass generation step. The set of all of these models will be referred to as the core models. In contrast to the original implementation, which uses MLPs as the core models, the ResNet architecture introduced in Ref. [55] is chosen for this work. The non-linear activation function is chosen as Swish-1, defined by setting $\beta = 1$ in equation (5.1), a functional form originally found in Ref. [63]. Similar to activation functions like $\mathrm{ReLU}(x) = \max(0, x)$, this function provides the benefit of avoiding the gradient becoming near zero for large inputs, as was the case for more classical activation functions like the sigmoid. Furthermore, it does not suffer the problem of not being continuously differentiable at $x = 0$; in fact, it is infinitely times differentiable at that point.

$$s_\beta(x) = \frac{x}{1 + \exp(-\beta x)} \tag{5.1}$$

The weights and biases for all core models are trainable parameters, which are updated by minimizing a loss function $\mathcal{L}$. Intuitive choices for this loss functions are given by the mean absolute error (MAE) or mean squared error (MSE). However, measuring the error in the L_2 norm puts heavy emphasis on outliers, which might affect model performance. On the other hand, the L_1 norm measure is not continuously differentiable at $q - \tilde{q} = 0$, where q refers to the underlying data and $\tilde{q}$ to the predictions made by the model. An "intermediate" choice is given by the log-cosh loss

$$\mathcal{L}_{\mathrm{log-cosh}} = \frac{1}{N_{\mathrm{atoms}}} \sum_{a=1}^{N_{\mathrm{atoms}}} \alpha \log \left[\cosh \left(\frac{q_a - \tilde{q}_a}{\alpha} \right) \right] \tag{5.2}$$

with α being a hyperparameter [64]. This loss function can be viewed as a smooth approximation to the MAE, which solves the differentiability problem at $q - \tilde{q} = 0$ since $\mathcal{L}_{\mathrm{log-cosh}}(q, \tilde{q}) \to \frac{|q-\tilde{q}|^2}{2}$ for prediction errors smaller than α. On the other hand,

$\mathcal{L}_{\log-\cosh}(q, \tilde{q}) \to |q - \tilde{q}|$ for prediction errors significantly larger than α, which avoids overemphasizing outliers.

After preliminary runs, in which the impacts of different layer widths, node feature vector dimensions and numbers of message- and electron-passing steps on model quality have been investigated, the following parameters are chosen: Models are constructed with a series of features with sizes 128:64:32 for the message and electron pass generation models and 64:32 for the update model. $n_{\max} = 4$ and $r_c = 4.0\,\text{Å}$ are chosen corresponding to the values for the dynamics model (section 5.2). The node feature dimension determining the size of the individual node feature vectors $\boldsymbol{v}_a$ is chosen as $d_v = 16$. Types are used as inputs to an extra embedding layer with an output dimension of $d_{\text{embed}} = 2$, which further extends the node description. In total, the local node description is given by the concatenation of the learned node states $\boldsymbol{v}_a$, the spherical Bessel descriptors and the type embedding. Therefore, the dimension of the node states is given by $d_v + d_d + d_{\text{embed}}$, where d_d refers to the total number of spherical Bessel descriptors for a single atom. These concatenated vectors are then used as inputs to the message generation, update and electron pass generation models. The number of message and electron passing steps is set to $S = T = 3$, as suggested in Ref. [6].

As the module implementing algorithm 1 is based on the neural network library FLAX, it is built as a stateless object. The parameters defining the state of a model are stored externally in PyTrees for each of the individual core networks, which are passed to the module upon calling it. Handling of model state is done via the functionalities of FLAX and additionally OPTAX, which provides tools for optimizers and updates.

It is noted that, while the network is initially introduced for predicting atomic charges, the implementation is structured to take any input shape and allow for use in modelling non-scalar quantities as well. Furthermore, since the spherical Bessel descriptors are added to the node features and thus a description of the immediate environment of the atoms is already given, the message passing phase can be skipped entirely, which allows for electron passing only. The performance impact of skipping the message passing phase is however not explored further in this work.

6 Results

6.1 Water Dimer

To investigate the energies for a system with well-known behaviour, a water dimer is chosen for preliminary experiments. Oxygen-oxygen distances are sampled in a range of 1.5 Å to 5.5 Å. Geometry optimization is performed to find the local minimum on the potential energy surface using the BFGS optimization method implemented in `scipy` [65] at these sampled points. The difference of the total energy of the the dimer and two individual H_2O molecules is then calculated. Calculations were performed using GPAW 22.8.0 [19, 20] in PAW mode with a PW cutoff energy of 750.0 eV and the LDA XC functional [13].

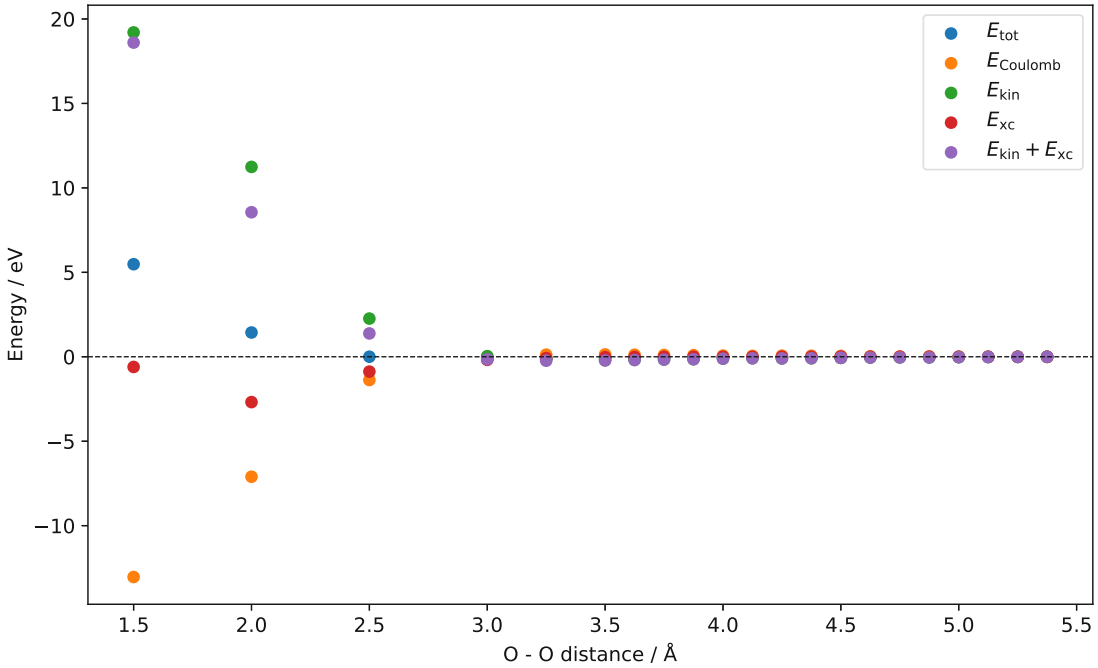


Figure 2: Total energy difference and individual contribution differences for a H_2O dimer compared to two isolated H_2O molecules depending on the distance between the oxygen atoms

The total energy and its contributions at the sampled oxygen-oxygen distances are visualized in figure 2. At O-O distances < 2.5 Å, a strong electrostatic attraction is observed due to H-O proximity. However, a large kinetic contribution still leads to a net energy increase. The minimum energy configuration among the sampled points is at an O-O distance of 2.5 Å with an energy difference of -0.13 eV between the dimer and the isolated H_2O molecules. At distances ≥ 3.5 Å the energy difference between the dimer and the isolated atoms is < 100 meV.

Furthermore, practically no short-range effects are observed anymore and the electrostatic energy can be modelled by the interaction of two classical dipoles. In this case the distance between the two dipoles $|\boldsymbol{r}_1 - \boldsymbol{r}_2| \gg |\boldsymbol{a}|$, where $|\boldsymbol{a}|$ is the distance between the two charges of a single, classical dipole. This is well approximated in the far-field limit, in which the potential $V_1(\boldsymbol{r})$ due to a dipole with moment $\boldsymbol{p}_1$ is given by

$$V_1(\boldsymbol{r}) = \frac{\boldsymbol{p}_1 \cdot \boldsymbol{r}}{4\pi\varepsilon_0 r^3} \tag{6.1}$$

wherefore equation 6.2 describes the interaction energy E due to a second dipole $\boldsymbol{p}_2$ in the field of the first dipole $\boldsymbol{p}_1$.

$$E = -\boldsymbol{p}_2(-\nabla V_1) = -\frac{1}{4\pi\varepsilon_0 r^3}\left[\boldsymbol{p}_1 \cdot \boldsymbol{p}_2 - 3\left(\boldsymbol{p}_1 \cdot \hat{\boldsymbol{r}}\right)\left(\boldsymbol{p}_2 \cdot \hat{\boldsymbol{r}}\right)\right] \tag{6.2}$$

This behaviour is recovered for large O-O distances, as evident in figure 3. Small deviations are observed due to the high convergence requirement posed upon the geometry optimization to resolve the small energy differences due to the dipole interaction only.

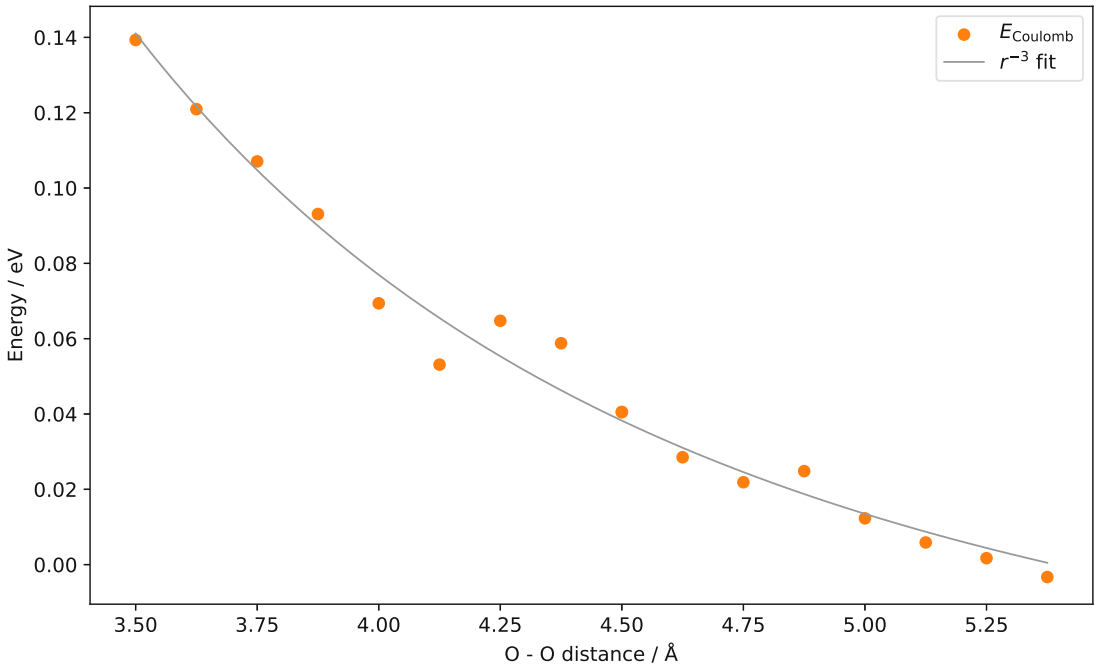


Figure 3: Electrostatic energy contribution difference for a H_2O dimer compared to two isolated H_2O molecules depending on the distance between the oxygen atoms. A r^{-3} fit models the interaction of two classical dipoles

6.2 Handling of Electrostatics in a Neural-Network Force-Field

As discussed in section 3.2, neural-network force fields by design do not account for long-range interactions beyond the cutoff radius r_c, so augmentations that introduce additional terms are introduced to the potential energy expression. To achieve this, the electron-passing neural network implementation outlined in section 5.3 can be used to generate charges used to evaluate a Coulomb term, which is added to the potential as given in equation (3.9).

The choice of the kind of underlying data used to construct the EPNN model is important: Since the potential energy and forces on atoms are the physical observables the model is supposed to predict directly, the most obvious choice is to also train an augmented NNFF on these sets alone. However, in an experiment conducted in Ref. [4], NeuralIL was augmented with an electrostatic term and charges inferred using a CENT-based partitioning scheme [36]. It has been shown that the improvement in the energies and forces is marginal and in this case comes at the cost of a global equilibration step, seriously compromising the scalability of the model. Furthermore, it is observed that charges tend to be predicted close to zero for all atoms, wherefore it is concluded that the local modelling approach was sufficient in this case.

Another possible choice is to provide a training data set that directly contains the partial charges the network is supposed to reproduce. While this gives a straightforward approach to train the electrostatic part independent of local part, the network implicitly gets trained to the methodology of the particular approach chosen to generate the training data. This is problematic since charge partitioning is not a uniquely defined task and different schemes exist, such as the ones described in Refs. [66, 52]. Although some charge partitioning schemes are designed to reproduce the electrostatic field, it is reasonable to expect that this approach negatively affects the transferability of the model. Further, these charges do not provide any information about quantities like the gradient of the electrostatic energy obtained from DFT.

The ultimate goal for this work therefore is to use the electrostatic energy $E_C[n]$ (see section 2.4) and its negative gradient $-\nabla_{\boldsymbol{R}^a} E_C[n]$, which will also be referred to as the electrostatic forces, for training the EPNN. Since the model is designed to predict charges, the outputs are first used to calculate an electrostatic energy, which is then in turn used to evaluate a loss function, similar to the procedure for NeuralIL [4]. The evaluation of the Coulomb term for modelling the long-range energy term is performed using the technique described in Ref. [22]:

$$E_{\text{elec}} = \sum_{a=1}^{N_{\text{atoms}}} \sum_{a'=1}^{N_{\text{atoms}}} \frac{\operatorname{erf}\left(\frac{|\boldsymbol{R}_a - \boldsymbol{R}_{a'}|}{\sqrt{2}\gamma_{aa'}}\right)}{|\boldsymbol{R}_a - \boldsymbol{R}_{a'}|} \frac{q_a q_{a'}}{4\pi\varepsilon_0} + \sum_{a=1}^{N_{\text{atoms}}} \frac{1}{\sqrt{4\pi}\sigma_a} \frac{q_a^2}{4\pi\varepsilon_0} \tag{6.3}$$

with $\gamma_{aa'} = \sqrt{\sigma_a^2 + \sigma_{a'}^2}$. The set of ionic radii $\{\sigma_a\}$ serves to approximate the shapes of the Gaussian charge density distributions.

While it is highly desirable to also be able to include the electrostatic forces to also provide local, node level, information for each node, this quantity is so far not available due to the problems discussed in detail in section 6.5. Alternatively, Born effective charges might serve as inputs to provide local information additionally to the global information given by the electrostatic energy.

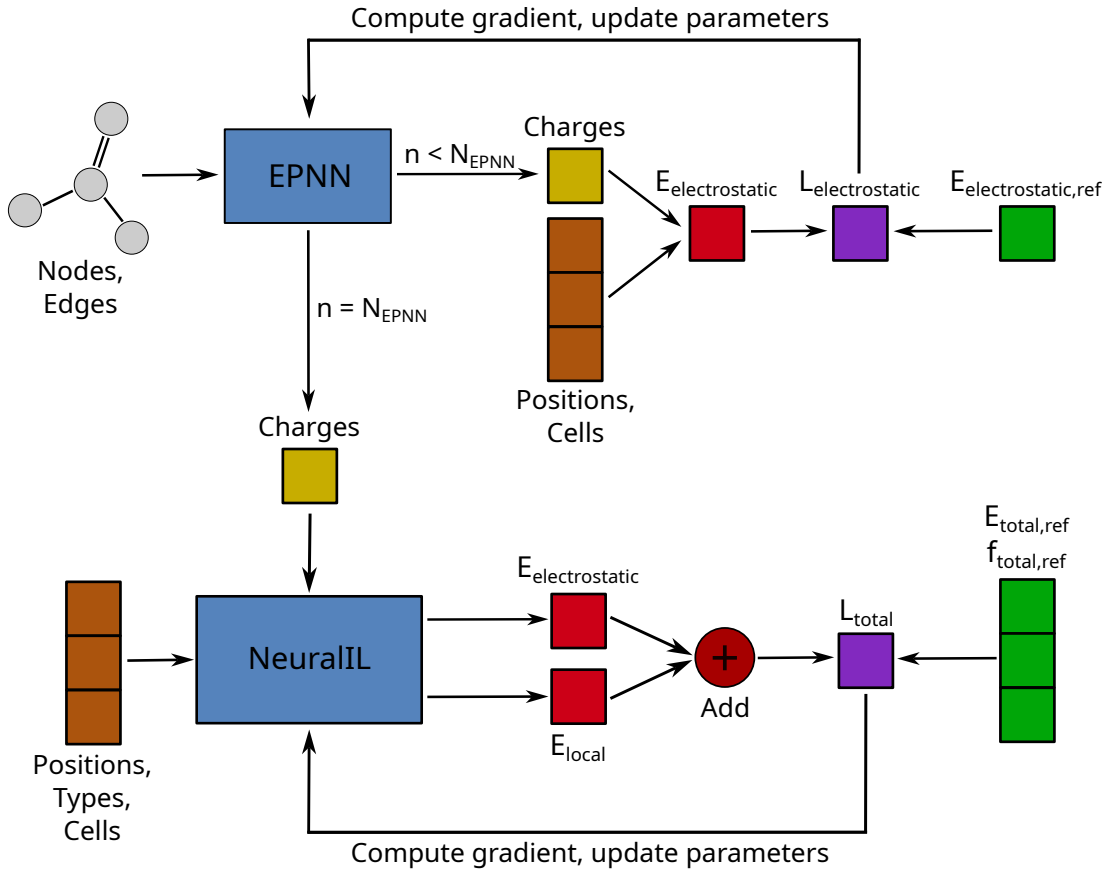


Figure 4: Visualization of the EPNN + NeuralIL training process. N_{EPNN} denotes the number of EPNN training epochs

The total training procedure is visualized in figure 4. The two models are trained in a sequential manner: First the EPNN is trained for N_{EPNN} epochs by using the predicted charges to compute the electrostatic energy according to equation (6.3). The error in the electrostatic energy prediction is then quantified using the log-cosh loss given in equation (5.2) with $\alpha = 0.02\,\text{eV}\,\text{atom}^{-1}$, which then allows to update the model parameters based on the gradient. Charges for the NeuralIL model, to which a long-range term has been added, are then generated using the converged EPNN model. The

training of the dynamics model is then performed as described in Ref. [4].

The problem of the charges being predicted close to zero reported in Ref. [4] is avoided by specifically training the EPNN to reproduce the electrostatic energy hypersurface. The sequential training method is chosen over a more combined, intervowen approach due to the electrostatic energy term also entering the NeuralIL output. Thus, the NeuralIL predictions depend both on the EPNN and NeuralIL parameters. Therefore, the EPNN parameters are optimized beforehand such that the electrostatic energy and forces contributions remain constant while optimizing the parameters for the local model.

6.3 EPNN Training on Charges Obtained by Density Partitioning

In a first set of runs, the EPNN implementation discussed in section 5.3 is used to train models based on charges obtained by partitioning the all-electron density (2.24). This allows for a direct comparison of the model predictions to the underlying training data without any need for post-processing, such as calculating a total electrostatic energy from the charge distribution. Individual models have been trained on the EAN, STO and H_2O cluster data sets respectively, with initial atomic charges $q_a^0 = 0\,\mathrm{e}$.

Figures 5 to 7 show comparisons of predicted charges to the underlying DDEC6 charges the models were trained on and root mean square error (RMSE) and mean absolute error (MAE) statistics for both the training and validation sets. Charges are predicted with very high accuracy in all cases, with the best results being obtained for the ionic liquid EAN. This can be attributed to the significantly larger size of the dataset, having approximately twice as many entries compared to the STO dataset and thirteen times as many entries compared to the H_2O dataset. However, even for the comparatively small set of H_2O cluster configurations, error statistics in the same order of magnitude as for the STO dataset are observed. Overall, the results confirm that the implementation described in section 5.3 is suitable for the prediction of atomic charges.

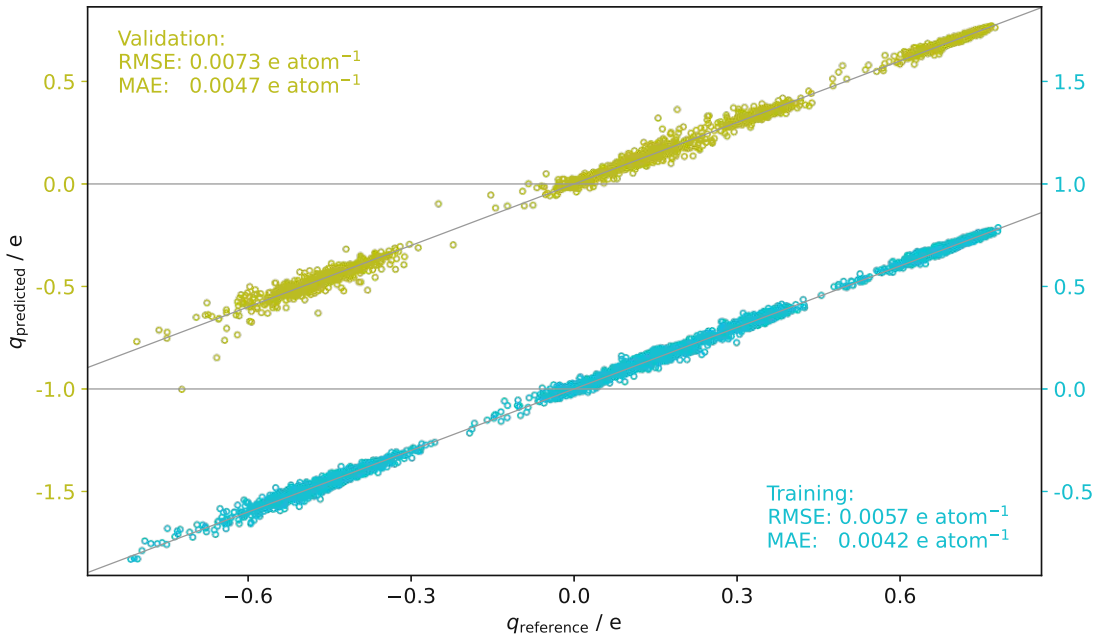


Figure 5: Parity plot comparing predicted charges to DDEC6 charges for EAN using an EPNN model trained directly on DDEC6 charges

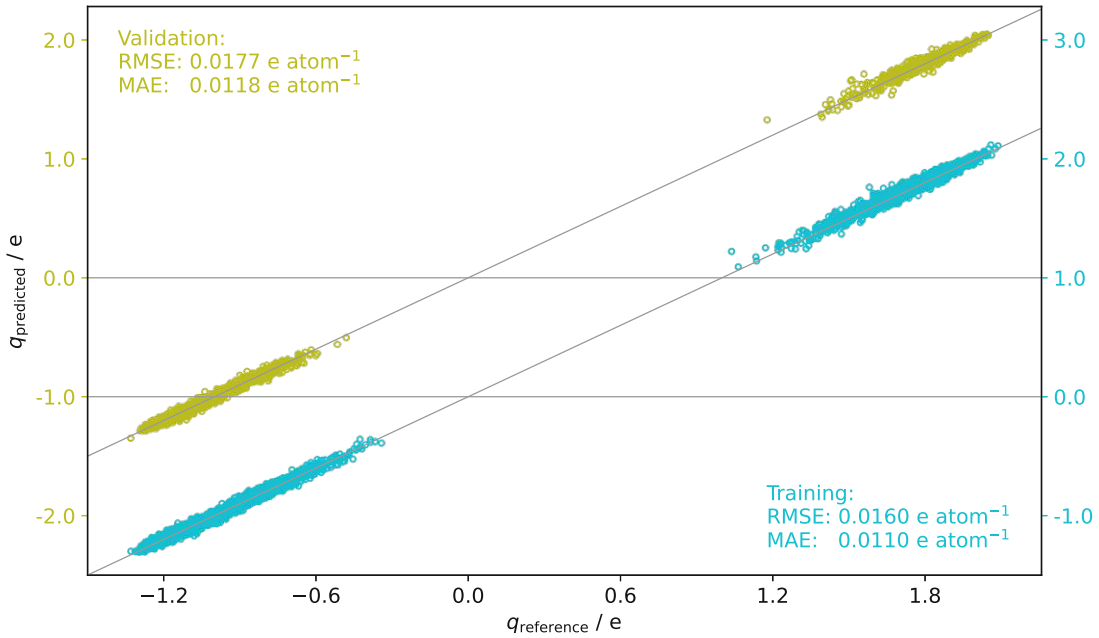


Figure 6: Parity plot comparing predicted charges to DDEC6 charges for STO using an EPNN model trained directly on DDEC6 charges

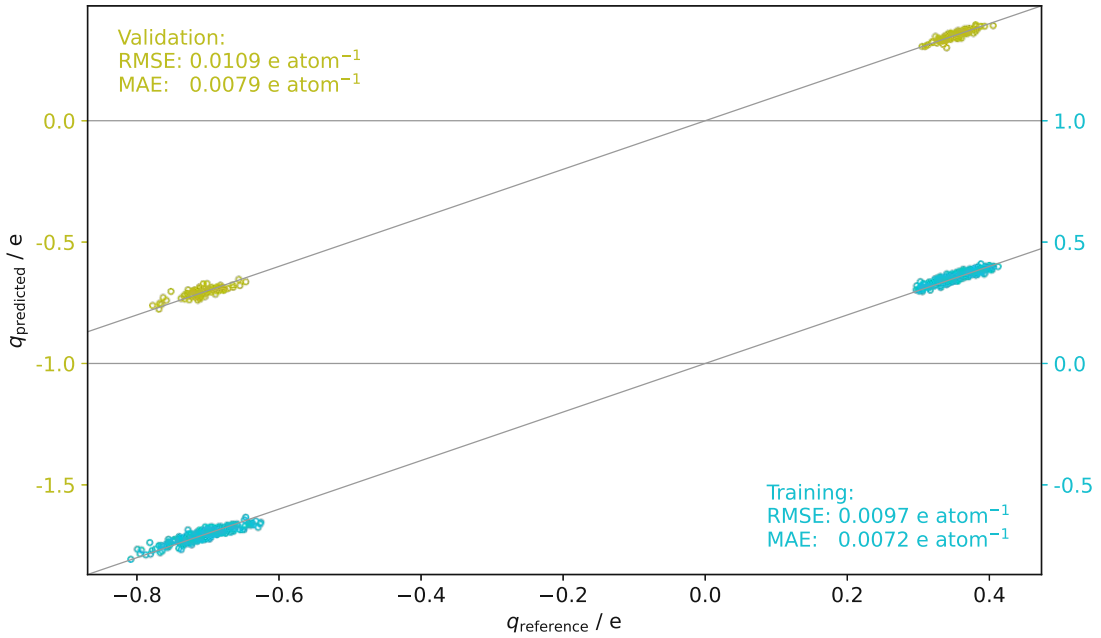
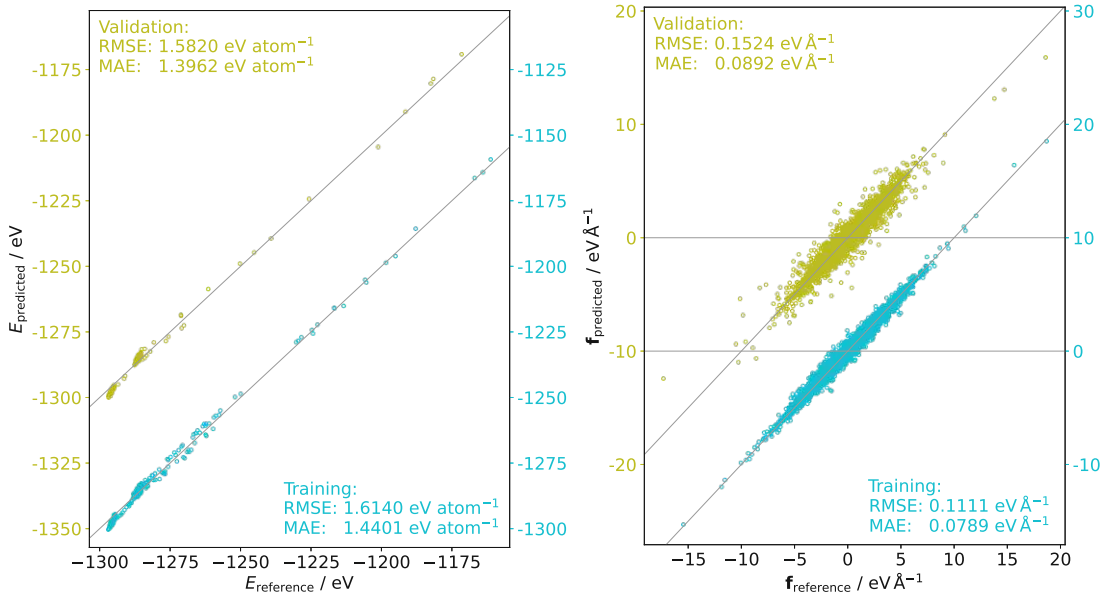


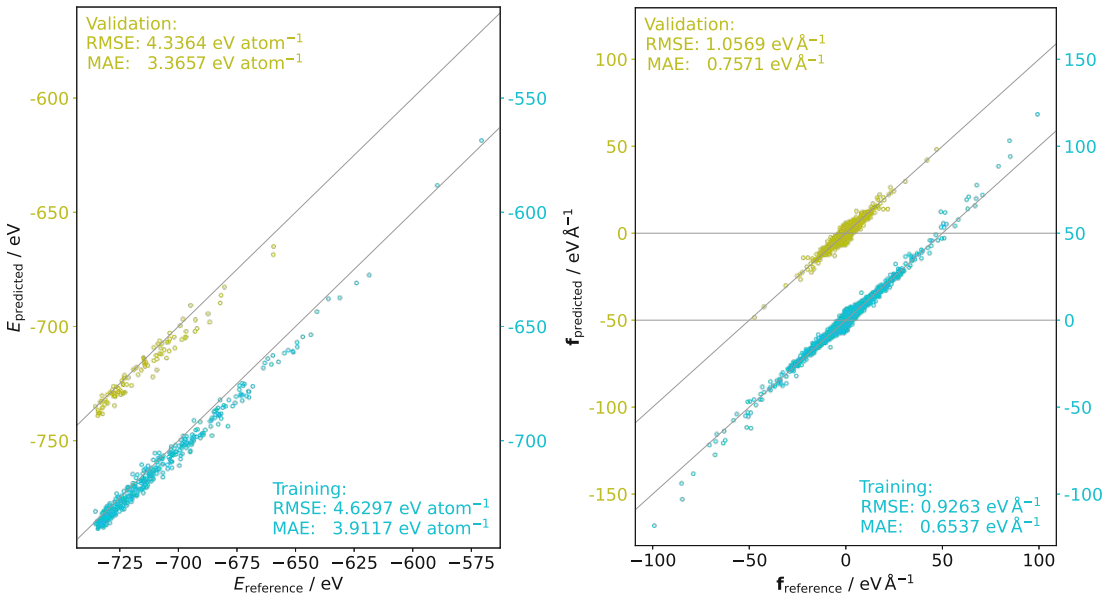
Figure 7: Parity plot comparing predicted charges to DDEC6 charges for H_2O clusters using an EPNN model trained directly on DDEC6 charges

The atomic charges predicted by the EPNN models considered in this section have been used as inputs to model the electrostatic contribution to the total energy in NeuralIL, as described in section 6.2. Comparisons of predicted energies and forces for the EAN, STO and H_2O cluster datasets are visualized in figures 8 to 10. In all cases, worse MAE and RMSE statistics are obtained for the force predictions for the validation sets compared to the training sets. This is most pronounced in the model trained for EAN, where a difference of $0.413\,\mathrm{eV\,Å^{-1}}$ is observed between the RMSEs (figure 8b). This difference is not observed for the energy prediction statistics and can be a consequence of using charges obtained from electron density partitioning for training the EPNN model: The DDEC6 scheme used to calculate the atomic charges aims to reproduce the electrostatic energy [52], but does not make a guarantee for the gradient. The augmented NeuralIL models are thus implicitly optimized to compensate for this error. However, one cannot expect this compensation to be systematic and thus the model gives worse predictions for forces it has not been trained on directly. This is in agreement with an experiment conducted in Ref. [4], in which the electrostatic forces obtained from OPLS-AA were subtracted from the DFT forces before training a NeuralIL model, causing a significant decrease in accuracy.



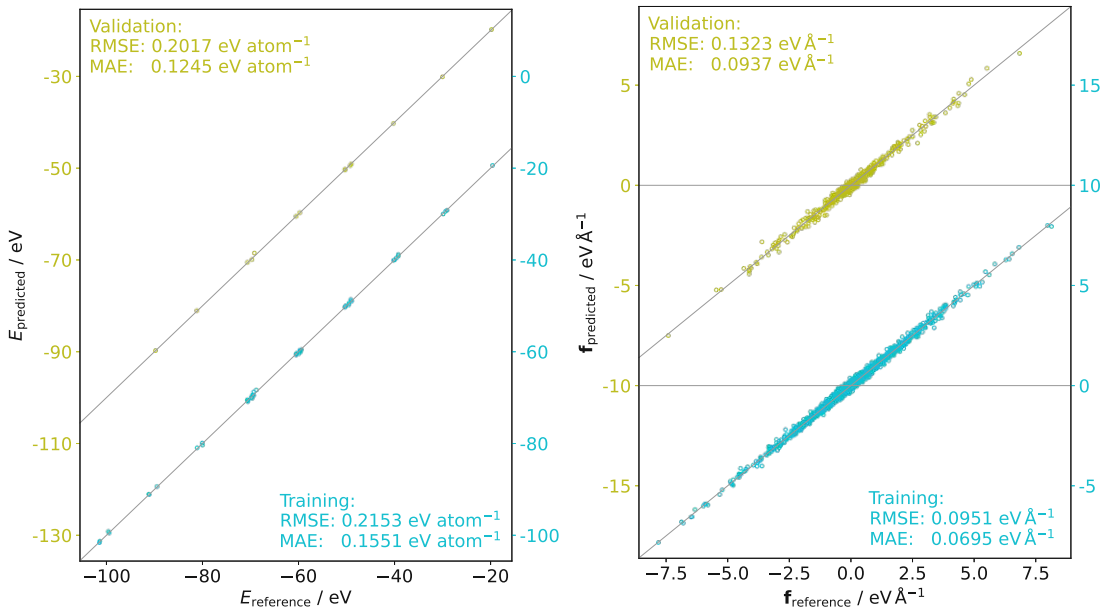
(a) Energies (b) Forces

Figure 8: Parity plots comparing predicted energies and forces to the reference quantities for EAN using an EPNN model trained directly on DDEC6 charges



(a) Energies (b) Forces

Figure 9: Parity plots comparing predicted energies and forces to the reference quantities for STO using an EPNN model trained directly on DDEC6 charges



(a) Energies (b) Forces

Figure 10: Parity plots comparing predicted energies and forces to the reference quantities for H_2O clusters using an EPNN model trained directly on DDEC6 charges

6.4 EPNN Training on Electrostatic Energies

As discussed in section 6.2 and shown in terms of the force predictions in section 6.3, charges obtained by partitioning of the electron density are not a suitable target quantity to reproduce for the inclusion of long-range interactions in NNFFs. EPNN models have therefore been trained to reproduce electrostatic energies: The predicted charges are used to compute the electrostatic energy given by the last term in equation (3.9), which is then passed as prediction input to the loss function. The model parameters are thus adjusted based on the difference between the predicted electrostatic energy and the electrostatic energy calculated using DFT. A single, trainable offset parameter was added to the model to be able to adjust for the origin of energies.

Results for models trained using initial atomic charges $q_a^0 = 0\,\mathrm{e}$ are visualized in figures 11 to 12. It is shown that the EPNN is able to predict charges that reproduce the total electrostatic energy of the system. However, the model for the H_2O cluster dataset appears not to be converged well, which can be attributed to a lack of training data. Since only the global electrostatic energy is available as information for training, the set is restricted to a total of 76 data points in total for training and validation.

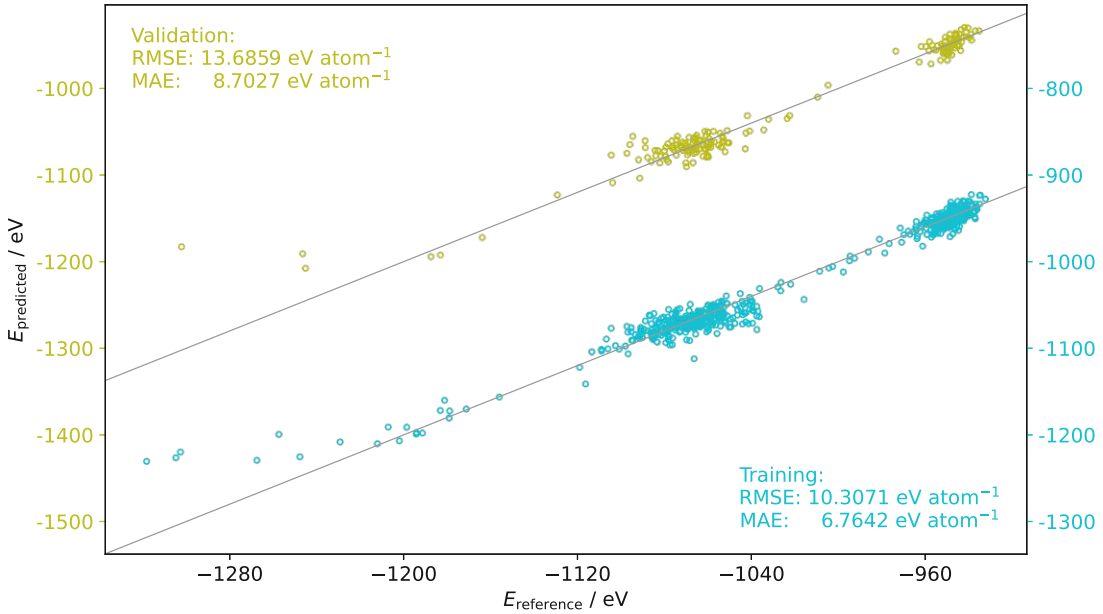


Figure 11: Parity plot comparing predicted energies to the reference energies for EAN using an EPNN model trained on electrostatic energies with initial atomic charges $q_a^0 = 0\,\mathrm{e}$

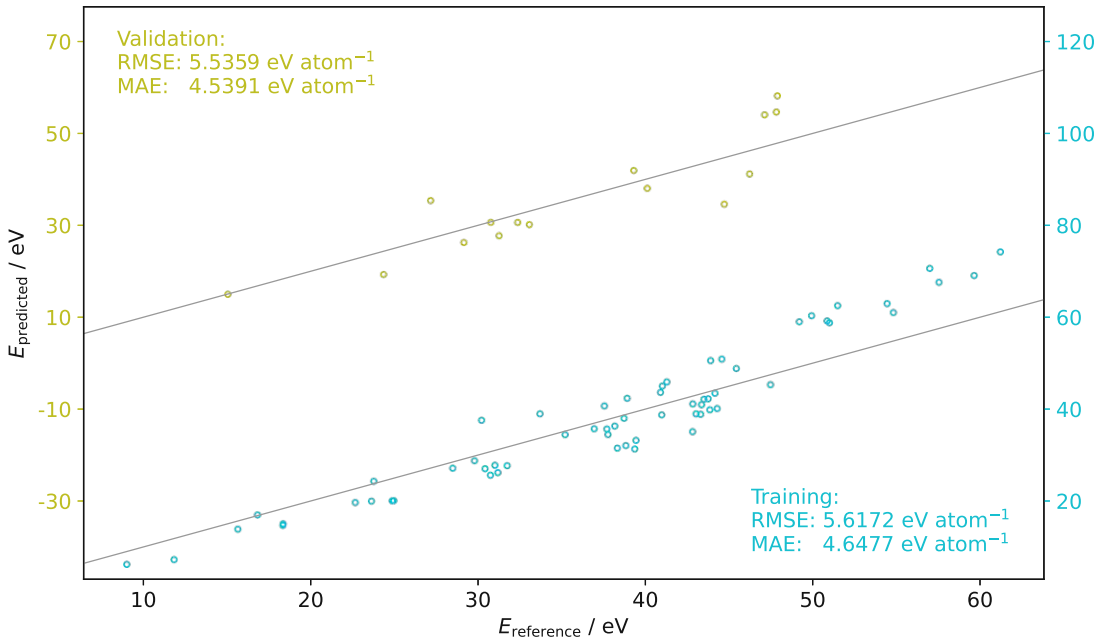
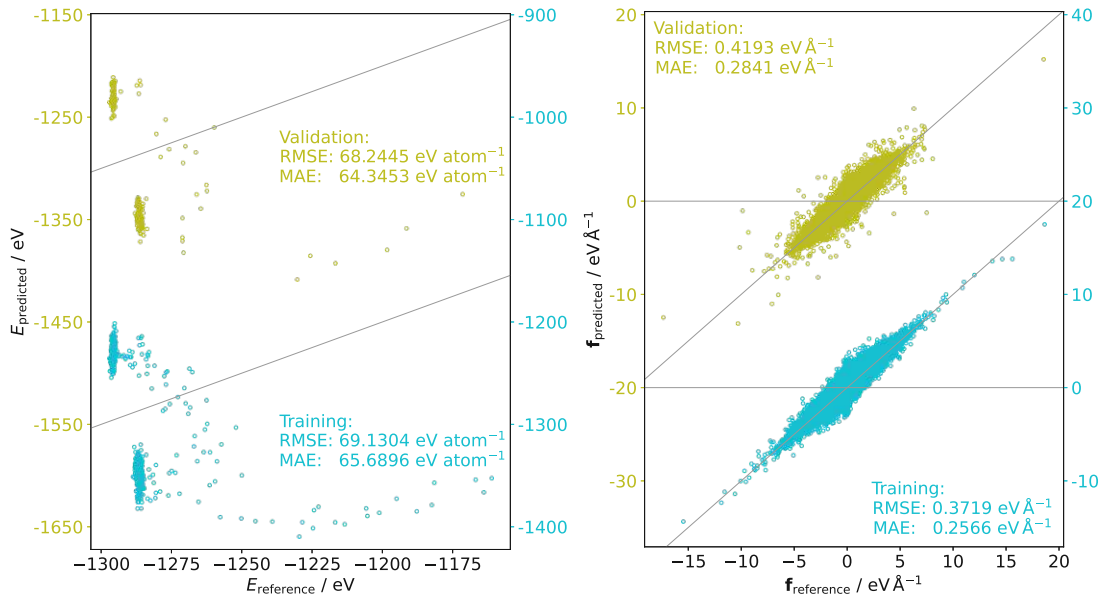


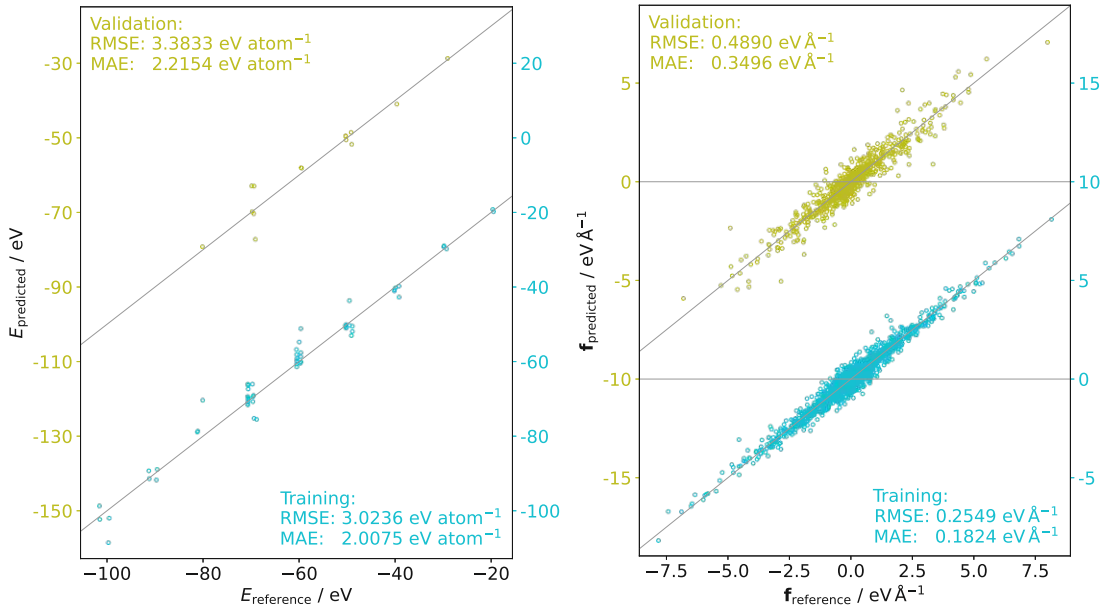
Figure 12: Parity plot comparing predicted energies to the reference energies for H_2O clusters using an EPNN model trained on electrostatic energies with initial atomic charges $q_a^0 = 0\,\mathrm{e}$

As in section 6.3, the EPNN models trained on electrostatic energies have again been used for the generation of inputs for augmented NeuralIL models. While the electrostatic energies themselves are predicted well, major errors arise for the total energy and force predictions. This is especially apparent in figure 13a, where a bipartite structure is observed in the energy predictions for both the training and validation sets. Only the global electrostatic energy is available as training information. Therefore, the optimization process is not guided in the direction of a specific minimum on the hypersurface in parameter space defined by the loss function. This is visualized in figure 15, in which the distributions of charges for all configurations in the EAN dataset are compared. While three different peaks in the relative frequencies are observed, the EPNN model trained with initial charges $q_a^0 = 0\,\mathrm{e}$ generates a distribution of charges that lead to cases, where the ionic charges of two different configurations are assigned an opposite sign. A global change of sign convention preserves electrostatic energies and forces, however, different conventions within a single batch lead to spurious pollution of the gradient of the loss function. Therefore, error statistics for force predictions are worse by a factor > 3 and a bipartite pattern in energy predictions is observed. One approach to break the degeneracy is to provide different initial charge states q_a^0, which has been tested by setting the DDEC6 charges as initial charges in the EPNN. However, while this leads to a different distribution of predicted charges, the same bipartite structure in the energy prediction is still observed.



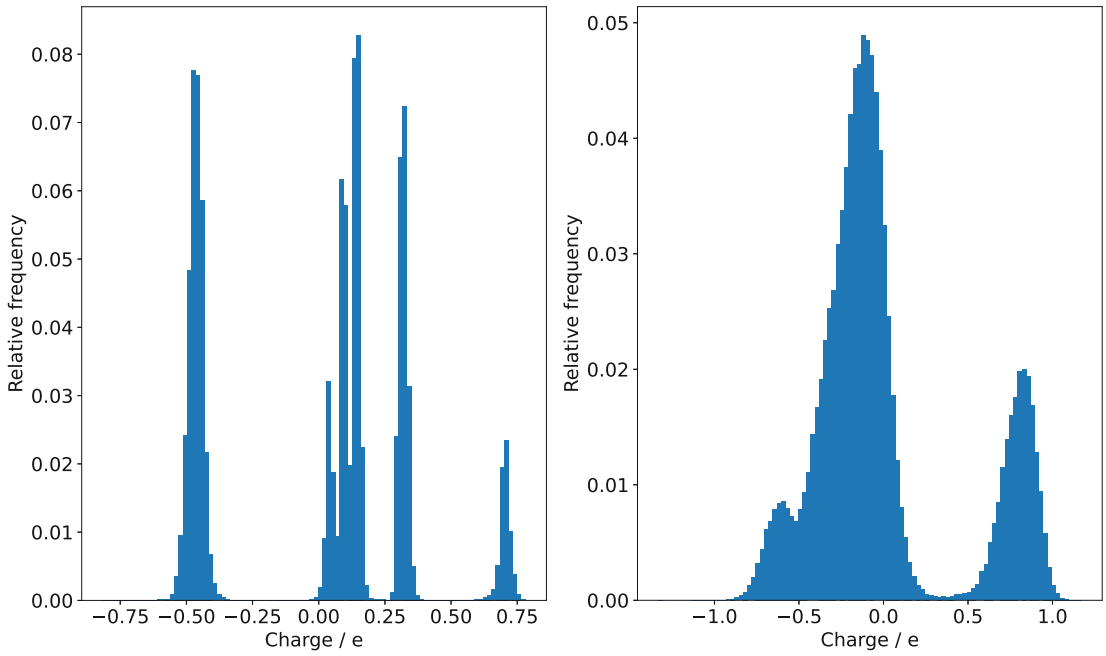
(a) Energies (b) Forces

Figure 13: Parity plots comparing predicted energies and forces to the reference quantities for EAN using an EPNN model trained on electrostatic energies with initial atomic charges $q_a^0 = 0\,\mathrm{e}$



(a) Energies (b) Forces

Figure 14: Parity plots comparing predicted energies and forces to the reference quantities for H_2O clusters using an EPNN model trained on electrostatic energies with initial atomic charges $q_a^0 = 0\,\mathrm{e}$



(a) DDEC6 charges (b) EPNN with initial $q_a^0 = 0\,\mathrm{e}$

Figure 15: Comparison of DDEC6 and EPNN predicted charge distributions for all EAN configurations

6.5 Electrostatic Forces in PAW

To provide local information, an expression for the electrostatic forces $\boldsymbol{f}_C^a = -\nabla_{\boldsymbol{R}^a} E_C$ is desirable to generate training data for integrating electrostatic interactions in neural-network force fields. Equation (2.45) is therefore analyzed in more detail with respect to the electrostatic energy. It is noted that the same system of units as in section 2 is used again in this section.

The term $\langle\tilde{\psi}_n|\nabla_{\boldsymbol{R}^a}\hat{E}_C|\tilde{\psi}_n\rangle$ is obtained by direct application of the gradient to the PAW energy expression derived in section 2.4:

$$\begin{aligned}
\nabla_{\boldsymbol{R}^a} E_C =& \nabla_{\boldsymbol{R}^a}\left(U_H[\tilde{\rho}] + \Delta E_C\left[\left\{D^a_{i_1 i_2}\right\}\right]\right) \\
=& \int \frac{\delta E_C}{\delta\tilde{n}(\boldsymbol{r}')}\nabla_{\boldsymbol{R}^a}\tilde{n}(\boldsymbol{r}')d\boldsymbol{r}' + \int\sum_{lm}\frac{\delta E_C}{\delta\tilde{g}^a_{lm}(\boldsymbol{r}')}\nabla_{\boldsymbol{R}^a}\tilde{g}^a_{lm}(\boldsymbol{r}')d\boldsymbol{r}' + \sum_{i_1 i_2}\frac{\partial E_C}{\partial D^a_{i_1 i_2}}\nabla_{\boldsymbol{R}^a}D^a_{i_1 i_2} \\
=& \int u_H(\boldsymbol{r}')\nabla_{\boldsymbol{R}^a}\tilde{n}^a_c(\boldsymbol{r}')d\boldsymbol{r}' + \int u_H(\boldsymbol{r}')\sum_{lm}Q_{lm}\nabla_{\boldsymbol{R}^a}\tilde{g}^a_{lm}(\boldsymbol{r}')d\boldsymbol{r}' \\
&+ \sum_{i_1 i_2}\frac{\partial E_C}{\partial D^a_{i_1 i_2}}\nabla_{\boldsymbol{R}^a}D^a_{i_1 i_2}
\end{aligned} \tag{6.4}$$

The last term in equation (6.4) remains to be treated. The gradient of the one-center density matrix with respect to atomic positions can be expressed directly via

$$\begin{aligned}
\nabla_{\boldsymbol{R}^a}D^a_{i_1 i_2} &= \nabla_{\boldsymbol{R}^a}\left(\sum_n f_n\langle\tilde{\psi}_n|\tilde{p}^a_{i_1}\rangle\langle\tilde{p}^a_{i_2}|\tilde{\psi}_n\rangle\right) \\
&= \sum_n f_n\left(\langle\tilde{\psi}_n|\nabla_{\boldsymbol{R}^a}\tilde{p}^a_{i_1}\rangle P^a_{ni_2} + P^{a*}_{ni_1}\langle\nabla_{\boldsymbol{R}^a}\tilde{p}^a_{i_2}|\tilde{\psi}_n\rangle\right)
\end{aligned} \tag{6.5}$$

The smooth charge density $\tilde{\rho}(\boldsymbol{r})$ also depends on the one-center density matrix through the smooth core charge density $\tilde{Z}^a(\boldsymbol{r})$, so the derivative of the electrostatic energy with respect to the elements of the one-center density matrix also has to include a term accounting for $\frac{\partial U_H}{\partial D^a_{i_1 i_2}}$:

$$\begin{aligned}\frac{\partial E_C}{\partial D^a_{i_1 i_2}} &= \frac{\partial U_H}{\partial D^a_{i_1 i_2}} + \frac{\partial \Delta E_C}{\partial D^a_{i_1 i_2}} = \frac{\delta U_H}{\delta \tilde{Z}^a}\frac{\partial \tilde{Z}^a}{\partial D^a_{i_1 i_2}} + \frac{\partial \Delta E_C}{\partial D^a_{i_1 i_2}} \\ &= \sum_{lm} \Delta^a_{lm,i_1 i_2} \int u_H[\tilde{\rho}](\boldsymbol{r}) \tilde{g}^a_{lm}(\boldsymbol{r}) d\boldsymbol{r} + \frac{\partial \Delta E_C}{\partial D^a_{i_1 i_2}}\end{aligned} \tag{6.6}$$

Finally, the derivative of the one-center correction tensor for the electrostatic energy (2.36) can also be directly evaluated:

$$\begin{aligned}\frac{\partial \Delta E_C}{\partial D^a_{i_1 i_2}} =& (\phi^a_{i_1}\phi^a_{i_2}|n^a_c) - (\tilde{\phi}^a_{i_1}\tilde{\phi}^a_{i_2}|\tilde{n}^a_c) - \mathcal{Z}^a \int \frac{\phi^a_{i_1}(\boldsymbol{r})\phi^a_{i_2}(\boldsymbol{r})}{r} d\boldsymbol{r} - \Delta^a(\tilde{\phi}^a_{i_1}\tilde{\phi}^a_{i_2}|\tilde{g}^a_{00}) \\ &- \sum_{lm} \Delta^a_{lm,i_1 i_2} \left[(\tilde{n}^a_c|\tilde{g}^a_{lm}) + \Delta^a(\tilde{g}^a_{00}|\tilde{g}^a_{00})\right] \\ &+ \sum_{i_3 i_4} \left[(\phi^a_{i_1}\phi^a_{i_2}|\phi^a_{i_3}\phi^a_{i_4}) - (\tilde{\phi}^a_{i_1}\tilde{\phi}^a_{i_2}|\tilde{\phi}^a_{i_3}\tilde{\phi}^a_{i_4})\right] D^a_{i_3 i_4} \\ &- \sum_{i_3 i_4} \left(\sum_{lm} (\tilde{\phi}^a_{i_1}\tilde{\phi}^a_{i_2}|\tilde{g}^a_{lm}) \Delta^a_{lm,i_3 i_4} + \Delta^a_{lm,i_1 i_2} (\tilde{\phi}^a_{i_3}\tilde{\phi}^a_{i_4}|\tilde{g}^a_{lm}) \right) D^a_{i_3 i_4} \\ &- \sum_{i_3 i_4} \left(\sum_{lm,l'm'} \Delta^a_{lm,i_1 i_2} (\tilde{g}^a_{lm}|\tilde{g}^a_{l'm'}) \Delta^a_{l'm',i_3 i_4} \right) D^a_{i_3 i_4}\end{aligned} \tag{6.7}$$

However, since the B_{mn} term is not known explicitly, the last term of equation (2.45) with respect to $\hat{E}_C$ instead of $\hat{\tilde{H}}$ remains unspecified. Other approaches, in particular density-functional perturbation theory [67], might provide a way of obtaining an expression for $\nabla_{\boldsymbol{R}^a} E_C$.

7 Conclusion and Outlook

Neural networks provide a promising framework for general and scalable parametrizations of force fields. Already in the form originally implemented in NeuralIL, energies and forces are predicted with DFT-level accuracy [4, 48], although the FF does not model the full physics of the system by construction. This validates the strategy of simply neglecting long-range contributions in situations where this leads to only a minimal loss of accuracy, for example in dense atomic arrangements. However, the inclusion of these interactions cannot be forgone in all cases, as for example with sparse arrangements or at, for example, liquid-vapor interfaces [5].

In this work, first steps towards the explicit inclusion of long-range electrostatic interactions in NeuralIL have been taken. It is shown that EPNNs are a viable strategy for the prediction of atomic charges. While EPNN models can be trained on charges obtained by partitioning of the electron density $n(\boldsymbol{r})$, it is preferable to use the network to rather model the electrostatic energy hypersurface. However, reproducing the correct behaviour leads to the challenge of finding the correct training data: While electrostatic energies can be obtained directly from DFT solvers, obtaining the corresponding forces $-\nabla_{\boldsymbol{R}^a} E_C$ proves to be a significant challenge in practical DFT schemes (section 6.5). Charges obtained from an EPNN model trained on the electrostatic energies only are observed to not follow a distribution which reproduces the correct corresponding forces.

One possible way to address this problem is to extend the training datasets with configurations obtained by applying small displacements to the atoms in the initial set of configurations. This adds configurations which lie close together in the space of atomic positions and thus emulates the inclusion of forces similar to the numeric computation of derivatives. A different approach is to use Born effective charges to add local information to the training process.

Furthermore, the modelling of electrostatic interactions as described in section 6.2 relies on prescribing the long-range behaviour using an unscreened Coulomb term. However, to achieve a more general and flexible method of modelling electrostatic interactions, a different functional form overall is necessary. Graph-based neural networks can still be of use here, for example by using such networks to predict the parameters for convolution kernels that account for long-range interactions.

References

[1] J. Behler and M. Parrinello. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Phys. Rev. Lett.* 98 (14 2007), p. 146401. DOI: 10.1103/PhysRevLett.98.146401.

[2] V. Botu et al. "Machine learning force fields: Construction, validation, and outlook". In: *J. Phys. Chem. C* 121 (1 2017), pp. 511–522. DOI: 10.1021/acs.jpcc.6b10908.

[3] J. S. Smith, O. Isayev, and A. E. Roitberg. "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost". In: *Chem. Sci.* 8 (4 2017), pp. 3192–3203. DOI: 10.1039/C6SC05720A.

[4] H. Montes-Campos et al. "A Differentiable Neural-Network Force Field for Ionic Liquids". In: *J. Chem. Inf. Model.* 62.1 (2022), pp. 88–101. DOI: 10.1021/acs.jcim.1c01380.

[5] S. Yue, M. C. Muniz, and M. F. Calegari Andrade. "When do short-range atomistic machine-learning models fall short?" In: *J. Chem. Phys.* 154 (3 2021). DOI: 10.1063/5.0031215.

[6] D. P. Metcalf et al. "Electron-Passing Neural Networks for Atomic Charge Prediction in Systems with Arbitrary Molecular Charge". In: *J. Chem. Inf. Model.* 61 (1 2021), pp. 115–122. DOI: 10.1021/acs.jcim.0c01071.

[7] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. "Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids". In: *J. Am. Chem. Soc.* 118 (45 1996), pp. 11225–11236. DOI: 10.1021/JA9621760.

[8] T. P. Senftle et al. "The ReaxFF reactive force-field: development, applications and future directions". In: *npj Comput. Mater.* 2.1 (2016), p. 15011. DOI: 10.1038/npjcompumats.2015.11.

[9] H. X. Zhou and X. Pang. "Electrostatic Interactions in Protein Structure, Folding, Binding, and Condensation". In: *Chem. Rev.* 118 (4 2018), pp. 1691–1741. DOI: 10.1021/acs.chemrev.7b00305.

[10] S. P. Niblett, M. Galib, and D. T. Limmer. "Learning intermolecular forces at liquid-vapor interfaces". In: *J. Chem. Phys.* 155 (16 2021). DOI: 10.1063/5.0067565.

[11] M. J. Gillan, D. Alfè, and A. Michaelides. "Perspective: How good is DFT for water?" In: *J. Chem. Phys.* 144 (13 2016), p. 130901. DOI: 10.1063/1.4944633.

[12] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136 (3B 1964), B864–B871. DOI: 10.1103/PhysRev.136.B864.

[13] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140 (4A 1965), A1133–A1138. DOI: 10.1103/PhysRev.140.A1133.

[14] K. Burke, J. Werschnik, and E. K. U. Gross. "Time-dependent density functional theory: Past, present, and future". In: *J. Chem. Phys.* 123 (6 2005), p. 062206. DOI: 10.1063/1.1904586.

[15] W. E. Pickett. "Pseudopotential methods in condensed matter applications". In: *Comput. Phys. Rep.* 9 (3 1989), pp. 115–197. DOI: 10.1016/0167-7977(89)90002-6.

[16] P. E. Blöchl. "Projector augmented-wave method". In: *Phys. Rev. B* 50 (24 1994), pp. 17953–17979. DOI: 10.1103/PhysRevB.50.17953.

[17] G. Kresse and D. Joubert. "From ultrasoft pseudopotentials to the projector augmented-wave method". In: *Phys. Rev. B* 59 (3 1999), pp. 1758–1775. DOI: 10.1103/PhysRevB.59.1758.

[18] A. Kiejna et al. "Comparison of the full-potential and frozen-core approximation approaches to density-functional calculations of surfaces". In: *Phys. Rev. B* 73 (3 2006), p. 035404. DOI: 10.1103/PhysRevB.73.035404.

[19] J. J. Mortensen, L. B. Hansen, and K. W. Jacobsen. "Real-space grid implementation of the projector augmented wave method". In: *Phys. Rev. B* 71 (3 2005), p. 035109. DOI: 10.1103/PhysRevB.71.035109.

[20] J. Enkovaara et al. "Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method". In: *J. Phys. Condens. Matter* 22 (25 2010), p. 253202. DOI: 10.1088/0953-8984/22/25/253202.

[21] J. Harris, R. O. Jones, and J. E. Müller. "Force calculations in the density functional formalism". In: *J. Chem. Phys.* 75 (8 1981), pp. 3904–3908. DOI: 10.1063/1.442546.

[22] T. W. Ko et al. "A fourth-generation high-dimensional neural network potential with accurate electrostatics including non-local charge transfer". In: *Nat. Commun.* 12 (398 2021). DOI: 10.1038/s41467-020-20427-2.

[23] A. P. Bartók and G. Csányi. "Gaussian Approximation Potentials: A Brief Tutorial Introduction". In: *Int. J. Quantum Chem.* 115 (16 2015), pp. 1051–1057. DOI: 10.1002/qua.24927.

[24] A. V. Shapeev. "Moment tensor potentials: A class of systematically improvable interatomic potentials". In: *Multiscale Model. Simul.* 14 (3 2016), pp. 1153–1173. DOI: 10.1137/15M1054183.

[25] J. Behler. "Atom-centered symmetry functions for constructing high-dimensional neural network potentials". In: *J. Chem. Phys.* 134 (7 2011). DOI: 10.1063/1.3553717.

[26] M. Rupp et al. "Fast and accurate modeling of molecular atomization energies with machine learning". In: *Phys. Rev. Lett.* 108 (5 2012), p. 058301. DOI: 10.1103/PhysRevLett.108.058301.

[27] A. P. Bartók, R. Kondor, and G. Csányi. "On representing chemical environments". In: *Phys. Rev. B* 87 (18 2013), p. 184115. DOI: 10.1103/PhysRevB.87.184115.

[28] E. Kocer, J. K. Mason, and H. Erturk. "Continuous and optimally complete description of chemical environments using Spherical Bessel descriptors". In: *AIP Adv.* 10 (1 2020). DOI: 10.1063/1.5111045.

[29] D. M. Anstine and O. Isayev. "Machine Learning Interatomic Potentials and Long-Range Physics". In: *J. Phys. Chem. A* (2023). DOI: 10.1021/acs.jpca.2c06778.

[30] Nongnuch Artrith and Alexander Urban. "An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO2". In: *Comput. Mater. Sci.* 114 (2016), pp. 135–150. DOI: 10.1016/J.COMMATSCI.2015.11.047.

[31] W. Li et al. "Study of Li atom diffusion in amorphous Li3PO4 with neural network potential". In: *J. Chem. Phys.* 147 (21 2017), p. 214106. DOI: 10.1063/1.4997242.

[32] Q. Li et al. "Development of robust neural-network interatomic potential for molten salt". In: *Cell Rep.* 2 (3 2021), p. 100359. DOI: 10.1016/j.xcrp.2021.100359.

[33] A. Y. Toukmaji and J. A. Board. "Ewald summation techniques in perspective: a survey". In: *Comput. Phys. Commun.* 95 (2-3 1996), pp. 73–92. DOI: 10.1016/0010-4655(96)00016-1.

[34] A. E. Sifain et al. "Discovering a Transferable Charge Assignment Model Using Machine Learning". In: *J. Phys. Chem. Lett.* 9 (16 2018), pp. 4495–4501. DOI: 10.1021/ACS.JPCLETT.8B01939.

[35] O. T. Unke and M. Meuwly. "PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges". In: *J. Chem. Theory Comput.* 15 (6 2019), pp. 3678–3693. DOI: 10.1021/ACS.JCTC.9B00181.

[36] E. R. Khajehpasha et al. "CENT2: Improved charge equilibration via neural network technique". In: *Phys. Rev. B* 105 (14 2022), p. 144106. DOI: 10.1103/PhysRevB.105.144106.

[37] L. Zhang et al. "A deep potential model with long-range electrostatic interactions". In: *J. Chem. Phys.* 156 (2022), p. 124107. DOI: 10.1063/5.0083669.

[38] A. Gao and R. C. Remsing. "Self-consistent determination of long-range electrostatics in neural network potentials". In: *Nature Comm.* 13 (1 2022), pp. 1–11. DOI: 10.1038/s41467-022-29243-2.

[39] A. Kosmala et al. *Ewald-based Long-Range Message Passing for Molecular Graphs.* 2023. DOI: 10.48550/arxiv.2303.04791.

[40] J. Gilmer et al. *Neural Message Passing for Quantum Chemistry.* 2017. DOI: 10.48550/arxiv.1704.01212.

[41] J. Zhou et al. *Graph Neural Networks: A Review of Methods and Applications.* 2018. DOI: 10.48550/arxiv.1812.08434.

[42] Z. Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Trans. Neural. Netw. Learn. Syst.* 32 (1 2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[43] D. Duvenaud et al. *Convolutional Networks on Graphs for Learning Molecular Fingerprints*. 2015. URL: https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

[44] T. N. Kipf and M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2016. DOI: 10.48550/arxiv.1609.02907.

[45] B. Anderson, T. Hy, and R. Kondor. *Cormorant: Covariant Molecular Neural Networks*. 2019. DOI: 10.48550/arXiv.1906.04015.

[46] D. Van Der Spoel et al. "GROMACS: Fast, flexible, and free". In: *J. Comput. Chem.* 26 (16 2005), pp. 1701–1718. DOI: 10.1002/JCC.20291.

[47] S. V. Sambasivarao and O. Acevedo. "Development of OPLS-AA force field parameters for 68 unique ionic liquids". In: *J. Chem. Theory Comput.* 5 (4 2009), pp. 1038–1050. DOI: 10.1021/CT900009A.

[48] R. Wanzenböck et al. "Neural-network-backed evolutionary search for SrTiO3(110) surface reconstructions". In: *Digital Discovery* 1 (5 2022), pp. 703–710. DOI: 10.1039/D2DD00072E.

[49] J. Řezáč et al. "Quantum Chemical Benchmark Energy and Geometry Database for Molecular Clusters and Complex Molecular Systems (www.begdb.com): A Users Manual and Examples". In: *Collect. Czechoslov. Chem. Commun.* 73 (10 2008), pp. 1261–1270. DOI: 10.1135/CCCC20081261.

[50] J. P. Perdew, K. Burke, and M. Ernzerhof. "Generalized Gradient Approximation Made Simple". In: *Phys. Rev. Lett.* 77 (18 1996), pp. 3865–3868. DOI: 10.1103/PhysRevLett.77.3865.

[51] B. Hammer, L. B. Hansen, and J. K. Nørskov. "Improved adsorption energetics within density-functional theory using revised Perdew-Burke-Ernzerhof functionals". In: *Phys. Rev. B* 59 (11 1999), pp. 7413–7421. DOI: 10.1103/PhysRevB.59.7413.

[52] T. A. Manz and N. G. Limas. "Introducing DDEC6 atomic population analysis: part 1. Charge partitioning theory and methodology". In: *RSC Adv.* 6 (53 2016), pp. 47771–47801. DOI: 10.1039/C6RA04656H.

[53] T. A. Manz and N. G. Limas. *Chargemol program for performing DDEC analysis*. Version 3.15. 2017. URL: http://ddec.sourceforge.net.

[54] J. Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[55] D. Chen et al. "Deep Residual Learning for Nonlinear Regression". In: *Entropy* 22 (2 2020), p. 193. DOI: 10.3390/e22020193.

[56] L. Metz et al. *VeLO: Training Versatile Learned Optimizers by Scaling Up*. 2022. DOI: 10.48550/arxiv.2211.09760.

[57] J. Carrete et al. *Deep Ensembles vs. Committees for Uncertainty Estimation in Neural-Network Force Fields: Comparison and Application to Active Learning*. 2023. DOI: 10.48550/arxiv.2302.08805.

[58] J. Godwin et al. *Jraph: A library for graph neural networks in jax.* Version 0.0.1.dev. 2020. URL: http://github.com/deepmind/jraph.

[59] J. Heek et al. *Flax: A neural network library and ecosystem for JAX.* Version 0.3.0. 2020. URL: http://github.com/google/flax.

[60] I. Babuschkin et al. *The DeepMind JAX Ecosystem.* 2020. URL: http://github.com/deepmind.

[61] A. G. Baydin et al. "Automatic Differentiation in Machine Learning: a Survey". In: *J. Mach. Learn. Res.* 18.153 (2018), pp. 1–43. DOI: 10.48550/arXiv.1502.05767.

[62] N. A. Spaldin. "A beginner's guide to the modern theory of polarization". In: *J. Solid State Chem.* 195 (2012), pp. 2–10. DOI: 10.1016/J.JSSC.2012.05.010.

[63] P. Ramachandran, B. Zoph, and Q. V. Le. *Searching for Activation Functions.* 2017. DOI: 10.48550/arXiv.1710.05941.

[64] Q. Wang et al. "A Comprehensive Survey of Loss Functions in Machine Learning". In: *Ann. Data Sci.* 9 (2 2022), pp. 187–212. DOI: 10.1007/S40745-020-00253-5.

[65] P. Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nat. Methods* 17 (3 2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[66] F. L. Hirshfeld. "Bonded-atom fragments for describing molecular charge densities". In: *Theor. Chem. Acc.* 44 (2 1977), pp. 129–138. DOI: 10.1007/BF00549096.

[67] S. Baroni et al. "Phonons and related crystal properties from density-functional perturbation theory". In: *Rev. Mod. Phys.* 73 (2 2001), pp. 515–562. DOI: 10.1103/RevModPhys.73.515.